Fast and Accurate Data-Driven Goal Recognition Using Process Mining Techniques

Zihang Su[a,*], Artem Polyvyanyy[a], Nir Lipovetzky[a], Sebastian Sardiña[b], Nick van Beest[c]

[a]*The University of Melbourne, Parkville, VIC 3010, Australia*
[b]*Royal Melbourne Institute of Technology, 124 La Trobe St., Melbourne, VIC 3000, Australia*
[c]*CSIRO, 41 Boggo Road, Dutton Park, QLD 4102, Australia*

Abstract

The problem of *goal recognition* requests to automatically infer an accurate probability distribution over possible goals an autonomous agent is attempting to achieve in the environment. The state-of-the-art approaches for goal recognition operate under full knowledge of the environment and possible operations the agent can take. This knowledge, however, is often not available in real-world applications. Given historical observations of the agents' behaviors in the environment, we learn *skill models* that capture how the agents achieved the goals in the past. Next, given fresh observations of an agent, we infer their goals by diagnosing deviations between the observations and all the available skill models. We present a framework that serves as an outline for implementing such data-driven goal recognition systems and its instance system implemented using process mining techniques. The evaluations we conducted using our publicly available implementation confirm that the approach is well-defined, i.e., all system parameters impact its performance, has high accuracy over a wide range of synthetic and real-world domains, which is comparable with the more knowledge-demanding state-of-the-art approaches, and operates fast.

Keywords: Goal recognition, data-driven, process mining, autonomous agent, fast and accurate

1. Introduction

Goal Recognition (GR) techniques aim to infer the intentions of an autonomous agent according to the observed actions of that agent [1, 2]. As humans can observe and understand another agent's intentions, GR techniques aim to mimic some extent of this human's ability. Three concepts are inherent to the understanding of GR: A *plan* is composed of actions that were or should be taken to achieve a certain goal; An *agent*, such as a robot or human, follows such plans to accomplish goals; A *GR system* is software that implements a GR technique capable of inferring the goals of agents based on partial knowledge about the plans. When a GR system analyzes actions executed by some agent, it aims to detect the plan that the agent is following and, hence, the goal that will be achieved after completing that plan.

Nowadays, many tasks traditionally assigned to humans are performed by robots or software. From smart houses to autonomously driving cars, understanding what other agents are trying

*Corresponding author
Email address: zihangs@student.unimelb.edu.au (Zihang Su)

Postprint, July 2023

to accomplish is paramount to implementing intelligent behavior and effective human-machine interaction. For example, an intelligent driving system should figure out other vehicles' intentions or next actions to ensure safety. A smart house ought to understand whether the household is trying to cook, watch a movie, or sleep to provide relevant help. GR techniques play an important role in these, and many other areas, including the support of adversarial reasoning [3, 4], trajectory/maneuver prediction [5, 6, 7], and human-computer collaboration [8].

The existing GR techniques can be broadly classified into the following three categories: (i) Observations of an agent's actions are "matched" to a plan (the one judged to be carried out by the agent) in a predefined library encoding the standard operational procedures of the domain [1, 2, 9], namely, plan-library-based GR approaches; (ii) Appealing to the principle of rational behavior, an agent is assumed to be taking the "optimal" plan to the goal: the more rational a behavior appears toward a goal, the more likely such goal is the agent's goal. Ramirez and Geffner [10, 11] have sparked a plethora of approaches not needing any a priori set of plans. These approaches perform GR by exploiting planning systems to automatically generate plans relative to a domain theory, namely, planning-based GR approaches; (iii) Learning-based GR approaches learn domain models or prediction models from data, for example, sequences of observed states, as illustrated by Amado et al. [12], perform GR over image sequences used to train an autoencoder learning the transition function, see the approach by Amado et al. [13] that learns Q-value functions to infer probabilities toward the goal candidates, and learn interpretable dynamics through deterministic finite automata, as in the approach by Shvo et al. [14].

The challenge for plan-library-based approaches is in obtaining or hand-coding a possible set of plans for achieving the candidate goals. In addition, these approaches do not accommodate uncertainty (i.e., they often can not generalize to the observations that are not pre-stored in the plan library). For planning-based approaches, even though specifying domain models could be less demanding than hand-coding of plans and "new" plans can be found, acquiring domain models is far from trivial due to the difficulty of defining models using standard declarative languages [15]. It is especially challenging to acquire domain models of real-world environments, which are subject to continuous changes [16]. As a result, planning-based approaches are difficult to apply in real-world scenarios. Finally, one of the main challenges of the learning-based approaches is determining the scope and obtaining a sufficient volume of data that can be used to train effective models for goal recognition.

In this paper, we present a framework, and its concrete instantiation, for implementing GR systems that do *not* require hand-coded plans or domain models for inference. Instead, the framework proposes to utilize *process mining* [17] techniques to automatically learn *process models* from *event logs* of historical observations of agents that encode the *skills* for achieving various goals in the environment, and to analyze deviations from the observed agent's behavior from the learned skill models for inference. A process, or a skill, model in this work is a Petri net (see Section 3.2.1), with its executions encoding action sequences (plans) executed by an agent for achieving a certain goal. The event logs are *not* plan libraries, and the learned process models stand for execution instances of some underlying "hidden" standard operational procedures of the domain. Therefore, our approach overcomes the challenges associated with handcrafting the plan libraries and domain models. Moreover, compared with the learning-based approaches, our GR approach learns models from sequences of actions, while the other approaches often require additional information, such as the environment states before and after the executed actions or the transition functions for every observed action [12, 13, 14]. We assume the agent is acting in an unknown environment that can be described, for instance, in a STRIPS [18] or PDDL [15] model that specifies the states and the dynamics of the environment. Learned Petri nets then

aim to describe a subset of goal-relevant action sequences—plans—in the environment. Note we do not use Petri nets to represent the underlying dynamic domain, and as such states in the net do not correspond to states in the domain. We argue that our learning-based proposal sits between plan-library-based approaches, which base reasoning on exemplary plans, and planning-based approaches, which base reasoning on cost differences between optimal and observed plans. The performed evaluations demonstrate that our GR system can make inferences fast, while its accuracy is comparable with the state-of-the-art GR techniques. The fast recognition speed brings advantages to applying our GR system to time-sensitive scenarios such as a smart home system [19], where the household expects a smooth user experience.

This paper is an extended version of our conference paper [20]. The conference paper made these contributions:

- It proposed a GR framework that describes the fundamental mechanisms for performing GR without predefined plan libraries and domain models;
- It discussed an implementation of a concrete GR approach based on process mining techniques that follows the proposed GR framework and relies on three parameters to construct a probability distribution over possible goals and to infer the most likely goal. The three parameters are a "smoothening" constant (ϕ) that flattens the probability distribution over possible goals, a consecutive mismatch suffix factor (λ) that detects whether the agent is deviating from a candidate goal, and a discount factor (δ) that emphasizes the recently observed actions to have more impact on the goal inferences;
- It presented experimental results for two traditional planning domains, namely "grids" and "blocks-world" from the International Planning Competition (IPC), and three real-world datasets, namely "daily activities," "building permit," and "environment permit" to demonstrate that our GR approach can overcome the limitations of the classical planning-based GR systems.

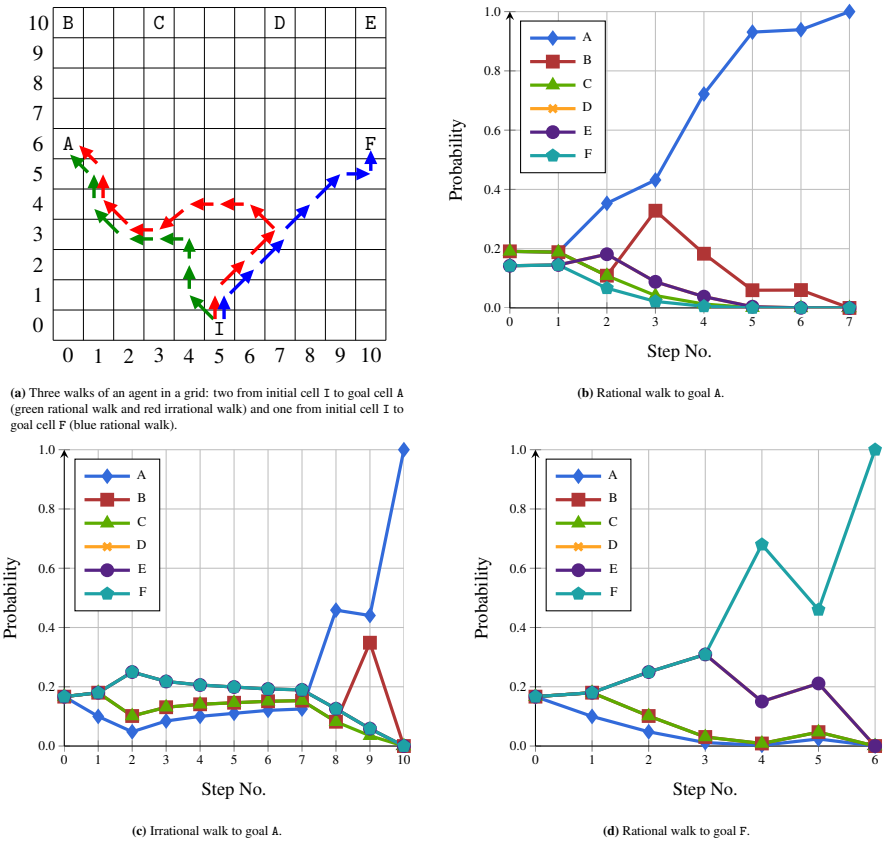
This paper extends the conference paper with the following contributions:

- It extends the original GR approach by introducing a new parameter, called *decision threshold* (θ), which controls the selection of the most likely goals;
- It presents results of a sensitivity analysis over 15 IPC domains and ten real-world domains that confirm that all four parameters (ϕ, λ, δ, and θ) have a significant impact on the performance of our GR approach;
- It presents a scenario discovery method for identifying parameters that lead to better performance of our GR approach;
- It confirms, via an evaluation over synthetic datasets generated by diverse planners, that our GR approach can accurately recognize goals even when trained on non-optimal plans;
- It summarizes the insights of a comprehensive comparison of the performance of our GR approach with the state-of-the-art techniques, which show that our approach achieves a comparable performance and is often faster;
- It demonstrates that our GR approach is applicable in real-world scenarios.

The next section presents motivating examples of goal recognition based on historical observations of agents' behaviors. Next, Section 3 presents the background in goal recognition and process mining required for understanding the subsequent discussions. Section 4 is devoted to presenting the GR framework and our GR system grounded in process mining. Subsequently, Section 5 presents the results of an evaluation of our implementation of the GR system, followed by a discussion of related work in Section 6. Finally, we summarize our contributions and conclude the paper in Section 7.

2. Motivating Examples

Figure 1a shows an 11x11 grid including the initial state in cell I and six goals represented by cells A to F; this grid is similar to the grid used in [11]. The grid also shows three observed walks of an agent, comprising a rational walk toward goal A (green), an irrational walk toward goal A (red), and a rational walk toward goal F (blue). To achieve a goal, the agent can perform horizontal and vertical steps at the cost of 1 and diagonal steps at the cost of $\sqrt{2}$.



(a) Three walks of an agent in a grid: two from initial cell I to goal cell A (green rational walk and red irrational walk) and one from initial cell I to goal cell F (blue rational walk).

(b) Rational walk to goal A.

(c) Irrational walk to goal A.

(d) Rational walk to goal F.

Figure 1: Inferred probability distributions (Fig. 1b, 1c, and 1d) over the six goals from Fig. 1a computed based on the observed behaviors shown in Fig. 2.

The green walk has a cost of $5 + 3\sqrt{2}$. As this cost is close to the cost of the optimal walk from I to A (i.e., $1 + 5\sqrt{2}$), we say that it is *rational*. The red walk starts by approaching goal F before diverting toward reaching target goal A, resulting in a cost of $5 + 6\sqrt{2}$. Hence, the red walk is *irrational*. Finally, the blue walk toward goal F has a cost of $3 + 4\sqrt{2}$, which is close to the cost of the optimal walk from cell I to cell F and is, therefore, rational.

Although the surrounding environment of an agent is often unknown, the observed action sequences executed by the agent can be used to learn models that explain the behavior for achieving different goals. For each of the six goals from Fig. 1a, Fig. 2 shows "footprints" of six observed walks of the agent from cell I to that goal. The thickness of an arrow in the figure indicates the frequency with which the corresponding step was taken. Using the observations, we

construct process models that describe *skills* required for accomplishing the goals. Subsequently, new observations of an agent in the environment can be compared with the acquired models to identify the discrepancies between the newly observed behavior and the skill models. Intuitively, the more discrepancies between the observed behavior and a skill model, the less likely the agent is attempting to achieve the corresponding goal. Using *process mining* techniques [17] (see Section 3.2 for details), we discover the skill models from the historical observations and identify and compare the discrepancies between the observed behavior and the skill models. We use the identified discrepancies to compute the probability distributions over the goals as functions of time (i.e., the number of steps) for each observed behavior.

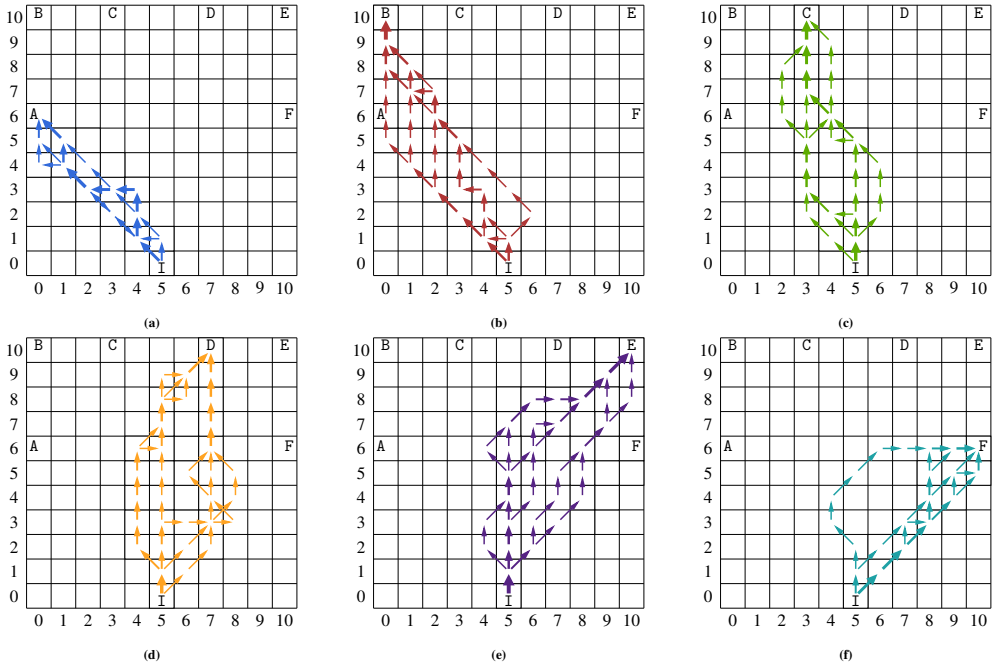


Figure 2: Observed agent behaviors from initial cell `I` to each of the goals `A`–`F`.

Figures 1b to 1d show the probability distributions over the candidate goals computed by our GR system for the three walks from Fig. 1a. As expected, along the green rational walk, goal `A` is consistently the most likely goal; see Fig. 1b. For the first seven steps of the red irrational walk, however, goals `E` and `F` prevail, while goal `A` is identified as the most likely goal only toward the end of the walk; see Fig. 1c. Finally, the blue rational walk toward goal `F` shows an equal probability toward goals `E` and `F` for the first three steps (the same confusion as for the first steps of the irrational walk), with the probability for goal `F` prevailing from step four onwards; see Fig. 1d. Empirical evidence suggests (refer to Section 5) that our GR system can be used to fast and accurately infer the intended goals of agents for a wide range of domains.

3. Background

This section provides the technical background required for understanding our contribution.

3.1. Goal Recognition

Goal Recognition (GR) is the problem of recognizing an agent's intention (i.e., its goal) according to its observed behavior.[1] The problem was first introduced by Schmidt et al. [21] in their psychological-based BELIEVER system and arguably first formalized by Kautz and Allen [2] as a logic-based (non-monotonic) deductive reasoning task over an action taxonomy encoding plan decompositions. A common feature of the early and traditional work in GR is the reliance on pre-defined *plan libraries* (e.g., a hierarchical tree or network of goals and actions) that are meant to encode the known "operational plans" of the domain [22]. Those plan libraries are then parsed and "matched" against the observed sequences of action observations. For example, in Kautz and Allen's approach, the plan library is represented as a hierarchical action graph (e.g., "Make Pasta Dish" and "Make Meat Dish" are the lower-level actions of "Prepare Meal") that allows high-level actions to be deduced from the observations of actions at the lower levels. Avrahami-Zilberbrand and Kaminka [23], in turn, use the so-called Feature Decision Tree (FDT) as a way for encoding the library of plans, whereas Pynadath and Wellman [24] use grammar representations augmented with probabilities and state information. The fact is that, in many domains, crafting those libraries could be costly or simply not feasible. In addition, GR approaches requiring the specification of plan libraries may not be able to adequately handle behavior outside the provided plans.

Possibly the first step toward plan-library-"free" GR was Hong's proposal [25], in which the so-called Goal Graph is constructed from the specification of a set of primitive actions and then analyzed against the observed actions to extract consistent goals and valid plans. It was then the work of Ramírez and Geffner [10, 11] that provided an elegant GR approach by leveraging on the representational and algorithmic techniques in AI model-based automated planning. Intuitively, the authors drew from the insight that a *rational* agent is expected to be taking the optimal, or close to optimal, plan to its (hidden) goal, a point also noted elsewhere [26, 27], so the probability of a plan can be linked to its *cost*. The main ingredient is that the relevant costs can be computed by automatically synthesizing adequate plans relative to the observations seen using planning technology [28]. By doing this, pre-defined plans are abandoned and replaced by – hopefully less onerous – *declarative models* of the world dynamics, which are well-studied in the automated planning and reasoning about action and change communities. The planning-based probabilistic goal recognition problem is defined as follows.

Definition 1 (Planning-based probabilistic goal recognition). Given a tuple $< F, A, I, \mathcal{G}, \tau >$, where F is a set a *fluents*, A is a set of *actions*, $I \subseteq F$ is the *initial state*, $\mathcal{G} \subseteq 2^F$ is a set of candidate *goals* (each given as a set of fluents that ought to be true in the corresponding goal state), and $\tau \in A^*$ is an *observation* given as a sequence of actions performed by an agent, the *planning-based probabilistic goal recognition* problem consists in obtaining a *posterior probability distribution* over the candidate goals that describes the true goal of the agent.

In their 2010 work, Ramírez and Geffner derive such probability distribution over the possible goals from Bayes' Rule based on the assumption that *the probability of a plan is inversely proportional to its cost*. Such assumption is encapsulated in the notion of *cost difference* between the (cost of the) optimal plan for a goal matching the observed actions and the optimal plan that could have been reached otherwise, that is, not embedding the observed actions. To compute those costs, planning systems are used over specific encodings of the domain that also account

[1] We refer to the problems of goal recognition, intention recognition, and plan recognition interchangeably.

for the observations. Ultimately, this yields a Boltzmann-like sigmoidal distribution with the important property that *the lower the cost difference, the higher the probability*.

Several works have subsequently elaborated Ramírez and Geffner's set-up (which we will refer to as R&G from now on), or grounded it to specific interesting settings, such as navigation. Here, we shall adopt the most recent elaboration by Masters and Sardiña [29, 30], which refined the original set-up to achieve a simpler and computationally less demanding GR approach, and one able to handle irrational agent behavior parsimoniously without counter-intuitive outcomes. Concretely, taking $optc(S, \tau, G)$ to denote the optimal cost of reaching goal G from state $S \subseteq F$ by *embedding* the sequence of observations τ, we first define the ***cost difference*** of reaching a goal G from S via observations τ as follows:[2]

$$costdiff(S, \tau, G) = optc(S, \tau, G) - optc(S, \epsilon, G).$$

When the agent is observed to act optimally for G, the cost difference is zero; as the agent becomes more suboptimal toward G, the cost difference increases. Here, the difference to the R&G approach is that the "cost difference" is calculated against the optimal cost (without knowing any observations) to reach the goals, rather than the optimal cost *without executing some observed actions*, which is known to be computationally demanding [29, 30].

Using the cost difference, and assuming for simplicity that all goals are initially equally likely, the probability of a candidate goal $G \in \mathcal{G}$ given observations τ from initial state I can be obtained as follows (note the denominator acts here as the normalization factor) [11, 31]:

$$\Pr(G \mid \tau) = \frac{e^{-\beta \times costdiff(I,\tau,G)}}{\sum_{G' \in \mathcal{G}} e^{-\beta \times costdiff(I,\tau,G')}}, \tag{1}$$

where β is a parameter that "allows the goal recognition system developers to soften the implicit assumption of the agent being rational" [11]. This account yields the principle that the more suboptimal an agent acts for a potential goal, the higher the cost difference, hence the lower the probability. Nonetheless, we adopt Masters and Sardiña [31, 32]'s approach—in turn, inspired in the cost-ratio used by Vered et al. [33]—that lifts the original requirement that agents ought to be rational (or close to rational) by *dynamically modulating* the β parameter using a *rationality measure* of the observed agent as follows:

$$\beta = \left(\max_{G \in \mathcal{G}} \frac{optc(I, \epsilon, G)}{optc(I, \tau, G)} \right)^{\eta}. \tag{2}$$

Intuitively, this β expresses the most optimistic rationality ratio among all goals, with $\beta = 1$ when the agent is seen fully rational toward *some* goal. By using this dynamic parameter, the more erratic the agent behavior (irrational to all goals), the more $\Pr(\cdot)$ approaches a uniform distribution. By doing this, the resulting GR system is capable of self-modulating its *confidence* as observations are gathered (η is a positive constant regulating how quickly confidence should drop when irrational behavior is seen).

In this paper, we adapt and instantiate the above approach to apply notions and techniques from *process mining* [17].

[2] Under no observations (i.e., $\tau = \epsilon$), $optc(S, \tau, G)$ reduces to optimal cost to G from state S.

3.2. Process Mining

Process mining studies methods, techniques, and tools to discover, monitor, and improve processes carried out by organizations using the knowledge accumulated in event logs recorded by information systems that support the execution of business processes [17].

An ***event log***, or *log*, is a collection of traces, where each ***trace*** consists of a sequence of timestamped ***events*** observed and recorded during the execution of a single case of a business process. Each event in such a log refers to an action executed by an agent at a particular time and for a particular case. Let $\mathcal{E}$ be a universe of events. Then an event log is defined as follows.

Definition 2 (Trace, Event log). A *trace* τ is a finite sequence of n events $\langle e_1, \ldots, e_n \rangle$, with $e_i \in \mathcal{E}$ and $i \in [1..n]$. An *event log*, or *log*, L is a finite collection of traces over $\mathcal{E}$.

For example, one can encode each collection of six sequences of actions toward each of the six goals from Fig. 2 in an event log of six traces. An event in a trace of such an event log encodes a move in the grid and can be specified as a pair of two cells: the source cell and the target cell. For example, by $\binom{5,0}{4,1}$, we denote the event of the agent moving from cell $(5,0)$ to cell $(4,1)$ in the grid. Let $L_{\mathtt{A}} = \{\tau_1, \ldots, \tau_6\}$ be the log that contains six traces, each capturing the moves from some walk toward goal `A` shown in Fig. 2a. The six traces in log $L_{\mathtt{A}}$ are specified in Fig. 3.

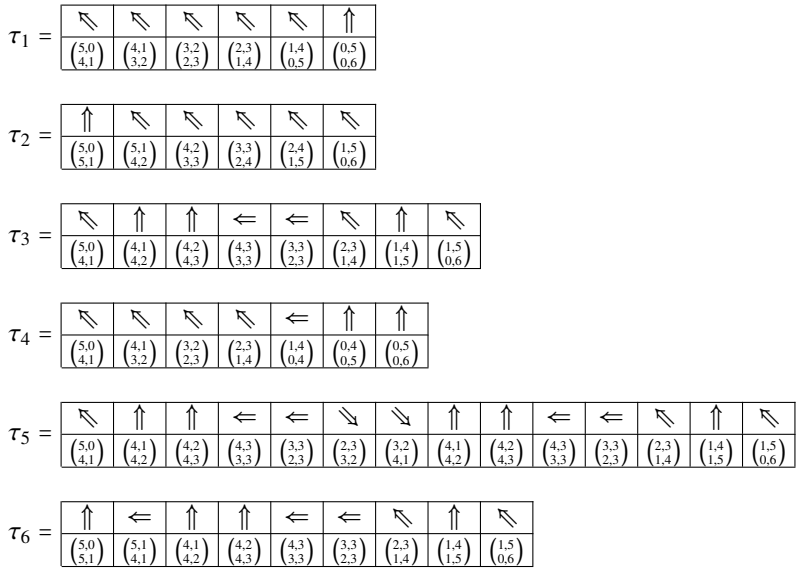


Figure 3: Event log $L_{\mathtt{A}}$ representing the walks to goal `A` shown in Fig. 2a.

