Author/s:
Su, Zihang

Title:
Evidence-Based Goal Recognition Using Process Mining Techniques

Date:
2024-05

Persistent Link:
https://hdl.handle.net/11343/354832

Evidence-Based Goal Recognition Using Process Mining Techniques

Zihang Su
ORCID: 0000-0003-1039-7462

Doctor of Philosophy

May, 2024

School of Computing and Information Systems
The University of Melbourne

Submitted in total fulfilment for the degree of Doctor of Philosophy

Abstract

Goal Recognition (GR) techniques aim to infer the intentions of an autonomous agent according to the observed actions of that agent. We introduce the evidence-based GR framework, regarded as the primary contribution of this thesis, which is designed to address goal recognition problems in both static and dynamic environments. Furthermore, we leverage the proposed framework to develop GR techniques aimed at addressing challenges in the transhumeral prosthesis scenario.

The evidence-based GR technique learns knowledge models using process discovery techniques. It then constructs conformance diagnostics between the learned models and a new observed action sequence executed by an agent. These diagnostics are then used to formulate a probability distribution over a range of possible goals of the agent, where the probabilities indicate the likelihood of the observed action sequence for achieving each possible goal candidate. The evaluation results confirm that the evidence-based GR approach grounded in process discovery and conformance checking techniques studied in process mining achieves comparable recognition accuracy to other state-of-the-art GR approaches and reacts faster. Notably, distinguishing itself from other GR approaches that rely on handcrafted domain knowledge, the evidence-based GR approach can automatically learn knowledge, making it applicable across diverse real-world scenarios. Additionally, the learned knowledge models are explainable, compared to models used in deep learning-based GR approaches.

This thesis emphasizes the challenge of GR in non-stationary environments, where the GR system is required to continuously solve GR tasks over time, during which the underlying environment may change. An adaptive GR framework is proposed as an extension of the evidence-based GR framework for static environments. This adaptive framework can detect changes in the behaviors of the observed agents and adapt the learned knowledge models accordingly. An evaluation of three specific solutions to the adaptive GR problem that are implemented as different control strategies of a GR system is presented and discussed. The evaluation is based on a collection of adaptive GR problem instances generated by the tool we designed and implemented, Goal Recognition Amidst Changing Environments (GRACE). The results demonstrate a trade-off between the GR performance over time and the effort

invested in adaptations of the learned knowledge models, showing that few well-planned adaptations can lead to a consistently high GR performance.

To verify the usefulness of the proposed GR techniques in real-world scenarios, their applicability in a powered transhumeral prosthetic scenario is assessed, where it is required to detect a person's intended movements based on electromyography and kinematic signals collected from their body. A powered transhumeral prosthesis aims to restore missing anatomical segments below the shoulder, including the hand, and is designed to assist patients with disabilities. It analyzes continuous, real-valued data from sensors to recognize patient target poses, or goals, and proactively move the artificial limb. Our GR techniques were evaluated using offline datasets and online human-in-the-loop experiments, comparing the results with state-of-the-art techniques such as linear discriminant analysis (LDA)-based and neural network-based approaches. The results demonstrate that the proposed GR techniques grounded in process mining achieve superior performance.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original, and have not been submitted in whole or in part for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 100,000 words including appendices, bibliography, footnotes, tables, equations, and figures.

Zihang Su
May, 2024

Preface

This thesis is submitted in total fullment of the requirements for the degree of Doctor of Philosophy at the University of Melbourne. The research presented here was primarily conducted at the School of Computing and Information Systems, The University of Melbourne, under the supervision of Associate Professor Artem Polyvyanyy and Associate Professor Nir Lipovetzky. External supervision was provided by Professor Sebastian Sardiña from RMIT University and Dr. Nick van Beest from Data61, CSIRO. Part of this research was conducted in collaboration with Ph.D. student Tianshi Yu from the Department of Mechanical Engineering, The University of Melbourne, who was supervised by Professor Denny Oetomo, Dr. Alireza Mohammadi, and Professor Ying Tan. Below is the list of publications and manuscripts arising from this thesis. I was the principal author of all papers and contributed more than 50% on each paper. My responsibilities included designing the algorithms and architectures, collecting datasets, implementing the systems, conducting experiments, and analyzing the experimental results. My co-authors contributed by providing feedback on the proposed algorithms and models, assisting with manuscript revisions, aiding in data collection from human participants, and helping set up the experimental devices. Ethics approval for the studies included in this thesis was granted by The University of Melbourne's human ethics committee (Chapter 5; IDs 11878 and 26442).

Part of the content from Chapter 3 has been published in the following paper:

- Zihang Su, Artem Polyvyanyy, Nir Lipovetzky, Sebastian Sardiña, and Nick van Beest. Fast and accurate data-driven goal recognition using process mining techniques. *Artificial Intelligence*, 323:103973, 2023. ISSN 0004-3702.

Part of the content from Chapter 4 has been published in the following papers:

- Zihang Su, Artem Polyvyanyy, Nir Lipovetzky, Sebastian Sardiña, and Nick van Beest. GRACE: A simulator for continuous goal recognition over changing environments. In *Proceedings of the Workshop on Process Management in the AI Era (PMAI@IJCAI)*, volume 3310 of CEUR Workshop Proceedings, pages 37 – 48, 2022

- Zihang Su, Artem Polyvyanyy, Nir Lipovetzky, Sebastian Sardiña, and Nick van Beest. Adaptive goal recognition using process mining techniques. *Engineering Applications of Artificial Intelligence*, 133:108189, 2024. ISSN 0952-1976.

Part of the content from Chapter 5 has been published or submitted for publication in the following papers:

- Zihang Su, Tianshi Yu, Nir Lipovetzky, Alireza Mohammadi, Denny Oetomo, Artem Polyvyanyy, Sebastian Sardiña, Ying Tan, and Nick van Beest. Data-driven goal recognition in transhumeral prostheses using process mining techniques. In *5th IEEE International Conference on Process Mining (ICPM)*, pages 25 – 32, 2023.
- Zihang Su, Tianshi Yu, Artem Polyvyanyy, Ying Tan, Nir Lipovetzky, Sebastian Sardiña, Nick van Beest, Alireza Mohammadi, and Denny Oetomo. Process mining over sensor data: Goal recognition for powered transhumeral prostheses. Submitted to *Information Systems, Elsevier*, 2024.

The publication listed below was completed prior to enrollment in the degree and serves as foundational work for the content in Chapter 3 of this thesis:

- Artem Polyvyanyy, Zihang Su, Nir Lipovetzky, and Sebastian Sardiña. Goal recognition using off-the-shelf process mining techniques. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 1072 – 1080, 2020.

I am grateful to acknowledge the following sources of funding:

- China Scholarship Council – University of Melbourne PhD Scholarship
- CSIRO Data61 Top-Up Scholarship

Acknowledgements

The sincere gratitude to my esteemed supervisors, Associate Professors Artem Polyvyanyy and Nir Lipovetzky, alongside my external supervisors, Professor Sebastian Sardiña and Dr. Nick van Beest. I must acknowledge that completing my PhD journey has been an immense challenge, both mentally and physically. However, it is thanks to the continuous support, patient guidance, and encouragement from my supervisors that I have been able to present my academic achievements in this thesis. I extend my heartfelt appreciation to Tianshi Yu and Professor Ying Tan, whose tremendous assistance enabled me to include the case study in this thesis. Additionally, I am grateful to Professor Denny Oetomo and Dr. Alireza Mohammadi for their support of the case study. I am profoundly grateful for the support, collaborative discussions, and the profound inspiration derived from the brilliance of my colleagues in the Process Science and Technology group, including Dr. Abel Armas Cervantes, Dr. Zahra Dasht Bozorgi, Hanan Alkhammash, Andrei Tour, Wenjun Zhou, Thakshila Dilrukshi, Qingtan Shen, Anandi Karunaratne, and Tian Li. A special expression of gratitude goes to Dr. Anubhav Singh from the AI and Autonomy Lab, whose invaluable assistance at the outset of my PhD, particularly during the challenging times of the COVID-19 pandemic, guided me onto the right path in my academic journey. I appreciate the support and companionship provided by the other cohort students in Melbourne Connect Level 4, including Dilang, Stella, Lillian, Tom, Poppy, Jia, Mingxin, and Mike, during the challenging final stages of my PhD journey. Their presence made the difficult days more manageable, especially in the aftermath of the COVID-19 pandemic. I would like to express my heartfelt gratitude to my dear housemate, Donghan, who has provided me with invaluable assistance in my daily life. Additionally, I extend my thanks to my dear friend Yuxuan, with whom I experienced many joyful moments during the dark days of the COVID-19 pandemic. Lastly, I am deeply grateful to my parents for their unwavering encouragement and support in every aspect of my life. Their presence has been a constant source of strength and inspiration throughout my academic journey.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Goal Recognition (GR), as a key aspect of the Theory of Mind [30, 15], has been extensively explored in the field of Artificial Intelligence [108, 117]. Its objective is to mimic, to some extent, the human ability to comprehend and interpret the intentions of others. The first formalized concept of GR is often attributed to Kautz and Allen [63], stating that GR techniques aim to deduce the intentions of autonomous agents by observing and analyzing their behaviors. Understanding GR is centered on three core concepts: A *plan* is composed of a series of actions that have been or should be executed to achieve a certain goal; An *agent*, such as a robot or a human, follows such plans to fulfill goals; A *GR system* refers to software implementing a GR technique capable of deducing agents' goals based on partial knowledge about the observed plans executed by these agents. When a GR system analyzes the actions executed by an agent, it aims to recognize the plan being pursued by the agent. Consequently, the system can also recognize the goal that will be achieved upon the completion of that plan.

In the contemporary era, tasks traditionally designated for humans are now carried out by robots or software, encompassing domains such as smart houses and autonomously driving cars. The core of achieving effective human-machine interaction and performing intelligent behavior in agents relies on understanding other agents' objectives. For instance, in the context of an autonomous driving system, the capability to discern the intentions or anticipated actions of other vehicles is crucial for ensuring safety. Similarly, a smart house needs to interpret whether the residents are engaged in cooking, watching a movie, or sleeping to provide relevant assistance. GR techniques play a pivotal role in these scenarios and various other domains, including support for adversarial reasoning [116, 66], trajectory/maneuver prediction [74, 36, 64], and human-computer collaboration [77].

1.1 Background

The majority of existing GR algorithms are designed to solve a single GR task at a specific time point, focusing on accurately inferring the most likely goal an observed agent is pursuing within a *static* environment. These approaches, which dominate the current research in the GR field, do not account for the continuous resolution of multiple GR tasks over time. Therefore, we refer to these methods as conventional GR and classify them as "single-shot" GR algorithms, as they address only one GR task without considering multiple GR problems that need to be solved over time in a dynamic environment. Expanding the scope of conventional GR involves enabling it to address multiple "single-shot" recognition challenges over an *extended* period, *during which the environment may change*. Importantly, as the environment changes, the GR system must adapt to these shifts to maintain accurate goal recognition performance. This problem setup is referred to as adaptive goal recognition (adaptive GR).

1.1.1 Conventional Goal Recognition

The objective of a "single-shot" GR task is to infer likely goal(s) from a set of goal candidates based on the currently observed action sequence executed by the agent. The existing conventional GR techniques can be broadly categorized into the following three groups: (I) The observed actions of an agent are "matched" to a plan (the one judged to be executed by the agent) in a pre-defined plan library that encodes the standard operational procedures of the domain [19, 63, 28]. This category is commonly known as plan library-based GR approaches. (II) By invoking the principle of rational behavior, an agent is assumed to adopt the "optimal" plan leading to the goal: the higher the perceived rationality of the behavior in achieving a goal, the more likely it is that the goal is the agent's objective. Ramirez and Geffner [101, 102] have initiated numerous approaches that do not require a pre-defined set of plans. Instead, these approaches perform GR by leveraging planning systems to generate plans automatically based on a domain theory. This category is commonly referred to as planning-based GR approaches. (III) Learning-based goal recognition approaches constitute a category of techniques capable of learning domain models or prediction models from data. These learned models can take various forms, such as Q-value functions, inferring probabilities towards goal candidates for each legal action at each state [6, 7]. Alternatively, models can be Long Short-Term Memory (LSTM) neural networks operating as classifiers. These models receive sequences of actions as input and subsequently identify the possible goals [88].

The challenge with plan library-based approaches lies in the necessity of obtaining or hand-coding a set of standard plans for achieving candidate goals. Furthermore, these approaches cannot accommodate uncertainty, as they often struggle to generalize to observations that are not pre-stored in the plan library. In planning-based approaches, even though the capability to recognize an unseen plan is a notable strength, and the process of specifying domain models can be less demanding than manually coding plans, the acquisition of domain models remains a non-trivial task. This difficulty arises from challenges associated with defining models using standard declarative languages, as highlighted in the work by Haslum et al. [51]. Furthermore, acquiring domain models for real-world environments, especially those prone to continuous changes, poses a significant challenge. Therefore, the application of planning-based approaches becomes notably challenging in real-world scenarios. One of the primary challenges linked to learning-based approaches is delineating the scope and acquiring a sufficient volume of data crucial for training effective knowledge models for goal recognition. Finally, these learned models are essentially deep neural networks, a characteristic that is often criticized for its lack of explainability.

1.1.2 Adaptive Goal Recognition

As mentioned above, the conventional GR focuses on solving a "single-shot" GR problem. However, in the real world, there is a need to address multiple GR problems over an extended period, during which the environment may change (also referred to as a non-stationary environment). Notably, when such environmental changes occur, the behavior of agents aiming to achieve their goals generally adapts as well [112]. For instance, in a navigational scenario, the emergence of new roadblocks or the creation of new shortcuts may prompt intelligent agents to take different routes to reach their destination locations. In turn, a smart home system performs goal recognition on the households to support their daily activities, and their behaviors may vary depending on the season. While people use heaters in winter to get comfortable temperatures, they tend to use cooling systems in summer instead (for the same goal). The behavior of the households is also expected to change upon changes in the house configuration (e.g., changing the location of the TV to another room). This phenomenon is also observed in business processes when the behavior of process participants changes to address new regulations, compliance rules, and innovative ways of doing business [140]. In all those cases, we would first like a GR system to recognize the changes in (observed) behavior and then adapt accordingly to maintain its recognition performance. We refer to the problem of solving multiple GR problems over a time interval during which the environment in which the agent operates may change as the *adaptive GR problem*.

Only a limited number of works are, to some extent, relevant to adaptive GR. Bryce et al. [17] proposed a technique for updating knowledge models through query-issued updates. This approach involves analyzing query answers to adapt models; however, instead of updating the models automatically, this technique relies on human-provided query answers. Chakraborti et al. [22] introduced a model reconciliation algorithm in which the reconciliation process modifies one of two models to achieve a single optimal plan for the same goal when different models result in distinct optimal plans.

1.2 Research Questions and Contributions

The main contribution of this thesis is the development of a GR approach that integrates the strengths of planning-based, plan library-based, and learning-based GR methodologies. Specifically, the proposed approach inherits the explainability of planning-based and plan library-based methods, as the models in the proposed GR approach are interpretable. However, conventional approaches require significant manual effort to design and craft these models. In contrast, the proposed GR approach also incorporates the advantages of learning-based methods by enabling automatic model learning, thus eliminating the need for manual model crafting. Unlike traditional learning-based approaches, which often rely on deep neural networks that lack explainability and require large amounts of training data, the proposed approach addresses these drawbacks by producing explainable learned models and reducing the dependence on extensive datasets. We refer to this approach as *Evidence-Based Goal Recognition* because the learned model is constructed from historical *evidence*, and the approach is capable of tracking interpretable *evidence* to infer potential goals. Experimental results illustrate that this approach is applicable in real-world scenarios, can perform well with limited training data, and can be extended to an adaptive GR system for handling environmental changes. The three research questions addressed in this thesis are presented as follows:

Research Question 1. How to do goal recognition in the absence of domain knowledge?

To answer Research Question 1 (RQ1), we propose a framework that leverages *process mining* techniques [123] to automatically acquire *process models* from *event logs* and analyze discrepancies between *process models* and *event logs*. These event logs keep records of historical observations of agents, encoding the *skills* required for accomplishing a range of goals within the environment. Subsequently, the framework infers the possible goal(s) by analyzing deviations in the observed agent's behavior from the skill models learned from the

event logs. Hence, this framework addresses the challenge of requiring pre-defined models and it avoids relying on non-explainable techniques such as reinforcement learning or deep neural networks. In this thesis, a process, skill, or behavior model is represented as a Petri net [104], whose executions encode action sequences (plans) executed by an agent to achieve a specific goal. The event logs do *not* function as plan libraries but as historical evidence. The learned skill models serve as instances of executions, revealing the underlying "hidden" standard operational procedures of the domain. Note that in this thesis, we will use similar terms such as GR framework, GR technique, and GR approach, each with slightly different meanings. The GR framework refers to the foundational structure of the system design, which can be used to implement or customize a concrete GR system. The GR approach refers to the specific method used to implement such a system. Meanwhile, a GR technique is a broader term that includes both specific GR approaches and higher-level frameworks that can be employed to develop concrete systems.

In contrast to learning-based approaches, the proposed GR approach acquires models only from sequences of actions. This ability distinguishes it from other GR techniques that require additional information, such as the environment states before and after the executed actions [6, 7] or the transition functions for every observed action [111]. The agent is assumed to operate within an unknown environment, which can be described, for instance, through a PDDL [51] or STRIPS [34] model (these models specify the dynamics and states of the environment). The learned process models, in the format of Petri nets, describe a subset of goal-relevant action sequences, commonly known as plans, within the environment. Note that Petri nets are not used to represent the underlying dynamic domain; therefore, the states in Petri nets cannot be considered as the states in the domain. We argue that our GR approach falls between planning-based approaches (which base reasoning on cost differences between optimal and observed plans) and plan-library-based approaches (which rely on exemplary plans for reasoning).

The evaluation results illustrate that our GR approach exhibits fast goal recognition, and its accuracy is comparable to the state-of-the-art GR techniques. The fast speed of inference provides advantages for time-sensitive applications. For example, to provide a seamless user experience, a smart house system may be required to anticipate household goals efficiently [138].

Specifically, this thesis makes these contributions to address RQ1:

- It proposes a GR framework that describes the fundamental mechanisms for performing GR without pre-defined models;

- It discusses an implementation of a concrete GR approach based on process mining techniques that follows the proposed GR framework and relies on four parameters to construct a probability distribution over possible goals and to infer the most likely goal. The four 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, a discount factor (δ) that emphasizes the recently observed actions to have more impact on the goal inferences, and a decision threshold (θ) that determines which goals should be inferred as likely goals;
- It presents results of a sensitivity analysis over 15 IPC domains[1] 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 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.

Research Question 2. How to do goal recognition in non-stationary environments?

In Research Question 2 (RQ2), we propose a process mining-based adaptive GR framework that can automatically learn and relearn knowledge models based on data generated while observing the behavior of the agent of interest. According to the framework, we instantiate three concrete adaptive GR systems to address the adaptive GR problem. Specifically, the "single-shot" GR system is controlled based on its performance to, for instance, request to relearn its knowledge base if the recognition accuracy drops. The three proposed concrete solutions are designed as open- and closed-loop control strategies applied to standard "single-shot" GR systems.

The research on the problem of GR is impeded by the lack of benchmarks for evaluating candidate solutions. Thus, we present a data-generating tool called the Goal Recognition Amidst Changing Environments (GRACE), which can modify the environments where agents

[1]The International Planning Competition (IPC) domains used in this thesis are the same as the domains in the paper by Pereira et al. [95].

work towards various goals and generate different behaviors under different environments. For the seed environments in which observed agents operate, we use the static IPC domains, which are widely used to evaluate conventional GR approaches. These environments are provided as input to GRACE, which subsequently generates problem instances in which the seed environments change over time according to the parameters supplied to the tool. The GRACE tool uses two aspects to characterize a change in an environment. Firstly, a change is characterized by the components of the environment that change. For example, such components are the initial and goal states of the agents and actions the agents can perform in the environment. Secondly, by interpreting an environment as a *signal* and, consequently, a change in the original environment as a change in the original signal, we characterize changes in the environment based on the different types of concept drift [140]. For example, a change in an environment can be sudden or such that it progresses gradually over time. Note that the GRACE tool incorporates a measure of the significance of environmental change, ensuring that the generated problem instances are defined over an environment that undergoes significant changes.

We use GRACE to generate synthetic adaptive GR problem instances and formulate real-world problem instances using BPI challenge event logs.[2] We conduct experiments using both synthetic and real-world problem instances to evaluate how well our GR systems address adaptive GR problems. The results show that, compared to a conventional GR system, adaptive GR systems recognize goals more accurately over time.

To achieve high recognition accuracy in solving the adaptive GR problem, the GR system needs to detect environmental changes or changes in the agent's behavior and determine the optimal timing for updating its knowledge models based on new evidence. We conduct experiments using synthetic and real-world datasets to illustrate the intrinsic trade-offs between the effort of updating the knowledge base and the recognition accuracy of adaptive GR systems.