In the figure, a trace is specified as a table with two rows, where a column encodes one event. The bottom row specifies the moves, while the top row visualizes the moves as arrows pointing in the directions of the moves. Thus, trace τ_1 in Fig. 3 consists of six events. The first five events encode diagonal north-west moves that take the agent from cell $(5,0)$ to cell $(0,5)$, and the last event encodes the move from cell $(0,5)$ to cell $(0,6)$, thus going north and reaching the goal.

3.2.1. Process Discovery

The behavior captured by the traces of an event log can be represented graphically by means of a process model, such as a Petri net, UML Activity Diagram, or an Event-Driven Process

Chain [17]. Process discovery is a technique that automatically constructs a process model from an event log [34]. Given a universe of logs $\mathbb{L}$ and a universe of process models $\mathbb{M}$, a ***process discovery technique*** π is a function that maps event logs onto process models, i.e., $\pi : \mathbb{L} \to \mathbb{M}$. We say that the model $\pi(L)$, $L \in \mathbb{L}$, is discovered from event log L using technique π.

Petri nets provide a convenient way to describe and analyze traces rigorously. A Petri net is a directed bipartite graph with two types of nodes: *places* (graphically denoted by circles) and *transitions* (graphically denoted by rectangles). Nodes of a Petri net are connected via directed edges called *arcs*. Transitions of a Petri net represent actions, or events, while places represent conditions. Let $\mathcal{L}$ be a universe of labels, then a Petri net is defined as follows [35].

Definition 3 (Petri net). A *Petri net* is a tuple (P, T, A, λ), where:

- P is a finite set of *places*,
- T is a finite set of *transitions*, such that $P \cap T = \varnothing$,
- $A \subseteq (P \times T) \cup (T \times P)$ is a set of *arcs*, and
- $\lambda : P \cup T \to \mathcal{L}$ is a *labeling* function.

A state M of a Petri net, often referred to as a *marking*, is a function that associates places of the Petri net with numbers, i.e., $M : P \to \mathbb{N}_0$. A *marked net*, or a *net*, is a tuple (P, T, A, λ, M_0), where (P, T, A, λ) is a Petri net and $M_0 : P \to \mathbb{N}_0$ is the *initial marking*. The *preset* of a node $y \in P \cup T$ is denoted by $\bullet y$ and is the set of all input nodes of y, i.e., $\bullet y = \{x \in P \cup T \mid (x, y) \in A\}$. The *postset* of a node $y \in P \cup T$ is denoted by $y\bullet$ and is the set of all output nodes of y, i.e., $y\bullet = \{x \in P \cup T \mid (y, x) \in A\}$. If $\forall p \in \bullet t : M(p) > 0$, transition $t \in T$ is said to be *enabled* in marking M. If t is enabled, the *firing* of t, denoted by $M \xrightarrow{t} M'$, leads to a new marking M', with $M'(p) = M(p) - 1$ if $p \in \bullet t \setminus t\bullet$, $M'(p) = M(p) + 1$ if $p \in t\bullet \setminus \bullet t$, and $M'(p) = M(p)$ otherwise. An *execution* σ of a net N is either the empty sequence, if no transitions are enabled in the initial marking, or a sequence of transitions $\langle t_1, t_2, \ldots, t_n \rangle$, $t_i \in T$, $i \in [1..n]$, such that $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} \ldots \xrightarrow{t_n} M_n$ and there are no enabled transitions in M_n.

Figure 4 shows the marked net discovered from log $L_{\mathtt{A}}$ shown in Fig. 3 using the Split miner discovery technique [36]. Note that in the discovered net, we label transitions with events to refer to the actions they represent, i.e., it holds that $\mathcal{E} \subset \mathcal{L}$. In the figure, the initial marking is denoted by the black dot in place `I`, specifying that in the initial marking, place `I` is associated with the number one (one black dot in the place), whereas every other place of the net is associated with the number zero (no black dots). The executions of the net describe (and generalize) the walks from the initial cell `I` toward goal `A` in the grid shown in Fig. 2a. In particular, the net generalizes the repetitive fragment in trace τ_5, via transitions $t_5, t_6, t_7, t_8, t_{10}$, and t_{11}. The transitions in the net encode steps in the grid. For example, transitions t_1 and t_5, despite both capturing a step to the north, describe two different steps in the grid, namely, $\binom{5,0}{5,1}$ and $\binom{4,1}{4,2}$, respectively; the cell references are not shown in the figure. Hence, execution $\langle t_3, t_4, t_5, t_6, t_7, t_8, t_{10}, t_{11}, t_5, t_6, t_7, t_8, t_9, t_{14}, t_{19}, t_{23} \rangle$ of the net describes trace τ_5 in the log from Fig. 3. Note that transitions t_4 and t_9 are *silent*, i.e., they are assigned silent labels that do not convey the domain semantics, shown as black rectangles in the figure.

3.2.2. Conformance Checking

Conformance checking measures and explains commonalities and discrepancies between traces in an event log and traces described in a process model. For example, conformance checking allows verifying the goodness of a model with respect to the log it is constructed from or the degree to which the observed behavior in the log corresponds with the allowed behavior specified in a normative model [37, 38, 39, 40].

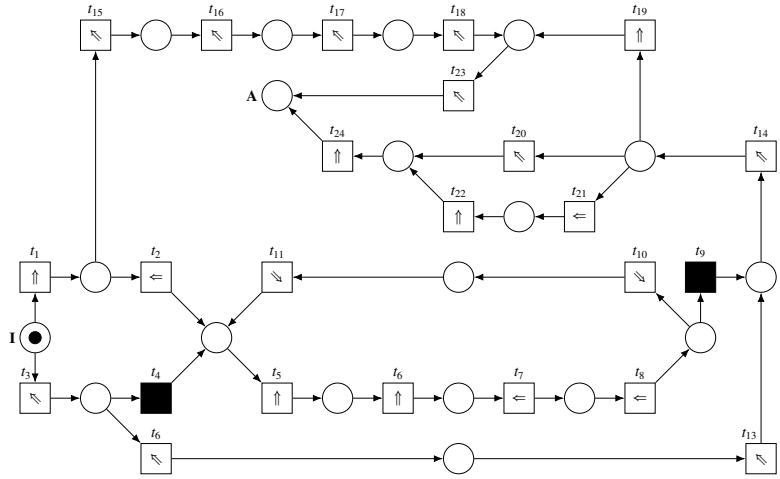


Figure 4: Net $N_{\mathtt{A}}$ discovered from the event log in Fig. 3 capturing the walks to goal A shown in Fig. 2a.

One of the central concepts in conformance checking is the concept of an ***alignment***. An alignment describes a relation between a trace and an execution of a process model as a sequence of *moves*, relating events in the log to transitions in the model [38]. In the context of nets, a move is a pair in which the first component refers to an element in a trace τ in a log L, and the second component refers to an element in an execution σ of a net N. Some elements in the trace may be not mimicked by elements in the execution (no move in the model) and vice versa (no move in the log). We denote "no move" by '$\gg$', a special symbol that is neither an event nor a transition.

Definition 4 (Move). A *move* over an execution σ of a net (P, T, A, λ, M_0) is a pair $(x, y) \in (((\mathcal{E} \cup \{\gg\}) \times (T \cup \{\gg\})) \setminus \{(\gg, \gg)\})$, where:

- (x, y) is a *synchronous move* if $x \in \mathcal{E}$, $y \in T$ and $x = \lambda(y)$;
- (x, y) is a *move on log* if $x \in \mathcal{E}$ and $y = \gg$; and
- (x, y) is a *move on model* if $x = \gg$ and $y \in T$.

A move (x, y) is a *legal* move if it is either a move on log, a move on model, or a synchronous move; otherwise move (x, y) is an *illegal* move. We also refer to moves on log and model as *asynchronous* moves. By $\mathbb{M}_{LM}$, we denote the set of all legal moves.

Definition 5 (Alignment). An *alignment* of a trace τ and an execution σ is a finite sequence $\gamma \in \mathbb{M}_{LM}^*$ of legal moves such that the first elements of the moves arranged in the order of the moves they come from without the '$\gg$' symbols yield τ and the second elements of the moves arranged in the order of the moves they come from without the '$\gg$' symbols yield σ.

Let $\delta : \mathbb{M}_{LM} \to \mathbb{N}_0$ be a function that assigns non-negative costs to moves. The cost of an alignment γ is denoted by $\delta(\gamma)$ and is equal to the sum of the costs of all its moves. As synchronous moves specify agreement between the trace and the execution, we use cost functions that assign zero costs to synchronous moves. In addition, as moves on model for *silent* transitions, e.g., transitions t_4 and t_9 in Fig. 4, do not demonstrate a disagreement with the trace, they are also assigned zero costs. Indeed, a silent transition does not represent a step of an agent and is present in a net for technical reasons only, i.e., to support the encoding of the desired executions.

Asynchronous moves capture the disagreement between the trace and the execution. Thus, we use cost functions that assign positive costs to asynchronous moves. Finally, an ***optimal alignment*** of a trace and a marked net is an alignment of the trace and some execution of the marked net that yields the lowest, among all possible alignments between the trace and executions of the marked net, cost. Intuitively, an optimal alignment characterizes minimal discrepancies between the trace and the net. A trace and a net are said to *agree perfectly* if there exists an optimal alignment of zero cost between the trace and some execution of the net.

Let us again consider the three walks of the agent from Fig. 1a. Alignments can be used to identify the discrepancies between the observed behavior of the agent in each of these walks and the models that represent the historical behavior toward the goals. Recall that net $N_{\mathtt{A}}$ in Fig. 4 models the observed behavior of the agent toward goal A from Fig. 2a. In addition, Fig. 5 depicts net $N_{\mathtt{F}}$ modeling the observed behavior of the agent toward goal F from Fig. 2f.

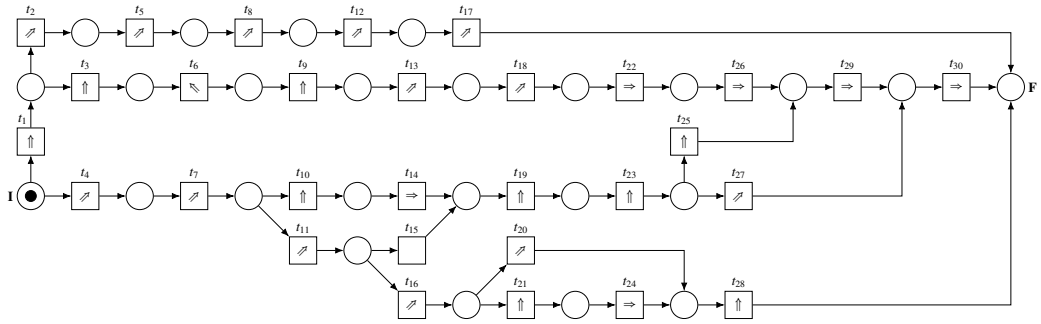


Figure 5: Net $N_{\mathtt{F}}$ discovered from the walks to goal F shown in Fig. 2f.

It is convenient to represent alignments as tables. Table 1 shows an optimal alignment γ_1 between the rational walk to goal A in Fig. 1a and net $N_{\mathtt{A}}$.[3] In the table, moves are encoded as columns, such that two successive columns refer to two successive moves in the alignment. Each column has five rows. The top two rows of each column correspond to the *trace contribution* to the move; they encode either an event from the trace (i.e., a step of the agent) or the no move symbol. The bottom three rows of each column correspond to the *model contribution* to the move; they encode either execution of an action in the model (i.e., an occurrence of a transition that describes a step of the agent) or, again, the no move symbol.

The first move in γ_1 is the synchronous move of transition t_3 in the net from Fig. 4 and event that encodes the step from cell $(5,0)$ to cell $(4,1)$ in the grid from Fig. 1a. Note that all moves in γ_1 except the second and the seventh move are synchronous. These are two model moves of silent transitions. Hence, the trace and model agree perfectly and $\delta(\gamma_1) = 0$. Note that the first elements in the moves in Table 1 without the "no move" symbols define the rational walk toward goal A shown in Fig. 1a, refer to Definition 5. Similarly, the second elements define execution $\langle t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{14}, t_{19}, t_{23} \rangle$ of the net in Fig. 4.

Table 2 shows optimal alignment γ_2 of the walk toward goal F in Fig. 1a and net $N_{\mathtt{F}}$ from Fig. 5. The first five moves in γ_2 are synchronous. The next move is, however, asynchronous. It is

[3]The logs that describe the walks toward the six goals shown in Fig. 2 and the nets discovered from these logs, captured using the XES standard (https://xes-standard.org/) and PNML standard (http://www.pnml.org/), respectively, can be accessed here: https://doi.org/10.26188/21749570.

$\gamma_1 =$

τ'	5,0 4,1		4,1 4,2	4,2 4,3	4,3 3,3	3,3 2,3		2,3 1,4	1,4 1,5	1,5 *A*
	↖	≫	↑	↑	←	←	≫	↖	↑	↖
$N_{\mathtt{A}}$	↖		↑	↑	←	←		↖	↑	↖
	5,0 4,1		4,1 4,2	4,2 4,3	4,3 3,3	3,3 2,3		2,3 1,4	1,4 1,5	1,5 *A*
	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{14}	t_{19}	t_{23}

Table 1: Optimal alignment between the rational walk τ' toward goal A in Fig. 1a and $N_{\mathtt{A}}$ in Fig. 4.

a move on model for transition t_{17} of the net. The last two steps in γ_2 are moves on log. As a result of these three asynchronous moves, two straight and one diagonal, it holds that $\delta(\gamma_2) = 2 + \sqrt{2}$. Note that we use the costs of straight and diagonal steps discussed in Section 2 as costs of the corresponding asynchronous alignment moves.

$\gamma_2 =$

τ''	5,0 5,1	5,1 6,2	6,2 7,3	7,3 8,4	8,4 9,5		9,5 10,5	10,5 *F*
	↑	↗	↗	↗	↗	≫	⇒	↑
	↑	↗	↗	↗	↗	↗	≫	≫
$N_{\mathtt{F}}$	5,0 5,1	5,1 6,2	6,2 7,3	7,3 8,4	8,4 9,5	9,5 *F*		
	t_1	t_2	t_5	t_8	t_{12}	t_{17}		

Table 2: Optimal alignment between the rational walk τ'' toward goal F in Fig. 1a and $N_{\mathtt{F}}$ in Fig. 5.

Table 3 shows optimal alignment γ_3 of the irrational walk toward goal A in Fig. 1a and net $N_{\mathtt{A}}$ from Fig. 4. The first five moves in γ_3 are moves on model, while the subsequent seven moves are moves on log. The final five moves demonstrate agreement between the walk and model. Considering cost function δ used to obtain costs of alignments γ_1 and γ_2 above, it holds that $\delta(\gamma_3) = 6 + 5\sqrt{2}$; again, the silent moves on model for transitions t_4 and t_9 are not penalized.

$\gamma_3 =$

τ'''							5,0 5,1	5,1 6,2	6,2 7,3	7,3 6,4	6,4 5,4	5,4 4,4	4,4 3,3	3,3 2,3		2,3 1,4	1,4 1,5	1,5 *A*
	≫	≫	≫	≫	≫	↑	↗	↗	↖	←	←	↙	←	≫	↖	↑	↖	
$N_{\mathtt{A}}$	↖		↑	↑	←	≫	≫	≫	≫	≫	≫	≫	≫	←		↖	↑	↖
	5,0 4,1		4,1 4,2	4,2 4,3	4,3 3,3									3,3 2,3		2,3 1,4	1,4 1,5	1,5 *A*
	t_3	t_4	t_5	t_6	t_7									t_8	t_9	t_{14}	t_{19}	t_{23}

Table 3: Optimal alignment between the irrational walk τ''' toward goal A in Fig. 1a and $N_{\mathtt{A}}$ in Fig. 4.

Finally, Table 4 shows optimal alignment γ_4 of the irrational walk toward goal A in Fig. 1a and net $N_{\mathtt{F}}$ from Fig. 5. Using the cost function δ, it holds that $\delta(\gamma_4) = 4 + 7\sqrt{2}$. Indeed, the first three moves are synchronous. However, among the subsequent eleven asynchronous moves, four represent straight steps and seven encode diagonal steps.

$\gamma_4 =$

τ'''	5,0 5,1	5,1 6,2	6,2 7,3				7,3 6,4	6,4 5,4	5,4 4,4	4,4 3,3	3,3 2,3	2,3 1,4	1,4 1,5	1,5 *A*
	↑	↗	↗	≫	≫	≫	↖	←	←	↙	←	↖	↑	↖
	↑	↗	↗	↗	↗	↗	≫	≫	≫	≫	≫	≫	≫	≫
$N_{\mathtt{F}}$	5,0 5,1	5,1 6,2	6,2 7,3	7,3 8,4	8,4 9,5	9,5 *F*								
	t_1	t_2	t_5	t_8	t_{12}	t_{17}								

Table 4: Optimal alignment between the irrational walk τ''' toward goal A in Fig. 1a and $N_{\mathtt{F}}$ in Fig. 5.

4. Data-Driven Goal Recognition

A solution to a data-driven GR problem uses data from historical solutions to the problem. In this paper, the data is a collection of previously observed sequences of actions that resulted in achieving the candidate goals. Concretely, we define the data-driven GR problem as follows.

Definition 6 (Data-driven probabilistic goal recognition). Given a tuple $< \mathcal{G}, A, D, \tau >$, where $\mathcal{G}$ is a set of candidate *goals*, A is a set of *actions*, $D \subseteq A^* \times \mathcal{G}$ is a set of pairs, each relating a historical sequence of actions to the goal achieved by executing this sequence of actions, and $\tau \in A^*$ is an *observation* given as a sequence of actions performed by an agent, the *data-driven probabilistic goal recognition* problem consists in obtaining a posterior probability distribution over the candidate goals that describes the true goal of the agent.

Section 4.1 presents our GR framework inspired by the principles from observational learning [41]. The framework can be seen as a collection of abstract components that, when instantiated, result in a concrete GR system. Then, in Section 4.2, we discuss an instantiation of the framework using process mining techniques to obtain a GR system for solving data-driven GR problems.

4.1. Goal Recognition Framework

Existing GR research focuses on developing approaches that merely solve one-shot recognition problems [16]. We present a framework for implementing intelligent agent systems that can continuously recognize goals according to newly observed actions and re-learn from the mistakes to improve the retained knowledge. The framework is inspired by the principles of *observational learning*, which considers the attitudes, values, and styles of thinking and behaving acquired by observing other examples [41]. Note that, compared with other well-developed cognitive architectures, the proposed framework shown in Fig. 6 is arguably one of the many components of a cognitive architecture, namely, a goal-intention recognition module (not a completed cognitive architecture). The framework is specifically proposed as an outline to implement process mining-based GR systems and consists of four corresponding stages: attention, retention, motivation, and recognition. The framework is given as a collection of UML Activity Diagrams that must be enacted simultaneously to support continuous learning and goal recognition. Next, we summarize each of these four stages.

1. The *attention* stage is responsible for determining whether an observed stimulus, e.g., a performed action, is relevant for learning a certain skill and capturing the relevant stimuli. This stage also regulates the selection of the relevant observed stimuli for a certain learning purpose and ignores irrelevant or noisy stimuli. In the top-left part of Fig. 6, the framework proposes to capture relevant actions s and w executed by agents A and B, respectively. Meanwhile, the framework also recognizes that agent A has achieved goal α. Hence, a GR system that instantiates the framework must "know" the possible goals agents may achieve in the environment and the conditions when these goals are fulfilled.

2. The *retention* stage (the top-right part of Fig. 6) is responsible for incorporating the observed stimuli into *skill models*, where a skill model describes how agents achieved one particular goal in the past; for example, the α-skill model records the historical traces to goal α. The framework receives information about the observed actions s and w and appends them to the corresponding currently constructed traces. Once the "*Goal completion recognized*" signal triggered by action α is captured, the corresponding trace of agent A is added to the *skill*

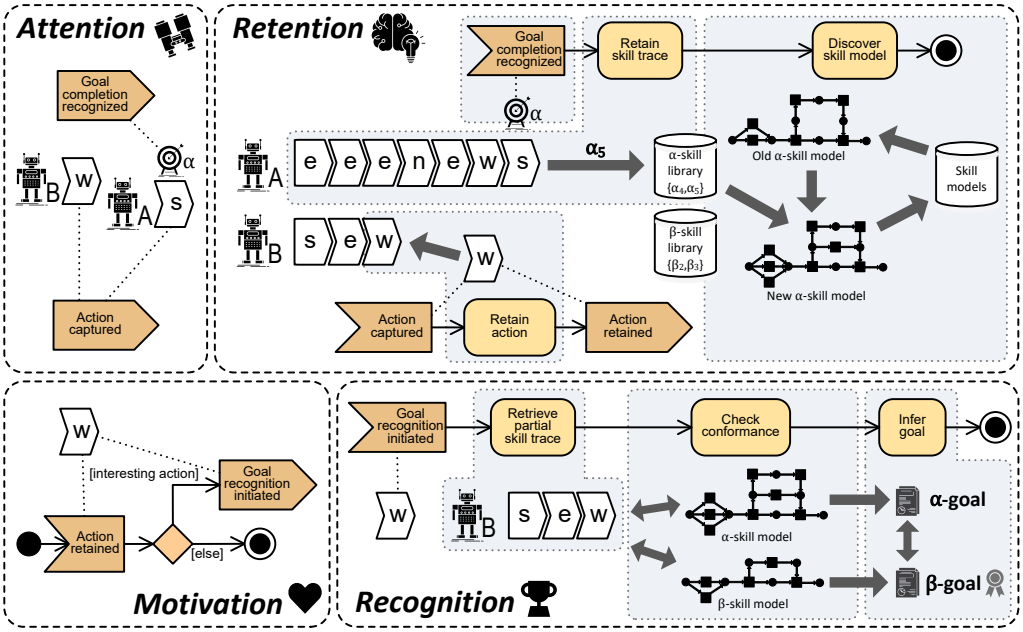


Figure 6: A schematic visualization of our goal recognition framework.

library, a collection of recently observed traces that lead to the same goal. For instance, in the figure, the "*Retain skill trace*" activity adds trace $\alpha_5 = \langle e, e, e, n, e, w, s \rangle$ to the α-skill library (α_5 is one of the traces to goal α). We use *process discovery* techniques [17, 42] from process mining to update old skill models based on the retained traces. Thus, a skill model aggregates and generalizes the observed behaviors for achieving the corresponding goal.

3. Based on the observed stimuli captured in the *attention* stage and stored in the *retention* stage, the *motivation* stage of the framework is responsible for triggering the subsequent goal recognition episodes. In the bottom-left part of Fig. 6, as a response to the "*Action retained*" signal triggered by action w, goal recognition is initiated by triggering the "Goal recognition initiated" signal. The implementation of the decision logic for triggering the recognition is outsourced to concrete instantiations of the framework.

4. The *recognition* stage is responsible for inferring the goals of the currently observed agents based on the retained skill models. This stage constructs the observed trace fragment, as shown in the bottom-right part of Fig. 6. For example, fragment $\langle s, e, w \rangle$ is performed by agent B and launches the "*Check conformance*" activity. The latter analyzes the commonalities and discrepancies between the trace fragment and all the available skill models to compute the distribution over the possible goals the agent may be striving to achieve. We use *conformance checking* techniques [38, 17, 43] from process mining to compute the commonalities and discrepancies between trace fragments and skill models. Finally, based on the performed analysis, the framework decides the goal of the agent. For instance, Fig. 6 suggests that β is the goal that agent B currently aims to achieve since $\langle s, e, w \rangle$ matches the β-skill model (the model for achieving goal β) better than the α-skill model (for goal α).