This thesis makes the following contributions to address RQ2:

- It formally defines the adaptive GR problem, which can be seen as an optimization problem that balances the cost of adapting knowledge models with the accuracy of GR;

- It introduces a tool called GRACE, capable of simulating significant changes in the environment in which agents strive to achieve goals and can generate problem instances for evaluating adaptive GR systems;

[2] https://www.tf-pm.org/resources/logs

- It introduces an adaptive GR framework as a control mechanism over a conventional "single-shot" GR system and presents three concrete adaptive GR systems instantiated from this framework;
- It uses synthetic and real-world datasets to evaluate three instantiated adaptive GR systems. The evaluation results confirm the effectiveness of these GR systems for solving adaptive GR problems.

Note that, to address RQ1, we propose a GR framework that employs process mining techniques to learn explainable models, and we refer to it as the Process Mining (PM-)based GR framework. For RQ2, derived from the PM-based GR approach, we introduce the adaptive GR framework that can adapt to environmental changes, namely, the adaptive GR framework. However, as both the PM-based GR and the adaptive GR can learn explainable models from historical evidence and track the evidence behind the system's inference of a specific goal, collectively, they fall within the scope of *Evidence-Based Goal Recognition*.

Research Question 3. Are evidence-based goal recognition techniques practically useful?

To answer Research Question 3 (RQ3), we conduct a case study to validate the efficacy of the evidence-based GR paradigm in addressing real-world challenges. Specifically, we explore the feasibility and potential of applying PM-based GR techniques to a *powered transhumeral prosthesis* scenario. A transhumeral prosthesis is designed to replace the missing anatomical parts of an arm below the patient's shoulder. A powered prosthesis is equipped with sensors and motors designed to recognize the user's intentions and provide assistance. The PM-based GR technique shows potential for recognizing the intended movements (goals) of subjects with disabilities by analyzing continuous, real-valued sensor measurements, such as surface electromyography (sEMG) and kinematic sensors. Therefore, this study contributes to the development of a powered transhumeral prosthesis, guided by PM-based GR techniques, to enhance the efficiency of prosthetic use for individuals with upper limb disabilities [75, 142, 141]. Note that the aim of this study is specifically focused on recognizing intended movements and identifying the target prosthetic pose that is desired to be reached. The aspect where actuators drive the prosthesis to complete movements relies on mechanical engineering and robotic techniques, which fall outside the scope of this thesis. Note that this case study focuses on the process mining-based GR framework for stationary environments, without considering the adaptive variant of the framework.

We implement two target pose recognition approaches using PM-based GR techniques and verify through experiments that these PM-based approaches can autonomously guide the artificial limb towards the targeted prosthetic pose, effectively achieving the intended goal.[3]

As PM-based GR techniques are designed to recognize goals from sequences of discrete events, the challenge arises since the sensor signals are continuous, real-valued measurements. Thus, PM-based target pose recognition approaches first require the transformation of continuous, real-valued sensor measurements into discrete events and then follow steps similar to the PM-based GR framework demonstrated in RQ1. The two data transformation methods we propose involve clustering algorithms and a classifier. We compare the PM-based target pose recognition approaches to three state-of-the-art baselines. One of the baseline approaches uses a linear discriminant analysis (LDA) classifier to identify target poses [142]. However, this classifier is trained with signals collected when the arm reaches the target pose and is held statically at that particular position. We refer to this baseline approach as *static LDA*. Additionally, we refined the static LDA baseline [141] by training it with signals collected during arm movements. This enhanced baseline is referred to as *dynamic LDA*. Another baseline for comparison is the LSTM neural network-based target pose recognition approach [57], referred to as *LSTM*.

The approaches including baselines (static LDA, dynamic LDA, and LSTM) were evaluated through two experimental settings: the offline experiment and the online human-in-the-loop (HITL) experiment. In the offline experiment, we used an existing dataset collected for the development of powered transhumeral prostheses. The dataset comprises data from ten non-disabled subjects in a virtual reality (VR) environment designed to emulate the behavior of patients using transhumeral prostheses [142]. Each subject was instructed to perform forward-reaching tasks involving three distinct elbow poses. They were required to extend their sound upper limb forward in a series of 30 iterations for each goal. The reaching targets were positioned along the parasagittal plane. A small sphere in the VR environment communicated a reaching target to the subject, and the subject had to reach it with their hand. The subjects were requested to stay still for approximately one second after reaching the goal. During each iteration towards a goal, 47 features were extracted at regular intervals from the measurements of kinematic and surface electromyography (sEMG) sensors attached to the subject. These sequences of features constitute time series of continuous, real-valued data that characterize the behavior of the subject in achieving the goal. We invited another six non-disabled subjects to participate in a human-in-the-loop (HITL) experiment, instructing them to perform the Refined Clothespin Relocation Task (RCRT), as documented in [58, 75]. This task involves eight distinct moving trajectories, with 59 features recorded for each

[3] In this study, we use the terms "goal" and "target pose" interchangeably.

trajectory. The HITL experiment is divided into two phases. During the first phase, we instructed the subjects to conduct the RCRT tasks for 10 iterations and collected sequences of induced signals. These signals were then used to train the GR system. In the second phase, the VR environment was used to simulate scenarios for patients with disabilities by deactivating two trackers (positioned at the upper arm and forearm). The participants were then supported by the trained GR system, which allowed them to interact with the prosthesis, when completing the RCRT tasks. The users were asked to perform the same RCRT tasks for five iterations. Subsequently, we assessed the performance based on how effectively the subjects could use the prosthesis device to execute the tasks. Thus, the HITL experiments serve as a robust measure for testing the effectiveness of the developed GR system. In the HITL experiments, we evaluated the two most effective GR approaches identified in the offline experiments.

The offline experiments reveal that two techniques stand out as the most effective: dynamic LDA and PM-based GR with a classifier. Specifically, the PM-based GR with classifier achieves the highest balanced accuracy, while dynamic LDA achieves the highest F_1 score. Human-in-the-loop (HITL) experiments assess the practicality of the two top offline techniques in real-time target identification. In these online HITL experiments, our PM-based GR with classifier achieves significantly higher F_1 score and balanced accuracy than the dynamic LDA approach. The results confirm that the target pose recognition approach, using process mining techniques (including discovered models and alignments between new observations and the models), performs effectively in the field of transhumeral prostheses. This finding can benefit further developments in powered transhumeral prostheses.

Concretely, this study makes the following contributions:

- It applies the PM-based GR framework to develop target pose recognition techniques for powered transhumeral prostheses, aiming to assist the daily activities of people with disabilities. The proposed techniques adapt the PM-based GR approach to operate with multi-dimensional, real-valued, continuous measurements that characterize the observed behavior of interest;

- It evaluates the proposed PM-based target pose recognition techniques by comparing them with state-of-the-art baselines, including static LDA, dynamic LDA, and LSTM, through an offline experiment;

- It conducts a human-in-the-Loop (HITL) experiment to assess the two best-performing approaches identified in the offline setting, the PM-based GR with a classifier and the dynamic LDA baseline, verifying that the PM-based approach outperforms the baseline,

thereby supporting the claim that the evidence-based GR framework is practically useful in real-world scenarios.

1.3 Thesis Outline

The remaining chapters of this thesis are arranged as follows. Chapter 2 provides a comprehensive literature review where we present key publications in the field of goal recognition. We discuss relevant techniques in the process mining field, including process discovery and conformance checking. These techniques are applied to implement evidence-based goal recognition systems. We review works related to adaptive goal recognition, including techniques such as model reconciliation, process model repair, and concept drift detection. Given that this thesis includes a case study involving the application of the evidence-based GR framework to guide the movement of a powered transhumeral prosthesis, we review state-of-the-art works in this robotic prostheses field. Finally, we introduce all the relevant tools used for implementing the GR systems and evaluating their performance. Chapter 3 addresses **RQ1** by presenting the evidence-based GR framework and illustrating it with examples of how process mining techniques are used to implement a concrete GR system based on the framework. We evaluate the GR performance of the implemented GR system grounded in process mining techniques and compare it with other state-of-the-art GR approaches. To address **RQ2**, Chapter 4 introduces the concept of the adaptive GR problem and defines it as an extension of the conventional single-shot GR problem. We propose the adaptive GR framework, an extension of the evidence-based GR framework, by adding two additional steps for collecting feedback and updating the model. We evaluate the adaptive GR systems instantiated from the framework using both synthetic and real-world problem instances. The evaluation results confirm that GR systems with adaptive mechanisms are effective in solving adaptive GR problems. Chapter 5 presents the transhumeral prosthetic study, which corresponds to addressing **RQ3**. It describes the experimental settings of the project, which aims to develop a powered transhumeral prosthesis to assist people with disabilities. We demonstrate how to apply our evidence-based GR technique to guide the movement of the prosthesis with detailed running examples. We conduct both offline and human-in-the-loop experiments to validate the effectiveness of our proposed approach compared to the state-of-the-art baselines. The results confirm that our proposed GR technique can provide better performance than existing baselines. Chapter 6 provides a summary of the thesis, including some final discussions on limitations and future work directions.

Chapter 2

Literature Review

In this chapter, we review the relevant literature across five key aspects. First, we discuss the key publications in the field of goal recognition (GR), evaluating their strengths and limitations. Second, since we propose an evidence-based GR framework that uses process mining techniques, we outline the relevant methods from the process mining field that are used or potentially applicable to our framework. Third, we expand the discussion of the literature that studies the conventional GR problem defined in a static environment to the approaches that tackle the adaptive GR problem, which focuses on enhancing the robustness of GR performance over time and accounts for changes in the underlying environment. Fourth, as this thesis includes a case study that demonstrates the applicability of the evidence-based GR approach to a powered transhumeral prosthesis, we review the existing state-of-the-art approaches in this area. Lastly, we introduce the various tools used in this research.

2.1 Goal Recognition

Goal recognition is a sub-area of research in the field of artificial intelligence (AI), aiming to identify the goals of an autonomous agent by observing and analyzing the sequence of actions executed by that agent. The term autonomous agent (or agent), as used in this thesis, refers to entities that can think and act intelligently to achieve specific goals. Examples of such agents can be humans, robots, or even software. In existing literature, the concept of plan recognition (PR) also appears frequently and shares a similar meaning with GR. A key difference between PR and GR lies in their objectives: while both disciplines aim to identify an agent's final goal, PR additionally attempts to forecast the specific actions that the agent will execute to achieve that goal [89]. The evidence-based GR techniques proposed in this thesis are primarily designed to identify the final goal, but they can also be used to infer the

actions needed to achieve the intended goal, as discussed in Chapter 3. In this thesis, we treat the terms PR and GR as equivalent.

The term AI was arguably first formally proposed during the 1956 Dartmouth Conference, which is commonly regarded as the starting point of the AI research field. Afterwards, key early research works related to plan and goal recognition began to emerge, including: the introduction of the General Problem-Solver by Newell et al. [91] in 1959, which aimed to solve a wide range of problems, including plan and goal recognition; the STRIPS model proposed by Fikes and Nilsson [34] in 1971, which became one of the foundational works in the planning and plan recognition research area; and, in 1978, Schmidt et al. [108] introduced the psychologically-based BELIEVER system, which initiated further research aimed at formalizing the plan and goal recognition problem. During those early years, researchers were attempting to identify, define, and formalize the GR problem, until 1986, when the work by Kautz and Allen [63] provided the first formal definition of the plan recognition problem in the field. This work involves encoding a set of plans into a hierarchical action graph, often known as a plan library, to represent knowledge. Given that many subsequent studies adopt a similar method for encoding such plan libraries, we refer to this class of GR approaches as the plan library-based approach, which will be discussed in the next section.

2.1.1 Plan Library-Based Goal Recognition

Kautz and Allen [63] introduced a plan recognition theory that uses circumscription to handle uncertainties and disjunctive information, enabling the interpretation of complex actions from observed behaviors. Their plan recognition approach uses an action taxonomy to logically deduce and predict future actions, making it possible to recognize plans from simultaneous and shared actions. They encode the standard plan library into a hierarchical action graph, where high-level actions can be composed into lower-level actions. For example, in Figure 2.1, the action “Prepare Meal” is decomposed into actions such as “Make Pasta Dish” and “Make Meat Dish.” By avoiding premature commitment to a single plan and allowing for multiple interpretations, this work significantly enhances the flexibility and applicability of plan recognition in dynamic, real-world environments. Additionally, it is arguably the first to formalize plan recognition as a logic-based (non-monotonic) deductive reasoning task, mapped over an encoded action taxonomy.

Vilain [134] explored the computational similarities between plan recognition and text parsing, highlighting their shared formal structures. This work mapped Kautz’s plan recognition formalism onto established grammatical frameworks, such as context-free grammars, to enhance the efficiency of plan parsing using well-known parsing techniques. Vilain assumed that with tailored constraints on plan hierarchies and specific observation types, plan recog-

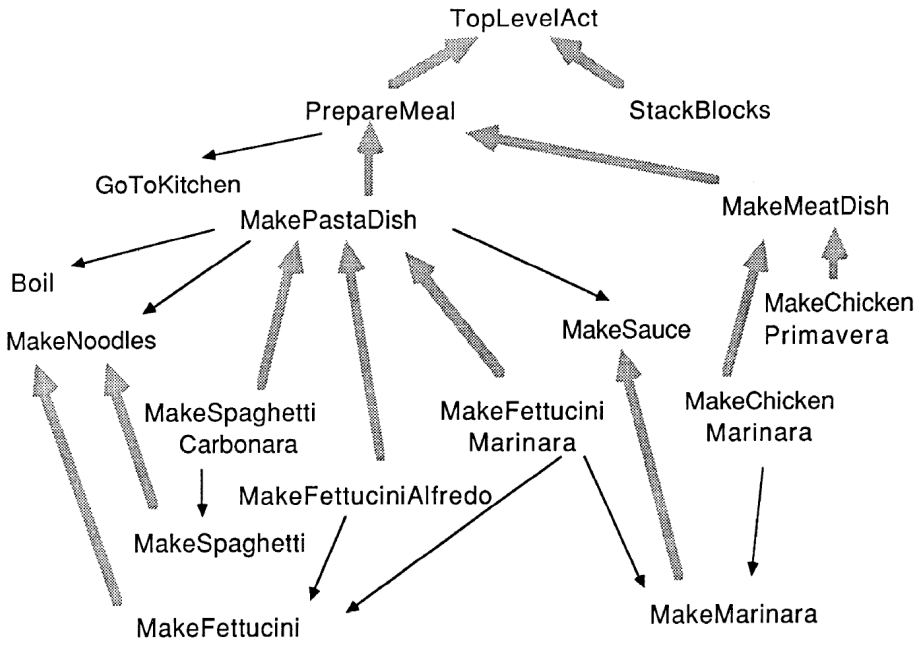


Figure 2.1 Action hierarchy for preparing a meal, illustrated by Kautz and Allen [63]

nition could be streamlined as a parsing challenge, thereby rendering some complex plan recognition tasks computationally manageable. Despite these advances, Vilain acknowledged a significant limitation: the finite nature of plan libraries, which lacked generativity and only encoded a limited array of possible plans, thus constraining the scope of recognizable plans.

The work by Geib and Goldman [46] introduced the PHATT algorithm, which used probabilistic grammars to model plan execution dynamics, similar to Vilain's approach in using formal grammars to represent plan libraries. Unlike conventional pattern matching, PHATT adeptly managed complex scenarios with multiple interleaved goals and actions that were partially observed or unobserved. This made it highly suitable for applications ranging from security monitoring to interactive systems. By incorporating temporal and parametric constraints, PHATT enhanced its ability to interpret the sequence and timing of actions accurately, as evidenced by robust theoretical analysis and empirical testing.

Charniak and Goldman [23] stated that plan recognition—the process of deducing an agent's plan from observations—primarily involves inference amidst uncertainty. To address this, they introduced a Bayesian network composed of a series of probabilistic grammars, which facilitated the modeling of various potential plans. This approach defined potential explanations for the observed behaviors, often focusing on the most promising ones, and structured these into a Bayesian network for plan recognition. This network represents the probability distribution across these explanations, and through Bayesian updating, the

method determines the most probable interpretation of the observed actions. This approach is specifically designed for understanding stories in natural language.

Avrahami-Zilberbrand and Kaminka [10], in turn, used the so-called Feature Decision Tree (FDT) to encode their plan library, effectively tackling the complexities of matching multi-featured observations to a hierarchical plan library. They introduced a machine-learning decision-tree that efficiently mapped observations to relevant plan steps, significantly reducing computational load. Furthermore, they developed lazy-commitment recognition algorithms that managed and updated recognition hypotheses incrementally, as needed, thus avoiding the inefficiencies of constant recalculations with each new observation. The algorithms distinguished between the agent's current state and its history of selected states, thereby enhancing efficiency and scalability. Through empirical testing with synthetic data, their approach proved to enhance efficiency and capabilities for applications in real-time monitoring and interactive systems. Many other works tackle goal recognition challenges by employing similar logic to propose various forms of plan library-like models. These include using probabilistic models for goal recognition [100, 45], employing Markov logic networks to infer players' goals in video game scenarios [48], and focusing on improving interactive systems by recognizing user goals and plans through dialogue interactions [18, 19]. We have listed several typical works that address the goal recognition problem using pre-defined plan library-like models, while there remains a vast number of studies that we cannot exhaustively introduce in this section.

Early works on plan library-based approaches demonstrated the potential of these GR techniques to address real-world problems. However, these approaches generally require knowledge encoding, which involves encoding the known "operational plans" of the domain into models (e.g., a hierarchical tree or network of goals and actions). These plan libraries are then parsed and "matched" against observed sequences of actions. The fact is that, in many domains, crafting these libraries can be costly or simply not feasible. Furthermore, GR approaches that rely on the specification of plan libraries may struggle to adequately handle behavior that falls outside the provided plans. With improvements in computational speed and power, some later studies attempt to address goal recognition from the reverse side of the *planning* problem, referred to as planning-based GR. We review key publications on planning-based GR in the following section.

2.1.2 Planning-Based Goal Recognition

Possibly the first step towards plan-library-"free" GR was Hong's proposal [56], 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 [101, 102] that provided an elegant GR approach by leveraging on the representational and algorithmic techniques in AI model-based automated planning, often referred to as "plan recognition as planning."

The 2009 work by Ramírez and Geffner [101] introduced a method that uses planning algorithms to infer goals from observed actions, eliminating the need for a traditional plan library. In 2010, they further advanced this concept by incorporating a probabilistic model that accounts for the cost differences in achieving goals, either with or without conforming to the observed actions [102]. This probabilistic extension allows the method to handle non-optimal and noisy behaviors of agents. 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, an idea also noted elsewhere [11, 94], 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 [47]. 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 declarative models are typically as formulated PDDL [51] or STRIPS [34] models and are used to synthesize possible planning goals.

Planning-based GR approaches rely on planners, which require the domain model as a prerequisite input. Below, we provide an example that explicitly illustrates the components and structure of a domain model. Figure 2.2 presents a simple example of a Sokoban problem, which is a widely studied puzzle that requires moving boxes within a constrained grid environment to achieve specific goal states. In this example, the grid consists of 3×3 cells. The robot starts at cell $(0,0)$, a box is initially located at cell $(1,1)$, and the goal is to move the box to the target location at cell $(2,2)$.

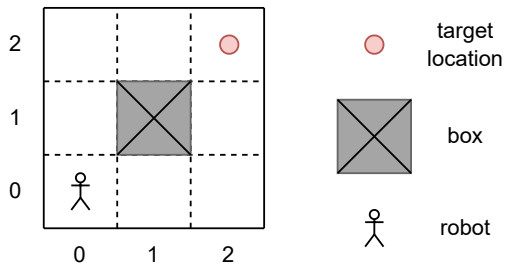


Figure 2.2 A Sokoban problem in 3 by 3 grid.

Below is an example of using the Planning Domain Definition Language (PDDL) to represent the Sokoban problem in Figure 2.2. The PDDL domain model contains a *domain definition* and a *problem definition* (specific problem instance). Listing 2.1 illustrates the

```
(define (domain sokoban)
  (:requirements :typing)
  (:types LOC DIR BOX)
  (:predicates
          (at-robot ?l - LOC)
          (at ?o - BOX ?l - LOC)
          (adjacent ?l1 - LOC ?l2 - LOC ?d - DIR)
          (clear ?l - LOC)
  )
  (:action move
    :parameters (?from - LOC ?to - LOC ?dir - DIR)
    :precondition (and (clear ?to) (at-robot ?from) (adjacent ?from ?to ?dir))
    :effect (and (at-robot ?to) (not (at-robot ?from)))
  )
  (:action push
    :parameters  (?rloc - LOC ?bloc - LOC ?floc - LOC ?dir - DIR ?b - BOX)
    :precondition (and (at-robot ?rloc) (at ?b ?bloc) (clear ?floc) (adjacent ?rloc ?
        bloc ?dir) (adjacent ?bloc ?floc ?dir))
    :effect (and (at-robot ?bloc) (at ?b ?floc) (clear ?bloc) (not (at-robot ?rloc))
        (not (at ?b ?bloc)) (not (clear floc)))
  )
)
```

Listing 2.1 Domain definition

domain definition for Sokoban, capturing the overarching rules, fundamental actions, and constraints of the puzzle. It specifies object types: `LOC` for grid locations, `DIR` for directional movements (e.g., up, down, left, and right), and `BOX` for movable boxes. Additionally, it defines four predicates relevant to this domain. The symbol `?` indicates a parameter that can be replaced by concrete values to form a valid predicate. The predicate `(at-robot ?l - LOC)` has one parameter, specifying the robot's exact location. The predicate `(at ?o - BOX ?l - LOC)` uses two parameters to denote which box is located at which cell. The predicate `(adjacent ?l1 - LOC ?l2 - LOC ?d - DIR)` includes three parameters to describe adjacency relationships between locations in a particular direction (i.e., which cell is adjacent to another and in which direction). Finally, the predicate `(clear ?l - LOC)` has one parameter that identifies unoccupied locations, indicating where the robot can move. The domain defines executable actions. Each action has parameters, which can be assigned specific values to form a concrete action. However, to determine whether a concrete action is executable, we must evaluate its preconditions. If all the predicates in the precondition are satisfied, the action can be executed. Executing an action results in its corresponding effects, which are essentially changes to predicates: some predicates may be added, while others may be removed as a consequence of the action's execution.