Apart from the four stages mentioned above, a feedback mechanism for learning based on the

recognition mistakes can be introduced. The need for such a feedback mechanism is motivated by the argument that observational learning could emerge from reinforcement learning [44].

4.2. Goal Recognition Framework Instantiation Using Process Mining Techniques

This section presents our approach to instantiating the recognition stage of the GR framework from Section 4.1. This stage is responsible for making GR decisions, while the other stages prepare for and trigger goal recognition. To implement a holistic GR experience, one can assume a basic implementation of the other stages. In particular, the attention and motivation stages can capture every action and trigger goal recognition for every captured action, while the retention phase can discover or update each skill model each time a fresh trace for the corresponding skill is recorded, possibly using only a specific number of the most recently observed traces. The detailed design of these three stages is a subject of future work. In this paper, we use the Directly Follows Miner discovery technique [45] to learn skill models. The Directly Follows Miner technique is more suitable for mining skill models of autonomous agents compared to other state-of-the-art miners, such as Split Miner [36] or Inductive Miner [46], which are primarily designed for mining business models. For example, the Directly Follows Miner tends to be more efficient in processing observations with many actions (long plans) without over-generalization.

Our instantiation of the recognition stage of the framework takes a trace fragment as input and constructs a GR probability distribution relative to a set of learned skill models. Section 3.1 explained how probabilistic GR could be realized by relying on the cost difference for each candidate goal. In line with that intuition, given a set of goals $\mathcal{G}$ and a sequence of observations τ, we adapt Eq. (1) by re-stating cost difference in terms of the level of misalignment between τ and each learned skill model α_G, with $G \in \mathcal{G}$. The less misalignment the observed behavior τ displays against the skill model α_G learned for goal G, the more likely it is that G is the goal of the agent. The level of misalignment is quantified by the alignment weight (defined below) of τ against model α_G, denoted by $\omega(\tau, \alpha_G)$. We follow Masters and Sardiña [29] in using a true Boltzmann distribution instead of a sigmoidal, and rewrite Eq. (1) as follows:

$$\Pr(G \mid \tau) = \frac{e^{-\beta \times \omega(\tau, \alpha_G)}}{\sum\limits_{G' \in \mathcal{G}} e^{-\beta \times \omega(\tau, \alpha_{G'})}}. \tag{3}$$

Here, $\omega(\tau, \alpha_G) \geq 0$, while the "temperature" β controls the level of confidence for GR, which can also be interpreted as the trust over the learned models. We define parameter β as follows:

$$\beta = \frac{1}{1 + \min\limits_{G \in \mathcal{G}} \omega(\tau, \alpha_G)}. \tag{4}$$

Equation (4) follows and simplifies Eq. (2) to account for the fact that the best case scenario is an alignment weight of zero, which implies that Eq. (4) inherits the confidence-based properties described in [31]. As the minimum (among all goals) alignment weight ω increases, the observed agent is arguably more "irrational", β approaches zero and the GR probability distribution more closely resembles a uniform one. Finally, the *alignment weight* between an observation trace $\tau = \langle e_1, \ldots, e_n \rangle$ and a skill model α_G captured as a marked net is defined as follows:

$$\omega(\tau, \alpha_G) = \phi + \lambda^m \times \sum_{i=1}^{n} \left(i^\delta \times c(\tau, \alpha_G, i) \right), \text{where:} \tag{5}$$

- $c(\tau, \alpha_G, i)$ is the cost of move for trace event e_i in an optimal alignment[4] between trace τ and model α_G;
- i^δ, with $\delta \geq 0$, is a discount factor that emphasizes that the more recent disagreements between the trace and the model impact the alignment weight more;
- $\phi \geq 0$ is the "smoothening" constant that flattens the likelihoods of the goals in the case of (close-to-)perfect alignments for all (most of) the skill models; and
- λ^m penalizes the suffix of the trace that deviates from the skill model, where $\lambda \geq 1$ is a constant and m is the number of consecutive asynchronous moves on trace at the end of the optimal alignment between trace τ and model α_G.

We use the example optimal alignments from Section 3.2.2 to demonstrate the computation of alignment weights. Differently from the cost functions used in conformance checking, which penalize moves on both model and log, for GR purposes, we assign the cost of move e_i, i.e., $c(\tau, \alpha_G, i)$, to be equal to one if it is an asynchronous move on log. All other moves are assigned a cost of zero. This costing scheme avoids penalizing partially observed traces since an optimal alignment of a partial trace tends to contain asynchronous moves on model. These moves on model, for example, describe how the trace can unfold in the future, not the discrepancies between the model and trace. A cost of one for asynchronous moves on trace is used as, in general, we assume no knowledge about the GR environment and the problem domain. Similarly, in all the evaluations of the approach we performed, when constructing alignments, we penalized all asynchronous moves, both moves on model and moves on trace, with a cost of one, and all the other moves were given no cost. This is different from our example alignments γ_1–γ_4 that were constructed using the cost of $\sqrt{2}$ for diagonal asynchronous moves. Note that picking different optimal alignments can affect the computation result of alignment weight, affecting the goal inference. Furthermore, in the example calculations, we use these default parameter settings: $\phi = 50$, $\lambda = 1.1$, and $\delta = 1$.

In alignment γ_1 from Table 1, all events in the trace are matched by model $\alpha_{\mathtt{A}}$, that is, net $N_{\mathtt{A}}$ from Fig. 4. Hence, each move has zero cost, i.e., $c(\tau', \alpha_{\mathtt{A}}, i) = 0$, for any position i in the trace. The number of consecutive asynchronous moves on trace in the suffix of the alignment is zero ($\lambda^m = 1.1^0$). Therefore, the alignment weight of γ_1 is 50; see the calculation below.

$$\omega(\tau', \alpha_{\mathtt{A}}) = \phi + \lambda^m \times \sum_{i=1}^{n} \left(i^\delta \times c(\tau', \alpha_{\mathtt{A}}, i) \right) = 50 + 1.1^0 \times 0 = 50$$

$\gamma_1 =$											
	τ'	⇖	≫	⇑	⇑	⇐	⇐	≫	⇖	⇑	⇖
	$\alpha_{\mathtt{A}}$	⇖		⇑	⇑	⇐	⇐		⇖	⇑	⇖
i^δ		1^1		2^1	3^1	4^1	5^1		6^1	7^1	8^1
$c(\tau', \alpha_{\mathtt{A}}, i)$		0		0	0	0	0		0	0	0

In alignment γ_2 from Table 2, the first five events in the trace are matched by model $\alpha_{\mathtt{F}}$, that is, net $N_{\mathtt{F}}$ from Fig. 5. However, the sixth and the seventh events result in asynchronous

[4] As there can exist multiple optimal alignments between a trace and a model, in this work, we rely on a procedure proposed by the authors of the original alignment technique that chooses one such optimal alignment deterministically [38, 47].

moves on log. Hence, each of the two corresponding moves in the alignment incurs the cost of one. As the alignment closes with two consecutive asynchronous moves on log, it holds that $\lambda^m = 1.1^2$. Consequently, the alignment weight of γ_2 is 65.73, see the detailed calculation below; the red columns in this and subsequent examples denote the asynchronous moves on log that are penalized when calculating alignment weight.

$$\omega(\tau'', \alpha_{\mathtt{F}}) = \phi + \lambda^m \times \sum_{i=1}^{n} \left(i^{\delta} \times c(\tau'', \alpha_{\mathtt{F}}, i)\right) = 50 + 1.1^2 \times 13 = 65.73$$

$\gamma_2 =$	τ''	⇑	⇗	⇗	⇗	⇗	≫	⇒	⇑
	$\alpha_{\mathtt{F}}$	⇑	⇗	⇗	⇗	⇗	⇗	≫	≫
i^{δ}		1^1	2^1	3^1	4^1	5^1		6^1	7^1
$c(\tau'', \alpha_{\mathtt{F}}, i)$		0	0	0	0	0		1	1

In γ_3 from Table 3, the first seven events in the trace are not matched by model $\alpha_{\mathtt{A}}$. Thus, the costs of the corresponding seven asynchronous moves in the alignment are equal to one. The last four moves are synchronous. As there are no asynchronous moves at the end of the alignment, it holds that $\lambda^m = 1.1^0$. Therefore, the alignment weight of γ_3 is 78, as shown below.

$$\omega(\tau''', \alpha_{\mathtt{A}}) = \phi + \lambda^m \times \sum_{i=1}^{n} \left(i^{\delta} \times c(\tau''', \alpha_{\mathtt{A}}, i)\right) = 50 + 1.1^0 \times 28 = 78$$

$\gamma_3 =$	τ'''	≫	≫	≫	≫	≫	⇑	⇗	⇗	⇖	⇐	⇐	⇙	⇐	≫	⇖	⇑	⇖
	$\alpha_{\mathtt{A}}$	⇖		⇑	⇑	⇐	≫	≫	≫	≫	≫	≫	≫	⇐		⇖	⇑	⇖
i^{δ}							1^1	2^1	3^1	4^1	5^1	6^1	7^1	8^1		9^1	10^1	11^1
$c(\tau''', \alpha_{\mathtt{A}}, i)$							1	1	1	1	1	1	1	0		0	0	0

Finally, in alignment γ_4 from Table 4, the first three events in the trace are matched by the model, while the remaining eight events are not and, thus, it holds that $\lambda^m = 1.1^8$. The weight of γ_4, consecutively, amounts to 178.62, see below.

$$\omega(\tau''', \alpha_{\mathtt{F}}) = \phi + \lambda^m \times \sum_{i=1}^{n} \left(i^{\delta} \times c(\tau''', \alpha_{\mathtt{F}}, i)\right) = 50 + 1.1^8 \times 60 \approx 178.62$$

$\gamma_4 =$	τ'''	⇑	⇗	⇗	≫	≫	≫	⇖	⇐	⇐	⇙	⇐	⇖	⇑	⇖
	$\alpha_{\mathtt{F}}$	⇑	⇗	⇗	⇗	⇗	⇗	≫	≫	≫	≫	≫	≫	≫	≫
i^{δ}		1^1	2^1	3^1				4^1	5^1	6^1	7^1	8^1	9^1	10^1	11^1
$c(\tau''', \alpha_{\mathtt{F}}, i)$		0	0	0	0	0	0	1	1	1	1	1	1	1	1

Our GR system uses Eq. (3) and weights of the alignments between the observed trace and all the skill models to construct a probability distribution over all the available goals. Subsequently, the GR system infers the possible goal(s) of the agent based on the constructed probability distribution and a given selection threshold θ. First, the candidate goal with the highest computed probability (Pr^+) is included in the resulting set of goals. Next, every goal with a computed probability greater than $\theta \times Pr^+$ is added to the resulting set. For instance, if it holds that $Pr^+ = 0.5$ and $\theta = 0.8$, then if any of the candidate goals has the probability of at least $0.8 \times 0.5 = 0.4$, it is included in the resulting set. In our implementation of the GR system, we set the default value of the selection threshold θ to be equal to 0.8.

Putting everything together, Eq. (3) provides a novel middle ground between traditional plan-library-based goal recognition and the more recent approaches from the planning literature grounded in cost differences between plans. Indeed, observations are matched to a sort of library of plans, implicitly represented and aggregated in a collection of skill models that generalize the original plans they are discovered from by re-interpreting plan cost as the level of misalignment between the observations and the skill models. Our GR system can be, in principle, used for plan recognition. Instead of inferring the goals, it could return skill models and remove the branches that are not optimally aligned with the observed action sequence, from which plan(s) can be inferred. Learning skill models from spurious (missing and noisy) observations is also possible but may lead to two issues: (i) An action should be included in the model but is missing. This phenomenon will lead to an *asynchronous move on model*, which may increase the alignment weight and decrease the recognition accuracy. (ii) An action should be excluded from the model but is included nonetheless. This phenomenon will cause an *asynchronous move on log* which we do not penalize, hence no impact on the accuracy.

Our GR system has four parameters that impact the inferences it produces, namely δ, λ, ϕ, and θ. In the next section, we demonstrate that each of these parameters impacts the performance of the system and suggest an approach for configuring them to maximize the GR performance.

5. Evaluation

In this section, we present our experimental setup (Section 5.1) and the results of the conducted experiments that allow us to answer the following five research questions:

RQ1: Do all the parameters of our GR system impact its performance (Section 5.2)?
RQ2: How to configure our GR system for better performance (Section 5.3)?
RQ3: Does the choice of a particular collection of traces for learning skill models impact goal recognition time and accuracy (Section 5.4)?
RQ4: How does the performance of our GR system compare with the performance of other state-of-the-art GR systems (Section 5.5)?
RQ5: Is our GR system applicable in real-world scenarios (Section 5.6)?

5.1. Experimental Setup

This section presents the experimental setup, including the datasets used in the experiments[5], the implementation of the PM-based GR system embedded in a simulation tool, and the quality measures used to assess the performance of all GR systems participating in the experiments.

5.1.1. Synthetic Datasets

We evaluated the performance of our GR system over 15 synthetic datasets, or *domains*, summarized by Pereira et al. [48]. Other techniques also use these domains for testing GR performance [11, 49], which allows for comparing performance across different experiments. Each of these synthetic domains consists of several problem instances. Each instance contains a list of candidate goals, the true goal, and an observed sequence of actions toward the true goal (an

[5]The datasets used in the experiments are available here: https://doi.org/10.26188/21749570.

observation). An observation can be full (when 100% of actions in the sequence were observed) or partial (for example, when only 10%, 30%, 50%, or 70% of all actions were observed).

Our approach requires historical observations for learning skill models. However, the mentioned domains only provide the rules of how agents can act in the environment. Therefore, we used planners to generate plans (traces) that resemble historical observations toward the candidate goals. Our GR system does not consider the frequencies of observed traces when it learns the skill models so that every generated trace in the dataset is unique.

To generate the traces, two planners were used: the top-k planner [50] and the diverse planner [51]. The top-k planner generates a set of k different *cost-optimal* traces toward a given goal. Such cost-optimal traces simulate possible rational behaviors of an agent toward the goal. On the other hand, the diverse planner generates a set of *divergent* traces such that each trace is significantly different from others according to a *stability* diversity metric [52, 53]. The stability between two plans is the number of the co-occurring actions in both plans over the total number of the actions, refer to Eq. (6). The diverse planner generates a set of plans such that the stability between every two plans is equal to or is less than a specified number. Note that $\mathcal{A}(\pi)$ and $\mathcal{A}(\pi')$ in Eq. (6) represent the sets of actions in plan π and plan π', respectively.

$$stability(\pi, \pi') = \frac{|\mathcal{A}(\pi) \cap \mathcal{A}(\pi')|}{|\mathcal{A}(\pi) \cup \mathcal{A}(\pi')|} \tag{6}$$

A planner does *not* guarantee that it can generate an arbitrary number of distinct traces toward a given goal. Therefore, if a planner failed to generate the requested number of traces for a given problem, we removed that problem from the dataset. We attempted to generate 100 traces toward each candidate goal in each problem instance in all 15 synthetic domains using the top-k planner and the diverse planner. If a planner did *not* generate 100 traces toward a candidate goal within one hour, we marked the GR problem instance containing that candidate goal as failed. The diverse planner failed to generate 100 traces for all the problem instances from the Campus and Kitchen domains; these were excluded from the analysis accordingly. For the same reason, we excluded several problem instances in the remaining domains, for both planners. For each planner, the summary of the excluded instances is provided in Appendix A. The excluded instances were not used in our experiments.

5.1.2. Real-World Datasets

We evaluated the performance of our GR system over ten real-world datasets, or *domains*. Most of these domains are made publicly available by the IEEE Task Force on Process Mining. Each of these ten domains is a real-world log of business processes comprising action traces to achieve a particular business goal. In seven of these logs, the labels of the traces for achieving goals are missing. Hence, we used the preprocessed datasets provided by Teinemaa et al. [54], in which the traces in the original logs are classified into categories using Linear Temporal Logic (LTL) classifiers. Each obtained cluster contains traces belonging to some business sub-goal. The domains and the classifiers are presented as follows:

- The BPIC 2011 event log includes records of patients' medical treatments from a Dutch Hospital (DOI: 10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54). Four GR problems were formulated using the LTL classifiers to capture different treatment outcomes.
- The five BPIC 2015 event logs record the application processes to acquire building permits in five Dutch municipalities (DOI: 10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1),

one municipality per log. The traces in each event log were classified according to whether the action "send confirmation receipt" was executed before the action "retrieve missing data".

- The BPIC 2017 event log records the loan application process of a Dutch financial institute (DOI: 10.4121/12705737.v2). This event log was partitioned into six clusters that formed three GR problems of identifying whether an application is accepted, rejected, or canceled. Each problem consists of two subsets of traces: one with the corresponding outcome ("accepted", "rejected", or "canceled") and the other one with all the remaining traces ("not accepted", "not rejected", or "not canceled", respectively).

- The Hospital Billing event log records the execution of billing procedures for medical services (DOI: 10.4121/uuid:76c46b83-c930-4798-a1c9-4be94dfeb741). Two GR problems were formulated based on this log: to recognize whether the billing package was eventually closed and to recognize whether the case was reopened.

- The Production event log records activities for producing items in a manufacturing scenario (DOI: 10.4121/uuid:68726926-5ac5-4fab-b873-ee76ea412399). The traces were classified into two sub-logs according to whether the number of rejected orders was zero or not.

- The Sepsis Cases event log from a Dutch hospital records laboratory tests of patients who have sepsis conditions (DOI: 10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460). Three GR problems were formulated using the LTL classifiers: whether a patient will return to the emergency room within 28 days of discharge, whether a patient is admitted for intensive care services, and whether a patient is discharged due to a reason other than *Release*.

- The Traffic Fines event log records events related to fines from an Italian local police force (DOI: 10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5). The traces were classified into two groups according to whether the fine was fully repaid or not.

The resulting GR problems are binary choices between two candidate goals. The traces for training skill models and the observed traces for testing GR performance are provided by Teinemaa et al. [54]. The statistics of these binary-choice datasets are shown in Table 5. We also tested our GR system on multi-class problems that have more than two candidate goals. To this end, we used three real-world datasets that were also used in [20], namely Activities of Daily Living, Building Permit Applications, and Environmental Permit Applications. The statistics of these multi-class datasets are shown in Table 6. These datasets were divided into 80% and 60% of traces for training the skill models and the remaining 20% and 40% of traces, respectively, for testing GR performance. These three domains are summarized below:

- The Activities of Daily Living event logs record activities executed by four individuals during weekdays and weekends separately (DOI: 10.4121/uuid:01eaba9f-d3ed-4e04-9945-b8b302764176). Therefore, eight event logs were used to formulate the GR problem of identifying who and on which day (weekday or weekend) executed a given action sequence.

- The Building Permit Applications is the same dataset as *BPIC 2015* mentioned above. It contains event logs that record the processes of build permit applications handled by five

Domain	Sub-set	# Train traces	# Obs traces	# Candidates	Avg Len	Min Len	Max Len	Std Dev
BPIC 2011	1	516	58	2	56.66	1	824	113.50
	2	1025	115	2	131.34	1	1814	202.60
	3	1008	113	2	62.93	1	1368	134.44
	4	1025	115	2	81.64	1	1432	142.54
BPIC 2015	1	626	70	2	41.34	2	101	17.22
	2	677	76	2	54.71	1	132	19.04
	3	1194	134	2	43.29	3	124	15.35
	4	518	59	2	42.00	1	82	14.52
	5	945	106	2	51.91	5	134	15.11
BPIC 2017	1	28270	3143	2	38.15	10	180	16.70
	2	28270	3143	2	38.15	10	180	16.70
	3	28271	3142	2	38.15	10	180	16.70
Hospital Billing	1	69772	7753	2	5.53	2	217	2.32
	2	69771	7754	2	5.27	2	217	1.97
Production		197	23	2	11.31	1	78	10.13
Sepsis Cases	1	702	80	2	16.78	5	185	12.10
	2	703	79	2	13.97	4	60	5.05
	3	702	80	2	15.94	4	185	12.18
Traffic Fines		116652	12963	2	3.55	2	20	1.37

Table 5: Statistics of the real-world datasets for binary choice problems. Train traces: the traces for training skill models; Obs traces: the observed traces for testing GR performance.

Dutch municipalities. Instead of classifying the traces in each event log into two subgroups, we formulated GR problems to identify which municipality handled which action sequence.

- The five Environmental Permit Applications event logs record the environmental permit application processes in five Dutch municipalities (DOI: 10.4121/uuid:26aba40d-8b2d-435b-b5af-6d4bfbd7a270). As for the Building Permit Applications domain, the GR problems were formulated to recognize which municipality processed which environmental application.

Domain	# Traces	# Candidates	Avg Len	Min Len	Max Len	Std Dev
Activities of Daily Living	148	8	75.26	20	134	23.49
Building Permit Applications	1000	5	45.61	1	154	19.67
Environmental Permit Applications	1000	5	43.89	2	108	17.39

Table 6: Statistics of the real-world datasets for multi-class problems.

5.1.3. Implementation

Our GR system was implemented and is available as part of an open-source simulation tool[6] capable of automatically solving batches of GR problem instances formulated in a given domain. When solving a single problem instance, our tool takes a set of parameters and an observed trace as input. As a result, it returns a list of inferred goals, their likelihoods, and the time spent solving the instance. All the problem instances were run on a single core of an Intel Xeon Processor (Skylake, IBRS) @ 2.0GHz. Note that solving a single problem instance requires less than 4GB of RAM.

[6] https://github.com/zihangs/GR_system

5.1.4. Quality Measures

We used precision, recall, and accuracy to evaluate the performance of GR systems. Four terms are used to compute these measures. True Positive (*TP*) is the number of correct goals inferred by a GR system. True Negative (*TN*) is the number of incorrect goals that were not inferred. False Positive (*FP*) is the number of incorrect goals inferred by a GR system. Finally, False Negative (*FN*) is the number of correct goals that were not inferred. Given the above terms, *precision* is the fraction of the correctly inferred goals among all the inferred goals.

$$f_{precision} = \frac{TP}{TP + FP} \tag{7}$$

Recall is defined as the fraction of the correctly inferred goals among all the true goals.

$$f_{recall} = \frac{TP}{TP + FN} \tag{8}$$

Finally, *accuracy* is the ratio of the correct positive and negative inferences to the total positive and negative inferences.

$$f_{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

Suppose our GR system observes an agent that executes a sequence of actions. The agent works toward a true hidden goal ($g1$) among ten candidate goals ($g1$ to $g10$). For example, according to the observed action sequence, our GR system infers $g1$ and $g2$ as two possible goals the agent is trying to achieve. Thus, $g1$ and $g2$ are two positive goals, and the other candidate goals are negative goals. In this scenario, for the two positive goals, $g1$ is the true hidden goal correctly inferred by the GR system. Hence, *TP* is equal to one. Goal $g2$ is not the true hidden goal (it is falsely inferred by the GR system). Thus, *FP* is also equal to one. For the eight negative goals ($g3$ to $g10$), none of them is the true hidden goal. As such, *TN* equals eight because our GR system made the correct decision not to infer these goals. Finally, *FN* is equal to zero because none of the true hidden goals are missed (our GR system correctly recognized the true hidden goal). Hence, in this example scenario, precision, recall, and accuracy are equal to 0.5, 1.0, and 0.9, respectively. Note that in our experiments $TP, FN \in \{0, 1\}$, as there is only one true goal per instance, while $TN, FP \in \{0, \ldots, |\mathcal{G}| - 1\}$, where $\mathcal{G}$ stands for the set of candidate goals.

5.2. Variance-Based Sensitivity Analysis

A sensitivity analysis explores how much each input (independent) variable contributes to the variance of a target (dependent) variable [55]. In our GR experiments, we used the *average accuracy* across different problems as the target variable and the parameters of our GR system (i.e., ϕ, λ, δ, and θ) as the input variables. Consequently, we conducted a sensitivity analysis to verify whether the parameters of our GR system have a significant impact on its performance in terms of accuracy. The sensitivity analysis relies on sampling input variable values and evaluating the corresponding target variable values to understand whether all the input variables influence the target variable.