In this example, the Sokoban domain defines two actions: `move` and `push`. The `move` action allows the robot to move between adjacent locations if the target location is clear, updating the robot's position accordingly. The `push` action enables the robot to push a box from one location to an adjacent location, provided the target location is clear and the robot

```
(define (problem sokobanProblem)
  (:domain sokoban)
    (:objects
      up down left right - DIR
      box0 - BOX
      f0-0f f0-1f f0-2f
      f1-0f f1-1f f1-2f
      f2-0f f2-1f f2-2f - LOC
    )
    (:init
      (adjacent f0-0f f0-1f right)
      (adjacent f0-0f f1-0f down)
      (adjacent f0-1f f0-0f left)
      (adjacent f0-1f f0-2f right)
      (adjacent f0-1f f1-1f down)
      (adjacent f0-2f f0-1f left)
      (adjacent f0-2f f1-2f down)
      (adjacent f1-0f f1-1f right)
      (adjacent f1-0f f0-0f up)
      (adjacent f1-0f f2-0f down)
      (adjacent f1-1f f1-0f left)
      (adjacent f1-1f f1-2f right)
      (adjacent f1-1f f0-1f up)
      (adjacent f1-1f f2-1f down)
      (adjacent f1-2f f1-1f left)
      (adjacent f1-2f f0-2f up)
      (adjacent f1-2f f2-2f down)
      (adjacent f2-0f f2-1f right)
      (adjacent f2-0f f1-0f up)
      (adjacent f2-1f f2-0f left)
      (adjacent f2-1f f2-2f right)
      (adjacent f2-1f f1-1f up)
      (adjacent f2-2f f2-1f left)
      (adjacent f2-2f f1-2f up)
      (at box0 f1-1f)
      (clear f0-0f)
      (clear f0-1f)
      (clear f0-2f)
      (clear f1-0f)
      (clear f1-2f)
      (clear f2-0f)
      (clear f2-1f)
      (clear f2-2f)
      (at-robot f0-0f)
    )
    (:goal (at box1 f2-2f))
)
```

Listing 2.2 Problem definition

is positioned next to the box, facing the box in the same direction as the movement from the robot to the box and from the box to the target location. This action updates the positions of both the robot and the box.

The Sokoban domain provides a reusable framework, while the problem instance specifies particular configurations, including the initial state and desired goal state, as shown in Listing 2.2. In this specific task, the grid is a 3×3 layout where the robot begins in the bottom-left corner `f0-0f` and a box (`box0`) is located at the center `f1-1f`. The domain

defines objects such as directions (up, down, left, right), locations (`f0-0f to f2-2f`), and the box. The initial state encodes adjacency relationships for valid movements, specifies that the robot starts at f0-0f and the box at f1-1f, and marks all locations as clear except the one occupied by the box. The goal state requires moving the box to the top-right corner `f2-2f`.

The research presented by Pereira et al. [95, 96] leveraged planning landmarks combined with probabilistic models to develop efficient heuristics for predicting user goals, where a landmark is a critical element or condition that must be realized at some point in any valid plan that aims to achieve a specific goal. In their 2017 work [95], the use of planning landmarks as heuristic guides allowed the system to quickly and accurately infer an agent's objectives without needing to simulate every possible action or outcome, thus speeding up goal recognition. Building on this foundation, their 2020 work [96] refined these landmark-based heuristics to enhance goal recognition accuracy. This later work includes adjustments that better account for uncertainty and variability in human behavior and planning, thereby improving the system's robustness and reliability in dynamic real-world scenarios.

The work by Santos et al. [107] introduced a novel approach to goal recognition using linear programming (LP) within an operator-counting framework. This method efficiently handles partial and noisy observations by incorporating observation-counting constraints, which enhanced the accuracy and robustness. The framework uses heuristic functions to estimate uncertainties and manage noisy data, showing superior performance in agreement ratio, accuracy, and spread compared to the existing approaches. Besides, many other studies [130, 99, 132] tackle the GR problem by assuming that agents follow an optimal (or near-optimal) plan to achieve their goals. These studies employ planning techniques and probability models to infer the likely goals of an agent by comparing observed action sequences with optimal plans generated by planners. Additionally, some works also consider handling irrational behaviors [83, 144].

A common drawback of planning-based GR techniques is that declarative models, such as PDDL or the STRIPS model, are generally designed to represent synthetic scenarios (domains). It is challenging to define effective and precise models for many real-world scenarios. For example, it is difficult for planning-based GR approaches to accurately define domain models that describe customer behaviors in real-world scenarios like banks or supermarkets, and then to infer the goals of these customers. Moreover, even in synthetic domains, defining such domain models requires considerable effort from domain experts, and these models are not generalizable to every domain. When a new domain arises, domain experts need to define it anew. We review learning-based GR techniques in the following section.

2.1.3 Leaning-Based Goal Recognition

The development of deep learning has provided mature techniques for handling many pattern recognition problems, and recent works in the goal recognition field have also applied deep learning algorithms to train neural network models to address GR challenges. The work by Min et al. [88] proposes a framework for player goal recognition in open-world digital games using Long Short-Term Memory (LSTM) networks. In this study, the sequence of player actions is analyzed to predict the player's next gameplay objective. Their approach includes encoding player actions, where discrete actions are transformed into continuous vector spaces. The LSTM network automates the process of feature extraction, which not only reduces the time and effort involved in model development but also enhances the model's ability to generalize across different gaming scenarios without the need for fine-tuning. Amado et al. [6] aim to address the GR problem without relying on pre-defined domain knowledge provided by domain experts. They aim to predict the next images from a sequence of images using a variational autoencoder (VAE) to encode these images into a latent space. This latent representation captures essential features of the raw data that are relevant for planning and goal recognition. From the latent representations, they automatically generate a planning domain, which includes defining an action library in PDDL. Another work by Amado et al. [7] proposed a GR framework that integrates model-free reinforcement learning (RL) techniques. In this system, policies or utility functions for each potential goal are obtained using tabular Q-learning. These learned policies are then applied to infer the agent's goals by comparing the observed trajectory against the learned policies, thereby identifying the most likely goal. Chiari et al. [24] addressed GR as a classification task by introducing GRNet, a Recurrent Neural Network (RNN) designed to process action sequences and determine the class (goal) of each sequence. Through experiments, they demonstrated that GRNet achieves superior performance in terms of speed and accuracy compared to the landmark-based GR approach [96] across various domains.

Learning-based approaches do not rely on pre-defined planning domain models or plan libraries. Although these approaches require historical datasets for training, data collection and model training can be automated, which minimizes human effort. Moreover, the trained models are generalizable and can predict sequences not present in the historical data. However, a common drawback of neural network-based models is their "black-box" nature, which obscures the underlying logic behind their inferences. In this thesis, the "evidence-based GR framework" we propose not only automatically learns models but also acquires interpretable models, such as Petri nets, enhancing the transparency of the inferred logic. A few works are similar to our proposal. However, they require more than just historical action sequences. For instance, the GR approach proposed by Sohrabi et al. [114] requires sequences of *states* as

input and uses pre-defined domain knowledge as mapping functions to map state sequences to plans. Another work by Shvo et al. [111] presented a GR approach that can learn interpretable dynamics, specifically through deterministic finite automata. However, this approach requires additional knowledge of the transition functions.

In summary, plan library-based approaches are explainable, but defining a generalized plan library is challenging and requires significant human effort. Planning-based approaches, utilizing STRIPS or PDDL models, can exhaustively represent possible plans within a specific domain, yet defining such a domain model also demands considerable human effort. On the other hand, learning-based GR approaches, such as those using deep neural network models, generally lack explainability. We state that our GR approach primarily aligns with learning-based GR. Moreover, our proposal retains the explainability advantages, as detailed in Table 2.1, which outlines the advantages and disadvantages.

	Human effort-free	Generalizable	Explainable
Plan library-based GR	✘	✘	✔
Planning-based GR	✘	✔	✔
Learning-based GR	✔	✔	✘
Our GR framework	✔	✔	✔

Table 2.1 The capabilities of GR approaches.

2.2 Process Mining Techniques

The study of workflow executions began in the late 1990s, and the concept of *process mining* was first introduced by van der Aalst [121]. Process mining is an analytical discipline aimed at discovering, monitoring, and improving real processes by extracting knowledge from event logs. It includes three main research aspects: process discovery, conformance checking, and enhancement [122]. In this thesis, the proposed evidence-based GR framework incorporates process discovery and conformance checking techniques.

2.2.1 Process Discovery

Process discovery is a critical component of process mining that focuses on automatically extracting process models from event logs. These event logs contain detailed records of process activities captured by information systems. The process models are representations of the sequences and dependencies between activities within a business process. Common modeling notations include Petri nets, BPMN (Business Process Model and Notation), and

Causal nets. We review the key publications on process discovery techniques, including the Alpha Miner, Inductive Miner, Heuristics Miner, Split Miner, and Directly Follows Miner.

The Alpha Miner [124] is one of the foundational algorithms in process mining. It uses the ordering relations between events to construct a workflow net (a type of Petri net specifically tailored for modeling workflows) that captures the control flow of the process. The algorithm identifies causal dependencies between activities. For instance, if activity a_1 is often followed by activity a_2, a causal relation $a_1 \rightarrow a_2$ is inferred. The Alpha Miner algorithm follows three main steps. First, it identifies relations by determining direct succession, causality, parallel, and choice relations between activities based on the event log. Second, it constructs places using these identified relations to create states in the Petri net. Finally, it builds the net by connecting activities with places to form a Petri net that accurately reflects the observed process behavior. However, the Alpha Miner can be sensitive to noise and may produce incorrect models if the event log contains infrequent or erroneous events. Additionally, it struggles with accurately discovering parallel activities, especially in complex and highly concurrent processes.

The Inductive Miner [71] uses a divide-and-conquer strategy to recursively split an event log into smaller subsets based on identified patterns such as sequences, parallelisms, choices, and loops. It guarantees the production of sound workflow nets, ensuring that the discovered process models are free from deadlocks and other anomalies. One of the key strengths of the Inductive Miner is its robustness against noise, making it particularly suitable for real-world event logs that often contain irregularities and infrequent behaviors. Additionally, it achieves a good balance between fitness and precision, making it well-suited for constructing process models from structured and repeated behaviors.

The Heuristics Miner [136] constructs process models by identifying frequent and significant patterns in event logs, effectively handling noise and incomplete data. It produces heuristics nets, which represent the sequence, parallelism, and frequency of activities within a process, enabling organizations to visualize and optimize their workflows. The Flexible Heuristics Miner [137] (FHM) is an enhanced version of this algorithm, developed to address the limitations of the traditional Heuristics Miner in dealing with low-structured and noisy event logs. FHM utilizes augmented Causal nets (augmented-C-nets) to provide a more detailed and accurate representation of complex control-flow constructs. By incorporating split and join frequency tables and focusing on the most significant patterns, FHM achieves improved accuracy and understandability, making it a robust and flexible tool for discovering process models from complex and noisy data.

The Split Miner [9] addresses common issues in existing process discovery methods such as producing complex, spaghetti-like models and balancing model quality dimensions

like fitness, precision, generalization, and simplicity. By filtering the directly-follows graph induced by an event log and identifying combinations of split gateways, Split Miner captures concurrency, conflict, and causal relations effectively. It guarantees the creation of deadlock-free process models without being restricted to block-structured models. Empirical studies demonstrate that Split Miner achieves a high and balanced fitness and precision while maintaining low complexity and faster execution times compared to other state-of-the-art methods.

The Directly Follows Miner (DFM) [72] constructs process models by directly capturing the transitions between activities as they appear in the event logs. It primarily focuses on the directly-follow relationships, creating a model that precisely reflects the sequences in the events log. The DFM technique is designed to create clear and user-friendly process maps, which can be easily interpreted by stakeholders. Unlike more complex academic tools that often produce models with intricate details like concurrency and inclusive choices, DFM aims for simplicity and clarity. It helps organizations to visualize their processes and identify performance measures such as execution times and bottlenecks. However, one of the challenges with directly-follows models is ensuring their accuracy and soundness, as overly simplistic models can lead to incorrect conclusions. To address this, DFM incorporates conformance checking techniques to validate the discovered models against the event logs, ensuring that the models accurately reflect the recorded process behavior.

2.2.2 Conformance Checking

Conformance checking is a technique that demonstrates the alignment between observed behavior (event logs) and modeled behavior (process models). In this thesis, the evidence-based GR framework employs techniques drawn from two existing works [1, 126] to construct optimal alignments between new observations and the learned process models. The work by Adriansyah et al. [1] focuses on a cost-based approach to assess fitness in conformance checking. This method quantifies the cost of deviations between the model and the log, providing a monetary or impact-based measure of non-conformance. Additionally, the work by van der Aalst et al. [126] elaborates on the concept of replaying history on process models. This approach involves mapping events in an event log to elements within a process model, enabling detailed analysis of where and how actual process executions deviate from the model. In Chapter 3, we elaborate on the conformance checking using a concrete running example to demonstrate how alignments are utilized to compute probabilities, which are then used to infer the goal.

2.2.3 Predictive Process Monitoring

In the area of business process management, several existing works have objectives similar to goal recognition. These works aim to predict the business goal of an incomplete business process (i.e., a partially observed trace), a task referred to as outcome-oriented predictive process monitoring [118] (PPM). From a process mining perspective, PPM is an analogous research area to GR in the planning community. The existing methods [82, 69, 125, 39, 133, 2] 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. [118], the outcome-oriented predictive process monitoring methods first 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) [82], the state-based approaches [69, 125], and the clustering-based approaches [39, 133]. 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.

2.3 Adaptive Goal Recognition

We extend the single-shot GR problem to the adaptive GR problem. Lesh [76] proposed an approach for selecting the best-performing adapted GR system to solve a single-shot GR problem, given a collection of allowed GR system adaptations. However, different from the work by Lesh [76], our study focuses on retraining a data-driven GR system to maintain high performance over an extended period, as demonstrated in Chapter 4. Adaptive GR demands GR systems to tackle multiple single-shot GR tasks over a period of time. When the environment changes, the GR systems need to detect the changes and subsequently adapt their GR inference procedures. The adaptive GR problem can be decomposed into subproblems of three types: a conventional GR problem, a knowledge model updating problem, and a concept drift detection problem. The related works for conventional GR have already been reviewed in Section 2.1. In this section, we explore related works that focus on updating the

knowledge model and detecting concept drift. Additionally, we discuss potential applications that can benefit from adaptive GR techniques.

The work by Bryce et al. [17] proposed a technique for issuing queries for updating the knowledge models. This technique adapts the models by analyzing query answers. However, instead of updating the models automatically, this technique relies on human-provided query answers. Chakraborti et al. [22] proposed a model reconciliation algorithm. The idea of model reconciliation is that if two different models result in different optimal plans for the same goal, the reconciliation technique attempts to modify one of the models to get a single optimal plan for both models. Consequently, when reconciling the models, one updates them according to a single trace. In contrast, our relearning mechanism updates the models based on an arbitrary number of traces. The algorithms for process model repair can be used for updating the process models of the PM-based GR systems [33, 97]. In this thesis, however, the PM-based GR systems relearn new process models from scratch. Specifically, we use DFM [72] to learn the initial models and then relearn the up-to-date models. The problem of concept drift detection in process mining studies ways to detect behavioral changes by analyzing the chronologically ordered traces in an event log [140, 29, 55]. The solutions to the concept drift detection problem can be used to trigger the relearning of knowledge models of adaptive GR systems.

The adaptive GR systems can be deployed in scenarios such as human-robot teaming, where robots continuously update their estimation of the plans and goals of the human(s) in the loop with every new observation. The domain knowledge of an adaptive GR system can evolve over time due to changing human goals and a changing environment. An early example using the planning-based GR system [102] and an abstraction technique called "resource profiles" is presented by Chakraborti et al. [21], where the evolving nature of the environment was extracted from the output of the GR algorithm to feed into the ultimate plan generator. Similar approaches have recently been built on this principle of abstraction in settings that require higher-order representations driving the adaptive plan/goal recognition and plan generation cycle [60].

2.4 Powered Transhumeral Prosthesis

In the field of powered transhumeral prosthetics, linear discriminant analysis (LDA) is the most commonly used classifier algorithm in prosthesis control and is also used in gesture recognition scenarios via Myoelectric interfaces [59, 80]. Due to the lightweight and low-complexity nature of the LDA classifier, it can achieve high control accuracy with short training and processing times [93]. Machine learning classifiers are commonly implemented

in upper-limb prostheses [44]. Given an input signal from sensors, a classifier predicts the output signal and the intended movement of the patient. In the work by Shehata et al. [110], it was demonstrated that neural networks achieve high goal recognition accuracy because they can learn complex dependencies between signal inputs and control outputs, although they require extensive training. For instance, Huang et al. [57] successfully employed LSTM neural networks to predict target poses based on time series of electromyography signals.

Chapter 5 explores how the evidence-based GR framework can contribute to the development of powered transhumeral prostheses. To this end, we implement a system based on the evidence-based GR framework, designed to analyze repeated, real-valued, continuous data captured by typical sensors and to infer the users' movement intentions (goals). As existing literature has demonstrated that using varied features customized for individual subjects enhances the accuracy of identifying intended movements [141]. Such customization introduces complexity into the development of ideal plan libraries or domain models, necessitating unique plans or models for each individual subject. We state that the evidence-based GR framework is data-driven and particularly well-suited for customization, as it can learn *personalized* patterns from a patient's historical behavior. Therefore, we conducted both offline and online experiments on the individual level. In the offline setting, the GR techniques are compared using pre-recorded data. In the online setting, we assess the performance of the GR techniques through human-in-the-loop control experiments. This experimental setup aims to address concerns raised in the literature about the conflicting results regarding the performance correlation between these two conditions [92, 81, 52].

2.5 Tools

This thesis develops and evaluates process mining-based GR systems using several tools. The work by Speck et al. [115] introduced the Symbolic Top-K Planner, aimed at generating a set of the k most cost-effective plans for a given planning task, referred to as the *top-k planner*. The top-*k* planner utilizes representations of state spaces through binary decision diagrams, which can handle sets of states. This approach enhances efficiency and scalability when exploring multiple potential solutions, and the top-*k* planner is both sound and complete. The decision problem associated with determining whether there exists a set of k plans for a planning task is theoretically PSPACE-complete. This places top-*k* planning equal to classical planning in terms of computational complexity. The work by Katz and Sohrabi [61] aims to generate multiple plans that not only meet quality standards but also exhibit diversity. They propose the Forbid Iterative (FI) Approach, a planning algorithm that modifies the planning task rather than the planner itself, which is adaptable to any planner.

Similar to the top-*k* planner, the FI approach can first generate a large set of plans and then choose a subset based on configurable diversity metrics. Therefore, we refer to it as a *diverse planner*. Additional tools relevant to this thesis include *ProM Tools*,[1] which offer an expandable framework accommodating diverse process mining algorithms through plug-ins. The *Exploratory Modeling and Analysis (EMA) Workbench*[2] is a tool we used for sensitivity analysis and parameter optimization, as detailed in Chapter 3 Section 3.5. *LAPKT* provides straightforward interfaces, providing flexibility by decoupling parsers from problem representations and algorithms [103]. *Tarski* facilitates the modeling and manipulation of AI planning problems [38]. In Chapter 4 Section 5.4, the experimental data for evaluating adaptive GR systems consist of a series of historical agents' traces generated in different environments. We developed a tool called GRACE to generate such experimental data. Given a GR problem instance in PDDL format, GRACE utilizes the Tarski and LAPKT tools to parse and manipulate the input environment models, simulating different drifts. Additionally, it leverages the top-*k* and diverse planners to generate sequences of agent actions towards different goals in the drifting environments.