We integrated our GR system and the performance measures discussed in Section 5.1.4 into a publicly available sensitivity analysis tool called "Exploratory Modelling and Analysis (EMA) Workbench" [56]. We subsequently conducted the sensitivity analysis using the Sobol variance decomposition method [57], namely Sobol sensitivity analysis, implemented by the EMA Workbench to obtain the first-order and total effects. The first-order effect captures the

direct influence of each parameter by measuring whether changes in a single parameter (while keeping all other parameters unchanged) affect the performance significantly [55]. The total effect captures both the direct and the indirect influence, where indirect influence measures change produced by every single parameter due to interactions with other parameters being changed at the same time [55]. Note that the index of the total effect is greater than or equal to the index of the first-order effect. The difference between these two indices expresses the indirect impact of the parameter on the GR performance. The total effect of a parameter measures the percentage of the output variance contributed by that parameter (directly and indirectly). Following Zhang et al. [58], a parameter can be considered to have a significant impact if the total effect index is greater than 0.05. The Sobol analysis also provides a confidence interval (CI) for each sensitivity index. The sensitivity index is expected to fall within the range defined by the CI. Therefore, we considered a parameter to have a significant impact only if it is expected to be above 0.05, where the lower bound of the CI falls above the 0.05 mark.

We conducted the Sobol sensitivity analysis on all synthetic domains discussed in Section 5.1.1. The results of the Sobol sensitivity analysis in the blocks-world domain for the GR system trained by the cost-optimal traces and the divergent traces are displayed in Fig. 7. The *S1* values express the first-order effects, and *ST* values represent the total effects. The black lines are the confidence intervals for the estimated indices. The lower bounds of the *ST* confidence intervals for these four parameters are greater than 0.05. These results indicate that, for the blocks-world domain, all four parameters, directly and indirectly, impact the GR performance (accuracy) regardless of which set of traces is used for training the GR system.

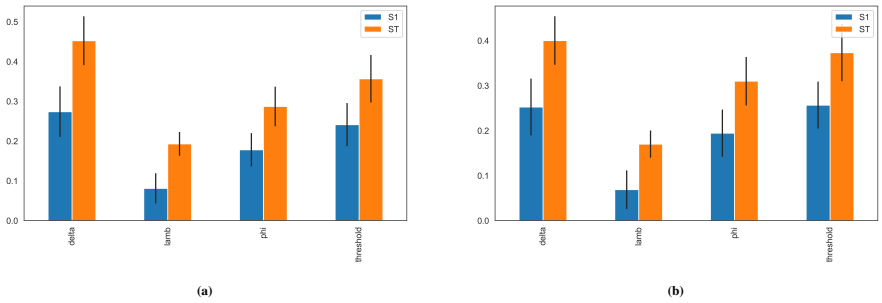


Figure 7: Sobol sensitivity analysis indices for the first-order effects and the total effects for the blocks-world domain. The GR system was trained by (a) the cost-optimal traces and (b) the divergent traces.

The Sobol sensitivity analysis results for all the other synthetic domains are listed in Appendix B. For the Sokoban domain with the divergent traces, the lower bound of the *ST* confidence interval for parameter ϕ is less than 0.05 (non-significant). Except for this special case, all other lower bounds are greater than 0.05, indicating that all four parameters significantly impact the performance.

5.3. Scenario Discovery

Our sensitivity analysis concluded that all four parameters impact the target performance measure. Consequently, we need to consider all four parameters for scenario discovery. Scenario discovery is a technique to identify the scenarios of interest from a collection thereof. We define a *scenario* as an experiment in which the parameters of our GR system are configured and lead to corresponding performance measurements (e.g., accuracy). In general, scenarios establish

the relationship between configuration parameters and performance measurements. We identify interesting scenarios as those in the top 10-percentile with respect to accuracy. We used the Latin Hypercube sampling method to fairly distribute the sampled parameter configurations in the multi-dimensional sampling space [59]. It is desirable to have N simulations (parameter configurations), where N is much larger than the number of parameters [55] (our GR system has four parameters). We followed the example from [60] and sampled 1,000 simulations for scenario discovery, given the small computational cost of running each simulation. Subsequently, we conducted the GR experiments based on these parameter configurations to obtain 1,000 scenarios. The parameter configurations that result in interesting scenarios constitute a *region*. Intuitively, configuring our GR system with parameters in that region should yield better performance than randomly selected parameters.

The Patient Rule Induction Method (PRIM) is an algorithm to iteratively seek a smaller (denser) region in which the mean of the target value is significantly higher than the mean value outside that region [61, 60]. As such, PRIM can be used to iteratively peel away some scenarios to find a high-density region, a small region that contains many interesting scenarios. Meanwhile, the region should have high *coverage*, which explains the percentage of interesting scenarios located in the region. In general, there is a trade-off between finding a region with high *density* of interesting scenarios versus a region with high coverage of interesting scenarios. Therefore, we aim to find a balance between the *density* and the *coverage*. Figure 8 shows the PRIM peeling trajectory for 1,000 generated scenarios in the *driverlog* domain (traces generated by the cost-optimal top-k planner). Each point on the trajectory represents a region of scenarios with the corresponding density and coverage values. As expected, the density decreases when the coverage increases.

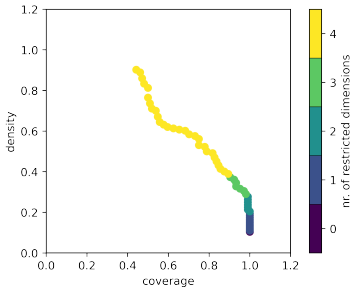


Figure 8: Peeling trajectory for the *driverlog* domain (trained by the cost-optimal traces).

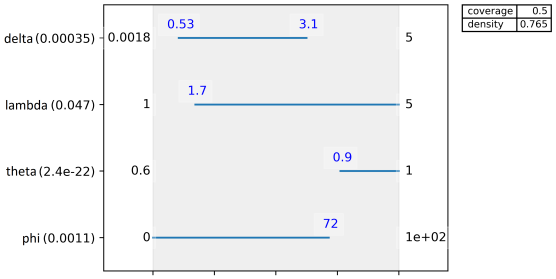


Figure 9: Visualization of the parameter ranges recommended by the PRIM algorithm.

Each region on a peeling trajectory is characterized by four attributes corresponding to the parameter ranges ϕ, λ, δ, and θ. Figure 9 shows these parameter ranges for a selected region from the peeling trajectory in Fig. 8 with a density of 0.765 and coverage of 0.5. For this region, parameter δ ranges from 0.53 to 3.1, λ from 1.7 to 5.0, ϕ from 0.0 to 72.0, and θ from 0.9 to 1.0. Note that 50% of the interesting scenarios stem from the parameter configurations that fall into these ranges, as indicated by the coverage.

To select a representative region on the peeling trajectory, we aim to choose a region that induces narrow parameter ranges that are significant, i.e., the *quasi p-value* or *qp-value* of the parameter ranges is less than 0.05. In Fig. 9, the qp-value for each parameter appears between parentheses. However, one cannot guarantee to find a point (region) on the peeling trajectory

with qp-values of all parameter ranges less than 0.05. Therefore, we choose the region of highest coverage with four restricted dimensions and all qp-values less than or equal to 0.05; otherwise, we choose the region of maximal density. Figure 9 shows a representative region with the restricted parameter ranges and the qp-values of each parameter range. The qp-values for δ is 0.00035, for λ is 0.047, for θ is 2.4e−22, and for ϕ is 0.0011.

We applied the PRIM algorithm for all the domains with skill models trained by the cost-optimal and divergent traces. The parameter ranges for representative regions for all the domains are shown in Tables 7 and 8. The $*$ indicates a qp-value of less than 0.05, and the $**$ indicates a qp-value of less than 0.001. A parameter range has lower (Min) and upper (Max) bounds. If at least one bound, lower or upper, is significantly restricted, it is annotated with $*/**$. Parameter ranges without any $*/**$ symbol indicate that no significant restriction has been applied, where interesting scenarios are evenly distributed in the initial sampling parameter range.

Domain	Delta (δ)		Lambda (λ)		Phi (ϕ)		Threshold (θ)	
	Min	Max	Min	Max	Min	Max	Min	Max
Blocks-world	0.31	4.54	1.00	5.00	0.00	13.50**	0.81**	1.00
Campus	2.19**	5.00	2.36**	5.00	0.00	95.50	0.64	1.00
Depots	0.20	3.72*	1.72	4.65	0.00	58.50**	0.93**	1.00
Driverlog	0.53	3.12**	1.69*	5.00	0.00	71.50*	0.90**	1.00
DWR	0.03	2.19**	1.00	5.00	0.00	82.50	0.94**	1.00
Easy-ipc-grid	0.44	5.00	2.19**	5.00	11.00	73.50*	0.86**	0.98
Ferry	0.35	2.49**	1.25	4.93	0.00	95.00	0.94**	1.00
Intrusion-detection	0.32	3.25	1.00	5.00	0.00	68.50	0.95**	1.00
Kitchen	0.13	3.52	1.62	4.56	0.00	30.50**	0.86**	1.00
Logistics	0.24	3.94	1.31	4.97	0.00	94.50	0.96**	1.00
Miconic	0.22	4.00**	1.00	5.00	0.00	100.00	0.96**	1.00
Rovers	0.00	5.00	1.17	4.99	0.00	85.50	0.97**	1.00
Satellite	0.93*	3.10**	1.03	5.00	0.00	89.50	0.94**	1.00
Sokoban	0.23	1.98**	1.21	3.41**	5.00	92.50	0.87**	1.00
Zeno-travel	0.15	4.18*	1.00	5.00	0.00	45.50**	0.94**	1.00

Table 7: The recommended ranges for parameters of the GR system (cost-optimal traces). $*$: qp-value ≤ 0.05; $**$: qp-value ≤ 0.001.

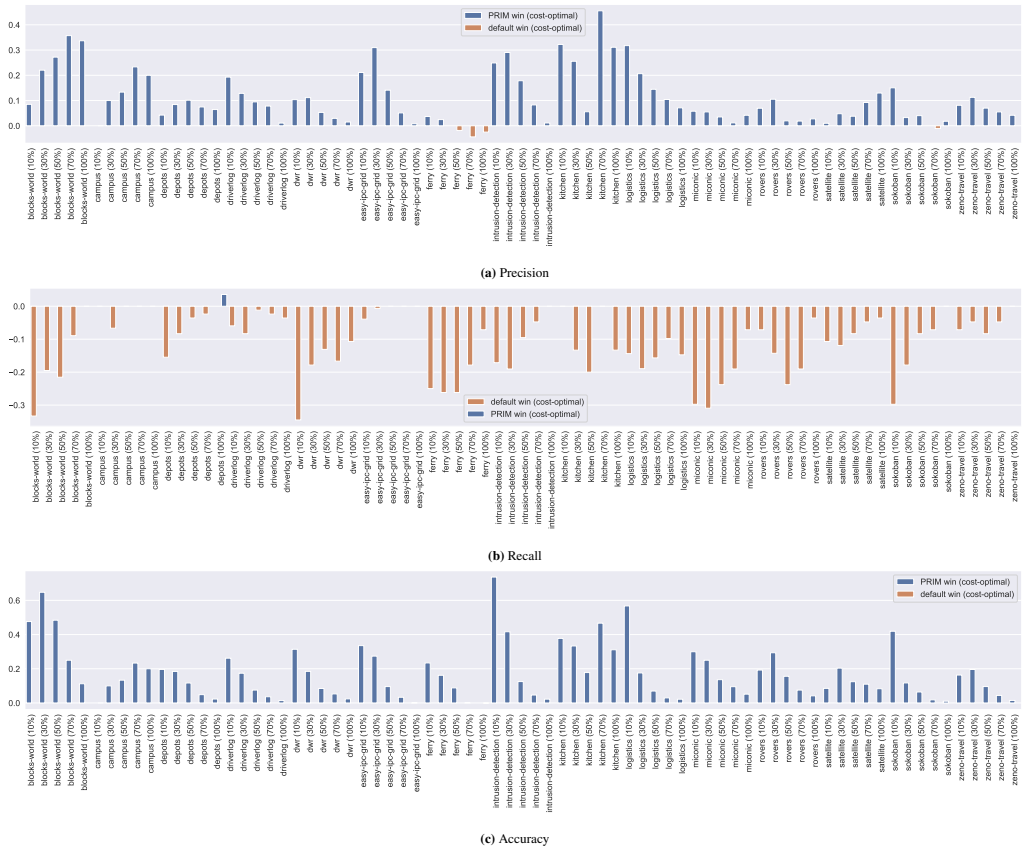
Domain	Delta (δ)		Lambda (λ)		Phi (ϕ)		Threshold (θ)	
	Min	Max	Min	Max	Min	Max	Min	Max
Blocks-world	0.00	5.00	1.64	4.56	0.00	18.50**	0.87**	1.00
Depots	0.27	1.97**	1.34	4.92	0.00	92.00	0.92**	1.00
Driverlog	0.61	2.74**	1.45	5.00	0.00	74.50	0.94**	1.00
DWR	0.33	3.11**	1.17	5.00	0.00	100.00	0.96**	1.00
Easy-ipc-grid	1.01*	4.29*	2.54**	5.00	0.00	79.50*	0.78**	1.00
Ferry	0.44	4.32*	1.00	5.00	0.00	95.50	0.96**	1.00
Intrusion-detection	0.30	3.90*	2.41*	5.00	0.00	79.50*	0.94**	1.00
Logistics	0.21	2.58**	1.23	5.00	0.00	100.00	0.95**	1.00
Miconic	0.27	2.21**	1.00	4.95	0.00	87.50	0.94**	1.00
Rovers	0.00	1.90**	1.08	5.00	0.00	89.50	0.91**	1.00
Satellite	0.26	2.52**	1.00	5.00	0.00	39.50**	0.89**	1.00
Sokoban	0.19	2.41**	1.31*	2.97**	0.00	86.50	0.87**	1.00
Zeno-travel	0.40	2.25**	1.00	5.00	0.00	92.50	0.93**	1.00

Table 8: The recommended ranges for parameters of the GR system (divergent traces). $*$: qp-value ≤ 0.05; $**$: qp-value ≤ 0.001.

Intuitively, PRIM narrows the parameter ranges to smaller ones that are more likely to contain the top performance scenarios. Thus, for each domain, we use the middle points of the

parameter ranges discovered by PRIM to configure our GR system to obtain good performance. For example, in Fig. 9, the middle point of the range for parameter δ is 1.82. Therefore, we configured the δ of our GR system to be 1.82. Similarly, other parameters are configured with the middle points of the corresponding ranges identified by PRIM, namely the PRIM parameters. We performed GR experiments with the PRIM parameters for all the synthetic domains and compared the performance with the GR experiments based on the default parameters (ϕ = 50, λ = 1.1, δ = 1.0, θ = 0.8). The results of precision, recall, accuracy, and execution time for each domain on each level of observation are listed in Appendix C.

We use the precision, recall, accuracy, and execution time from the GR system configured by the PRIM parameters and subtract the corresponding values from the GR system configured by the default parameters. Figures 10 and 11 show the performance differences between the GR system configured with the PRIM parameters and that with the default parameters. The blue bars, "PRIM win," represent that the GR system configured with the PRIM parameters performs better than that configured with the default parameters. The height of the bars represents how much one is better than the other. Contrarily, the orange bars, "default win," represent the default parameters' win over the PRIM parameters.



(a) Precision

(b) Recall

(c) Accuracy

Figure 10: The performance differences between the GR system configured with the PRIM parameters and that configured with the default parameters (both systems are trained by the cost-optimal traces).

(a) Precision

(b) Recall

(c) Accuracy

Figure 11: The performance differences between the GR system configured with the PRIM parameters and that configured with the default parameters (both systems are trained by the divergent traces).

The comparison results show that the PRIM parameters are likely to yield high precision and accuracy of our GR system, while the default parameters yield high recall. A GR system with higher precision and accuracy is more convincing to be a good system than that with higher recall, as one can always obtain high recall by inferring all the candidate goals. Therefore, we conclude that the PRIM parameters lead to better performance, and we recommend using the PRIM analysis presented here to identify configuration parameters of our GR system.

5.4. Impacts of Training Traces

In our experiments, the traces accepted as historical observations of agents are generated by planners, namely the top-*k* planner and the diverse planner, see Section 5.1.1, which may impact learning the skill models and, consequently, the GR performance. We compared the GR systems trained by different sets of traces (configured with the PRIM parameters). The performance of the GR systems trained by the cost-optimal and divergent traces is shown in Appendix C.

The results of comparing GR performance on different sets of training traces are shown in Fig. 12. The blue bars represent the GR system trained by divergent traces win over those trained by the cost-optimal traces. The orange bars represent the cost-optimal traces win. The height

of the bars represents how much one is better than the other. The divergent traces yield higher precision than the cost-optimal traces (except for the domains of DWR, Intrusion-detection, and Logistics), as well as a higher accuracy (except for the domains of DWR, Intrusion-detection, and Sokoban). The cost-optimal traces yield higher recall for all the domains. The recognition times of the GR system trained by the cost-optimal traces are shorter than those trained by the divergent traces. Intuitively, compared with the cost-optimal traces, the divergent traces tend to be longer traces that consist of more actions. As Table 9 shows, the skill models (Petri nets) learned from these traces tend to have more places, transitions, and arcs, which suggests that the skill models tend to cover a larger state space. Consequently, the GR system trained by the divergent traces yields higher precision and accuracy. However, computing the optimal alignment between a trace and a skill model that covers a smaller state space is relatively easier, such that the recognition time of the GR system trained by the cost-optimal traces is shorter.

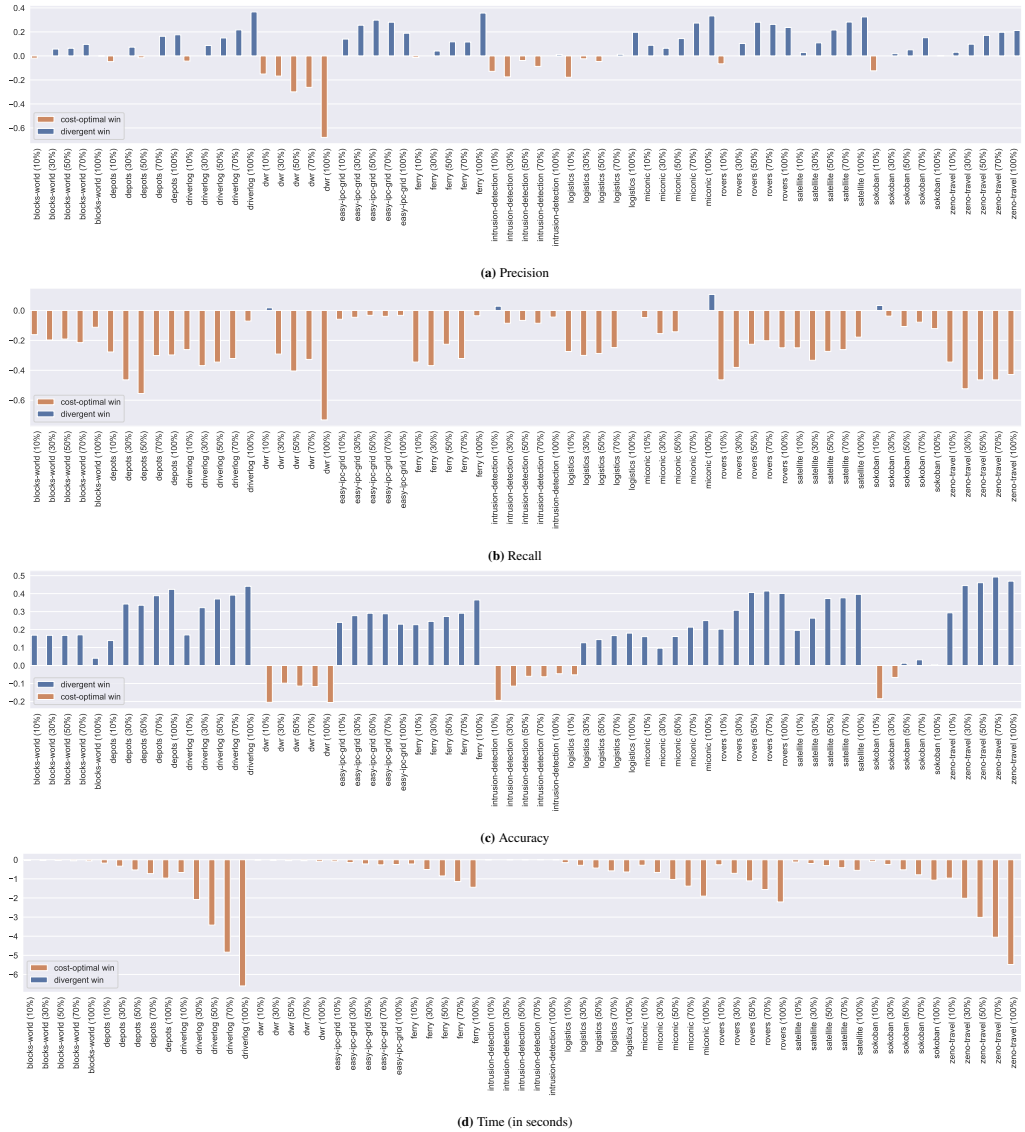
Domain	Cost-optimal Traces			Divergent Traces		
	Transitions	Places	Arcs	Transitions	Places	Arcs
Blocks-world	63.08	28.08	126.16	120.59	58.25	241.18
Depots	32.36	12.73	64.72	389.94	121.4	779.87
Driverlog	23.52	8.55	47.04	591.32	263.12	1182.64
DWR	78.43	33.90	156.86	134.93	53.71	269.86
Easy-ipc-grid	39.74	21.80	79.49	151.05	86.54	302.10
Ferry	28.22	13.79	56.45	381.15	127.02	762.30
Intrusion-detection	73.23	18.71	146.46	89.20	18.60	178.40
Logistics	35.63	14.73	71.25	275.95	102.38	551.91
Miconic	53.05	30.25	106.11	375.52	182.67	751.08
Rovers	24.35	8.55	48.70	387.68	135.92	775.35
Satellite	32.78	15.72	65.57	294.33	138.58	588.67
Sokoban	134.65	81.96	269.30	358.85	186.03	717.71
Zeno-travel	19.33	8.51	38.66	678.79	284.67	1357.58

Table 9: The average number of transitions, places, and flow arcs over the skill models learned from the cost-optimal traces and the divergent traces for each domain.

5.5. Comparison with Other Approaches

We compared our PM-based GR approach with state-of-the-art GR approaches. Section 5.5.1 presents the comparison between our approach and domain knowledge-based GR techniques: Ramirez and Geffner's approach (R&G) [11], the landmark-based approach [48], and the LP-based GR approach [62]. Section 5.5.2 shows the comparison between our approach and the long short-term memory networks (LSTM-)based GR approach [63]. For the R&G approach, we experimented with two embedded planners: the Greedy LAMA planner [64] and the state-of-the-art planner from the international planning competition, DUAL-BFWS [65]. For the landmark-based approach, we used *uniqueness* as the heuristic and set the threshold to 20%. For the LP-based approach, we used a combination of three heuristics, which are *landmarks*, *state equation*, and *post-hoc*. Our GR system and the LSTM-based approach are trained using the divergent traces for all except Campus and Kitchen. Note that the diverse planner failed to generate traces for these two domains (cf. Section 5.1.1). Therefore, we used the cost-optimal traces to train our GR system and the LSTM-based approach for these domains. We configured our GR system using the PRIM parameters (cf. Section 5.3).