[1] https://promtools.org/

[2] https://github.com/quaquel/EMAworkbench

Chapter 3

Process Mining-Based Goal Recognition

In this chapter, we address *RQ1: How to do goal recognition in the absence of domain knowledge?* We introduce the evidence-based goal recognition framework, which is inspired by the principles of observational learning [12]. This framework is also referred to as the Process Mining-based Goal Recognition (PM-based GR) framework because it is primarily designed to use process mining techniques to instantiate concrete GR systems. These systems can automatically learn process models from event logs of historical agent observations. The learned models encode the skills for achieving various goals in the environment, and the GR system analyzes deviations between the new observed agent behavior and the learned skill models to make inferences. The GR system is expected to output a probability distribution over a pre-defined set of goal candidates, such a problem referred to as *Probabilistic Goal Recognition*. We assume that the agent operates in a static environment and receives only a sequence of observations as input, which is a common assumption in the field of goal recognition research.[1] The PM-based GR framework serves as the foundation of this thesis, with the adaptive GR (in Chapter 4) and the case study of the robotic prosthesis scenario (in Chapter 5) built upon the findings of this chapter.[2]

Concretely, this chapter makes the following contributions:

- It proposes a GR framework that describes the fundamental mechanisms for performing GR without pre-defined models;
- It discusses an implementation of a concrete GR approach based on process mining techniques that follows the proposed GR framework and relies on four parameters to construct a probability distribution over possible goals and to infer the most likely

[1]The terms "single-shot" and "conventional" GR, which are mentioned repeatedly in the rest of this thesis, actually refer to probabilistic GR.

[2]This chapter is an adaptation of a previously published work, "Fast and accurate data-driven goal recognition using process mining techniques." Artificial Intelligence, 323:103973, 2023. ISSN 0004-3702.

goal. The four 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, a discount factor (δ) that emphasizes the recently observed actions to have more impact on the goal inferences, and a decision threshold (θ) that determines which goals should be inferred as likely goals;

- It presents the 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 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 introduces the probabilistic goal recognition problem by means of a motivating example. Section 3.2 is devoted to presenting the GR framework. Section 3.3 introduces the process mining techniques used to implement a concrete PM-based GR system, which is subsequently presented in detail in Section 3.4. Finally, Section 3.5 presents the results of an evaluation of our implementation of the GR system.

3.1 Probabilistic Goal Recognition

A definition of probabilistic goal recognition that can be symbolically formulated requires some preliminary knowledge about planning, as goal recognition is akin to the reverse side of an automated planning problem. To this end, we use symbolic terms from planning to define the probabilistic goal recognition problem. Firstly, we introduce several important sets: (i) Let $\mathcal{F}$ be a set of *fluents*, where a fluent refers to a condition or property of the world that can change over time. A fluent can be represented in first-order logic by a predicate. In the PDDL model example presented in Chapter 2, Listing 2.2, each row in the "`:init`" section describes a property of the initial state. For example, the predicate `(at-robot f0-0f)` specifies the robot's location and represents a fluent; (ii) Let $\mathcal{A}$ be the set of all possible *actions*, where each action $a \in \mathcal{A}$ has preconditions $Pre(a) \subseteq \mathcal{F}$, additive effects $Add(a) \subseteq \mathcal{F}$, and delete effects $Del(a) \subseteq \mathcal{F}$. Additive effects refer to the outcomes of actions that introduce new fluents to the current state of the world without removing or conflicting with existing facts. In contrast, delete effects represent the negative consequences of an action, removing

fluents from the current state of the world; (iii) Let $\mathcal{O} = \mathcal{A}^*$ represent the set of all possible finite-length action sequences formed by concatenating actions from the set $\mathcal{A}$. Then, the probabilistic goal recognition problem can be defined as follows.

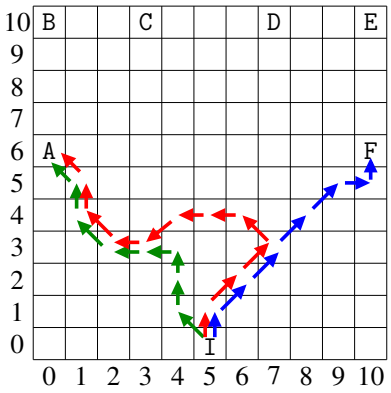
Definition 1 (Probabilistic goal recognition)**.** *Given a tuple* $< \mathcal{F}, \mathcal{A}, I, \mathcal{G}, O >$*, where* $\mathcal{F}$ *is a set of* fluents*,* $\mathcal{A}$ *is a set of* actions*,* $I \subseteq \mathcal{F}$ *is the* initial state*,* $\mathcal{G} \subseteq 2^{\mathcal{F}}$ *is a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

In this thesis, the definition of the single-shot, or conventional, GR problem refers to the definition of probabilistic GR, which serves as the foundation (the adaptive GR problem in Chapter 4 is derived from this definition).

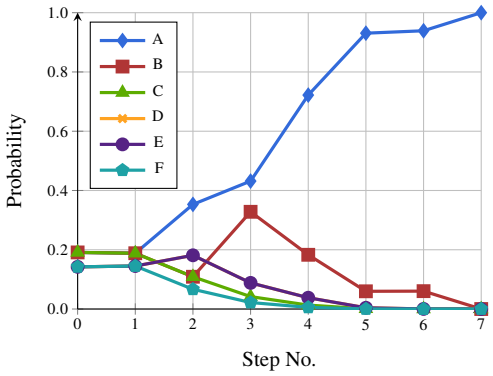
We present an example of a probabilistic GR problem in Figure 3.1a. The figure depicts an 11x11 grid, including the initial state in cell `I` and six goals represented by cells `A` to `F` (this grid is similar to the grid used in [102]). The grid also shows three observed walks of an agent, comprising a rational walk towards goal `A` (green), an irrational walk towards goal `A` (red), and a rational walk towards 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 $\sqrt{2}$.

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 towards reaching target goal `A`, resulting in a cost of $5+6\sqrt{2}$. Hence, the red walk is *irrational*. Finally, the blue walk towards 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.

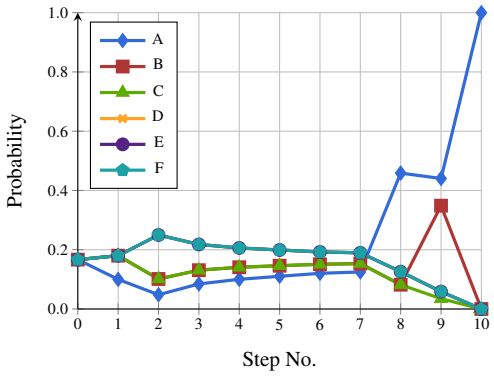
In Section 3.4, we present our approach to obtaining the probability distributions over goal candidates, referring to our GR system for now. Figures 3.1b to 3.1d show the probability distributions over the candidate goals computed by our GR system for the three walks from Figure 3.1a. As expected, along the green rational walk, goal `A` is consistently the most likely goal; see Figure 3.1b. 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 towards the end of the walk; see Figure 3.1c. Finally, the blue rational walk towards goal `F` shows an equal probability towards 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 Figure 3.1d. Empirical evidence suggests (refer to Section 5.4) that our GR system can be used to quickly and accurately infer the intended goals of agents for a wide range of domains.



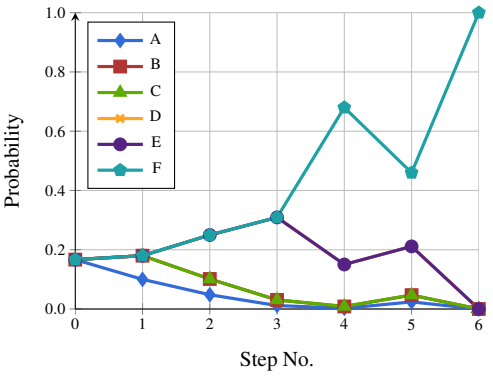
(a) Three walks of an agent in a grid: two from initial cell `I` to goal cell `A` (green rational walk and red irrational walk) and one from initial cell `I` to goal cell `F` (blue rational walk).



(b) Rational walk to goal `A`.



(c) Irrational walk to goal `A`.



(d) Rational walk to goal `F`.

Figure 3.1 Inferred probability distributions (Figure 3.1b, 3.1c, and 3.1d) over the six goals from Figure 3.1a computed based on the observed behaviors shown in Figure 3.3.

Ramírez and Geffner [102] derive 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 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 a most recent elaboration by Masters and Sardiña [84, 86],

which refined the original set-up to achieve a simpler and computationally less demanding GR approach that can handle irrational agent behavior parsimoniously without counter-intuitive outcomes. Concretely, taking $optc(St, O, G)$ to denote the optimal cost of reaching goal $G \in \mathcal{G}$ from state $St \subseteq \mathcal{F}$ by *embedding* the sequence of observations $O \in \mathcal{O}$, we first define the ***cost difference*** of reaching the goal G from St via observations O as follows:[3]

$$costdiff(St, O, G) = optc(St, O, G) - optc(St, \varepsilon, G).$$

When the agent is observed to act optimally for G, the cost difference is zero; as the agent becomes more suboptimal towards 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 [84, 86].

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 O from initial state I can be obtained as follows (note the denominator acts here as the normalization factor) [102, 85]:

$$\Pr(G \mid O) = \frac{e^{-\beta \times costdiff(I,O,G)}}{\sum\limits_{G' \in \mathcal{G}} e^{-\beta \times costdiff(I,O,G')}}, \tag{3.1}$$

where β is a parameter that "allows the goal recognition system developers to soften the implicit assumption of the agent being rational" [102]. 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 [85, 83]'s approach—in turn, inspired in the cost-ratio used by Vered et al. [131]—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:

$$\beta = \left(\max_{G \in \mathcal{G}} \frac{optc(I, \varepsilon, G)}{optc(I, O, G)} \right)^{\eta}. \tag{3.2}$$

Intuitively, this β expresses the most optimistic rationality ratio among all goals, with $\beta = 1$ when the agent is fully rational towards *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

[3] Under no observations (i.e., $O = \varepsilon$), $optc(St, O, G)$ reduces to optimal cost to G from state St.

its *confidence* as observations are gathered (η is a positive constant regulating how quickly confidence should drop when irrational behavior is seen).

The existing solutions that implement this approach require the declarative model of the planning domain. Our Research Question 1 explores how to perform GR in the absence of domain models. We assume the full sets of fluents $\mathcal{F}$ and actions $\mathcal{A}$ are not available, and the initial state I is also unknown. Instead, we collect a set of action sequences that resulted in achieving the candidate goals in the past, denoted by $D \subseteq \mathcal{A}^* \times \mathcal{G}$. We propose a solution based on *process mining* [123] techniques, which are used to learn skill models for achieving different goal candidates from historically observed traces. We then analyze the deviations between the learned skill models and the newly observed action sequence of the agent to infer its goal. Concretely, given a tuple $< D, \mathcal{G}, \tau >$, where $D \subseteq \mathcal{A}^* \times \mathcal{G}$ is a set of pairs, each relating a historical sequence of actions to the goal achieved by executing this sequence, $\mathcal{G}$ is a set of goal candidates, and $\tau \in \mathcal{A}^*$ is an *observation* represented as a sequence of actions performed by an agent, the problem consists of obtaining a posterior probability distribution over the goal candidates. Section 3.2 presents our GR framework inspired by the principles of observational learning [12]. The framework can be seen as a collection of abstract components that, when instantiated, result in a concrete GR system. Then, in Section 3.4, we discuss an instantiation of the framework using process mining techniques to implement a GR system.

3.2 Goal Recognition Framework

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 [12]. Note that, compared with other well-developed cognitive architectures, the proposed framework shown in Figure 3.2 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.

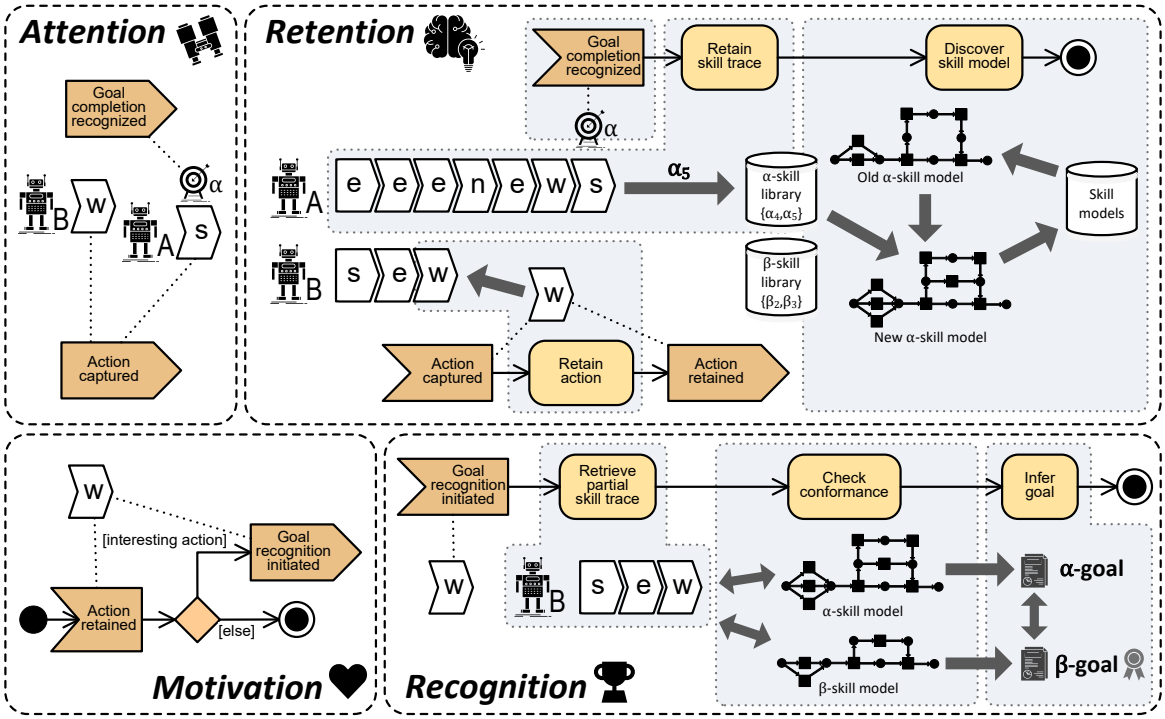


Figure 3.2 A schematic visualization of the evidence-based goal recognition framework.

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 Figure 3.2, 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 Figure 3.2) 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 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 [123, 127] 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 Figure 3.2, as a response to the "*Action retained*" signal triggered by action w, goal recognition is initiated by triggering the "Goal recognition initiated" signal. The motivation stage filters actions, waiting to accumulate sufficient meaningful subsequences before triggering the recognition stage for deeper computation. If the observed actions lack insightful patterns, the system delays processing to gather more information, conserving computational resources. However, in this thesis, we do not instantiate the motivation stage when implementing the concrete GR system; instead, the recognition stage is triggered at every step upon receiving a new action. According to the principles of observation learning, all four stages, including the motivation stage, should ideally be incorporated. Thus, integrating the motivation stage to develop an evidence-based GR system remains a direction for future work.

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 Figure 3.2. 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 [126, 123, 73] 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, Figure 3.2 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 [14].

3.3 Process Mining Techniques

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 [123].

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 a 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* $\mathcal{E}$*.*

In relation to Definition 1, a trace τ can be understood as an observation O. For the remainder of this chapter, we will use the symbol τ to represent the observation O.

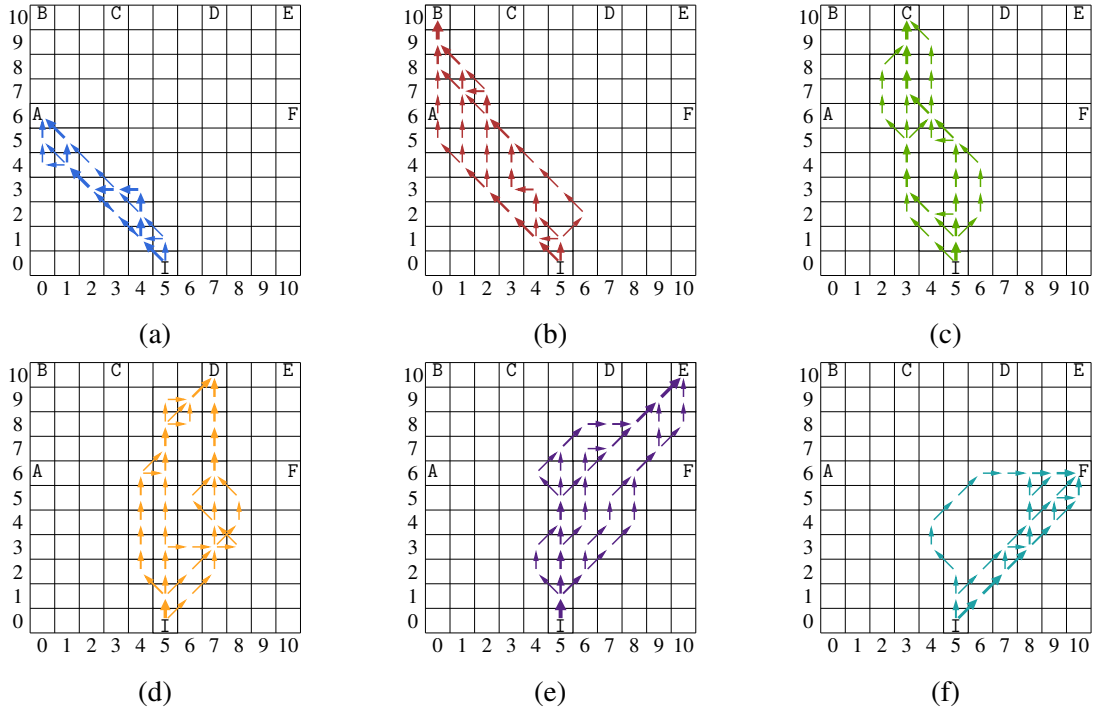


Figure 3.3 Observed agent behaviors from initial cell `I` to each of the goals `A`–`F`.

Continuing with the example in Figure 3.1a, we assume that the surrounding environment (domain model) of the agent is unknown. However, 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 Figure 3.1a, each sub-figure in Figure 3.3 shows "footprints" of six observed walks of the agent from cell `I` to one of the six goals. The thickness of an arrow in the sub-figures indicates the frequency with which the corresponding step was taken. We can encode each collection of six sequences of actions towards each of the six goals from Figure 3.3 in an event log of six traces. An event in a trace of such an event log encodes a step in the grid and can be specified as a pair of two cells: the source cell

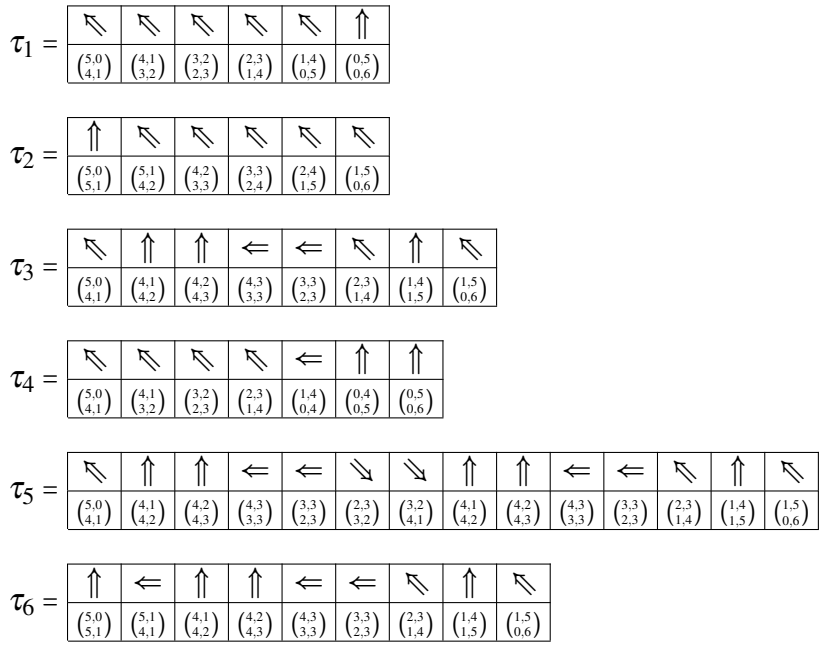


Figure 3.4 Event log $L_{\mathtt{A}}$ representing the walks to goal A shown in Figure 3.3a.

and the target cell. For example, by $\binom{5,0}{4,1}$, we denote the event of the agent moving from the initial state in 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 towards goal A shown in Figure 3.3a. These six traces from log $L_{\mathtt{A}}$ are specified in Figure 3.4. In the figure, a trace is defined 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 Figure 3.4 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.

Process Discovery

The behavior captured by the traces in an event log can be graphically represented using various process models, such as Petri nets, Directly-Follows Graphs (DFG), Business Process Model and Notation (BPMN), UML Activity Diagrams, or Event-Driven Process Chains [123]. In this thesis, we utilize Petri nets due to their formal mathematical framework for representing systems. Their concepts of tokens, places, and transitions align closely with the notions of transitions and states commonly used in the planning community. Additionally, Petri nets are particularly well-suited for applying conformance checking. However, BPMN, DFG, or other models could also serve as viable alternatives. Process discovery is a technique that automatically constructs a process model from an event log [4, 124]. 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 [120].

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 3.5 shows the marked net discovered from log $L_{\mathtt{A}}$ shown in Figure 3.4 using the Split miner discovery technique [9]. 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` towards goal `A` in the grid shown in Figure 3.3a. 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 Figure 3.4. 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.

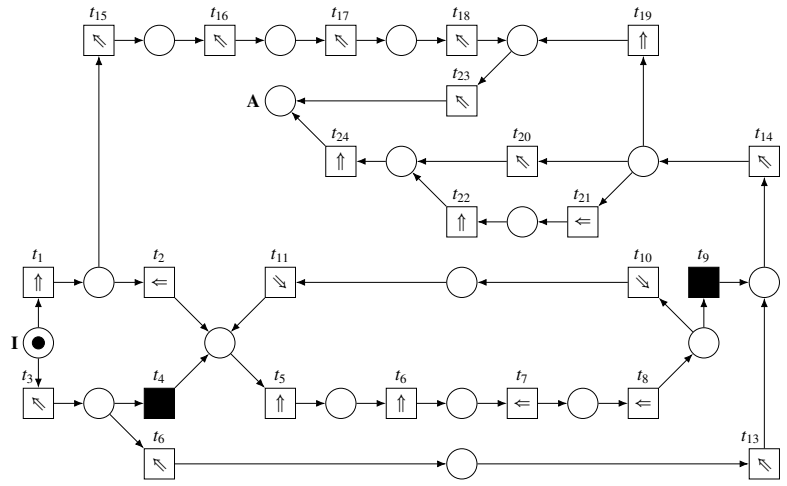


Figure 3.5 Net $N_{\mathtt{A}}$ discovered from the event log in Figure 3.4 capturing the walks to goal A shown in Figure 3.3a.

Using process discovery techniques, we obtain skill models from historical observations that describe the *skills* required for accomplishing the goals. Subsequently, new observations of an agent in the environment can be compared with the acquired models to identify 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. We use *conformance checking* techniques to identify discrepancies and use the identified discrepancies to compute probability distributions over the goal candidates.

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 [135, 126, 20, 98, 3].

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 [126]. 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 moves on 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 Figure 3.5, do not demonstrate a disagreement with traces, 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 an optimal alignment of zero cost exists between the trace and some execution of the net.

Let us again consider the three walks of the agent from Figure 3.1a. 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 towards the goals. Recall that net

$N_{\mathtt{A}}$ in Figure 3.5 models the observed behavior of the agent towards goal A from Figure 3.3a. In addition, Figure 3.6 depicts net $N_{\mathtt{F}}$ modeling the observed behavior of the agent towards goal F from Figure 3.3f.

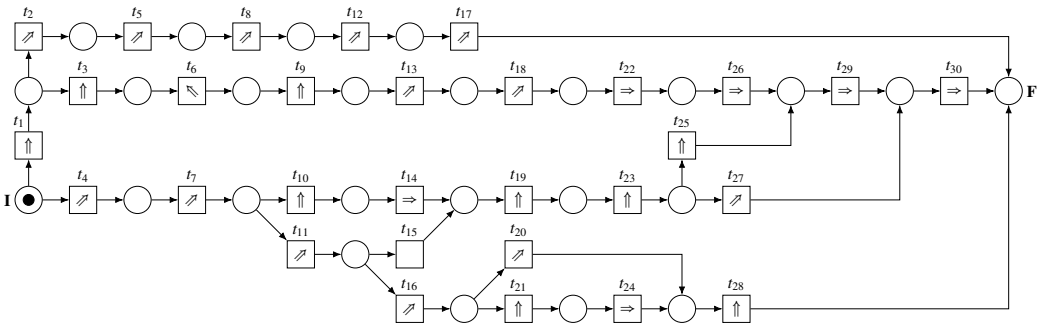


Figure 3.6 Net $N_{\mathtt{F}}$ discovered from the walks to goal F shown in Figure 3.3f.

It is convenient to represent alignments as tables. Table 3.1 shows an optimal alignment γ_1 between the rational walk to goal A in Figure 3.1a and net $N_{\mathtt{A}}$.[4] 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.

$\gamma_1 =$

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 3.1 Optimal alignment between the rational walk τ' towards goal A in Figure 3.1a and net $N_{\mathtt{A}}$ in Figure 3.5.

The first move in γ_1 is the synchronous move of transition t_3 in the net from Figure 3.5 and event that encodes the step from cell $(5,0)$ to cell $(4,1)$ in the grid from Figure 3.1a. 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

[4]The logs that describe the walks towards the six goals shown in Figure 3.3 and the nets discovered from these logs, captured using the XES standard (https://xes-standard.org/) and PNML standard (http://www.pnml.org/) notations, respectively, can be accessed here: https://doi.org/10.26188/21749570.

$\delta(\gamma_1) = 0$. Note that the first elements in the moves in Table 3.1 without the "no move" symbols define the rational walk towards goal A shown in Figure 3.1a, 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 Figure 3.5.

Table 3.2 shows optimal alignment γ_2 of the walk towards goal F in Figure 3.1a and net N_F from Figure 3.6. The first five moves in γ_2 are synchronous. The next move is, however, asynchronous. It is 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 3.1 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_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 3.2 Optimal alignment between the rational walk τ'' towards goal F in Figure 3.1a and net N_F in Figure 3.6.

Table 3.3 shows optimal alignment γ_3 of the irrational walk towards goal A in Figure 3.1a and net N_A from Figure 3.5. 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_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 3.3 Optimal alignment between the irrational walk τ''' towards goal A in Figure 3.1a and net N_A in Figure 3.5.

Finally, Table 3.4 shows optimal alignment γ_4 of the irrational walk towards goal A in Figure 3.1a and net N_F from Figure 3.6. 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_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 3.4 Optimal alignment between the irrational walk τ''' towards goal A in Figure 3.1a and net N_F in Figure 3.6.

3.4 Framework Instantiation Using Process Mining

This section presents our approach to instantiating the recognition stage of the GR framework from Section 3.2. The attention and motivation stages can capture every action and trigger goal recognition for every captured action, while the retention phase can discover each skill model from a set of observed traces (historical observations). Figure 3.7 presents the architecture of a PM-based GR system, demonstrating the computation and data flow within the system. The PM-based GR system consists of three steps: the first two are process discovery and conformance checking, which are explained in detail in Section 3.3. Note that we use the Directly Follows Miner discovery technique [72] to learn skill models.[5] 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 [9] or Inductive Miner [71], 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.

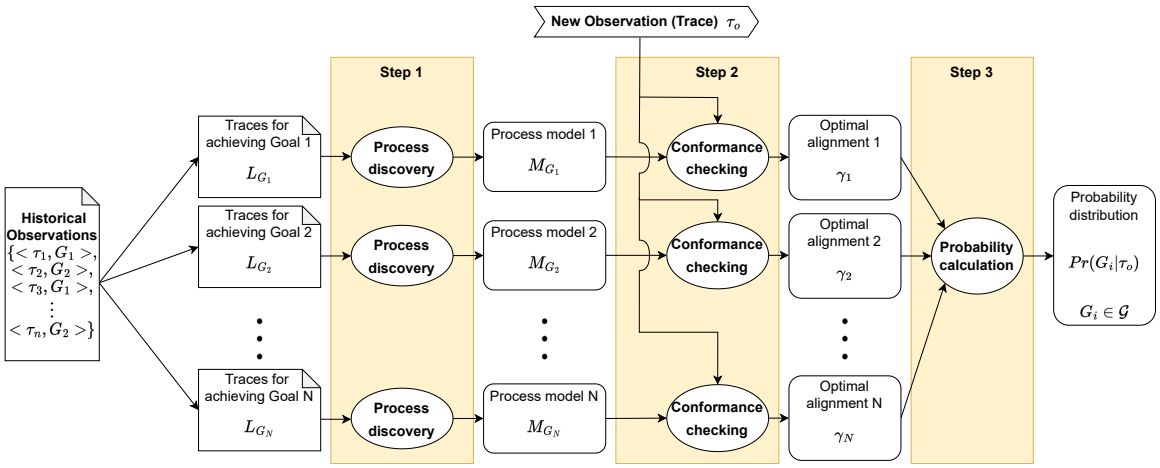


Figure 3.7 Architecture of the PM-based GR system.

[5] A skill model is denoted as α_G as shown in Figure 3.2, or as M_G as shown in Figure 3.7.

The third step in the architecture, probability calculation, focuses on instantiating the recognition stage of the evidence-based GR framework. The GR system 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 Equation 3.1 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 [84] in using a true Boltzmann distribution instead of a sigmoidal, and rewrite Equation 3.1 as follows:

$$\Pr(G \mid \tau) = \frac{e^{-\beta \times \omega(\tau, \alpha_G)}}{\sum\limits_{G' \in \mathcal{G}} e^{-\beta \times \omega(\tau, \alpha_{G'})}}. \tag{3.3}$$

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:

$$\beta = \frac{1}{1 + \min\limits_{G \in \mathcal{G}} \omega(\tau, \alpha_G)}. \tag{3.4}$$

Equation 3.4 follows and simplifies Equation 3.2 to account for the fact that the best case scenario is an alignment weight of zero, which implies that Equation 3.4 inherits the confidence-based properties described in [85]. 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:

$$\omega(\tau, \alpha_G) = \phi + \lambda^m \times \sum_{i=1}^{n} \left(i^{\delta} \times c(\tau, \alpha_G, i) \right), \text{where:} \tag{3.5}$$

- $c(\tau, \alpha_G, i)$ is the cost of move for trace event e_i in an optimal alignment[6] between trace τ and model α_G;

[6] 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 [1, 126].

- 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 3.3 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 3.1, all events in the trace are matched by model $\alpha_{\mathtt{A}}$, that is, net $N_{\mathtt{A}}$ from Figure 3.5. 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'	$\nwarrow$	$\gg$	$\Uparrow$	$\Uparrow$	$\Leftarrow$	$\Leftarrow$	$\gg$	$\nwarrow$	$\Uparrow$	$\nwarrow$
	$\alpha_{\mathtt{A}}$	$\nwarrow$		$\Uparrow$	$\Uparrow$	$\Leftarrow$	$\Leftarrow$		$\nwarrow$	$\Uparrow$	$\nwarrow$
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 3.2, the first five events in the trace are matched by model α_F, that is, net N_F from Figure 3.6. However, the sixth and the seventh events result in asynchronous 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.

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

$\gamma_2 =$	τ''	⇑	↗	↗	↗	↗	≫	⇒	⇑
	α_F	⇑	↗	↗	↗	↗	↗	≫	≫
i^δ		1^1	2^1	3^1	4^1	5^1		6^1	7^1
$c(\tau'', \alpha_F, i)$		0	0	0	0	0		1	1

In γ_3 from Table 3.3, the first seven events in the trace are not matched by model α_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.

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

$\gamma_3 =$	τ'''	≫	≫	≫	≫	≫	⇑	↗	↗	↖	⇐	⇐	↙	⇐	≫	↖	⇑	↖
	α_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
$c(\tau''', \alpha_A, i)$							1	1	1	1	1	1	1	0		0	0	0

Finally, in alignment γ_4 from Table 3.4, 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.

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

$\gamma_4 =$	τ'''	⇑	↗	↗	≫	≫	≫	↖	⇐	⇐	↙	⇐	↖	⇑	↖
	α_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
$c(\tau''', \alpha_F, i)$		0	0	0	0	0	0	1	1	1	1	1	1	1	1

Our GR system uses Equation 3.3 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, Equation 3.3 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.

3.5 Evaluation

In this section, we present our experimental setup (Section 3.5.1), and then we use sensitivity analysis to verify whether all the parameters of our GR system impact its performance (Section 3.5.2). Subsequently, we illustrate how to configure the parameters in our GR system for better performance (Section 3.5.3). Following this, we investigate whether the choice of a particular collection of traces for learning skill models impacts goal recognition

time and accuracy (Section 3.5.4). Next, we compare our approach with other methods and explore how the performance of our GR system compares with that of other state-of-the-art GR systems (Section 3.5.5). Finally, we assess the applicability of our GR system in real-world scenarios (Section 3.5.6).

3.5.1 Experimental Setup

This section presents the experimental setup, including the datasets used in the experiments,[7] the implementation of the PM-based GR system, and the quality measures used to assess the performance of all GR systems participating in the experiments.

Synthetic datasets

We evaluated the performance of our GR system over 15 synthetic datasets, or *domains*, summarized by Pereira et al. [96]. Other techniques also use these domains for testing GR performance [102, 95], 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 towards the true goal (an *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 towards 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 [62] and the diverse planner [61]. The top-k planner generates a set of k different *cost-optimal* traces towards a given goal. Such cost-optimal traces simulate possible rational behaviors of an agent towards 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 [37, 25]. 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 Equation 3.6. 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 Equation 3.6 represent the sets of actions in plan π and plan π', respectively.

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

$$stability(\pi, \pi') = \frac{|\mathcal{A}(\pi) \cap \mathcal{A}(\pi')|}{|\mathcal{A}(\pi) \cup \mathcal{A}(\pi')|} \tag{3.6}$$

A planner does *not* guarantee that it can generate an arbitrary number of distinct traces towards 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 towards 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 towards 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.1. The excluded instances were not used in our experiments.

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. [118], 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). 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. [118]. The statistics of these binary-choice datasets are shown in Table 3.5. 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, namely Activities of Daily Living, Building Permit Applications, and Environmental Permit Applications. The statistics of these multi-class datasets are shown in Table 3.6. 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

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 3.5 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.

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 (10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1). It contains event logs that record the processes of build permit applications handled by five Dutch municipalities. Instead of classifying the traces in each event log into two sub-groups, 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 3.6 Statistics of the real-world datasets for multi-class problems.

Implementation

Our GR system was implemented and is available as part of an open-source simulation tool[8] 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.

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.

$$f_{precision} = \frac{TP}{TP + FP} \tag{3.7}$$

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

$$f_{recall} = \frac{TP}{TP + FN} \tag{3.8}$$

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

$$f_{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.9}$$

Suppose our GR system observes an agent that executes a sequence of actions. The agent works towards 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

[8] https://doi.org/10.26188/23807415

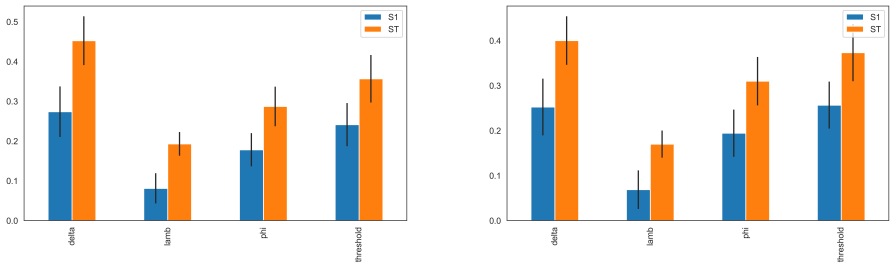
such, *TN* equals eight because our GR system made the correct decision not to infer these goals. Finally, *FN* is equal to zero because none of the true hidden goals are missed (our GR system correctly recognized the true hidden goal). Hence, in this example scenario, precision, recall, and accuracy 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.

3.5.2 Variance-Based Sensitivity Analysis

A sensitivity analysis explores how much each input (independent) variable contributes to the variance of a target (dependent) variable [106]. 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 3.5.1 into a publicly available sensitivity analysis tool called "Exploratory Modelling and Analysis (EMA) Workbench" [67]. We subsequently conducted the sensitivity analysis using the Sobol variance decomposition method [113], 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 [106]. 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 [106]. 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. [143], 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 3.5.1. 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 Figure 3.8. 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.



(a) (b)

Figure 3.8 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 A.2. 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.

3.5.3 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 [53]. It is desirable to

have N simulations (parameter configurations), where N is much larger than the number of parameters [106] (our GR system has four parameters). We followed the example from [16] 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 [40, 16]. 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 3.9 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.

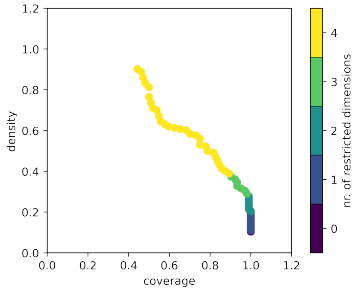


Figure 3.9 Peeling trajectory for the *driverlog* domain (trained by the cost-optimal traces).

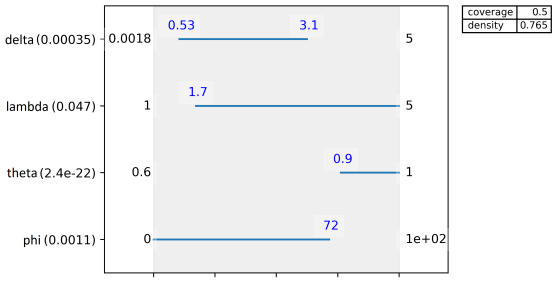


Figure 3.10 Visualization of the parameter ranges recommended by the PRIM algorithm.

Each region on the peeling trajectory is characterized by four attributes corresponding to the parameter ranges ϕ, λ, δ, and θ. Figure 3.10 shows these parameter ranges for a selected region from the peeling trajectory in Figure 3.9 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 Figure 3.10, 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 3.10 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 3.7 and 3.8. 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

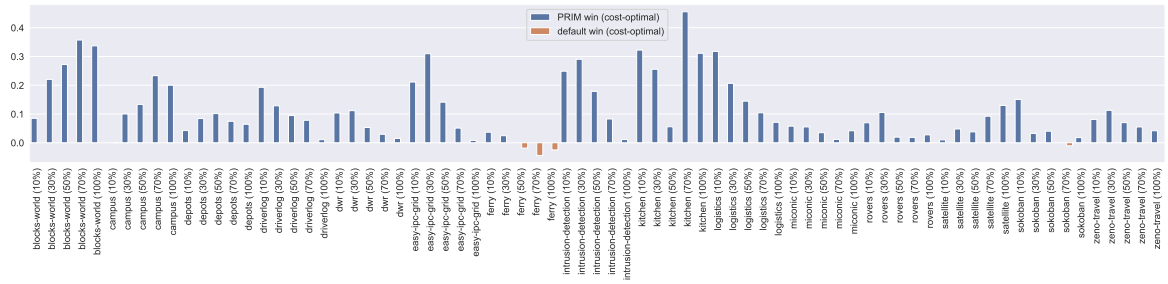
Table 3.7 The recommended ranges for parameters of the GR system (cost-optimal traces). $*$: qp-value ≤ 0.05; $**$: qp-value ≤ 0.001.

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

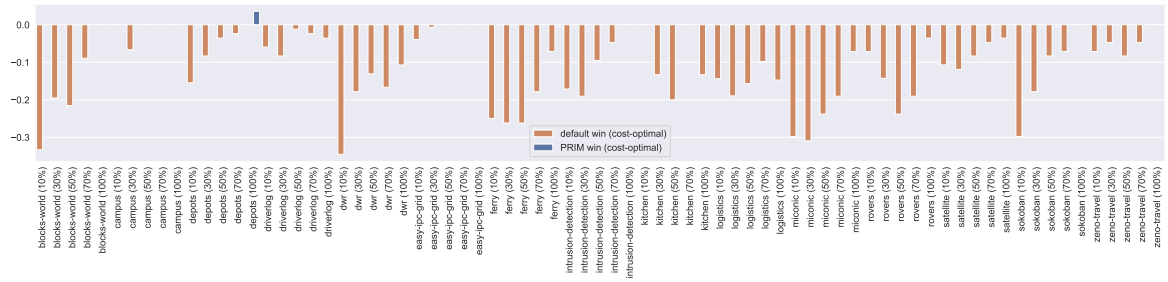
Table 3.8 The recommended ranges for parameters of the GR system (divergent traces). $*$: qp-value ≤ 0.05; $**$: qp-value ≤ 0.001.

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 Figure 3.10, 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 ($\phi = 50$, $\lambda = 1.1$, $\delta = 1.0$, $\theta = 0.8$). The results of precision, recall, accuracy, and execution time for each domain on each level of observation are listed in Appendix A.3.