5.5.1. Comparison with Domain Knowledge-Based GR Approaches



(a) Precision

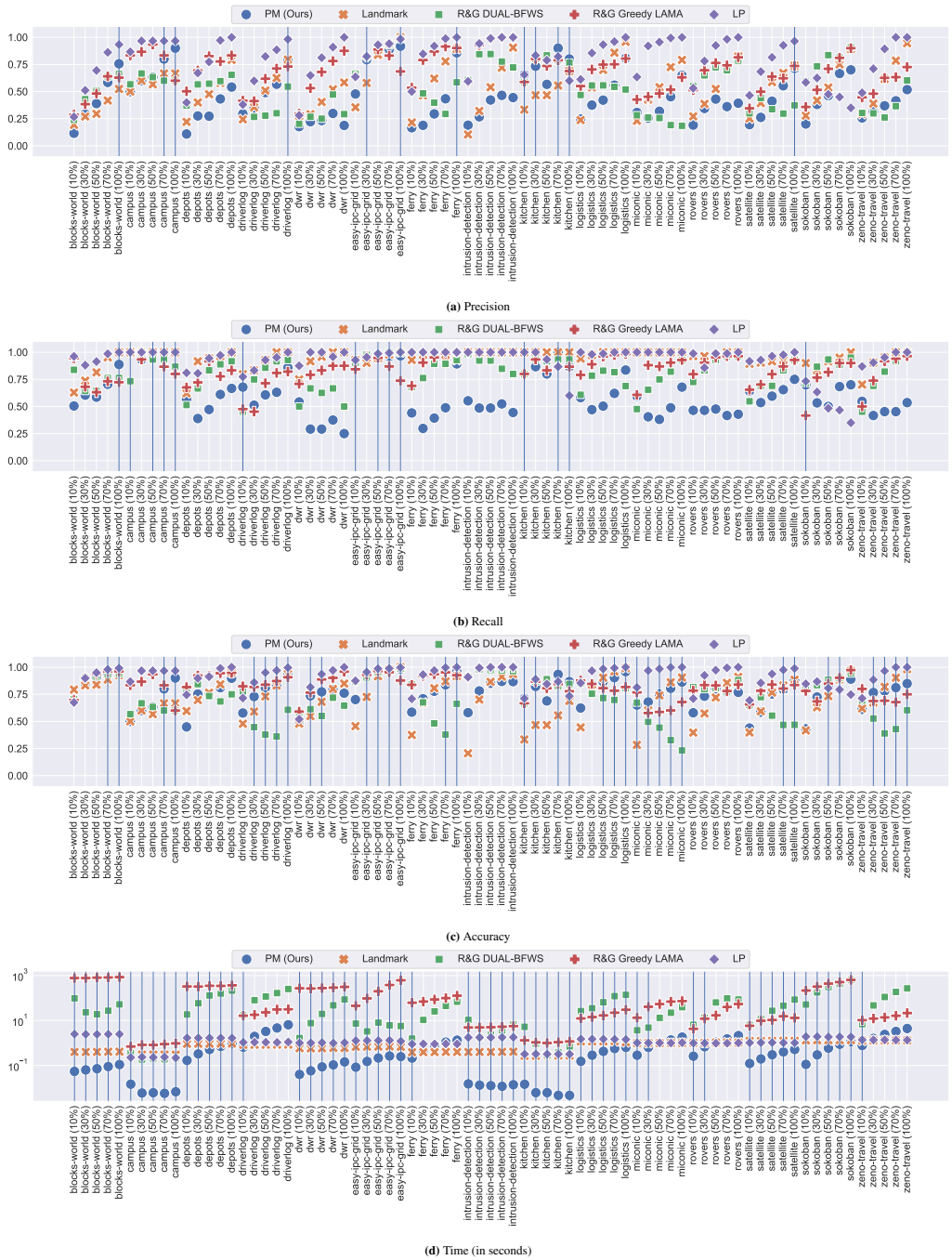
(b) Recall

(c) Accuracy

(d) Time (in seconds)

Figure 12: The differences of the GR performance between the system trained by the cost-optimal traces and the divergent traces.

The GR performance for domain knowledge-based approaches is listed in Appendix D. Figure 13 plots the precision, recall, accuracy, and time for different GR approaches in each domain for each level of observation. The blue dots represent the GR performance of our approach. We calculated the average of the precision, recall, and accuracy over the other approaches and compared it with the performance of our approach. A blue line represents that the performance of our approach is better than the mean of the other GR approaches for the corresponding domain and observation level. For the comparison of the execution time, the blue lines represent that our approach is the fastest in that domain with the level of observation.



(a) Precision

(b) Recall

(c) Accuracy

(d) Time (in seconds)

Figure 13: The performances of different GR approaches. For the precision, recall, and accuracy, the blue lines indicate that the performance of our approach is better than the average performance of the other GR approaches. For the time, the blue lines indicate that our approach uses the shortest recognition time (the fastest approach).

In Fig. 13, the plots show that our approach uses the shortest recognition time in 57 out of 75 cases. The precision of our GR approach is higher than the average of the other approaches in 12 out of 75 cases, and in 14 out of 75 the recall of our approach is higher than the average. As PRIM identified parameters to maximize accuracy, in approximately half of the cases (36 out of 75 cases), our accuracy is higher than the average of the other approaches. As other approaches (R&G variants) can access the full domain model, they can compute the cost difference between the optimal plan and any observed plan to infer the goal. However, our PM-based approach learns skill models based on a few rational (optimal or close to optimal) plans. If an observed plan has a large distance from plans in the learning dataset, the recognition accuracy of our PM-based approach tends to decrease. Hence, for some domains and some levels of observation, other GR approaches can outperform our PM-based approach. For example, in Section 2, the skill model for achieving goal `A` is learned from six rational traces (see Fig. 2a). If the first seven steps of the red irrational walk (in Fig. 1a) are observed, our PM-based GR system is unlikely to infer that the red walk intends to achieve goal `A`, because only the first step can match the steps in Fig. 2a. However, for the planning-based GR approaches, after seven steps of the red walk, the agent is located in cell $(3, 3)$, which is close to goal `A`. As a result, it is likely for these systems to infer goal `A`. In short, our PM-based GR system can recognize a goal accurately if it has observed some similar traces before (regardless of the rationality).

GR approach	Precision		Recall		Accuracy		Time	
	avg	std	avg	std	avg	std	avg	std
PM (Ours)	4.24	0.94	4.53	0.72	3.52	0.96	1.44	0.80
Landmark	3.55	1.02	2.17	0.85	3.75	1.26	2.03	0.71
R&G (DUAL-BFWS)	3.36	1.30	3.60	0.99	3.67	1.31	4.24	0.73
R&G (Greedy LAMA)	2.39	0.85	3.23	0.93	2.56	1.02	4.64	0.48
LP	1.45	0.96	1.47	1.05	1.51	1.02	2.65	0.60

Table 10: The average ranks of performance (avg) for each GR approach and the standard deviations (std) of the ranks.

Table 10 shows the average ranks of GR performance (precision, recall, accuracy, and time) for the five approaches mentioned above. Despite only using skill models learned from historical observations and without access to full domain knowledge, our approach achieves an accuracy level that is comparable to other GR approaches. Furthermore, our approach shows a clear performance advantage over existing GR approaches in terms of recognition speed. We note that, potentially, the R&G variants may use some form of *precomputation* to speed up the recognition by, for example, precomputing the probabilities "heatmaps" for each state or the so-called Radius of Maximum Probability (RMP) for each possible goal, as proposed by Masters and Sardiña [31, 30]. However, those techniques have been proposed for the special case of navigational grid-world settings, which enjoy a uniform and manageable state space. In our work, on the other hand, we deal with task-planning domains (beyond navigation) with arbitrary state space, so precomputation is less applicable or practical. In particular, RMP only provides the tipping point boundary in which a goal becomes the most probable but would not provide probabilities or ranks outside that boundary. More generally, precomputation is arguably a different and orthogonal issue to all approaches. As such, all the evaluated techniques have performed the recognition from scratch to allow meaningful and fair comparisons.

The relatively lower values for precision and recall reflect our PRIM parameter settings optimizing for accuracy. When no domain models are available but only historical traces, our GR system is the only approach among those evaluated here that can still be used.

5.5.2. Comparison with LSTM-based GR Approach

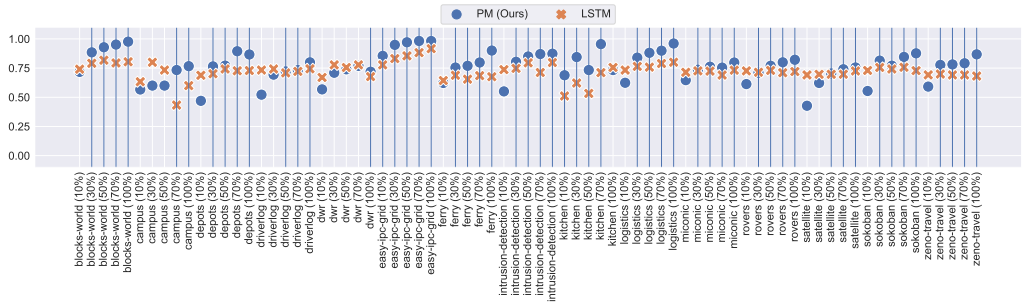
The implementation of the LSTM-based GR approach uses the configuration recommended by Min et al. [63]. The LSTM model, with a dropout rate of 0.75, comprises three layers: an embedding layer that converts actions to a 20-dimensional vector space, a layer with 100 self-connected memory cells (units), and a softmax layer for probability distribution over goal candidates, with the highest probability indicating the inferred goal. To handle actions that appear in testing traces but not in training traces, we set the number of distinct embeddings in the embedding layer to be the number of unique actions in the training traces, with an additional label for "unknown" actions. During the training of the LSTM model, we utilize a mini-batch size of 8, employ the cross entropy loss function, apply the stochastic gradient descent optimizer, and train for a total of 100 epochs. We evaluated the PM- and LSTM-based systems with two datasets: a small dataset of 10 traces for achieving each goal and a large dataset with 100 traces per goal. Note that the training datasets generated by the diverse planner and the top-k planner differ from the standard testing dataset provided by Pereira et al. [48]. The detailed GR performance results for the PM- and LSTM-based approaches trained with small and large datasets are listed in Table E.18 included in Appendix E. For three performance metrics of precision, recall, and accuracy, Table 11 shows the percentage and the number of cases (out of 75 total cases) where the PM-based system strictly outperforms the LSTM-based system.

	Precision	Recall	Accuracy
10 Traces	89% (67)	94% (71)	72% (54)
100 Traces	56% (42)	85% (64)	33% (25)

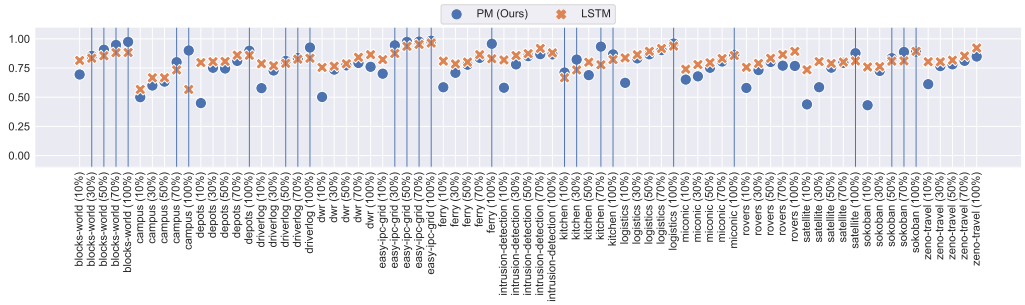
Table 11: The percentage (number) of cases out of 75 cases in which the PM-based system outperformed the LSTM-based system.

Figure 14 plots the accuracies of the PM-based and LSTM-based GR approaches trained with 10 and 100 traces per goal for all 75 cases. The blue vertical lines denote the cases in which the PM-based approach is more accurate than the LSTM-based approach. When trained with 10 traces per goal, the PM-based approach is more accurate in 54 out of 75 cases than the LSTM-based approach. However, the latter is more accurate more often when trained with 100 traces per goal. For precision and recall, regardless of the size of the training dataset, our approach outperforms the LSTM-based GR. In Appendix E, Figure E.17 plots all the comparison results for precision and recall on 10 and 100 training traces. The reason why our approach performs better on the precision and recall, while the LSTM-based approach performs better (with a large training dataset) on the accuracy, is that LSTM tends to have high true negative scores (TN, refer to Section 5.1.4). The problem instances in our dataset contain multiple goal candidates but only one true goal. If LSTM tends to infer only one goal (or few goals) among many goal candidates, the TN score is high, even if LSTM always infers a wrong goal. In contrast, our approach tends to infer more goals to increase the possibility of containing the true goal. Hence, the TN score of our approach tends to be low, especially in the situations of uncertainty, like when only a few observations have been made. We conclude that our approach performs better than the LSTM-based approach if the training dataset size is small and has comparable performance to the LSTM-based approach when the dataset is relatively large.

The explainability is another merit of the PM-based GR system. While the logic behind the GR inference is a black box for the LSTM-based GR, the GR results of the PM-based approach can be explained using alignments between observations and the learned Petri nets. The user can explore the commonalities and discrepancies between the observations and skill models for each goal and study how they contributed to the resulting probabilities assigned to each candidate



(a) Accuracy (trained with 10 traces per goal).



(b) Accuracy (trained with 100 traces per goal).

Figure 14: The accuracies of the PM-based GR approaches and the LSTM-based GR approach. The blue lines indicate that the accuracy of our approach is better than that of the LSTM-based GR approach.

goal; refer to the examples discussed in Section 3.2.2. In addition, the learned Petri nets explain the behavior of the agents toward candidate goals and, thus, give a general view of what agents are doing when striving for different goals. In contrast, the LSTM network only provides tuned parameters, which hardly delivers any insights into the inference.

5.6. Performance in Real-World Scenarios

To verify whether our GR system is applicable in real-world scenarios, we conducted the GR experiments using the real-world dataset mentioned in Section 5.1.2, and introduced the *random guess baseline* to compare with our GR performance. We considered our GR system applicable in the real-world if it outperforms the baseline. Note that we applied a PRIM analysis in the real-world domains to identify the best configuration parameters. The random guess baseline represents the performance of a GR approach that randomly returns any candidate goal or any combination of candidate goals. For example, if a GR problem instance has two candidate goals ($g1$ and $g2$), the random guess approach can randomly return $\{g1\}$, $\{g2\}$, or $\{g1, g2\}$.

The performance of the binary choice real-world GR problems is shown in Table 12. As mentioned in Section 5.1.2, we clustered the event log of each domain into subsets of traces to formulate sub-problems. The notation "BPIC 2011 (1)" in the table represents the first sub-problem in the *BPIC 2011* domain. The *Production* and *Traffic Fines* domains only have one sub-problem. Since binary choice problems only have two candidate goals, precision equals accuracy. Hence, we only show the values of precision to represent both precision and accuracy. In Table 12, the majority of precision and recall values are greater than the corresponding expected precision and recall of the random guess baseline GR approach, except for 15 out

of 190 cases (highlighted in red). The recognition time tends to increase as more actions are observed. Note that, if a recognition time is less than 0.01 seconds, we consider that GR problem instance to be recognized immediately, and the time is denoted with ε accordingly.

	BPIC 2011 (1)			BPIC 2011 (2)			BPIC 2011 (3)			BPIC 2011 (4)			BPIC 2015 (1)		
%O	p	r	t	p	r	t	p	r	t	p	r	t	p	r	t
10	0.47	0.84	0.32	0.56	0.96	1.98	0.59	0.94	0.48	0.55	0.92	1.05	0.53	0.90	0.45
30	0.65	0.91	0.72	0.60	0.89	5.75	0.66	0.88	1.26	0.62	0.88	2.86	0.65	0.74	1.72
50	0.62	0.86	1.40	0.67	0.92	11.00	0.63	0.83	2.18	0.62	0.88	4.64	0.76	0.80	3.51
70	0.59	0.84	1.94	0.70	0.91	14.37	0.81	0.96	2.95	0.66	0.88	6.85	0.71	0.74	7.31
100	0.69	0.86	2.58	0.66	0.85	15.61	0.79	0.97	3.11	0.84	0.97	8.41	0.82	0.83	10.30
Baseline	0.50	0.67		0.50	0.67		0.50	0.67		0.50	0.67		0.50	0.67	
	BPIC 2015 (2)			BPIC 2015 (3)			BPIC 2015 (4)			BPIC 2015 (5)			BPIC 2017 (1)		
%O	p	r	t	p	r	t	p	r	t	p	r	t	p	r	t
10	0.56	0.87	0.70	0.63	0.93	0.28	0.53	0.81	0.32	0.69	0.97	0.98	0.50	1.00	ε
30	0.56	0.71	2.07	0.45	0.59	1.63	0.65	0.97	1.16	0.54	0.79	2.68	0.52	0.98	ε
50	0.55	0.61	5.96	0.76	0.91	3.57	0.79	0.93	2.37	0.72	0.90	4.87	0.57	0.98	ε
70	0.58	0.64	6.63	0.85	0.90	4.66	0.91	0.93	4.05	0.84	0.89	7.30	0.59	0.97	ε
100	0.68	0.75	7.29	0.90	0.94	6.04	0.95	0.97	4.86	0.82	0.87	7.70	0.77	0.97	ε
Baseline	0.50	0.67		0.50	0.67		0.50	0.67		0.50	0.67		0.50	0.67	
	BPIC 2017 (2)			BPIC 2017 (3)			Hospital Billing (1)			Hospital Billing (2)			Production		
%O	p	r	t	p	r	t	p	r	t	p	r	t	p	r	t
10	0.50	1.00	ε	0.50	1.00	ε	0.50	1.00	ε	0.50	0.99	ε	0.50	1.00	0.01
30	0.52	0.97	ε	0.50	1.00	ε	0.68	0.99	ε	0.12	0.20	ε	0.52	1.00	0.01
50	0.56	0.96	ε	0.51	1.00	ε	0.97	0.98	ε	0.50	0.89	ε	0.43	0.83	0.01
70	0.59	0.94	ε	0.51	0.99	ε	0.98	0.98	ε	0.51	0.91	ε	0.52	0.87	0.01
100	0.78	0.97	ε	0.99	1.00	ε	0.99	0.99	ε	0.91	0.93	ε	0.50	0.87	0.01
Baseline	0.50	0.67		0.50	0.67		0.50	0.67		0.50	0.67		0.50	0.67	
	Sepsis Cases (1)			Sepsis Cases (2)			Sepsis Cases (3)			Traffic Fines					
%O	p	r	t	p	r	t	p	r	t	p	r	t			
10	0.49	0.97	ε	0.49	0.99	ε	0.49	0.97	ε	0.50	1.00	ε			
30	0.55	0.97	ε	0.46	0.87	ε	0.47	0.85	ε	0.62	0.80	ε			
50	0.59	0.96	ε	0.45	0.85	ε	0.50	0.89	ε	0.64	0.95	ε			
70	0.57	0.96	0.01	0.54	0.96	ε	0.47	0.91	0.01	0.68	0.99	ε			
100	0.61	0.94	0.01	0.91	0.97	ε	0.55	0.94	0.01	0.74	0.98	ε			
Baseline	0.50	0.67		0.50	0.67		0.50	0.67		0.50	0.67				

Table 12: GR performance of the binary-choice real-world GR problems; %O: the level of observation, p: precision, r: recall, t: time (in seconds), ε: *time* < 0.01. The performance worse than the random guess baseline is highlighted in red.

The performance of the multi-class real-world GR problems is shown in Table 13. The "Activities," "Build Prmt," and "Env Prmt" represent the datasets of *Activities of Daily Living*, *Building Permit Applications*, and *Environmental Permit Applications*, respectively. The 80%/20% split represents that the skill models were learned from 80% of the traces in that domain, and the remaining 20% of the traces were used for testing the GR performance. The (60%/40%) means that 60% of traces were used for learning and 40% of traces for testing. For the domains of "Activities" and "Build Prmt," all precision, recall, and accuracy values are greater for the PM-based GR system than for the random guess baseline. For the domain of "Env Prmt," there are two cases where the recall values are slightly lower than the baseline (annotated in red), while the corresponding precision and accuracy values are significantly higher than the baseline. These two cases indicate that our GR system tends to infer fewer goals, resulting in higher precision and accuracy, which, in turn, impacts recall. There are two accuracy values slightly lower than the baseline (annotated in red), which might indicate that 10% observations may not contain enough information to identify the true goal.

The precision values are sometimes low (even for 100% observations). The testing observations are not seen during the learning stage, and the GR precision relies on the ability of the learned Petri nets to generalize to unforeseen traces. If the learned Petri nets do not generalize

	Activities (80%/20%)				Build Prmt (80%/20%)				Env Prmt (80%/20%)			
%O	p	r	a	t	p	r	a	t	p	r	a	t
10	0.27	0.97	0.52	0.14	0.34	0.77	0.59	2.00	0.30	0.72	0.47	1.42
30	0.34	0.77	0.71	0.28	0.53	0.65	0.76	6.21	0.40	0.59	0.69	4.99
50	0.35	0.61	0.81	0.47	0.58	0.69	0.80	11.29	0.41	0.52	0.73	9.62
70	0.42	0.74	0.82	0.62	0.58	0.64	0.82	15.88	0.43	0.54	0.72	13.64
100	0.55	0.81	0.88	0.67	0.71	0.76	0.88	20.80	0.55	0.69	0.78	15.20
Baseline	0.13	0.50	0.50		0.20	0.52	0.49		0.20	0.52	0.49	
	Activities (60%/40%)				Build Prmt (60%/40%)				Env Prmt (60%/40%)			
%O	p	r	a	t	p	r	a	t	p	r	a	t
10	0.29	1.00	0.58	0.04	0.33	0.72	0.58	1.70	0.32	0.74	0.46	0.76
30	0.43	0.71	0.82	0.08	0.56	0.65	0.79	5.37	0.43	0.60	0.72	2.92
50	0.54	0.73	0.88	0.12	0.60	0.70	0.81	9.74	0.37	0.42	0.72	5.75
70	0.50	0.67	0.87	0.15	0.61	0.65	0.83	13.66	0.39	0.48	0.72	8.17
100	0.50	0.60	0.87	0.18	0.65	0.69	0.86	17.85	0.59	0.69	0.81	8.93
Baseline	0.13	0.50	0.50		0.20	0.52	0.49		0.20	0.52	0.49	

Table 13: GR performance of the multi-classes real-world GR problems; %O: the level of observation, p: precision, r: recall, a: accuracy, t: time (in seconds), 80%/20%: 80% of traces used for learning and 20% of traces used for testing, 60%/40%: 60% of traces used for learning and 40% of traces used for testing. The performance worse than the random guess baseline is highlighted in red.

well, our GR system may not infer the true goal, and the precision decreases. If the learned Petri nets manage to over-generalize, not distinguishing between different traces for different goals, then our GR system may infer multiple goals, decreasing the precision.

Our GR system outperforms the random guess baseline for binary choice and multi-class real-world GR problems. We conclude that our GR system is applicable in real-world scenarios.

6. Related Work

The typical GR approaches rely on predefined models derived either from plan libraries [2, 66, 23, 67] or domain knowledge [10, 11, 48, 62]. Some domain knowledge-based GR approaches consider handling irrational behaviors [32, 68]. Our work, on the other hand, aims to solve the GR problem by learning models from historical observations, relying on sets of observed plans. Similar work by Sohrabi et al. [69] also relies on sets of plans to solve GR problems. However, this approach takes a sequence of *states* rather than actions as input. The observed state sequence is mapped to each plan in the set according to the domain knowledge, and the mapping process identifies the noisy and missing observations. Taking the noisy and missing observations into account, the likelihood of the observed state sequence to match each plan in the set and the likelihood of the agent following that plan to achieve the goal are computed. However, this approach still requires domain knowledge for mapping a state sequence to an action sequence.

Some works in (statistical) learning propose performing GR based on historical behavior data. For example, one can learn the underlying domain transition model to support planning-based GR [12, 13, 70]. Alternatively, one can learn the decision-making model of the observed agent when executing an Hierarchical Task Network style plan library (which is known as a priori) or perform the end-to-end learning from observed behavior to the intended goal [63, 71]. Like our work, their overarching objective is to ease the traditional requirement of hand-crafting the observed agent model. However, those approaches require large training datasets and yield *black-box* type GR systems. In contrast, our approach can perform well even if the training datasets are small. In addition, our approach produces judgments that can be directly interpreted by relying on the structured processes synthesized and identified process misalignments.

A method proposed by Shvo et al. [14] trains interpretable *classifiers* that can solve GR problems in the absence of pre-defined models. This method learns deterministic finite automata (DFA) from training traces. Similar to our previous work [20], they replace Petri nets with DFAs and target the use case of early prediction of goals. The learned DFAs are graphical encodings of the training traces, and each DFA is associated with a class label. The probability distribution over class labels is computed based on the observed trace against the DFAs. This approach relies on the training traces and a transition function to assign the trace prefixes to the states in DFA. However, our approach purely relies on the traces.

In the business process management area, some existing works aim to predict the business goal of an incomplete business process (i.e., partially observed trace), which is referred to as outcome-oriented predictive process monitoring [54]. The existing methods [72, 73, 74, 75, 76] predict the class label (business goal) of a given trace based on trained classifiers. Similar to our GR approach, these works require an offline phase for learning classifiers and an online phase for predicting. As summarized by Teinemaa et al. [54], the outcome-oriented predictive process monitoring methods extract and filter traces from an event log to obtain the prefixes of the traces. Next, these methods divide the trace prefixes into multiple buckets for training several classifiers. Several bucketing approaches are used, such as the k-nearest neighbors (KNN) [72], the state-based approaches [73, 74], and the clustering-based approaches [75, 76]. Subsequently, the trace prefixes in each bucket are encoded to feature vectors, since training the classifiers requires the fixed-length feature vectors as input. Finally, the classifiers are trained with commonly used classification algorithms such as decision tree (DT), random forest (RF), or support vector machine (SVM). In the online predicting phase, the trained classifier assigns a class label to an observed trace. However, these outcome-oriented predictive process monitoring approaches are also non-interpretable artifacts. In contrast, we construct interpretable artifacts such that both skill models and alignments are interpretable.