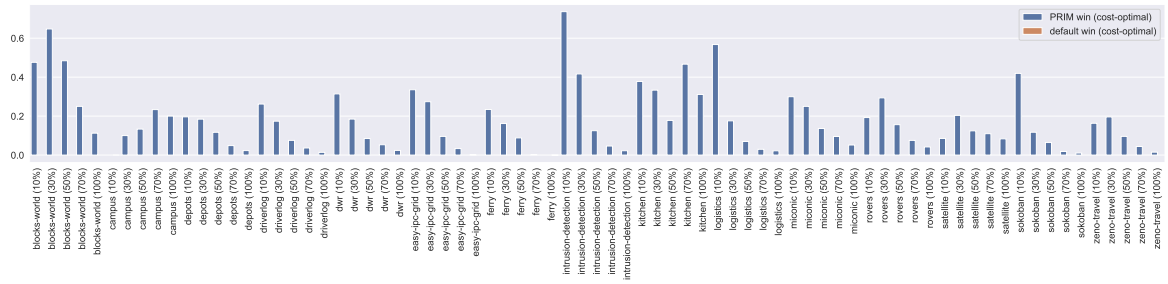
We use the precision, recall, accuracy, and execution time from the GR system configured by the PRIM parameters and subtract the corresponding values from the GR system configured by the default parameters. Figures 3.11 and 3.12 show the performance differences between the GR system configured with the 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

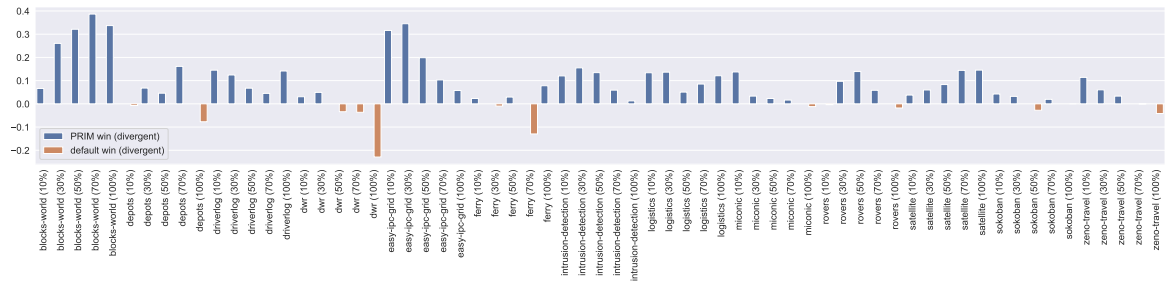


(b) Recall

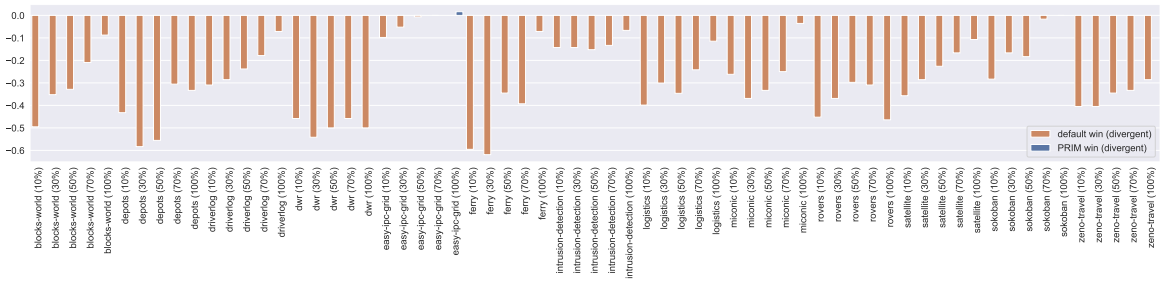


(c) Accuracy

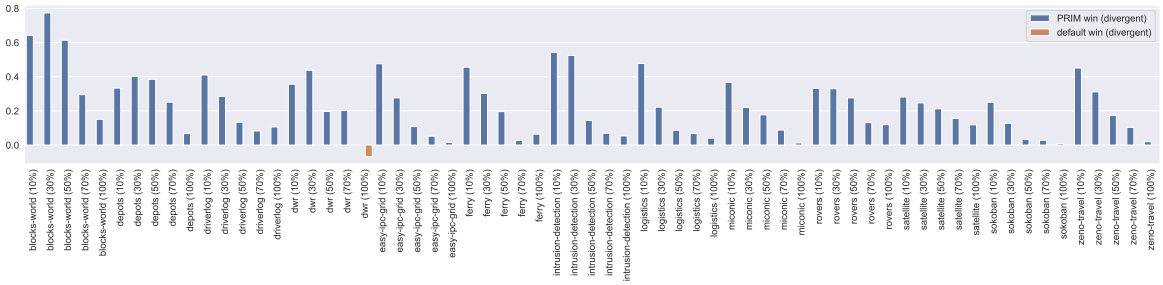
Figure 3.11 The performance differences between the GR system configured with the PRIM parameters and that configured with the default parameters (both systems are trained by the cost-optimal traces).



(a) Precision



(b) Recall



(c) Accuracy

Figure 3.12 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).

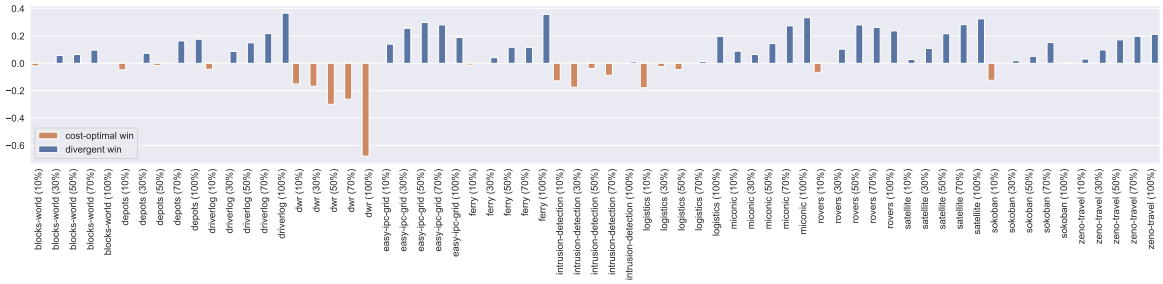
The comparison results show that the PRIM parameters are likely to yield high precision and accuracy of our GR system, while the default parameters yield high recall. A GR system with higher precision and accuracy is more convincing to be a good system than that with higher recall, as one can always 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.

3.5.4 Impacts of Training Traces

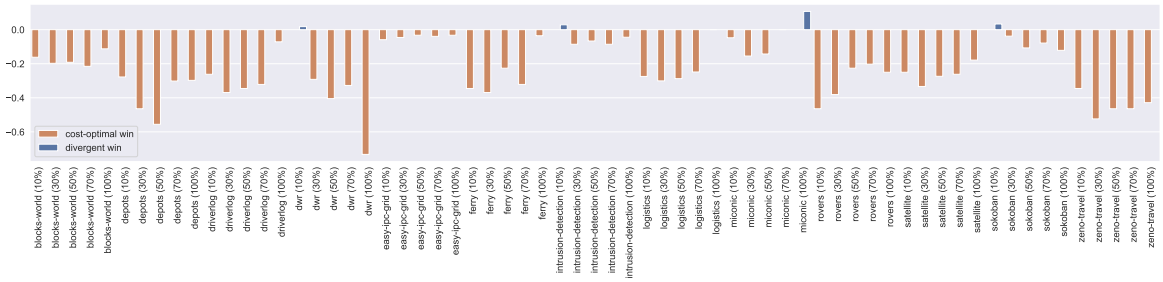
In our experiments, the traces accepted as historical observations of agents are generated by planners, namely the top-k planner and the diverse planner, see Section 3.5.1, which may impact learning the skill models and, consequently, the GR performance. We compared the GR systems trained by different 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 A.3.

The results of comparing GR performance on different sets of training traces are shown in Figure 3.13. The blue bars represent the GR system trained by divergent traces win over

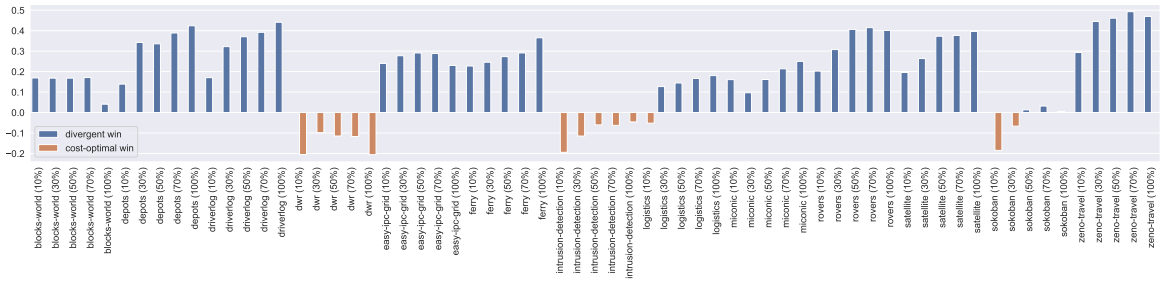
those trained by the cost-optimal traces. The orange bars represent the cost-optimal traces win. The height of the bars represents how much one is better than the other.



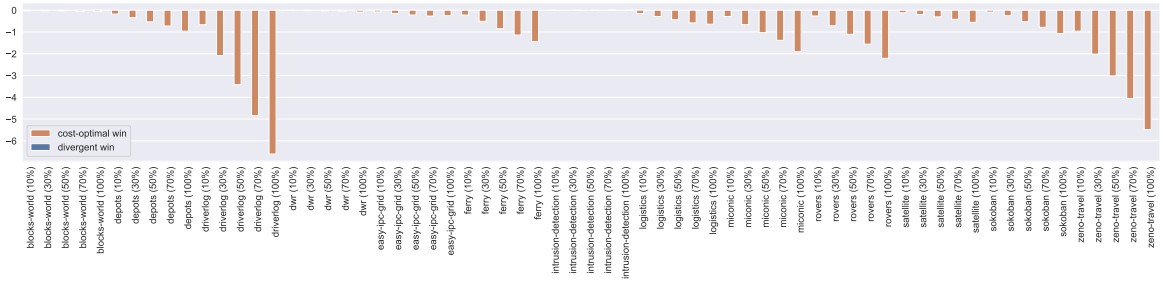
(a) Precision



(b) Recall



(c) Accuracy



(d) Time (in seconds)

Figure 3.13 The differences of the GR performance between the system trained by the cost-optimal traces and the divergent traces.

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 3.9 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 3.9 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.

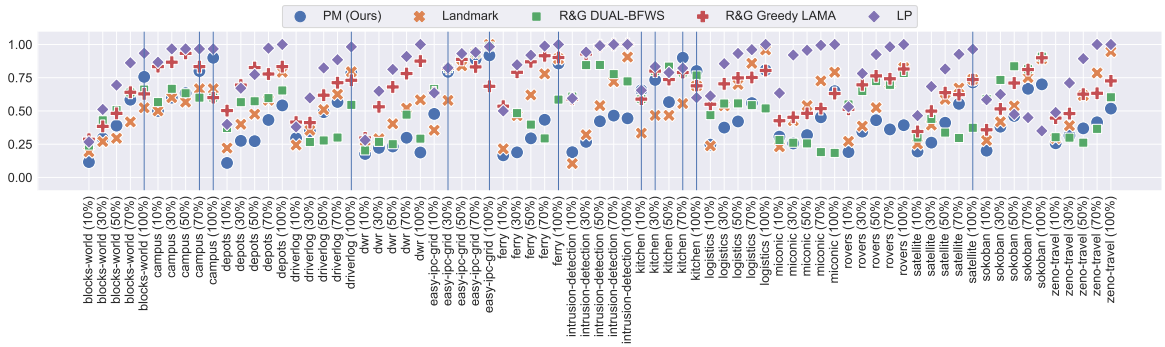
3.5.5 Comparison with Other Approaches

We compared our PM-based GR approach with state-of-the-art GR approaches. Section 3.5.5 presents the comparison between our approach and domain knowledge-based GR techniques: Ramirez and Geffner's approach (R&G) [102], the landmark-based approach [96], and the LP-based GR approach [107]. Section 3.5.5 shows the comparison between our approach and the long short-term memory networks (LSTM-)based GR approach [88]. For the R&G approach, we experimented with two embedded planners: the Greedy LAMA planner [105] and the state-of-the-art planner from the international planning competition, DUAL-BFWS [79]. 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

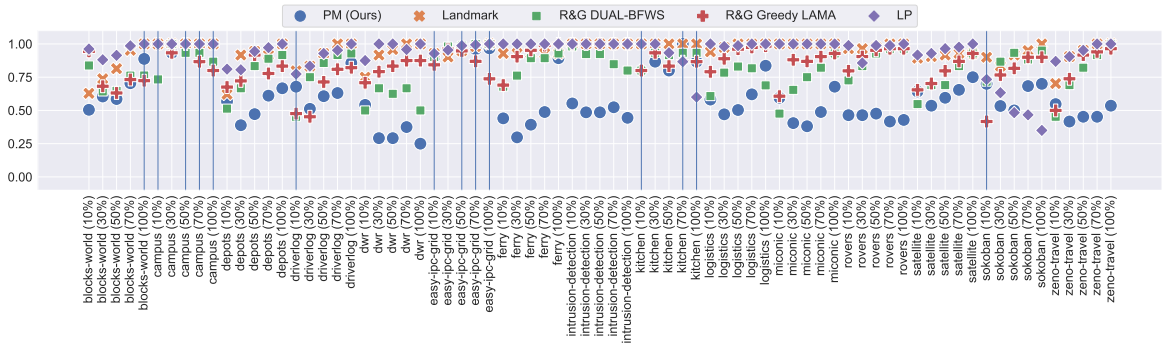
landmarks, *state equation*, and *post-hoc*. Our GR system and the LSTM-based approach are trained using the divergent traces for all except Campus and Kitchen. Note that the diverse planner failed to generate traces for these two domains (cf. Section 3.5.1). Therefore, we used the cost-optimal traces to train our GR system and the LSTM-based approach for these domains. We configured our GR system using the PRIM parameters (cf. Section 3.5.3).

Comparison with Domain Knowledge-Based GR Approaches

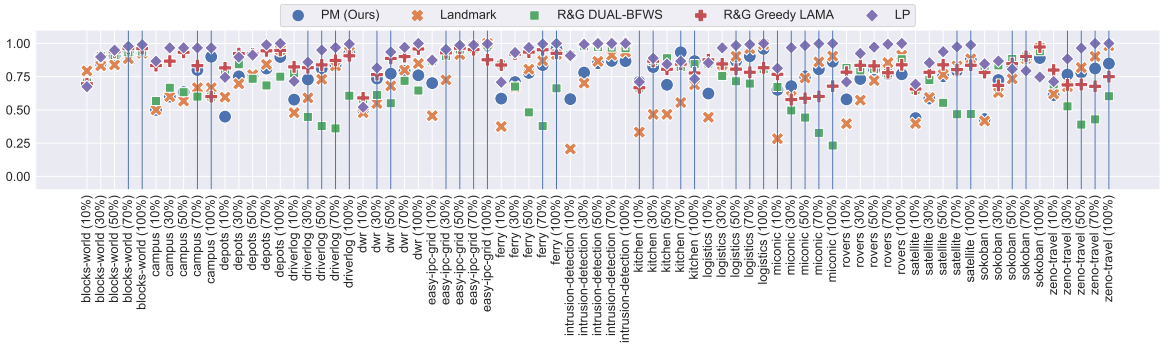
The GR performance for domain knowledge-based approaches is listed in Appendix A.4. Figure 3.14 plots the precision, recall, accuracy, and time for different GR approaches in each domain for each level of observation. The blue dots represent the GR performance of our approach. We calculated the average 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.



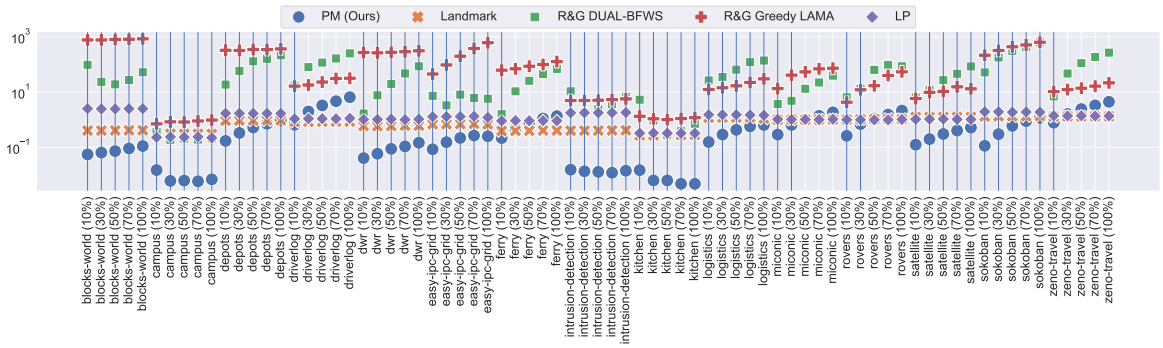
(a) Precision



(b) Recall



(c) Accuracy



(d) Time (in seconds)

Figure 3.14 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 Figure 3.14, 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 3.1, the skill model for achieving goal `A` is learned from six rational traces (see Figure 3.3a). If the first seven steps of the red irrational walk (in Figure 3.1a) 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 Figure 3.3a. 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 3.10 The average ranks of performance (avg) for each GR approach and the standard deviations (std) of the ranks.

Table 3.10 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 [85, 86]. 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 does 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, which are optimized 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.

Comparison with LSTM-based GR Approach

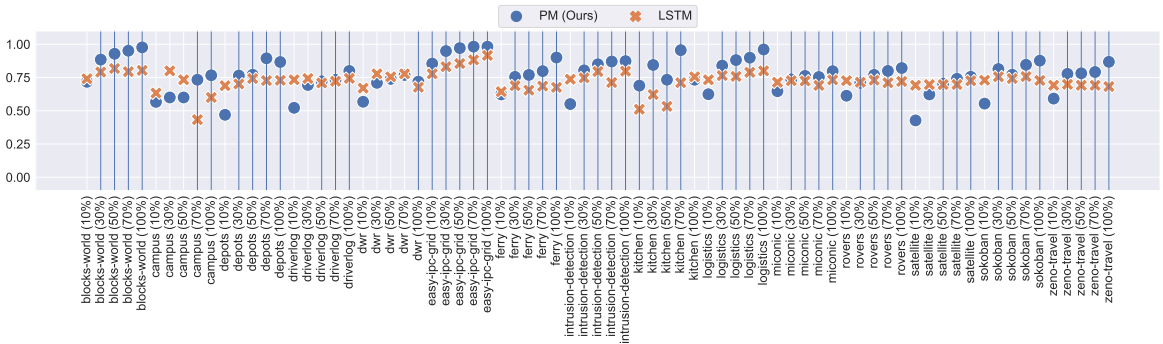
The implementation of the LSTM-based GR approach uses the configuration recommended by Min et al. [88]. The LSTM model, with a dropout rate of 0.75, comprises three layers: an embedding layer that converts actions to a 20-dimensional vector space, a layer with 100 self-connected memory cells (units), 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. [96]. The detailed GR performance results for the PM- and LSTM-based approaches trained with small and large datasets are listed in Table A.5 included in Appendix A.5. For three performance metrics of precision, recall, and accuracy, Table 3.11 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)

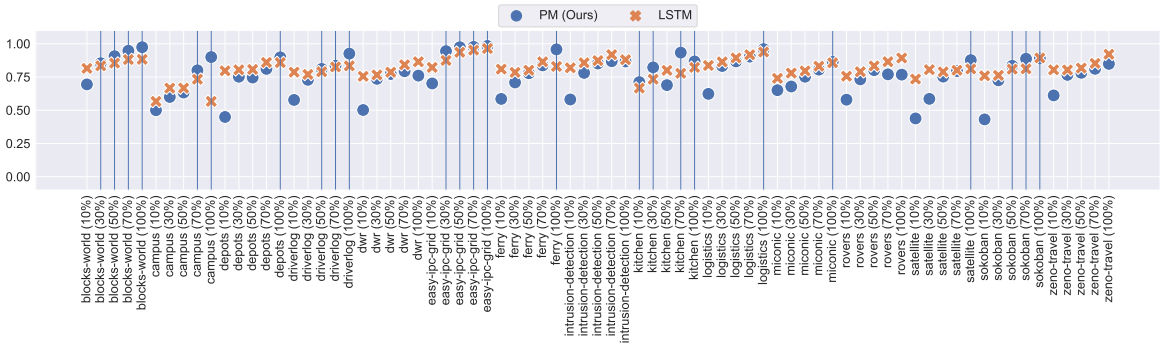
Table 3.11 The percentage (number) of cases out of 75 cases in which the PM-based system outperformed the LSTM-based system.

Figure 3.15 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 A.5, Figure A.3 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 3.5.1). 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.



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



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

Figure 3.15 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.

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 goal; refer to the examples discussed in Section 3.3. In addition, the learned Petri nets explain the behavior of the agents towards 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.

3.5.6 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 3.5.1, 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 3.12. As mentioned in Section 3.5.1, 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 3.12, 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.

The performance of the multi-class real-world GR problems is shown in Table 3.13. 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

	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 3.12 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.

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 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.

	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 3.13 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.

We acknowledge the limitations in the evaluation metrics, particularly the accuracy metric. Notably, the accuracy metric tends to yield high True Negative scores in scenarios where there is only one true goal but multiple goal candidates, potentially leading to an inflated accuracy score. Nevertheless, we use the same metric for consistency when evaluating and comparing different approaches to ensure a fair comparison. In the subsequent chapters, we address this bias by using balanced accuracy as our primary evaluation metric.

Chapter 4

Adaptive Goal Recognition with Process Mining Techniques

The PM-based GR framework described in Chapter 3 is designed to address the "single-shot" GR problem (also referred to as the conventional GR problem). That approach concentrates on observing a single sequence of actions and then identifying the agent's intended goal based on that action sequence. However, some real-world scenarios require addressing multiple GR problems over an extended period, during which the environment may change. When such environmental changes occur, the behavior of agents aiming to achieve their goals generally adapts [112]. For example, in navigation, the emergence of new obstacles or shortcuts may lead intelligent agents to choose alternative pathways. These scenarios, characterized by changes in the underlying environment, are known as *concept drift*. Such drifts are commonly observed in business processes when the behavior of process participants changes to address new regulations, compliance rules, and innovative business practices [140]. In all these scenarios, we expect a GR system to first recognize changes in observed behavior and then adapt accordingly to maintain its recognition performance. We refer to the problem of solving multiple GR problems over a time interval during which the environment in which the agent operates may change as the *adaptive GR* problem. This chapter formally introduces and defines the challenge. It also presents an adaptive GR framework based on process mining (PM) techniques and outlines three adaptive GR systems that extend from the established framework.[1]

[1]This chapter is an adaptation of two of our previously published works: (1) "GRACE: A simulator for continuous goal recognition over changing environments." In PMAI@IJCAI, volume 3310 of CEUR Workshop Proceedings, pages 37-48. CEUR-WS.org, 2022. (2) "Adaptive goal recognition using process mining techniques." Engineering Applications of Artificial Intelligence, 133:108189, 2024. ISSN 0952-1976.

In this chapter, we address *RQ2: How to do goal recognition in non-stationary environments?* Our objective is to apply the goal recognition (GR) technique in scenarios where the GR system is required to continuously observe agents' action sequences. However, the underlying environment may change during the observation period, potentially influencing the agents' behaviors in achieving their goals. Throughout this period, it is assumed that agents execute multiple action sequences over several GR episodes, with environmental changes occurring exclusively between these GR episodes. We develop an experimental data generator named the Goal Recognition Amidst Changing Environments (GRACE) tool, capable of synthesizing problem instances. Each problem instance is a concept drift containing multiple action sequences, ordered by the time each sequence occurs. Finally, we propose an adaptive GR framework designed to continuously solve GR tasks in periods where the underlying environment is non-stationary. We instantiated three adaptive GR systems from the proposed adaptive GR framework and evaluated them using both synthetic and real-world domains. This evaluation results confirm that the adaptive GR systems indeed improve average GR performance compared to conventional GR systems, which cannot adapt to environmental changes.

Concretely, this chapter makes the following contributions:

- It formally defines the adaptive GR problem, which can be seen as an optimization problem that balances the cost of adapting knowledge models with the accuracy of GR;
- It introduces GRACE, a tool designed to simulate significant environmental changes affecting agents' behaviors and generate publicly available drift behavior problem instances[2] for evaluating adaptive GR systems;
- It introduces an adaptive GR framework as a control mechanism over a conventional GR system grounded in process mining techniques, and presents three concrete adaptive GR systems instantiated from this framework;
- It uses synthetic and real-world datasets to evaluate the three instantiated adaptive GR systems. The evaluation results confirm the effectiveness of these GR systems for solving adaptive GR problems.

[2]https://doi.org/10.26188/21802081

4.1 Adaptive Goal Recognition

An *adaptive* algorithm aims to improve its performance based on the inputs and changing conditions. Such an algorithm is useful, for instance, when the problem it solves is defined over a dynamic or uncertain environment. We define the adaptive GR problem based on the conventional probabilistic GR problem, as outlined in chapter 3, Definition 1. In the conventional probabilistic GR problem, $\mathcal{G}$ denotes a set of candidate goals, $\mathcal{F}$ denotes a set of fluents, $\mathcal{A}$ denotes a set of actions, I denotes the initial state, and $\mathcal{O}$ denotes an observation, which is a sequence of actions performed by an agent. $\mathcal{F}$, $\mathcal{A}$, and I together represent the available knowledge, $\mathcal{K}$, about an agent and the environment. Therefore, given a triple $< \mathcal{G}, \mathcal{K}, O >$, the probabilistic GR problem involves obtaining a posterior probability distribution over $\mathcal{G}$ based on $\mathcal{K}$ and O that describes the likelihood of the agent achieving the different goals.
Different instantiations of this definition of the GR problem have been proposed in the literature. They differ in how the elements of the problem and its solution are represented. In the approach by Ramírez and Geffner [102], the knowledge is represented by a domain theory and the information on when the agent's actions are applicable in the environment and how they transform it, while in our proposed PM-based GR framework (Chapter 3), the knowledge is given as collections of historical action sequences towards goals. The observations usually capture a time-ordered sequence of *actions* performed by the agent thus far. Finally, the inferred goals can be identified by a set $G \subseteq \mathcal{G}$, for instance, as a subset of the most likely goals of the agent.

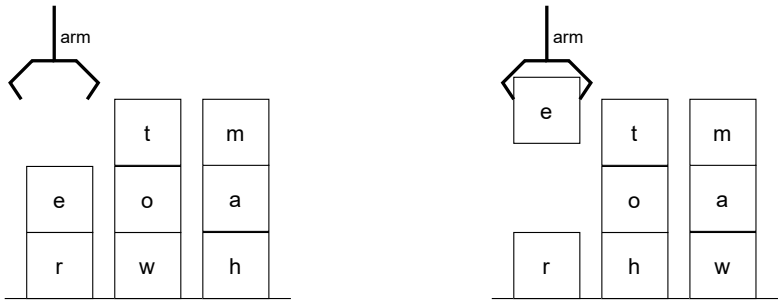
Let $\mathcal{P}$, $\mathcal{S}$, and $\mathcal{T}$ denote the universes of all GR problems, solutions to these problems (for instance, probability distributions over candidate goals), and (true) goals, respectively. A goal recognition algorithm α maps problems to solutions, that is, $\alpha : \mathcal{P} \to \mathcal{S}$, where $\alpha(p)$ is the solution to problem $p \in \mathcal{P}$ by algorithm α. Given a solution $\alpha(p)$ to problem p and the *true* goal $g \in \mathcal{T}$ of the agent from p, one can assess the quality of the solution using standard measures such as precision, recall, and accuracy. Given a sequence X of length n, by $X[i]$, $i \in [0..n]$, we denote the prefix of X of length i. By X_i, we denote the element of sequence X at position i.

Then, the adaptive GR problem is defined as follows.

Definition 6 (Adaptive goal recognition)**.** *A triple $< P, S, T >$, where $P \in \mathcal{P}^*$, $S \in \mathcal{S}^*$, $T \in \mathcal{T}^*$, $|P| = n$, and $|S| = n - 1 = |T|$, is an* adaptive goal recognition *problem of size n that consists in solving GR problem P_n if either (i) $n = 1$ or (ii) $n > 1$, S_i is a solution to problem P_i, T_i is the true goal of the agent from problem P_i, $i \in [1..n-1]$, and $< P[n-1], S[n-2], T[n-2] >$ is an adaptive GR problem of size $n - 1$.*

The adaptive GR problem reduces to the conventional GR problem when $n = 1$ and is a generalization of the latter. The adaptive GR problem can also be captured as a special GR problem with the candidate goals and observations of P_n and the knowledge given by P, S, and T. However, the explicit inductive structure of Definition 6 suggests that each subsequent goal recognition can happen under new conditions and, thus, it is possible that $P_i = P_j$ but $S_i \neq S_j$, or $S_i = S_j$ but $P_i \neq P_j$. The former situation can occur when a solution to the same problem learns from its previous solution and adapts it, aiming to improve the quality of the new solution. The latter situation may arise when different knowledge of the environment or observations of the agent confirm the same inferred goal.

An algorithm for solving adaptive GR problems can use available information on the quality of previous GR solutions to improve its performance. Interestingly, we do not require all the solutions in S to stem from the same algorithm. However, in practice, one can often expect that S_{n-1} is obtained by the same algorithm that is used to solve P_n when solving the adaptive problem of size $n-1$ defined by the prefixes of P, S, and T. Finally, the performance of a GR algorithm that tackles an adaptive GR problem can be assessed over time, that is, over all the induced problems of sizes from one to n.



(a) Environment 0 (b) Environment 1

Figure 4.1 Two environments in the blocks-world domain.

Figure 4.1 shows two environments in the blocks-world domain. An environment in this domain consists of a set of blocks, each either lying on the table or stacked on top of another block, and an arm that can either be empty or hold a single block. For example, in Figure 4.1a, the arm is empty, blocks *r*, *w*, and *h* are on the table, block *e* is stacked on top of block *r*, block *o* is stacked on top of block *w*, block *t* is stacked on top of block *o*, block *a* is stacked on top of block *h*, and block *m* is stacked on top of block *a*. When empty, the arm, controlled by an agent, can pick up single block x from the table, using action `(pick-up` x`)`, or unstack x from block y, using action `(unstack` x y`)`, if no other blocks are stacked on top of block x. After picking up a block, the arm is no longer empty, as it now holds the block, and cannot pick up other blocks. The arm can put block x it is holding back onto the table, using action `(put-down` x`)`, or stack it on top of block y, using action `(stack` x y`)`.

Let the environment in Figure 4.1a be the initial environment from which an agent starts to control the arm and attempts to achieve one of two candidate goals: G_1 or G_2. Goal G_1 is to build the blocks that form the word "tower" where there are no blocks on *t*, block *r* is on the table, *t* stacks on *o*, *o* stacks on *w*, *w* stacks on *e*, and *e* stacks on *r*. Goal G_2 is to build the blocks and form the word "mother" where there are no blocks on *m*, block *r* is on the table, *m* stacks on *o*, *o* stacks on *t*, *t* stacks on *h*, *h* stacks on *e*, and *e* stacks on *r*. The actions of the agent are the ability to use the arm to pick up and put down blocks. The knowledge about the environment and agent determines which blocks the arm can pick up and where it can put the block it holds.

One can capture the described situation as the probabilistic goal recognition problem $< \{G_1, G_2\}, \mathcal{K}, O >$, where $\mathcal{K}$ describes the environment in Figure 4.1a and the actions the agent can perform in different circumstances, and O is an observation of the agent's historical actions. For example, one can use the sequence $\langle$`(unstack t o)`, `(put-down t)`, `(unstack o w)`, `(put-down o)`, `(pick-up w)`, `(stack w e)`$\rangle$ to specify that six consecutive actions performed by the agent were observed. A solution to this problem can state, for instance, that the agent aims for G_1 with the probability of 0.7 and G_2 with the probability of 0.3.

In practice, an agent may need to achieve the same goal repeatedly, for example, to practice or learn different operation strategies or to accomplish multiple instances of the same task. In general, the environment and the behavior of the agent may change over time. For example, the initial state of the environment in which the agent attempts to achieve the goals can change from the one shown in Figure 4.1a to the one depicted in Figure 4.1b. Other changes that may affect the agent's performance include changes in the candidate goals, changes in the knowledge about the principles that govern the evolution of the environment, and the evolution of the skills of the agent to operate in the environment. Such ongoing changes to the environment can be referred to as *concept drifts*. Figure 4.2 visualizes an example gradual drift of the initial state of the environment from environment 0 to environment 1 generated by the GRACE tool[3] that happens between time steps 50 and 100. In this type of drift, the initial state defined by environment 0 is observed up to and including time step 67. Subsequently, between time steps 67 and 88, the initial state gradually changes from environment 0 to environment 1, with environment 1 appearing more frequently over time until from time step 88 onward the initial state is entirely defined by environment 1.

A data-driven GR algorithm, like the PM-based GR system described in Chapter 3, often has a training stage. During training, such an algorithm can use historical observations and feedback from existing GR solutions to learn how to carry out future goal recognition

[3]https://github.com/zihangs/GRACE

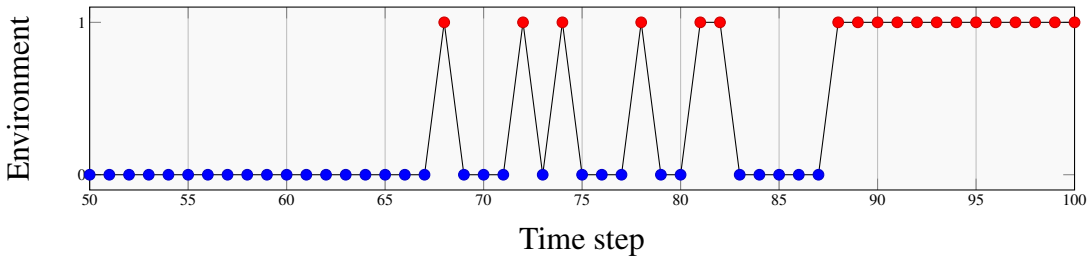


Figure 4.2 An example gradual drift from environment 0 to environment 1 generated by GRACE.

practices best. Such training is often associated with a cost, for instance, computational resources and time spent for training. While there is a desire to minimize these costs, frequent training may be necessary when the environment, goals, or the agent's behavior drifts. These conflicting aims call for exploring different strategies to minimize training costs while maximizing GR quality over time.

Given two sequences X and Y, $X \circ Y$ denotes their concatenation. Let $P \in \mathcal{P}^*$ be a non-empty sequence of GR problems of length n. By $P(1)$, we denote the adaptive GR problem $< P[1], \langle\rangle, \langle\rangle >$. Then, by $P(i)$, $i \in [2..n]$, we denote the adaptive GR problem $< P[i], S, T >$, where $S = S' \circ \langle s \rangle$ and $T = T' \circ \langle g \rangle$, such that S' and T' are, respectively, the solutions and goals of $P(i-1)$ and s and g are, respectively, the solution of problem $P(i-1)$ and the true goal of the agent from problem $P(i-1)$. Then, an *adaptive GR strategy* is a solution to the optimization problem that maximizes the quality of solutions to GR problems while minimizing the cost invested in training the GR algorithm to obtain the solutions.

Definition 7 (Adaptive GR strategy)**.** *Given a GR algorithm $\alpha : \mathcal{P} \to \mathcal{S}$ and a non-empty sequence of GR problems $P \in \mathcal{P}^*$ of length n, an* adaptive GR strategy *is a set of positions I in P, such that for each position $i \in I$ algorithm α is trained on the solutions and true goals of $P(i)$ before it is applied to solve problems $\{P(k) \mid k \in [i..n]\}$, aiming to maximize an objective function that praises the GR solutions of high quality and condemns training of α of high costs.*

To exemplify the trade-off between the performance of GR inference and the number of times the system is trained, Figure 4.3 compares the performance of the conventional PM-based GR system and the same system controlled by the open-loop adaptive strategy, which regularly triggers the system to relearn its knowledge models based on which it carries out the inference. The inference is made for the two goals of building words "tower" and "mother" starting from one of the two environments shown in Figure 4.1. The initial state of the environment for each time step is determined by the drift depicted in Figure 4.2. In the figure, green boxes denote the accuracy of the adaptive GR system that relearns every ten time steps using the data collected from the ten most recent inferences. Vertical blue dashed

lines in the figure denote the time steps when the system is retrained. In contrast, red crosses denote the accuracy of the conventional GR system that relies on the same knowledge models throughout the entire inference period. The adaptive GR system achieves an average accuracy of 0.90 over the period between time steps 50 and 100 while being trained four times, while the conventional GR system achieves an average accuracy of 0.75 without updates of its knowledge models, showing the value of the adaptive strategy.

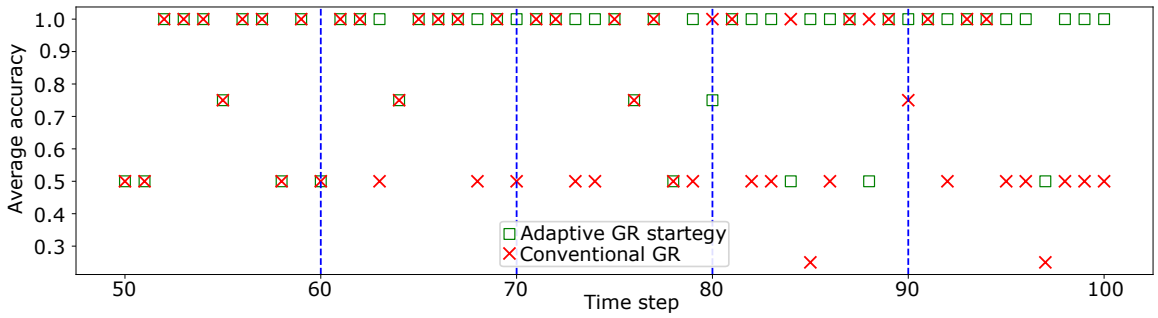
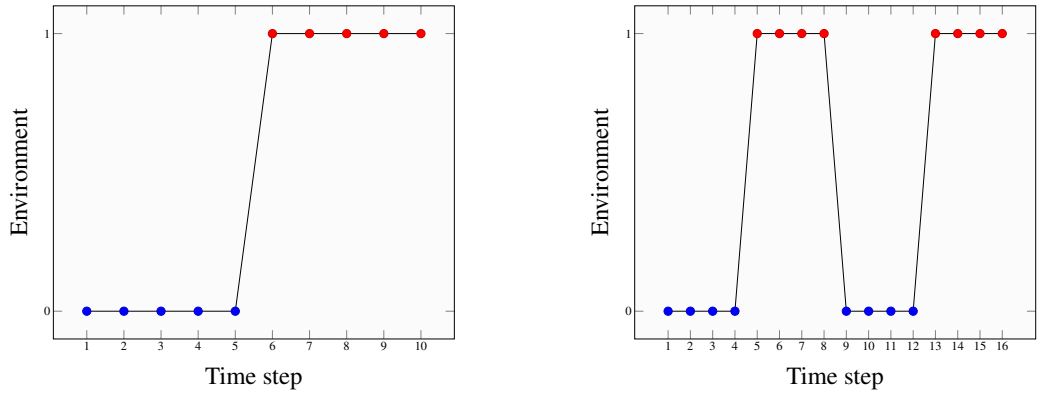


Figure 4.3 Accuracy of a conventional GR system and a GR system that implements an open-loop adaptive strategy.

Each time step in Figures 4.2 and 4.3 defines an adaptive GR problem, cf. Definition 6, and reports the quality of its solution. Specifically, for a time step i, the corresponding adaptive GR problem is defined by $< P^i, S^i, T^i >$, where P^i is the sequence of all GR problems requested to be solved up to, and including, time step i, S^i is the sequence of all solutions to the GR problems solved up to, but excluding, time step i, and T^i is the sequence of all true goals the agent achieved in the past up to, but excluding, time step i. The accuracy of the solutions in S^i with respect to the true goals from T^i is reported in Figure 4.3. The GR problem P^i_j can be captured as triplet $< \{G_1, G_2\}, \mathcal{K}_j, O_j >$, where $\mathcal{K}_j$ is the knowledge model defined by the environment at time step j in Figure 4.2 and O_j are the available observations of the agent in that environment. Besides the gradual drift, the GRACE tool can also simulate other types of concept drifts such as sudden drift and reoccurring drift. Sudden drift, see Figure 4.4a, concerns the situation when the original environment (*env0*) changes to the new environment (*env1*) over a short period. Reoccurring drift, see Figure 4.4b, concerns the situation when two distinct environments *env0* and *env1* repeatedly alternate, where each environment is observed for some period. Note that reoccurring drift is different from gradual drift as there is no incremental transition from *env0* and *env1* and the transitions between the environments happen abruptly.



(a) Sudden (b) Reoccurring

Figure 4.4 Other types of concept drift.

4.2 Adaptive Goal Recognition Architecture

The adaptive GR system extends the conventional PM-based GR system by incorporating additional control mechanisms governed by adaptive GR strategies. These mechanisms trigger the relearning of process models, which is crucial for adapting the system to handle new GR problems when the environment changes. As presented in Chapter 3, Figure 3.7 illustrates the three steps in a PM-based GR system for solving single-shot GR problems: process discovery, conformance checking, and probability calculation. The architecture of the adaptive version of the PM-based GR system consists of a conventional PM-based GR system and two additional steps, *feedback collection* and *relearn decision*, as shown in Figure 4.5.

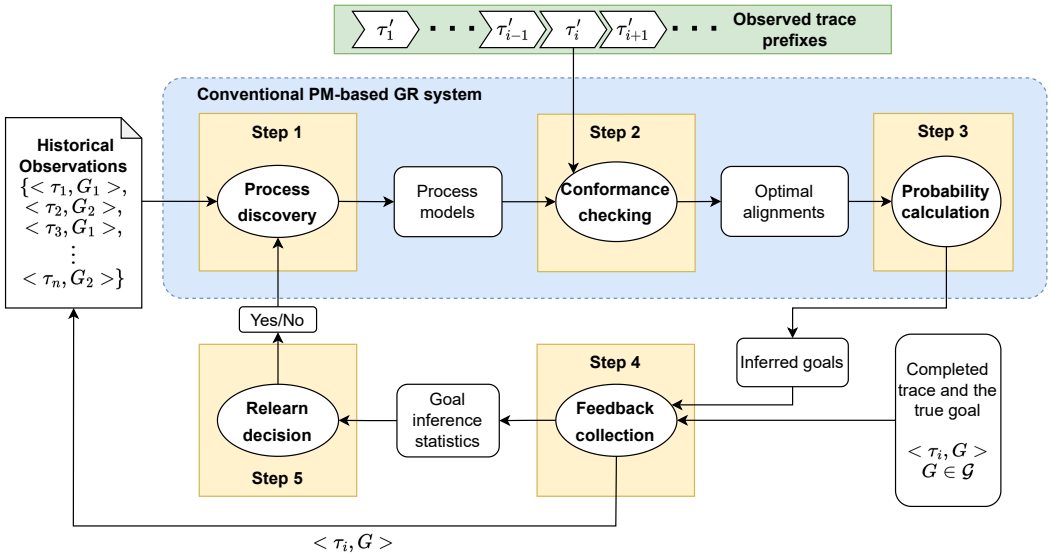


Figure 4.5 Architecture of the adaptive PM-based GR system.