Process discovery resembles action model learning [77, 78, 79, 80]. Whereas the aim of action model learning is to learn the dynamics of an underlying environment (e.g., PDDL models), process discovery aims to learn models that compactly describe sets of action sequences (goal-relevant plans, in our case) without relying on any information about the states of the environment. Hence, unlike existing works on learning action models, process discovery has fewer data requirements in that it does not require information about domain states. As such, places in our Petri nets do not represent domain states but rather the states of plans in a generalized manner. Importantly, also, discovery techniques are designed to work with human-driven processes and, thus, are made to be robust toward missing or noisy data. At the technical level, Petri nets and planning models (PDDL or STRIPS) have indeed been related in both directions, but for different purposes and needs than ours. For example, planning models were translated to Petri nets to perform concurrent planning using known Petri net unfolding techniques [81, 82], while Petri nets have been compiled into planning models to facilitate process analysis using planning technology [83, 84]. As stated, our Petri net models do not aim to represent dynamic systems or be used to perform planning. Instead, we use Petri nets as convenient compact representations of sets of plans and leverage significant existing work and tools to perform process discovery and alignment analysis.

The research on cognitive architectures [85, 86, 87, 88] attempts to model the core capabilities of humans, including, but not limited to, perception, attention mechanisms, action selection, learning, memory, reasoning, and metareasoning [89]. Our GR framework can be seen as a goal-intention recognition module of a cognitive architecture, such as ACT-R [85] and Soar [86]. Note also that the proposed in this work GR framework is specifically designed as an outline for

implementing process mining-based GR systems.

7. Conclusion

We presented a solution to the goal recognition problem that does *not* require pre-defined models of behavior in the domain of concern, as is often the case in many existing goal recognition approaches in the form of plan libraries or domain dynamic descriptions (e.g., planning domains). Instead, our GR approach leverages recorded past behaviors (captured as collections of event traces) to automatically learn skill models using process discovery techniques and, subsequently, infer the goal of the agent by checking conformance, or aligning, observations against the skill models. This perspective takes advantage of the fact that logs of past behaviors exist or can be collected with reasonable effort. In contrast, plan libraries or domain descriptions may not be readily available in many real-world domains (or are costly to produce).

More concretely, we recast the principled planning-based GR approaches based on the rationality assumption of agents within our set-up of learned skill models and process alignment to obtain a probabilistic GR based on seen traces of behavior. As our approach contains four parameters, we conducted a sensitivity analysis to verify that all four parameters have a significant impact on the GR accuracy and used the Patient Rule Induction Method (PRIM) to identify the parameters that yield high GR accuracy. We showed, experimentally, that despite relying on a limited number of past behaviors, our approach achieved an accuracy level comparable to other state-of-the-art GR approaches with full access to domain knowledge and that, in addition, the recognition speed of our approach is often faster. We also provided experimental results on real-world datasets for which no domain knowledge is readily available but for which logs of traces do exist. Such results provide evidence that the approach is able to perform GR quickly and accurately and *without predefined plan libraries or domain models*. Finally, we argue that our GR approach can be used to instantiate a GR framework inspired by the principles of observational learning from social cognitive learning theory, which constitutes a collection of components that can be selectively replaced to tune the performance of the system.

We acknowledge a range of limitations of our work thus far. For instance, we have not tested with a wider range of process discovery techniques, including learning and using stochastic process models that encode frequencies of the traces they were discovered from and models constructed from spurious observations. Also, other conformance checking techniques beyond alignments used in this work can be explored, and we have only considered alignment moves that involve trace actions. These limitations give rise to future work. Another interesting direction to pursue in future work is looking at non-stationary environments, that is, environments that change over time. Finally, future work will address the detailed design and implementation of the attention, retention, and motivation phases of the proposed goal recognition framework.

References

[1] S. Carberry, Techniques for plan recognition, User Modeling and User-Adapted Interaction 11 (2001) 31–48.

[2] H. A. Kautz, J. F. Allen, Generalized plan recognition, in: AAAI, 1986, pp. 32–37.

[3] G. Sukthankar, K. P. Sycara, A cost minimization approach to human behavior recognition, in: AAMAS, 2005, pp. 1067–1074.

[4] A. Kott, W. McEneaney, Adversarial reasoning: Computational approaches to reading the opponent's mind, CRC Press, 2006.

[5] S. Lefèvre, D. Vasquez, C. Laugier, A survey on motion prediction and risk assessment for intelligent vehicles, Robomech Journal 1 (2014) 1–14.

[6] J. Firl, Q. Tran, Probabilistic maneuver prediction in traffic scenarios, in: ECMR, 2011, pp. 89–94.
[7] J. F. P. Kooij, N. Schneider, F. Flohr, D. M. Gavrila, Context-based pedestrian path prediction, in: ECCV (6), volume 8694 of *LNCS*, 2014, pp. 618–633.
[8] N. Lesh, C. Rich, C. L. Sidner, Using plan recognition in human-computer collaboration, in: UM, 1999, pp. 23–32.
[9] R. Demolombe, E. Hamon, What does it mean that an agent is performing a typical procedure? A formal definition in the situation calculus, in: AAMAS, 2002, pp. 905–911.
[10] M. Ramírez, H. Geffner, Plan recognition as planning, in: IJCAI, 2009, pp. 1778–1783.
[11] M. Ramírez, H. Geffner, Probabilistic plan recognition using off-the-shelf classical planners, in: AAAI, 2010, pp. 1121–1126.
[12] L. Amado, R. F. Pereira, J. P. Aires, M. C. Magnaguagno, R. Granada, F. Meneguzzi, Goal recognition in latent space, in: IJCNN, 2018, pp. 1–8.
[13] L. Amado, R. Mirsky, F. Meneguzzi, Goal recognition as reinforcement learning, in: AAAI, 2022, pp. 9644–9651.
[14] M. Shvo, A. C. Li, R. T. Icarte, S. A. McIlraith, Interpretable sequence classification via discrete optimization, in: AAAI, 2021, pp. 9647–9656.
[15] P. Haslum, N. Lipovetzky, D. Magazzeni, C. Muise, An Introduction to the Planning Domain Definition Language, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers, 2019.
[16] Z. Su, A. Polyvyanyy, N. Lipovetzky, S. Sardiña, N. van Beest, GRACE: A simulator for continuous goal recognition over changing environments, in: PMAI@IJCAI, volume 3310 of *CEUR Workshop Proceedings*, 2022, pp. 37–48.
[17] W. M. P. van der Aalst, Process Mining — Data Science in Action, Second Edition, Springer, 2016.
[18] R. Fikes, N. J. Nilsson, STRIPS: A new approach to the application of theorem proving to problem solving, Artificial Inteligence 2 (1971) 189–208.
[19] N. Wilken, H. Stuckenschmidt, Combining symbolic and statistical knowledge for goal recognition in smart home environments, in: PerCom Workshops, 2021, pp. 26–31.
[20] A. Polyvyanyy, Z. Su, N. Lipovetzky, S. Sardiña, Goal recognition using off-the-shelf process mining techniques, in: AAMAS, 2020, pp. 1072–1080.
[21] C. F. Schmidt, N. S. Sridharan, J. L. Goodson, The plan recognition problem: An intersection of psychology and artificial intelligence, Artificial Inteligence 11 (1978) 45–83.
[22] G. Sukthankar, C. Geib, H. H. Bui, D. Pynadath, R. P. Goldman, Plan, activity, and intent recognition: Theory and practice, Newnes, 2014.
[23] D. Avrahami-Zilberbrand, G. A. Kaminka, Fast and complete symbolic plan recognition, in: IJCAI, 2005, pp. 653–658.
[24] D. V. Pynadath, M. P. Wellman, Probabilistic state-dependent grammars for plan recognition, in: UAI, 2000, pp. 507–514.
[25] J. Hong, Goal recognition through goal graph analysis, Journal of Artificial Intelligence Research 15 (2001) 1–30.
[26] C. L. Baker, R. Saxe, J. B. Tenenbaum, Action understanding as inverse planning, Cognition 113 (2009) 329–349.
[27] D. Pattison, D. Long, Accurately determining intermediate and terminal plan states using Bayesian goal recognition, in: D. Pattison, D. Long, C. Geib (Eds.), GAPRec 2011. Proceedings of the First Workshop on Goal, Activity and Plan Recognition, 2011, pp. 32–37.
[28] M. Ghallab, D. S. Nau, P. Traverso, Automated planning — Theory and practice, Elsevier, 2004.
[29] P. Masters, S. Sardiña, Cost-based goal recognition for path-planning, in: AAMAS, ACM, 2017, pp. 750–758.
[30] P. Masters, S. Sardiña, Cost-based goal recognition in navigational domains, Journal of Artificial Intelligence Research 64 (2019) 197–242.
[31] P. Masters, S. Sardiña, Goal recognition for rational and irrational agents, in: AAMAS, 2019, pp. 440–448.
[32] P. Masters, S. Sardiña, Expecting the unexpected: Goal recognition for rational and irrational agents, Artificial Inteligence 297 (2021) 103490.
[33] M. Vered, G. A. Kaminka, S. Biham, Online goal recognition through mirroring: Humans and agents, in: Advances in Cognitive Systems, 2016.
[34] W. M. P. van der Aalst, T. Weijters, L. Maruster, Workflow mining: Discovering process models from event logs, IEEE Transactions on Knowledge and Data Engineering 16 (2004) 1128–1142.
[35] W. M. P. van der Aalst, The application of Petri nets to workflow management, Journal of Circuits, Systems and Computers 8 (1998) 21–66.
[36] A. Augusto, R. Conforti, M. Dumas, M. La Rosa, A. Polyvyanyy, Split miner: Automated discovery of accurate and simple business process models from event logs, Knowledge and Information Systems 59 (2019) 251–284.
[37] M. Weidlich, A. Polyvyanyy, N. Desai, J. Mendling, M. Weske, Process compliance analysis based on behavioural profiles, Information Systems 36 (2011) 1009–1025.
[38] W. M. P. van der Aalst, A. Adriansyah, B. F. van Dongen, Replaying history on process models for conformance checking and performance analysis, WIREs Data Mining and Knowledge Discovery 2 (2012) 182–192.
[39] J. Carmona, B. F. van Dongen, A. Solti, M. Weidlich, Conformance Checking — Relating Processes and Models,

Springer, 2018.
[40] A. Polyvyanyy, A. Solti, M. Weidlich, C. Di Ciccio, J. Mendling, Monotone precision and recall measures for comparing executions and specifications of dynamic systems, ACM Transactions on Software Engineering and Methodology 29 (2020) 17:1–17:41.
[41] A. Bandura, Observational Learning, American Cancer Society, 2008.
[42] J. M. E. M. van der Werf, A. Polyvyanyy, B. R. van Wensveen, M. J. S. Brinkhuis, H. A. Reijers, All that glitters is not gold: Four maturity stages of process discovery algorithms, Information Systems 114 (2023) 102155.
[43] S. J. J. Leemans, W. M. P. van der Aalst, T. Brockhoff, A. Polyvyanyy, Stochastic process mining: Earth movers' stochastic conformance, Information Systems 102 (2021) 101724.
[44] D. Borsa, N. Heess, B. Piot, S. Liu, L. Hasenclever, R. Munos, O. Pietquin, Observational learning by reinforcement learning, in: AAMAS, 2019, pp. 1117–1124.
[45] S. J. J. Leemans, E. Poppe, M. T. Wynn, Directly follows-based process mining: Exploration & a case study, in: ICPM, 2019, pp. 25–32.
[46] S. J. J. Leemans, D. Fahland, W. M. P. van der Aalst, Process and deviation exploration with inductive visual miner, in: BPM Demos, volume 1295 of *CEUR Workshop Proceedings*, 2014, p. 46.
[47] A. Adriansyah, N. Sidorova, B. F. van Dongen, Cost-based fitness in conformance checking, in: ACSD, 2011, pp. 57–66.
[48] R. F. Pereira, N. Oren, F. Meneguzzi, Landmark-based approaches for goal recognition as planning, Artificial Inteligence 279 (2020).
[49] R. F. Pereira, N. Oren, F. Meneguzzi, Landmark-based heuristics for goal recognition, in: AAAI, 2017, pp. 3622–3628.
[50] M. Katz, S. Sohrabi, O. Udrea, D. Winterer, A novel iterative approach to top-k planning, in: ICAPS, 2018, pp. 132–140.
[51] M. Katz, S. Sohrabi, Reshaping diverse planning, in: AAAI, 2020, pp. 9892–9899.
[52] M. Fox, A. Gerevini, D. Long, I. Serina, Plan stability: Replanning versus plan repair, in: ICAPS, 2006, pp. 212–221.
[53] A. Coman, H. Muñoz-Avila, Generating diverse plans using quantitative and qualitative plan distance metrics, in: AAAI, 2011.
[54] I. Teinemaa, M. Dumas, M. La Rosa, F. M. Maggi, Outcome-oriented predictive process monitoring: Review and benchmark, ACM Transactions on Knowledge Discovery from Data 13 (2019) 17:1–17:57.
[55] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, S. Tarantola, Global sensitivity analysis: The primer, John Wiley & Sons, 2008.
[56] J. H. Kwakkel, The exploratory modeling workbench: An open source toolkit for exploratory modeling, scenario discovery, and (multi-objective) robust decision making, Environmental Modelling & Software 96 (2017) 239–250.
[57] I. Sobol, Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates, Mathematics and Computers in Simulation 55 (2001) 271–280.
[58] X. Y. Zhang, M. N. Trame, L. J. Lesko, S. Schmidt, Sobol sensitivity analysis: A tool to guide the development and evaluation of systems pharmacology models, CPT: Pharmacometrics & Systems Pharmacology 4 (2015) 69–79.
[59] J. Helton, F. Davis, Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems, Reliability Engineering & System Safety 81 (2003) 23–69.
[60] B. P. Bryant, R. J. Lempert, Thinking inside the box: A participatory, computer-assisted approach to scenario discovery, Technological Forecasting and Social Change 77 (2010) 34–49.
[61] J. H. Friedman, N. I. Fisher, Bump hunting in high-dimensional data, Statistics and Computing 9 (1999) 123–143.
[62] L. R. A. Santos, F. Meneguzzi, R. F. Pereira, A. G. Pereira, An lp-based approach for goal recognition as planning, in: AAAI, 2021, pp. 11939–11946.
[63] W. Min, B. W. Mott, J. P. Rowe, B. Liu, J. C. Lester, Player goal recognition in open-world digital games with long short-term memory networks, in: IJCAI, 2016, pp. 2590–2596.
[64] S. Richter, M. Helmert, M. Westphal, Landmarks revisited, in: AAAI, 2008, pp. 975–982.
[65] N. Lipovetzky, H. Geffner, Best-first width search: Exploration and exploitation in classical planning, in: AAAI, 2017, pp. 3590–3596.
[66] E. Charniak, R. P. Goldman, A Bayesian model of plan recognition, Artificial Inteligence 64 (1993) 53–79.
[67] C. W. Geib, R. P. Goldman, Partial observability and probabilistic plan/goal recognition, in: Proceedings of the International workshop on modeling other agents from observations (MOO-05), volume 8, 2005, pp. 1–6.
[68] T. Zhi-Xuan, J. L. Mann, T. Silver, J. Tenenbaum, V. Mansinghka, Online Bayesian goal inference for boundedly rational planning agents, in: NeurIPS, 2020.
[69] S. Sohrabi, A. V. Riabov, O. Udrea, Plan recognition as planning revisited, in: IJCAI, 2016, pp. 3258–3264.
[70] R. F. Pereira, M. Vered, F. Meneguzzi, M. Ramírez, Online probabilistic goal recognition over nominal models, in: IJCAI, 2019, pp. 5547–5553.
[71] W. Min, E. Ha, J. P. Rowe, B. W. Mott, J. C. Lester, Deep learning-based goal recognition in open-ended digital

games, in: AIIDE, 2014.
[72] F. M. Maggi, C. Di Francescomarino, M. Dumas, C. Ghidini, Predictive monitoring of business processes, in: CAiSE, volume 8484 of *LNCS*, 2014, pp. 457–472.
[73] G. T. Lakshmanan, S. Duan, P. T. Keyser, F. Curbera, R. Khalaf, Predictive analytics for semi-structured case oriented business processes, in: BPM Workshops, volume 66 of *LNBIP*, 2010, pp. 640–651.
[74] W. M. P. van der Aalst, V. A. Rubin, H. M. W. Verbeek, B. F. van Dongen, E. Kindler, C. W. Günther, Process mining: A two-step approach to balance between underfitting and overfitting, Software and Systems Modeling 9 (2010) 87–111.
[75] C. D. Francescomarino, M. Dumas, F. M. Maggi, I. Teinemaa, Clustering-based predictive process monitoring, IEEE Transactions on Services Computing 12 (2019) 896–909.
[76] I. Verenich, M. Dumas, M. La Rosa, F. M. Maggi, C. Di Francescomarino, Complex symbolic sequence clustering and multiple classifiers for predictive process monitoring, in: BPM Workshops, volume 256 of *LNBIP*, 2015, pp. 218–229.
[77] R. García-Martínez, D. Borrajo, An integrated approach of learning, planning, and execution, Journal of Intelligent & Robotic Systems 29 (2000) 47–78.
[78] D. Shahaf, E. Amir, Learning partially observable action schemas, in: AAAI, 2006, pp. 913–919.
[79] L. Lamanna, A. Saetti, L. Serafini, A. Gerevini, P. Traverso, Online learning of action models for PDDL planning, in: IJCAI, 2021, pp. 4112–4118.
[80] V. Mehta, B. Paria, J. Schneider, S. Ermon, W. Neiswanger, An experimental design perspective on model-based reinforcement learning, in: ICLR, 2022.
[81] B. Bonet, P. Haslum, S. L. Hickmott, S. Thiébaux, Directed unfolding of Petri nets, Transactions on Petri Nets and Other Models of Concurrency 1 (2008) 172–198.
[82] S. L. Hickmott, S. Sardiña, Optimality properties of planning via Petri net unfolding: A formal analysis, in: ICAPS, 2009.
[83] S. Edelkamp, S. Jabbar, Action planning for directed model checking of Petri nets, in: MoChArt@CONCUR/SPIN, volume 149 of *Electronic Notes in Theoretical Computer Science*, 2005, pp. 3–18.
[84] M. de Leoni, G. Lanciano, A. Marrella, Aligning partially-ordered process-execution traces and models using automated planning, in: ICAPS, 2018, pp. 321–329.
[85] J. R. Anderson, How Can the Human Mind Occur in the Physical Universe?, Oxford University Press, 2007.
[86] J. E. Laird, Extending the soar cognitive architecture, in: AGI, volume 171 of *Frontiers in Artificial Intelligence and Applications*, 2008, pp. 224–235.
[87] E. Gat, Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots, in: AAAI, 1992, pp. 809–815.
[88] R. J. Firby, R. E. Kahn, P. N. Prokopowicz, M. J. Swain, An architecture for vision and action, in: IJCAI, 1995, pp. 72–81.
[89] I. Kotseruba, J. K. Tsotsos, 40 years of cognitive architectures: Core cognitive abilities and practical applications, Artificial Intelligence Review 53 (2020) 17–94.

Appendix A. Skipped GR Problems

The total number of GR problem instances, the number of skipped instances, and the number of remaining instances used in our evaluations, for each synthetic domain, are shown in Table A.14. The number of skipped instances and remaining instances for the top-k planner and the diverse planner are recorded in separate columns. In the domains of Blocks-world, Depots, DWR, and Sokoban, the top-k or the diverse planner skip a portion of the total instances. For the domains in which some instances are skipped, in Table A.15, we display the total number of instances and the numbers of skipped instances for each level of observations (10%, 30%, 50%, 70%, and 100%).

Domain	Instances	top-k planner		diverse planner	
		skipped	remaining	skipped	remaining
Blocks-world	1076	12	1064	152	924
Campus	75	0	75	75	0
Depots	364	0	364	203	161
Driverlog	364	0	364	0	364
DWR	364	0	364	260	104
Easy-ipc-grid	673	0	673	0	673
Ferry	364	0	364	0	364
Intrusion-detection	465	0	465	0	465
Kitchen	75	0	75	75	0
Logistics	673	0	673	0	673
Miconic	364	0	364	0	364
Rovers	364	0	364	0	364
Satellite	364	0	364	0	364
Sokoban	364	0	364	104	260
Zeno-travel	364	0	364	0	364

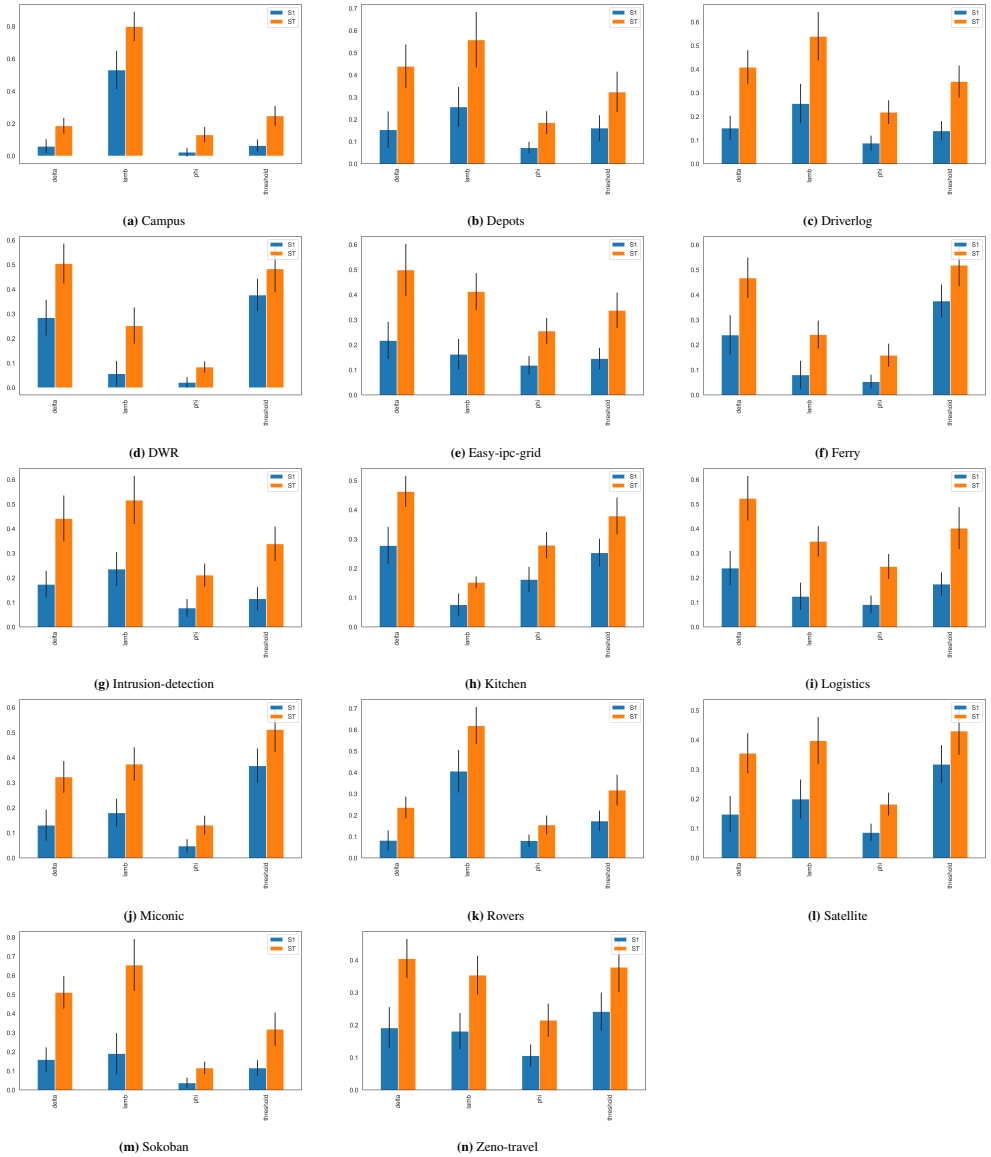
Table A.14: The number of skipped problems in synthetic domains for the top-k planner and the diverse planner.

Domain (planner)	10%		30%		50%		70%		100%	
	total	skipped	total	skipped	total	skipped	total	skipped	total	skipped
Blocks-world (top-k)	246	0	246	0	246	0	246	0	92	12
Blocks-world (diverse)	246	36	246	36	246	36	246	36	92	12
Depots (diverse)	84	47	84	47	84	47	84	47	28	16
DWR (diverse)	84	60	84	60	84	60	84	60	28	20
Sokoban (diverse)	84	24	84	24	84	24	84	24	28	8

Table A.15: The number of skipped problems for each level of observations (for the domains with some, and not all, skipped instances).

Appendix B. Sobol Sensitivity Analysis

The results of the Sobol sensitivity analysis on the synthetic domains (excluding the domain of Blocks-world discussed in Section 5.2) with the cost-optimal traces generated by the top-k planner are shown in Fig. B.15.



(a) Campus **(b)** Depots **(c)** Driverlog

(d) DWR **(e)** Easy-ipc-grid **(f)** Ferry

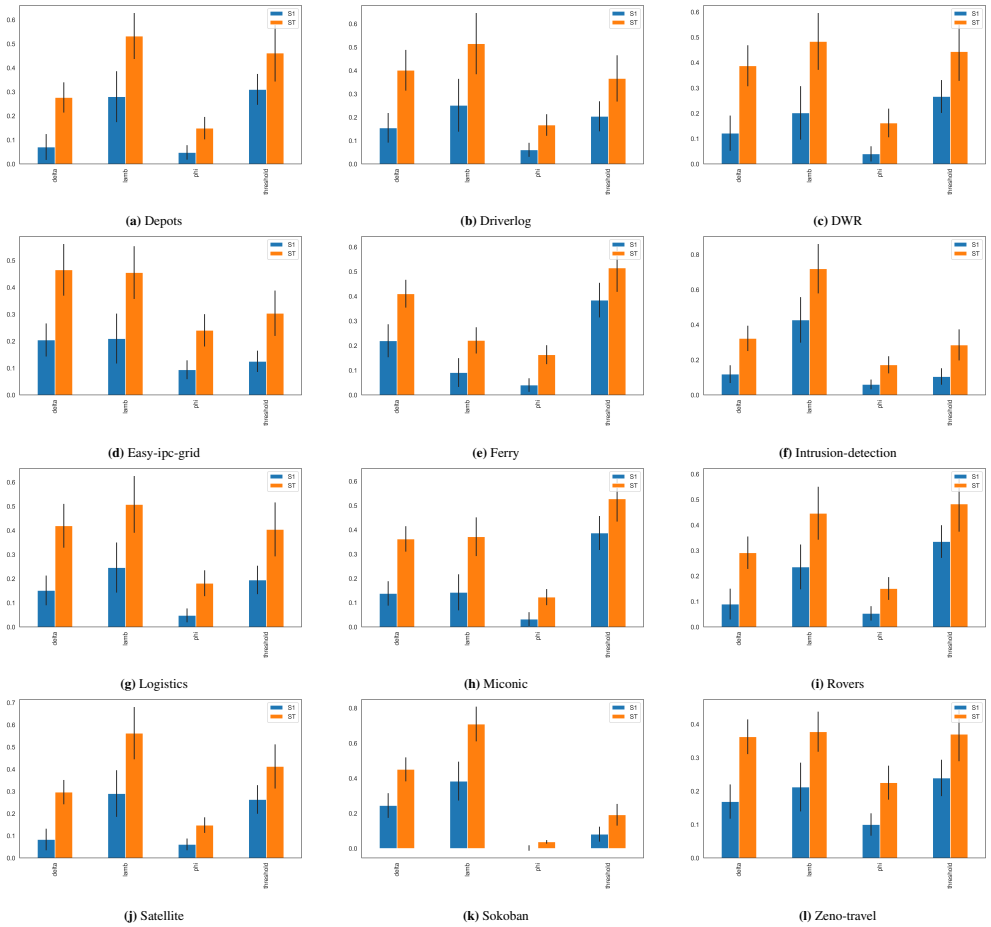
(g) Intrusion-detection **(h)** Kitchen **(i)** Logistics

(j) Miconic **(k)** Rovers **(l)** Satellite

(m) Sokoban **(n)** Zeno-travel

Figure B.15: Sobol sensitivity analysis (the GR system was trained with the cost-optimal traces).

The results of the Sobol sensitivity analysis on the synthetic domains (excluding the domain of Blocks-world discussed in Section 5.2) with the divergent traces generated by the diverse planner are shown in Fig. B.16.



(a) Depots **(b)** Driverlog **(c)** DWR

(d) Easy-ipc-grid **(e)** Ferry **(f)** Intrusion-detection

(g) Logistics **(h)** Miconic **(i)** Rovers

(j) Satellite **(k)** Sokoban **(l)** Zeno-travel

Figure B.16: Sobol sensitivity analysis (the GR system was trained with the divergent traces).

Appendix C. Performance of GR Approaches Trained Using Cost-Optimal and Divergent Traces and Configured Using PRIM and Default Parameters

Domain	%O	PRIM (cost-optimal traces)				Default (cost-optimal traces)				PRIM (divergent traces)				Default (divergent traces)			
		p	r	a	t	p	r	a	t	p	r	a	t	p	r	a	t
blocks-world	10	0.13	0.67	0.53	0.04	0.05	1.00	0.05	0.05	0.12	0.50	0.69	0.07	0.05	1.00	0.05	0.07
	30	0.25	0.79	0.70	0.04	0.03	0.99	0.05	0.06	0.31	0.60	0.87	0.07	0.05	0.95	0.10	0.07
	50	0.33	0.77	0.74	0.04	0.05	0.99	0.26	0.04	0.39	0.58	0.91	0.08	0.07	0.91	0.30	0.08
	70	0.49	0.91	0.78	0.05	0.13	1.00	0.53	0.05	0.58	0.70	0.95	0.10	0.20	0.90	0.65	0.10
	100	0.76	1.00	0.93	0.05	0.43	1.00	0.82	0.06	0.76	0.89	0.98	0.12	0.43	0.97	0.82	0.12
campus	10	0.50	1.00	0.50	0.01	0.50	1.00	0.50	0.01	—	—	—	—	—	—	—	—
	30	0.60	0.93	0.60	5.96E-3	0.50	1.00	0.50	6.81E-3	—	—	—	—	—	—	—	—
	50	0.63	1.00	0.63	6.57E-3	0.50	1.00	0.50	6.85E-3	—	—	—	—	—	—	—	—
	70	0.80	1.00	0.80	5.51E-3	0.57	1.00	0.57	6.14E-3	—	—	—	—	—	—	—	—
	100	0.90	1.00	0.90	6.53E-3	0.70	1.00	0.70	6.95E-3	—	—	—	—	—	—	—	—
depots	10	0.16	0.85	0.31	0.01	0.11	1.00	0.11	0.01	0.11	0.57	0.45	0.19	0.12	1.00	0.12	0.15
	30	0.21	0.88	0.41	0.01	0.13	0.96	0.23	0.01	0.28	0.42	0.75	0.35	0.22	1.00	0.35	0.26
	50	0.26	0.92	0.44	0.01	0.16	0.95	0.32	0.01	0.25	0.36	0.77	0.55	0.20	0.92	0.39	0.41
	70	0.29	0.94	0.45	0.02	0.22	0.96	0.40	0.02	0.46	0.64	0.84	0.74	0.30	0.94	0.58	0.54
	100	0.41	0.96	0.48	0.01	0.34	0.93	0.46	0.01	0.58	0.67	0.91	0.98	0.66	1.00	0.84	0.73
driverlog	10	0.34	0.94	0.41	9.75E-3	0.15	1.00	0.15	0.01	0.29	0.68	0.58	0.68	0.15	0.99	0.17	0.69
	30	0.26	0.88	0.41	8.43E-3	0.13	0.96	0.24	8.80E-3	0.35	0.51	0.74	2.09	0.22	0.80	0.45	2.13
	50	0.34	0.94	0.45	8.83E-3	0.24	0.95	0.37	7.67E-3	0.48	0.60	0.82	3.43	0.42	0.83	0.68	3.48
	70	0.34	0.94	0.45	9.51E-3	0.27	0.96	0.41	9.65E-3	0.56	0.62	0.84	4.85	0.52	0.80	0.76	4.85
	100	0.42	0.93	0.48	7.85E-3	0.41	0.96	0.47	7.32E-3	0.79	0.86	0.93	6.61	0.64	0.93	0.82	6.82
dwr	10	0.33	0.52	0.71	0.02	0.22	0.87	0.39	0.02	0.18	0.54	0.50	0.04	0.15	1.00	0.15	0.04
	30	0.39	0.58	0.83	0.04	0.28	0.76	0.65	0.04	0.22	0.29	0.74	0.06	0.17	0.83	0.30	0.06
	50	0.55	0.74	0.89	0.05	0.50	0.87	0.81	0.05	0.25	0.33	0.78	0.10	0.29	0.83	0.58	0.09
	70	0.54	0.70	0.90	0.07	0.51	0.87	0.85	0.07	0.28	0.38	0.79	0.12	0.31	0.83	0.59	0.11
	100	0.80	0.86	0.95	0.07	0.79	0.96	0.92	0.08	0.12	0.12	0.74	0.16	0.35	0.62	0.81	0.16
easy-ipc-grid	10	0.34	0.96	0.46	0.01	0.13	1.00	0.13	0.01	0.48	0.90	0.70	0.09	0.16	1.00	0.23	0.09
	30	0.54	0.99	0.67	0.02	0.23	1.00	0.39	0.02	0.79	0.95	0.94	0.17	0.45	1.00	0.67	0.17
	50	0.59	1.00	0.68	0.02	0.45	1.00	0.59	0.02	0.89	0.97	0.97	0.24	0.69	0.97	0.87	0.23
	70	0.61	1.00	0.69	0.02	0.56	1.00	0.66	0.02	0.89	0.96	0.98	0.29	0.79	0.96	0.92	0.29
	100	0.73	1.00	0.75	0.02	0.72	1.00	0.75	0.02	0.92	0.97	0.98	0.27	0.86	0.95	0.97	0.27
ferry	10	0.17	0.75	0.37	0.01	0.13	1.00	0.13	0.01	0.16	0.40	0.59	0.23	0.13	1.00	0.14	0.24
	30	0.13	0.65	0.45	9.97E-3	0.11	0.92	0.29	0.01	0.18	0.29	0.70	0.52	0.18	0.90	0.40	0.52
	50	0.18	0.64	0.51	0.01	0.20	0.90	0.42	0.01	0.29	0.42	0.78	0.86	0.26	0.76	0.58	0.86
	70	0.29	0.77	0.54	0.01	0.33	0.95	0.54	0.01	0.41	0.45	0.83	1.15	0.54	0.85	0.81	1.16
	100	0.48	0.93	0.59	0.01	0.51	1.00	0.59	0.01	0.84	0.89	0.95	1.45	0.76	0.96	0.89	1.43
intrusion-detection	10	0.32	0.52	0.78	0.01	0.07	0.70	0.04	0.01	0.19	0.55	0.58	0.02	0.07	0.70	0.04	0.02
	30	0.43	0.58	0.89	0.01	0.14	0.77	0.47	0.02	0.26	0.50	0.77	0.01	0.10	0.64	0.25	0.01
	50	0.46	0.55	0.91	0.02	0.28	0.65	0.79	0.02	0.42	0.49	0.85	0.01	0.29	0.64	0.71	0.01
	70	0.55	0.61	0.93	0.02	0.47	0.66	0.89	0.02	0.47	0.52	0.87	0.01	0.41	0.66	0.80	0.01
	100	0.43	0.49	0.91	0.02	0.42	0.49	0.89	0.02	0.44	0.44	0.87	0.02	0.43	0.51	0.81	0.01
kitchen	10	0.66	1.00	0.71	0.01	0.33	1.00	0.33	0.01	—	—	—	—	—	—	—	—
	30	0.73	0.87	0.82	6.02E-3	0.48	1.00	0.49	5.84E-3	—	—	—	—	—	—	—	—
	50	0.57	0.80	0.69	5.72E-3	0.51	1.00	0.51	5.97E-3	—	—	—	—	—	—	—	—
	70	0.90	1.00	0.93	5.09E-3	0.44	1.00	0.47	5.40E-3	—	—	—	—	—	—	—	—
	100	0.80	0.87	0.87	4.15E-3	0.49	1.00	0.56	4.55E-3	—	—	—	—	—	—	—	—
logistics	10	0.41	0.86	0.67	0.01	0.10	1.00	0.10	0.01	0.24	0.58	0.61	0.16	0.10	0.98	0.14	0.17
	30	0.40	0.78	0.71	0.02	0.20	0.97	0.53	0.01	0.38	0.48	0.83	0.31	0.24	0.78	0.61	0.30
	50	0.48	0.81	0.72	0.02	0.33	0.97	0.65	0.02	0.43	0.52	0.87	0.45	0.38	0.87	0.78	0.45
	70	0.56	0.88	0.74	0.02	0.45	0.98	0.71	0.02	0.57	0.63	0.91	0.60	0.48	0.88	0.84	0.60
	100	0.62	0.84	0.78	0.02	0.55	0.98	0.76	0.02	0.82	0.84	0.96	0.66	0.70	0.95	0.92	0.67
miconic	10	0.22	0.64	0.49	0.01	0.16	0.94	0.19	0.01	0.31	0.60	0.65	0.30	0.17	0.86	0.28	0.31
	30	0.19	0.56	0.59	0.02	0.14	0.87	0.34	0.02	0.26	0.40	0.68	0.68	0.22	0.77	0.46	0.68
	50	0.20	0.56	0.60	0.02	0.16	0.80	0.46	0.02	0.34	0.42	0.76	1.05	0.32	0.75	0.58	1.05
	70	0.20	0.50	0.61	0.02	0.19	0.69	0.51	0.02	0.48	0.50	0.82	1.41	0.46	0.75	0.73	1.41
	100	0.35	0.61	0.63	0.02	0.31	0.68	0.57	0.02	0.68	0.71	0.88	1.92	0.70	0.75	0.86	1.94
rovers	10	0.24	0.93	0.36	8.74E-3	0.17	1.00	0.17	9.04E-3	0.17	0.46	0.56	0.27	0.17	0.92	0.23	0.28
	30	0.22	0.86	0.40	7.43E-3	0.11	1.00	0.11	7.36E-3	0.32	0.48	0.71	0.72	0.22	0.85	0.38	0.72
	50	0.15	0.74	0.39	6.10E-3	0.13	0.98	0.23	6.26E-3	0.43	0.51	0.79	1.11	0.29	0.81	0.52	1.12
	70	0.15	0.70	0.38	7.83E-3	0.14	0.89	0.30	6.87E-3	0.42	0.50	0.79	1.57	0.36	0.81	0.66	1.55
	100	0.19	0.71	0.39	6.46E-3	0.16	0.75	0.35	6.19E-3	0.43	0.46	0.79	2.22	0.45	0.93	0.67	2.20
satellite	10	0.17	0.89	0.24	0.01	0.16	1.00	0.16	0.01	0.20	0.64	0.44	0.13	0.16	1.00	0.16	0.13
	30	0.15	0.88	0.32	0.01	0.11	1.00	0.11	9.41E-3	0.26	0.55	0.58	0.21	0.20	0.83	0.33	0.21
	50	0.19	0.87	0.38	9.44E-3	0.15	0.95	0.25	0.01	0.41	0.60	0.75	0.32	0.32	0.82	0.54	0.32
	70	0.27	0.92	0.42	9.64E-3	0.18	0.96	0.31	0.01	0.55	0.65	0.79	0.43	0.41	0.82	0.64	0.43
	100	0.39	0.93	0.48	7.29E-3	0.26	0.96	0.40	8.11E-3	0.71	0.75	0.88	0.56	0.57	0.86	0.76	0.56
sokoban	10	0.33	0.67	0.62	0.04	0.17	0.96	0.20	0.04	0.20	0.70	0.43	0.12	0.16	0.98	0.18	0.12
	30	0.36	0.57	0.79	0.07	0.33	0.75	0.67	0.07	0.38	0.53	0.72	0.32	0.35	0.70	0.60	0.33
	50	0.41	0.61	0.82	0.11	0.37	0.69	0.76	0.11	0.46	0.50	0.83	0.64	0.49	0.68	0.80	0.64
	70	0.51	0.76	0.86	0.14	0.53	0.83	0.84	0.14	0.67	0.68	0.89	0.93	0.65	0.70	0.86	0.93
	100	0.70	0.82	0.89	0.15	0.68	0.82	0.87	0.16	0.70	0.70	0.89	1.22	0.70	0.70	0.88	1.22
zeno-travel	10	0.23	0.93	0.31	9.45E-3	0.15	1.00	0.15	0.01	0.26	0.58	0.61	0.97	0.15	0.99	0.16	0.69
	30	0.22	0.95	0.32	8.44E-3	0.10	1.00	0.12	7.95E-3	0.31	0.43	0.77	2.02	0.25	0.83	0.45	1.32
	50	0.20	0.90	0.32	9.37E-3	0.13	0.99	0.22	7.57E-3	0.37	0.44	0.78	3.02	0.33	0.79	0.61	2.04
	70	0.22	0.92	0.32	9.77E-3	0.16	0.96	0.28	9.36E-3	0.41	0.45	0.82	4.06	0.42	0.79	0.71	2.71
	100	0.32	0.96	0.38	7.36E-3	0.28	0.96	0.37	7.24E-3	0.54	0.54	0.85	5.49	0.58	0.82	0.83	3.61
average	—	0.40	0.82	0.61	0.02	0.29	0.92	0.44	0.02	0.43	0.58	0.78	0.93	0.35	0.85	0.55	0.85

Table C.16: Performance of the GR systems with the PRIM parameters and the cost-optimal traces, the Default parameters and the cost-optimal traces, the PRIM parameters and the divergent traces, and the Default parameters and the divergent traces; the PRIM parameters: the middle points of the parameter ranges identified by the PRIM algorithm, the Default parameters: $\phi = 50$, $\lambda = 1.1$, $\delta = 1.0$, $\theta = 80\%$, %O: the level of observation, p: precision, r: recall, a: accuracy, t: time (in seconds).

Appendix D. Performance Comparison with the Domain Knowledge-Based GR Approaches

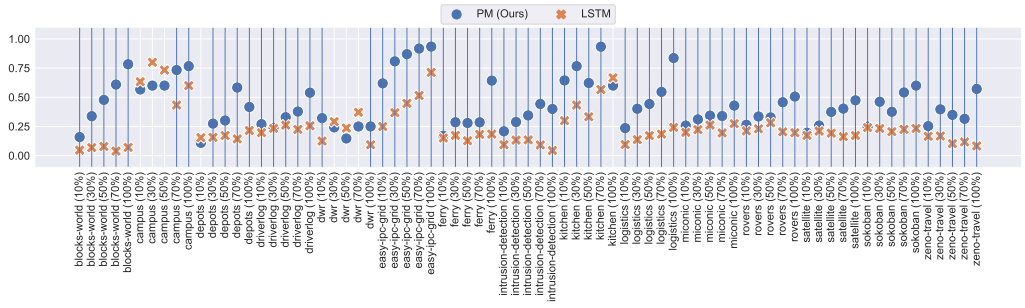
Domain	%O	PM-based (ours)				Landmark-based				R&G (DUAL-BFWS)				R&G (Greedy LAMA)				LP-based			
		p	r	a	t	p	r	a	t	p	r	a	t	p	r	a	t	p	r	a	t
blocks-world	10	0.12	0.50	0.69	0.05	0.19	0.63	0.79	0.40	0.24	0.84	0.71	97.17	0.29	0.94	0.70	773.25	0.27	0.96	0.67	2.51
	30	0.29	0.60	0.85	0.06	0.27	0.74	0.83	0.40	0.43	0.64	0.91	23.39	0.38	0.68	0.90	772.76	0.51	0.88	0.90	2.44
	50	0.39	0.59	0.91	0.07	0.29	0.81	0.84	0.41	0.51	0.65	0.91	19.03	0.48	0.63	0.94	806.92	0.69	0.91	0.95	2.44
	70	0.58	0.70	0.95	0.09	0.42	0.95	0.89	0.41	0.64	0.76	0.93	27.25	0.64	0.73	0.96	819.90	0.86	0.99	0.98	2.48
	100	0.76	0.89	0.97	0.11	0.52	1.00	0.93	0.41	0.66	0.76	0.94	52.18	0.63	0.72	0.96	848.38	0.93	1.00	0.99	2.49
campus	10	0.50	1.00	0.50	0.01	0.50	1.00	0.50	0.31	0.57	0.73	0.57	0.36	0.83	1.00	0.83	0.70	0.87	1.00	0.87	0.23
	30	0.60	0.93	0.60	6.02E-3	0.60	1.00	0.60	0.31	0.67	1.00	0.67	0.19	0.87	0.93	0.87	0.83	0.97	1.00	0.97	0.24
	50	0.63	1.00	0.63	6.19E-3	0.57	1.00	0.57	0.31	0.63	0.93	0.63	0.20	0.93	1.00	0.93	0.82	0.97	1.00	0.97	0.23
	70	0.80	1.00	0.80	5.82E-3	0.67	1.00	0.67	0.31	0.60	0.93	0.60	0.19	0.83	0.87	0.83	0.90	0.97	1.00	0.97	0.24
	100	0.90	1.00	0.90	6.84E-3	0.67	1.00	0.67	0.30	0.60	0.87	0.60	0.21	0.60	0.80	0.60	0.97	0.97	1.00	0.97	0.22
depots	10	0.11	0.57	0.45	0.17	0.22	0.62	0.60	0.92	0.37	0.51	0.78	18.61	0.50	0.68	0.82	326.27	0.40	0.81	0.75	1.70
	30	0.28	0.39	0.75	0.33	0.40	0.92	0.70	0.93	0.57	0.67	0.84	58.49	0.69	0.72	0.92	322.05	0.67	0.81	0.90	1.69
	50	0.27	0.47	0.75	0.52	0.48	0.94	0.76	0.94	0.57	0.83	0.73	130.20	0.83	0.92	0.92	349.64	0.77	0.94	0.91	1.69
	70	0.43	0.61	0.81	0.71	0.58	0.92	0.84	0.95	0.60	0.89	0.68	159.63	0.78	0.78	0.94	346.60	0.97	0.97	0.99	1.69
	100	0.54	0.67	0.90	0.93	0.79	1.00	0.95	0.95	0.65	0.92	0.75	214.83	0.83	0.83	0.95	367.32	1.00	1.00	1.00	1.67
driverlog	10	0.29	0.68	0.58	0.65	0.24	0.80	0.48	0.83	0.39	0.45	0.78	17.10	0.42	0.48	0.82	16.39	0.38	0.77	0.72	1.07
	30	0.35	0.51	0.73	1.98	0.35	0.83	0.59	0.84	0.27	0.75	0.45	80.30	0.41	0.45	0.82	18.03	0.60	0.83	0.86	1.08
	50	0.49	0.61	0.81	3.28	0.51	0.93	0.73	0.84	0.28	0.86	0.38	117.16	0.62	0.71	0.84	23.06	0.82	0.93	0.95	1.08
	70	0.57	0.63	0.84	4.67	0.63	1.00	0.83	0.85	0.30	0.92	0.36	165.37	0.71	0.81	0.87	30.97	0.88	0.95	0.97	1.07
	100	0.79	0.86	0.93	6.49	0.79	1.00	0.94	0.85	0.55	0.93	0.61	252.39	0.73	0.82	0.91	31.65	0.98	1.00	1.00	1.12
dwr	10	0.18	0.54	0.50	0.04	0.23	0.75	0.48	0.59	0.20	0.50	0.58	1.66	0.30	0.71	0.59	271.30	0.28	0.88	0.52	1.01
	30	0.22	0.29	0.74	0.06	0.29	0.92	0.55	0.58	0.27	0.67	0.61	7.65	0.53	0.79	0.77	264.35	0.65	1.00	0.82	1.02
	50	0.23	0.29	0.77	0.09	0.40	0.96	0.68	0.59	0.25	0.62	0.55	19.75	0.68	0.83	0.89	275.76	0.81	1.00	0.94	1.04
	70	0.30	0.38	0.79	0.11	0.52	1.00	0.80	0.61	0.47	0.67	0.72	47.24	0.78	0.88	0.90	288.99	0.91	0.96	0.97	1.00
	100	0.19	0.25	0.76	0.15	0.58	1.00	0.85	0.60	0.29	0.50	0.65	86.00	0.88	0.88	0.96	312.59	1.00	1.00	1.00	1.04
easy-ipc-grid	10	0.48	0.90	0.70	0.08	0.36	0.93	0.46	0.68	0.67	0.90	0.87	7.28	0.64	0.84	0.87	45.03	0.64	0.93	0.87	1.28
	30	0.79	0.95	0.94	0.15	0.58	0.90	0.73	0.69	0.82	0.98	0.91	3.30	0.82	0.96	0.93	96.23	0.82	0.95	0.95	1.29
	50	0.89	0.97	0.97	0.22	0.84	0.95	0.92	0.69	0.91	0.99	0.95	8.05	0.89	0.95	0.96	198.89	0.93	0.99	0.99	1.30
	70	0.89	0.96	0.98	0.27	0.94	0.98	0.98	0.71	0.94	1.00	0.96	6.12	0.83	0.87	0.95	379.05	0.94	0.99	0.99	1.32
	100	0.92	0.97	0.98	0.25	1.00	1.00	1.00	0.68	0.99	1.00	0.99	5.71	0.69	0.74	0.88	614.65	0.98	1.00	1.00	1.20
ferry	10	0.17	0.44	0.59	0.21	0.21	0.93	0.37	0.39	0.52	0.68	0.84	1.58	0.54	0.69	0.84	61.95	0.50	1.00	0.71	0.90
	30	0.19	0.30	0.71	0.49	0.46	0.93	0.67	0.39	0.48	0.76	0.68	10.62	0.79	0.90	0.92	69.77	0.85	1.00	0.93	0.92
	50	0.29	0.39	0.78	0.80	0.62	0.93	0.81	0.40	0.40	0.89	0.48	25.59	0.87	0.95	0.94	86.17	0.92	1.00	0.97	0.91
	70	0.43	0.49	0.84	1.09	0.78	0.93	0.87	0.40	0.29	0.89	0.38	44.89	0.92	0.99	0.95	99.55	0.99	1.00	1.00	0.92
	100	0.86	0.89	0.96	1.34	0.89	0.93	0.92	0.40	0.59	0.93	0.66	68.78	0.90	1.00	0.93	127.22	1.00	1.00	1.00	0.91
intrusion-detection	10	0.19	0.55	0.58	0.02	0.10	1.00	0.21	0.40	0.61	0.98	0.91	10.66	0.59	1.00	0.91	4.91	0.59	1.00	0.91	1.76
	30	0.27	0.49	0.78	0.01	0.32	1.00	0.70	0.40	0.84	0.92	0.98	1.80	0.93	1.00	0.99	4.95	0.94	1.00	0.99	1.77
	50	0.42	0.49	0.85	0.01	0.54	1.00	0.86	0.40	0.85	0.92	0.97	2.40	0.99	1.00	1.00	5.05	0.99	1.00	1.00	1.77
	70	0.47	0.52	0.87	0.01	0.72	1.00	0.91	0.41	0.78	0.85	0.97	3.69	1.00	1.00	1.00	5.32	1.00	1.00	1.00	1.79
	100	0.44	0.44	0.87	0.01	0.91	1.00	0.94	0.41	0.72	0.80	0.96	6.81	1.00	1.00	1.00	5.65	1.00	1.00	1.00	1.81
kitchen	10	0.66	1.00	0.71	0.01	0.33	1.00	0.33	0.27	0.59	0.80	0.67	5.27	0.59	0.80	0.67	1.32	0.66	1.00	0.71	0.32
	30	0.73	0.87	0.82	6.24E-3	0.47	1.00	0.47	0.27	0.80	0.93	0.87	0.92	0.80	0.93	0.87	1.07	0.83	1.00	0.89	0.33
	50	0.57	0.80	0.69	6.25E-3	0.47	1.00	0.47	0.28	0.83	0.92	0.89	0.29	0.74	0.83	0.81	1.00	0.79	0.93	0.84	0.32
	70	0.90	1.00	0.93	4.74E-3	0.56	1.00	0.56	0.28	0.79	0.93	0.82	0.38	0.79	0.87	0.84	1.09	0.82	0.87	0.87	0.32
	100	0.80	0.87	0.87	4.72E-3	0.69	1.00	0.69	0.28	0.77	0.93	0.84	0.71	0.69	0.87	0.78	1.18	0.60	0.60	0.73	0.32
logistics	10	0.25	0.58	0.62	0.15	0.24	0.94	0.44	1.18	0.47	0.61	0.85	26.33	0.55	0.79	0.88	12.41	0.61	1.00	0.85	1.51
	30	0.38	0.47	0.83	0.29	0.54	0.97	0.79	1.19	0.56	0.78	0.76	35.07	0.70	0.89	0.85	14.28	0.86	0.98	0.97	1.50
	50	0.42	0.50	0.87	0.43	0.70	1.00	0.90	1.19	0.56	0.83	0.72	64.76	0.75	0.96	0.81	16.81	0.93	0.99	0.98	1.50
	70	0.56	0.62	0.90	0.57	0.86	1.00	0.96	1.21	0.54	0.82	0.70	121.22	0.75	0.97	0.78	22.28	0.96	1.00	0.99	1.48
	100	0.81	0.84	0.96	0.64	0.96	1.00	0.99	1.09	0.52	0.69	0.81	139.02	0.80	0.98	0.82	30.15	1.00	1.00	1.00	1.44
miconic	10	0.31	0.60	0.65	0.29	0.23	1.00	0.28	0.99	0.28	0.48	0.67	3.68	0.43	0.61	0.77	13.35	0.63	1.00	0.81	1.03
	30	0.26	0.40	0.68	0.64	0.43	1.00	0.60	1.00	0.26	0.65	0.50	4.83	0.45	0.88	0.58	40.92	0.92	1.00	0.97	1.02
	50	0.32	0.38	0.75	1.00	0.54	1.00	0.74	1.00	0.26	0.75	0.44	13.21	0.48	0.87	0.59	53.85	0.96	1.00	0.98	1.02
	70	0.45	0.49	0.81	1.37	0.73	1.00	0.86	1.00	0.19	0.82	0.33	22.20	0.52	0.90	0.60	69.41	0.99	1.00	1.00	1.00
	100	0.65	0.68	0.86	1.84	0.79	1.00	0.90	1.01	0.18	0.93	0.23	39.31	0.63	0.93	0.68	73.11	1.00	1.00	1.00	1.04
rovers	10	0.19	0.46	0.58	0.26	0.27	0.96	0.40	1.01	0.55	0.73	0.82	6.43	0.51	0.80	0.78	4.28	0.53	0.99	0.71	0.99
	30	0.34	0.46	0.73	0.69	0.39	0.96	0.57	1.02	0.65	0.83	0.80	13.31	0.70	0.90	0.84	12.11	0.78	0.86	0.92	0.99
	50	0.43	0.48	0.80	1.09	0.52	0.98	0.72	1.05	0.72	0.93	0.81	63.57	0.76	0.95	0.83	16.98	0.92	0.99	0.97	0.98
	70	0.36	0.42	0.77	1.53	0.71	1.00	0.86	1.05	0.70	0.96	0.78	96.03	0.74	0.96	0.78	39.90	0.98	0.99	0.99	0.98
	100	0.39	0.43	0.77	2.14	0.83	1.00	0.91	1.05	0.79	0.96	0.88	85.87	0.82	0.96	0.84	54.18	1.00	1.00	1.00	1.02
satellite	10	0.20	0.64	0.44	0.12	0.25	0.89	0.40	1.24	0.30	0.55	0.65	6.37	0.35	0.65	0.66	5.83	0.46	0.92	0.69	1.03
	30	0.26	0.54	0.59	0.20	0.39	0.90	0.59	1.24	0.44	0.69	0.72	12.22	0.50	0.70	0.78	9.76	0.68	0.93	0.85	1.04
	50	0.41	0.60	0.75	0.30	0.59	0.92	0.77	1.25	0.54	0.69	0.55	27.00	0.64	0.80	0.84	10.57	0.82	0.96	0.97	1.05
	70	0.55	0.65	0.79	0.40	0.67	0.93	0.83	1.26	0.30	0.83	0.47	45.91	0.62	0.87	0.80	15.48	0.93	0.98	0.97	1.02
	100	0.71	0.75	0.88	0.51	0.74	0.93	0.88	1.25	0.37	0.93	0.47	84.69	0.74	0.93	0.84	13.11	0.96	1.00	0.99	1.02
sokoban	10	0.20	0.70	0.43	0.11	0.28	0.90	0.42	1.30	0.59	0.72	0.81	51.53	0.36	0.42	0.78	215.56	0.58	0.73	0.85	1.93
	30	0.38	0.53	0.72	0.30	0.42	0.83	0.63	1.32	0.73	0.87	0.83	179.13	0.52	0.77	0.68	321.13	0.62	0.63	0.87	1.93
	50	0.46	0.50	0.83	0.59	0.54	0.92	0.73	1.33	0.84	0.93	0.88	309.37	0.71	0.82	0.85	438.12	0.47	0.48	0.81	1.87
	70	0.67	0.68	0.89	0.88	0.75	0.95	0.89	1.33	0.81	0.88	0.90	415.83	0.81	0.90	0.90	510.07	0.45	0.47	0.79	1.89
	100	0.70	0.70	0.89	1.16	0.89	1.00	0.97	1.35	0.91	0.95	0.94	632.05	0.90	0.90	0.97	637.70	0.35	0.35	0.75	1.88
zeno-travel	10	0.26	0.55	0.61	0.78	0.29	0.70	0.62	1.28	0.30	0.45	0.69	6.97	0.45	0.50	0.80	10.39	0.49	0.87	0.71	1.39
	30	0.31	0.42	0.77	1.61	0.39	0.90	0.67	1.30	0.30	0.69	0.53	46.92	0.48	0.74	0.69	12.28	0.71	0.90	0.88	1.37
	50	0.37	0.45	0.78	2.42	0.62	0.95	0.82	1.30	0.26	0.82	0.39	111.15	0.63	0.92	0.69	13.78	0.89	0.95	0.97	1.38
	70	0.41	0.45	0.81	3.29	0.79	1.00	0.90	1.31	0.37	0.92	0.43	186.22	0.63	0.94	0.68	16.34	1.00	1.00	1.00	1.35
	100	0.52	0.54	0.85	4.44	0.95	1.00	0.98	1.31	0.60	1.00	0.60	268.10	0.73	0.96	0.75	21.54	1.00	1.00	1.00	1.36
average	—	0.46	0.62	0.77	0.74	0.54	0.94	0.72	0.80	0.54	0.81	0.71	65.73	0.68	0.84	0.83	157.42	0.79	0.93	0.91	1.24

Table D.17: Performance of different GR approaches; %O: the level of observation, p: precision, r: recall, a: accuracy, t: time (in seconds). The PM-based approach (ours) is configured with the PRIM parameters and trained with the divergent traces or with the cost-optimal traces if the divergent traces are not available. The landmark-based approach uses the *uniqueness* heuristic with $\theta = 20\%$. The two R&G approaches use the DUAL-BFWS planner and the Greedy LAMA planner, respectively. The LP-based approach uses a combination of three heuristics, which are landmarks, state equation, and post-hoc.

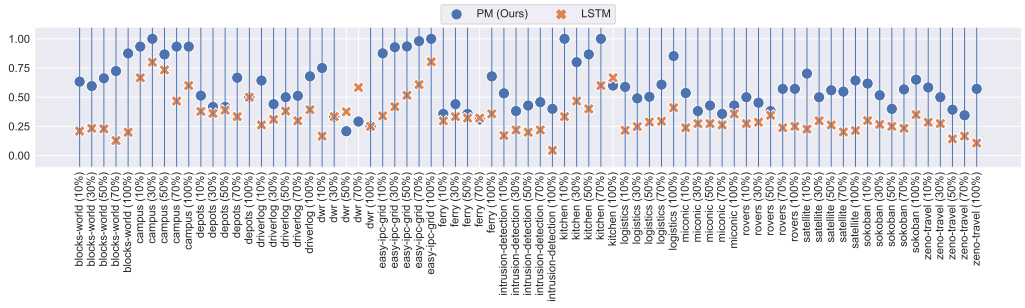
Appendix E. Performance Comparison with the LSTM-Based GR Approach

Domain	%O	PM (10)			LSTM (10)			PM (100)			LSTM (100)		
		p	r	a	p	r	a	p	r	a	p	r	a
blocks-world	10	0.16	0.63	0.72	0.05	0.21	0.74	0.12	0.50	0.69	0.08	0.18	0.81
	30	0.34	0.60	0.88	0.07	0.23	0.79	0.29	0.60	0.85	0.09	0.25	0.83
	50	0.48	0.66	0.93	0.08	0.23	0.82	0.39	0.59	0.91	0.15	0.29	0.85
	70	0.61	0.72	0.95	0.04	0.13	0.79	0.58	0.70	0.95	0.23	0.42	0.88
	100	0.78	0.88	0.98	0.07	0.20	0.80	0.76	0.89	0.97	0.23	0.42	0.88
campus	10	0.57	0.93	0.57	0.63	0.67	0.63	0.50	1.00	0.50	0.57	0.60	0.57
	30	0.60	1.00	0.60	0.80	0.80	0.80	0.60	0.93	0.60	0.67	0.67	0.67
	50	0.60	0.87	0.60	0.73	0.73	0.73	0.63	1.00	0.63	0.67	0.67	0.67
	70	0.73	0.93	0.73	0.43	0.47	0.43	0.80	1.00	0.80	0.73	0.73	0.73
	100	0.77	0.93	0.77	0.60	0.60	0.60	0.90	1.00	0.90	0.57	0.60	0.57
depots	10	0.11	0.51	0.47	0.15	0.38	0.69	0.11	0.57	0.45	0.14	0.14	0.80
	30	0.27	0.42	0.76	0.16	0.36	0.70	0.28	0.39	0.75	0.17	0.17	0.80
	50	0.30	0.42	0.77	0.17	0.39	0.74	0.27	0.47	0.75	0.18	0.19	0.81
	70	0.58	0.67	0.89	0.14	0.33	0.73	0.43	0.61	0.81	0.43	0.44	0.86
	100	0.42	0.50	0.87	0.22	0.50	0.73	0.54	0.67	0.90	0.42	0.42	0.86
driverlog	10	0.27	0.64	0.52	0.20	0.26	0.73	0.29	0.68	0.58	0.27	0.29	0.79
	30	0.24	0.44	0.69	0.23	0.31	0.74	0.35	0.51	0.73	0.23	0.24	0.77
	50	0.33	0.50	0.72	0.26	0.38	0.71	0.49	0.61	0.81	0.27	0.27	0.79
	70	0.38	0.51	0.73	0.22	0.30	0.72	0.57	0.63	0.84	0.40	0.42	0.83
	100	0.54	0.68	0.80	0.26	0.39	0.74	0.79	0.86	0.93	0.43	0.43	0.83
dwr	10	0.32	0.75	0.57	0.12	0.17	0.67	0.18	0.54	0.50	0.21	0.25	0.76
	30	0.24	0.33	0.71	0.29	0.33	0.78	0.22	0.29	0.74	0.25	0.25	0.77
	50	0.15	0.21	0.74	0.24	0.38	0.76	0.23	0.29	0.77	0.29	0.29	0.79
	70	0.25	0.29	0.77	0.37	0.58	0.78	0.30	0.38	0.79	0.46	0.46	0.84
	100	0.25	0.25	0.72	0.09	0.25	0.68	0.19	0.25	0.76	0.50	0.50	0.86
easy-ipc-grid	10	0.62	0.88	0.85	0.25	0.34	0.78	0.48	0.90	0.70	0.29	0.29	0.82
	30	0.81	0.93	0.95	0.37	0.42	0.83	0.79	0.95	0.94	0.51	0.54	0.87
	50	0.87	0.93	0.97	0.45	0.52	0.85	0.89	0.97	0.97	0.72	0.73	0.94
	70	0.92	0.98	0.98	0.52	0.61	0.88	0.89	0.96	0.98	0.77	0.77	0.95
	100	0.93	1.00	0.98	0.71	0.80	0.92	0.92	0.97	0.98	0.82	0.84	0.96
ferry	10	0.17	0.36	0.62	0.15	0.30	0.64	0.17	0.44	0.59	0.31	0.33	0.81
	30	0.29	0.44	0.75	0.17	0.33	0.69	0.19	0.30	0.71	0.23	0.23	0.78
	50	0.28	0.36	0.77	0.13	0.32	0.66	0.29	0.39	0.78	0.31	0.32	0.80
	70	0.29	0.31	0.80	0.18	0.32	0.69	0.43	0.49	0.84	0.51	0.54	0.86
	100	0.64	0.68	0.90	0.18	0.36	0.68	0.86	0.89	0.96	0.41	0.43	0.83
intrusion-detection	10	0.21	0.53	0.55	0.09	0.17	0.74	0.19	0.55	0.58	0.16	0.21	0.82
	30	0.29	0.38	0.80	0.13	0.22	0.75	0.27	0.49	0.78	0.31	0.34	0.86
	50	0.34	0.43	0.85	0.13	0.20	0.80	0.42	0.49	0.85	0.37	0.37	0.87
	70	0.44	0.46	0.87	0.09	0.22	0.71	0.47	0.52	0.87	0.59	0.59	0.92
	100	0.40	0.40	0.87	0.04	0.04	0.80	0.44	0.44	0.87	0.42	0.44	0.88
kitchen	10	0.64	1.00	0.69	0.30	0.33	0.51	0.66	1.00	0.71	0.50	0.53	0.67
	30	0.77	0.80	0.84	0.43	0.47	0.62	0.73	0.87	0.82	0.60	0.60	0.73
	50	0.62	0.87	0.73	0.33	0.40	0.53	0.57	0.80	0.69	0.70	0.73	0.80
	70	0.93	1.00	0.96	0.57	0.60	0.71	0.90	1.00	0.93	0.67	0.67	0.78
	100	0.60	0.60	0.73	0.67	0.67	0.76	0.80	0.87	0.87	0.73	0.73	0.82
logistics	10	0.24	0.59	0.62	0.09	0.22	0.73	0.25	0.58	0.62	0.19	0.20	0.84
	30	0.40	0.49	0.84	0.14	0.25	0.76	0.38	0.47	0.83	0.33	0.35	0.86
	50	0.44	0.50	0.88	0.17	0.29	0.76	0.42	0.50	0.87	0.45	0.48	0.89
	70	0.55	0.61	0.90	0.18	0.29	0.79	0.56	0.62	0.90	0.57	0.59	0.92
	100	0.84	0.85	0.96	0.24	0.41	0.80	0.81	0.84	0.96	0.66	0.69	0.94
miconic	10	0.26	0.54	0.65	0.20	0.24	0.71	0.31	0.60	0.65	0.23	0.24	0.74
	30	0.31	0.38	0.74	0.22	0.27	0.73	0.26	0.40	0.68	0.35	0.36	0.78
	50	0.34	0.43	0.76	0.26	0.27	0.72	0.32	0.38	0.75	0.39	0.39	0.80
	70	0.34	0.36	0.75	0.19	0.26	0.69	0.45	0.49	0.81	0.50	0.51	0.83
	100	0.43	0.43	0.80	0.27	0.36	0.73	0.65	0.68	0.86	0.57	0.57	0.86
rovers	10	0.26	0.50	0.61	0.21	0.27	0.73	0.19	0.46	0.58	0.28	0.30	0.76
	30	0.34	0.45	0.71	0.23	0.29	0.71	0.34	0.46	0.73	0.38	0.38	0.79
	50	0.33	0.38	0.77	0.28	0.35	0.73	0.43	0.48	0.80	0.51	0.51	0.83
	70	0.46	0.57	0.80	0.20	0.24	0.71	0.36	0.42	0.77	0.60	0.60	0.87
	100	0.51	0.57	0.82	0.20	0.25	0.72	0.39	0.43	0.77	0.68	0.68	0.89
satellite	10	0.20	0.70	0.43	0.17	0.23	0.69	0.20	0.64	0.44	0.18	0.19	0.73
	30	0.26	0.50	0.62	0.21	0.30	0.70	0.26	0.54	0.59	0.37	0.37	0.81
	50	0.37	0.56	0.70	0.19	0.26	0.70	0.41	0.60	0.75	0.33	0.35	0.79
	70	0.40	0.55	0.74	0.16	0.20	0.70	0.55	0.65	0.79	0.37	0.38	0.80
	100	0.47	0.64	0.76	0.17	0.21	0.73	0.71	0.75	0.88	0.39	0.39	0.81
sokoban	10	0.25	0.62	0.55	0.24	0.30	0.73	0.20	0.70	0.43	0.25	0.27	0.76
	30	0.46	0.52	0.81	0.23	0.27	0.76	0.38	0.53	0.72	0.23	0.23	0.76
	50	0.38	0.40	0.77	0.21	0.25	0.74	0.46	0.50	0.83	0.37	0.37	0.81
	70	0.54	0.57	0.85	0.23	0.23	0.76	0.67	0.68	0.89	0.39	0.42	0.81
	100	0.60	0.65	0.88	0.23	0.35	0.73	0.70	0.70	0.89	0.60	0.60	0.89
zeno-travel	10	0.25	0.58	0.59	0.17	0.29	0.69	0.26	0.55	0.61	0.36	0.37	0.80
	30	0.40	0.50	0.78	0.17	0.27	0.70	0.31	0.42	0.77	0.35	0.37	0.80
	50	0.35	0.39	0.78	0.10	0.14	0.69	0.37	0.45	0.78	0.40	0.40	0.82
	70	0.32	0.35	0.79	0.12	0.17	0.69	0.41	0.45	0.81	0.49	0.50	0.85
	100	0.57	0.57	0.87	0.08	0.11	0.68	0.52	0.54	0.85	0.75	0.75	0.92
average	—	0.44	0.59	0.77	0.24	0.33	0.72	0.46	0.62	0.77	0.41	0.43	0.82

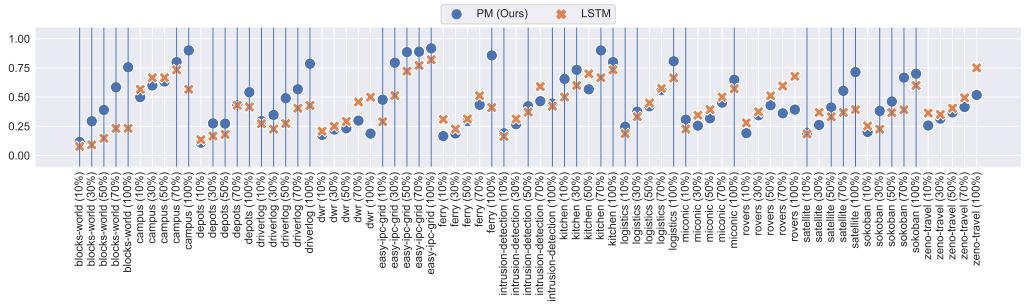
Table E.18: Performance of the PM-based GR approach (ours) and the LSTM-based approach; (10): trained with 10 traces per goal, (100): trained with 100 traces per goal, %O: the level of observation, p: precision, r: recall, a: accuracy. Both approaches are trained with the divergent traces or with the cost-optimal traces if the divergent traces are not available. Our approach is configured with the PRIM parameters.



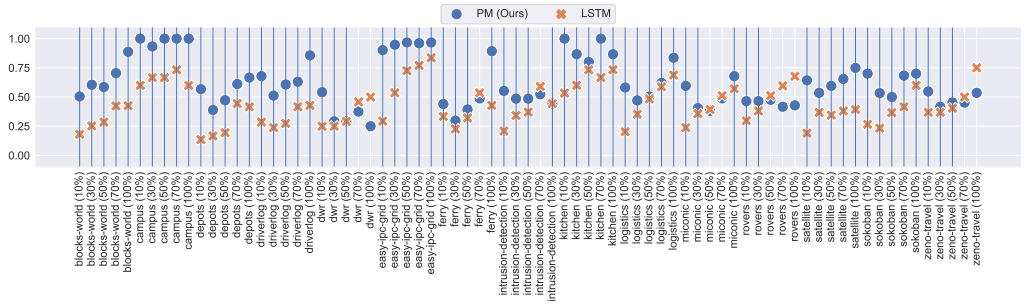
(a) Precision (trained with 10 traces per goal).



(b) Recall (trained with 10 traces per goal).



(c) Precision (trained with 100 traces per goal).



(d) Recall (trained with 100 traces per goal).

Figure E.17: Precision and recall of the PM-based (ours) and the LSTM-based GR approaches. The blue lines indicate cases when the PM-based approach outperforms the LSTM-based approach.