We use the example from Figure 4.1, which includes a transition from the original environment (environment 0) to environment 1, to describe all five steps of the adaptive PM-based GR system below.

Step 1: Starting from the initial state shown in Figure 4.1a, the agent controls the arm to achieve one of the two candidate goals: to form the word "tower" (G_1) and to form the word "mother" (G_2). Let us assume that we have already observed two distinct sets of traces (event logs), denoted by L_{G_1} and L_{G_2}, each containing five traces as presented in Tables 4.1 and 4.2.

τ_1	τ_2	τ_3	τ_4	τ_5
⟨(unstack t o), (put-down t), (unstack o w), (put-down o), (pick-up w), (stack w e), (pick-up o), (stack o w), (pick-up t), (stack t o)⟩	⟨(unstack t o), (stack t m), (unstack o w), (put-down o), (pick-up w), (stack w e), (pick-up o), (stack o w), (unstack t m), (stack t o)⟩	⟨(unstack t o), (put-down t), (unstack o w), (stack o m), (pick-up w), (stack w e), (unstack o m), (stack o w), (pick-up t), (stack t o)⟩	⟨(unstack t o), (put-down t), (unstack o w), (stack o t), (pick-up w), (stack w e), (unstack o t), (stack o w), (pick-up t), (stack t o)⟩	⟨(unstack t o), (stack t m), (unstack o w), (stack o t), (pick-up w), (stack w e), (unstack o t), (stack o w), (unstack t m), (stack t o)⟩

Table 4.1 Event log L_{G_1} comprising five traces τ_1 to τ_5 for achieving goal G_1.

τ_6	τ_7	τ_8	τ_9	τ_{10}
⟨(unstack m a), (put-down m), (unstack a h), (put-down a), (pick-up h), (stack h e), (unstack t o), (stack t h), (unstack o w), (stack o t), (pick-up m), (stack m o)⟩	⟨(unstack m a), (put-down m), (unstack a h), (put-down a), (pick-up h), (stack h e), (unstack t o), (stack t h), (unstack o w), (stack o t), (pick-up m), (stack m o), (pick-up a)⟩	⟨(unstack m a), (put-down m), (unstack a h), (put-down a), (pick-up h), (stack h e), (unstack t o), (stack t h), (unstack o w), (stack o t), (pick-up m), (stack m o), (pick-up w)⟩	⟨(unstack m a), (put-down m), (unstack a h), (put-down a), (pick-up a), (put-down a), (pick-up h), (stack h e), (unstack t o), (stack t h), (unstack o w), (stack o t), (pick-up m), (stack m o)⟩	⟨(unstack m a), (put-down m), (unstack a h), (stack a e), (unstack a e), (put-down a), (pick-up h), (stack h e), (unstack t o), (stack t h), (unstack o w), (stack o t), (pick-up m), (stack m o)⟩

Table 4.2 Event log L_{G_2} comprising five traces τ_6 to τ_{10} for achieving goal G_2.

The GR system can subsequently learn two process models from L_{G_1} and L_{G_2}, namely M_{G_1} and M_{G_2}, represented graphically as Petri nets in Figure 4.6 and Figure 4.7, respectively.[4]

[4]In Figure 4.6 and Figure 4.7, we abbreviate agent's actions as follows: `pu` = `pick-up`, `pd` = `put-down`, `st` = `stack`, `us` = `unstack`.

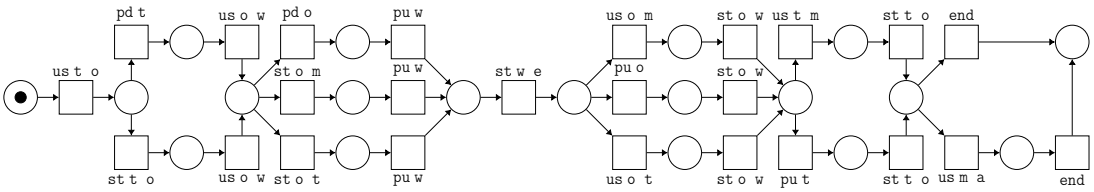


Figure 4.6 Petri net M_{G_1} learned from event log L_{G_1}.

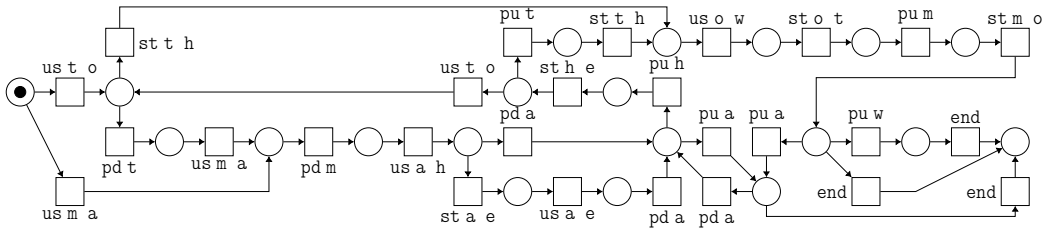


Figure 4.7 Petri net M_{G_2} learned from event log L_{G_2}.

Step 2: Table 4.3 shows two example optimal alignments, denoted by γ_1 and γ_2, between a partially observed trace τ_o and models M_{G_1} and M_{G_2}. Trace $\tau_o = \langle$(unstack t o), (put-down t), (unstack o w), (stack o m), (pick-up w), (stack w e), (unstack o m)$\rangle$ consists of seven actions, denoted by $a_1, \ldots, a_7$, where action indices correspond to their positions in the trace, which do not provide a complete sequence of actions from the initial state to the goal state. Let the ground truth be that the agent that performed τ_o aims to achieve goal G_1. Model M_{G_1} describes a trace that can match τ_o perfectly. However, the best-matched trace described by M_{G_2} can only align three actions, a_1, a_3, and a_5 with τ_o, while a_2, a_4, a_6, and a_7 are asynchronous moves denoted by the special skip symbol "≫". Intuitively, in the example in Table 4.3, the agent is more likely to work towards goal G_1. Next, the probability distribution over candidate goals G_1 to G_N is computed using optimal alignments γ_1 to γ_N that describes the likelihood of τ_o leading to the true goal.

		a_1	a_2	a_3	a_4	a_5	a_6	a_7
$\gamma_1 =$	τ_o	(unstack t o)	(put-down t)	(unstack o w)	(stack o m)	(pick-up w)	(stack w e)	(unstack o m)
	M_{G_1}	(unstack t o)	(put-down t)	(unstack o w)	(stack o m)	(pick-up w)	(stack w e)	(unstack o m)
i^δ		1^1	2^1	3^1	4^1	5^1	6^1	7^1
$c(\tau_o, M_{G_2}, i)$		0	0	0	0	0	0	0
$\gamma_2 =$	τ_o	(unstack t o)	(put-down t)	(unstack o w)	(stack o m)	(pick-up w)	(stack w e)	(unstack o m)
	M_{G_2}	(unstack t o)	>>	(unstack o w)	>>	(pick-up w)	>>	>>
i^δ		1^1	2^1	3^1	4^1	5^1	6^1	7^1
$c(\tau_o, M_{G_2}, i)$		0	1	0	1	0	1	1

Table 4.3 Optimal alignments between trace τ_o observed in environment 0 from Figure 4.1a and models M_{G_1} and M_{G_2}.

Step 3: The weight of the constructed optimal alignment between τ and model M_G is computed using Equation 3.5, as described in Chapter 3. The parameter values are configured

as $\phi = 50$, $\delta = 1.0$, $\lambda = 1.5$, and the cost of each asynchronous move on the trace is set to $c(\tau, M_G, i) = 1$. In Table 4.3, i^{δ} represents a discount factor, and $c(\tau, M_G, i)$ denotes the cost of the asynchronous move on the trace in position i in the alignment (for example actions a_2, a_4, a_6, and a_7 in γ_2). Thus, in the blocks-world example, the alignment weight $\omega(\tau_o, M_{G_1})$ is 50, since all the moves in γ_1 are synchronous and only the constant ϕ contributes to the weight. On the other hand, $\omega(\tau_o, M_{G_2}) = 50 + 1.5^2 \times (1 \times 2^1 + 1 \times 4^1 + 1 \times 6^1 + 1 \times 7^1) = 92.75$, since a_2, a_4, a_6, a_7 in γ_2 are asynchronous moves. When incorporating the alignment weights $\omega(\tau_o, M_{G_1})$ and $\omega(\tau_o, M_{G_2})$ into Equation 3.3, the probabilities of the agent achieving G_1 and G_2 are 0.69 and 0.31, respectively. The probabilities indicate that the observed trace τ_o is more likely to lead the agent to goal G_1, and, hence, the system infers G_1 as the goal of the agent.

The above three steps work perfectly in the original environment (environment 0) if the environment is static. However, if the environment changes from environment 0 to environment 1 (as shown in Figure 4.1), the agent will need to follow a significantly different trajectory to achieve the same goal. Assume that the new observed sequence of actions for the agent acting in environment 1 to achieve G_1 is $\tau'_i = \langle$ `(put-down e)`, `(unstack m a)`, `(put-down m)`, `(unstack t o)`, `(stack t m)`, `(unstack a w)`, `(put-down a)` $\rangle$. If we align τ'_i to the models learned for the environment 0, M_{G_1} and M_{G_2}, we obtain optimal alignments γ_3 and γ_4, as shown in Table 4.4. The alignment weights, according to Equation 3.5, are $\omega(\tau'_i, M_{G_1}) = 92.75$ and $\omega(\tau'_i, M_{G_2}) = 66$, respectively. According to Equation 3.3, the inferred probabilities of the agent to achieve G_1 and G_2 are 0.40 and 0.60, respectively. Consequently, if the GR system relies on the models learned for environment 0, the newly inferred probabilities indicate that the agent executing τ'_i is more likely working towards G_2, which is incorrect given that the true goal of the agent is G_1. This running example illustrates the limitations of the conventional GR system when the underlying environment changes.

		a_1	a_2	a_3	a_4	a_5	a_6	a_7
$\gamma_3 =$	τ'_i	`(put-down e)`	`(unstack m a)`	`(put-down m)`	`(unstack t o)`	`(stack t m)`	`(unstack a w)`	`(put-down a)`
	M_{G_1}	>>	>>	>>	`(unstack t o)`	`(stack t m)`	>>	>>
i^{δ}		1^1	2^1	3^1	4^1	5^1	6^1	7^1
$c(\tau'_i, M_{G_1}, i)$		0	0	0	0	0	0	0
$\gamma_4 =$	τ'_i	`(put-down e)`	`(unstack m a)`	`(put-down m)`	`(unstack t o)`	`(stack t m)`	`(unstack a w)`	`(put-down a)`
	M_{G_2}	>>	`(unstack m a)`	`(put-down m)`	>>	>>	>>	`(put-down a)`
i^{δ}		1^1	2^1	3^1	4^1	5^1	6^1	7^1
$c(\tau'_i, M_{G_2}, i)$		1	0	0	1	1	1	0

Table 4.4 Optimal alignments between trace prefix τ'_i observed in environment 1 from Figure 4.1a and models M_{G_1} and M_{G_2}.

Next, we discuss the two new steps of the adaptive version of the PM-based GR system.

Step 4: Once the agent reaches a goal, the information on completed trace τ_i, true goal G the agent has achieved by executing τ_i, and the previously inferred goal based on the corresponding trace prefix τ_i' are passed to the feedback collection component. We assume that once the agent reaches a goal, that is, completes a trace of actions, the corresponding true goal can be detected. The feedback collection component performs two tasks. First, it evaluates the inferred goal against the true goal. The evaluation results are then communicated to the relearn decision component through goal inference statistics to inform whether to retrain the GR system.

Second, the completed trace and the corresponding achieved goal are stored in the collection of historical observations, ready to be used to relearn the process models once the necessity arises.

Step 5: The relearn decision component uses the goal inference statistics to generate signals to instruct the GR system to update the discovered process models, that is, to implement adaptive GR strategies. In this work, we explore open- and closed-loop strategies. The open-loop strategy instructs the process discovery component to relearn the models after a specified number of performed GR tasks; the decision to relearn the models is triggered regularly. On the other hand, a closed-loop strategy evaluates the historical performance of GR inferences by the system summarized in the statistics generated by the feedback collection component to decide when to trigger the decision to relearn the process models. We use balanced accuracy to measure the GR performance, as explained in Section 4.3.2. Intuitively, a drop in the GR accuracy indicates that the environment may have changed. Consequently, the adaptive GR system may decide to relearn the process models from the recently observed historical traces.

In this work, we present and evaluate three adaptive GR strategies summarized below.

- **Open-loop adaptive GR strategy.** This strategy requests automatically relearning the process models from the most recent historical observations after a pre-configured number of GR inferences, regardless of the GR performance.

- **Closed-loop adaptive GR strategy based on average accuracy.** This strategy monitors the average accuracy over the pre-configured number of most recent GR inferences. If the average accuracy drops below a given threshold, it requests to relearn the process models from the most recent historical observations.

- **Closed-loop adaptive GR strategy based on accuracy trend.** This strategy uses linear regression to analyze the accuracy trend of the most recent GR inferences. If this trend suggests that the accuracy of a number of the subsequent GR inferences will drop below a

given threshold, the strategy requests to relearn the process models from the most recent historical observations.

4.3 Evaluation

This section presents an evaluation we conducted to study the performance of our adaptive GR systems. Section 4.3.1 describes the synthetic datasets we generated using the extended version of the GRACE tool and the real-world event logs we used to evaluate the GR performance. Section 4.3.2 presents the measures we used to assess the performance of our GR systems. Finally, Section 4.3.3 describes the experiments, while Section 4.3.4 discusses the results of the experiments.

4.3.1 Datasets

The synthetic dataset, a collection of adaptive GR problems, used in our experiments was generated using the extended version of the GRACE tool. The tool takes a static GR problem instance described in the Planning Domain Definition Language (PDDL) [51] and a configuration as input and generates a sequence of GR problems, each composed of a PDDL description of the environment and an observation of an agent accomplishing a goal in the environment, as output. The configuration specifies changes that should be applied in the environment of the input GR problem (e.g., a change of the initial state of the agent and objects in the environment) and a drift type according to which this input environment should evolve in the modified environment. Each problem in the generated sequence is either the original GR problem or the GR problem over the changed environment. Two consecutive problems in the sequence specify their temporal relation; the preceding problem should be solved before the succeeding problem. The order of the problems in the sequence implements the requested drift type.

To ensure that the GRACE tool generates environments significantly different from those in the input GR problems, we extended the tool to use the significance of change (SOC) measure. The SOC measure quantifies the differences between the original and modified environments. In the first implementation of the GRACE tool, the input environment is modified randomly. For example, a new initial state of the agent is discovered via a random walk from the original initial state of the environment. The new initial state discovered this way could be similar to the original initial state. To avoid such situations, we extended GRACE to use the differences between optimal plans in the original and the changed environments to quantify how significantly the environment has been modified. The

difference between two given plans is computed as their *Levenshtein edit distance* [78]. Let the original environment have n goal candidates, $G_1, \ldots, G_n$, and an initial state I. For each goal candidate G_i, $i \in [1..n]$, we find an optimal plan π_i that starts in I to G_i using the top-k planner [115]. The GRACE tool modifies the original environment to an environment with n candidate goals, namely $G'_1, \ldots, G'_n$, and one initial state I'. We compute optimal plans $\pi'_1, \ldots, \pi'_n$ that start in I' to goals $G'_1, \ldots, G'_n$, where π'_i is an optimal plan from I' to G'_i. By $dist(\pi_i, \pi'_i)$, we denote the Levenshtein edit distances between plans π_i and π'_i; note that the plans are captured as sequences of actions performed by the agent, that is, traces. If two traces, π_i and π'_i, are entirely different, the maximum possible Levenshtein edit distance between them can be obtained by summing their lengths, that is $|\pi_i| + |\pi'_i|$. Equation 4.1 defines the SOC measure between the original and modified environments.

$$SOC = \frac{\sum_{i=1}^{n} dist(\pi_i, \pi'_i)}{\sum_{i=1}^{n} (|\pi_i| + |\pi'_i|)} \tag{4.1}$$

It holds that SOC is between zero and one, inclusively, that is, $SOC \in [0, 1]$. The larger the SOC value, the more significant the difference between the original and modified environments.

We used conventional GR problem instances from 15 International Planning Competition (IPC) domains[5] as input to GRACE. For each input problem instance, GRACE was configured to generate the corresponding adaptive GR problem instance by following these four steps:

1. Modify the initial state of the environment of the input GR problem to generate a new GR environment that is significantly different from the input environment (the SOC value greater or equal to 0.25).
2. Use the top-k planner [115] to construct 1 000 plans from the initial state to every goal candidate state, both for the original and modified environment.
3. Use the original and modified environments to simulate sudden, gradual, and reoccurring drifts from the original to the modified environment as sequences of these two environments.
4. For each environment at every position in the sequence, select one generated plan towards each candidate goal from the set of constructed 1 000 plans to represent the behavior of the agent for accomplishing the goal in that environment at that time step.

We constructed and made publicly available 228 adaptive GR problem instances from 15 domains (76 problem instances times 3 types of drifts).[6] Note that the number of problem instances is determined by the number of different problems in the original IPC dataset.

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

[6] https://doi.org/10.26188/21802081

Figure 4.1 shows the original environment (environment 0) and the modified environment (environment 1) of one constructed adaptive GR problem instance from the constructed datasets, while Figure 4.2 shows how these two environments are arranged in a sequence by GRACE to simulate a gradual drift from environment 0 to environment 1.

To conduct experiments over real-world datasets, we used the preprocessed event logs provided by Teinemaa et al. [118], in which the traces are grouped by Linear Temporal Logic (LTL) classifiers. The traces within each group accomplish the same goal. For each event log, we sorted the traces based on the timestamps of their last events. Following this sorted chronological order, we selected an equal number of traces that accomplish each goal candidate to simulate a sequence of observed traces as inputs for GR tasks. We then extracted traces from the very beginning and the very end of each log to induce a drift. Intuitively, a time gap between these two trace groups may result in a noticeable concept drift. We then used the first ten time steps (where each time step is represented by the agent traces towards all the goal candidates) for discovering the initial process models. We used the subsequent 50 time steps (time steps 11 to 60) to simulate GR tasks within the initial environment and the last 50 time steps to simulate GR tasks after the changes in the initial environment. Finally, we conducted experiments using the conventional GR system that does not adapt to environmental changes over the constructed 100 testing time steps. If the average recognition accuracy over the first 50 time steps was significantly higher (by 10%) than the average accuracy over the last 50 time steps, we asserted that the concept drift indeed occurred. Using this approach, we selected two event logs from the real-world business domains that exhibit concept drift: BPIC 2011[7] and Hospital Billing.[8] Note that for the Sepsis Cases event log,[9] the number of traces in the event log is relatively small, resulting in 98 time steps after preprocessing. Thus, we examined the first and the last 20 time steps to ascertain whether there was a significant drop in GR accuracy over these two periods. The results revealed a 0.061 (8.6%) accuracy drop, but we included the Sepsis Cases event log in the subsequent experiments. Consequently, we obtained three event logs[6] from real-world business domains, specifically BPIC 2011, Hospital Billing, and Sepsis Cases, which we used for evaluating our adaptive GR systems.

4.3.2 Performance Measures

For each observation of an agent attempting to accomplish a goal supplied with the adaptive GR problem instance from the dataset, the GR performance is measured using the *balanced*

[7] https://doi.org/10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54
[8] https://doi.org/10.4121/uuid:76c46b83-c930-4798-a1c9-4be94dfeb741
[9] https://doi.org/10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460

accuracy measure [90], which requires four terms to compute. True Positive (*TP*) is the number of correct goals inferred by the GR system. True Negative (*TN*) is the number of incorrect goals not inferred. False Positive (*FP*) is the number of incorrect goals inferred by the GR system. Finally, False Negative (*FN*) is the number of correct goals not inferred. For example, consider a GR system observing an agent that executes a sequence of actions, working towards a true hidden goal (G_1) among ten candidate goals (G_1 to G_{10}). Suppose that, according to the observed action sequence, the GR system infers G_1 and G_2 as possible goals the agent is trying to achieve. Therefore, G_1 and G_2 are two positive goals, and the rest of the candidate goals are negative goals. In this scenario, for the two positive goals, G_1 is the true hidden goal correctly inferred by the GR system. Thus, *TP* is equal to one. Goal G_2 is not the true hidden goal; it is falsely inferred by the GR system. Thus, *FP* equals one. None of the eight negative goals (G_3 to G_{10}) is the true hidden goal. Hence, *TN* equals eight because the 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; the GR system correctly recognized the true hidden goal. 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. Given the four terms (*TP*, *TN*, *FP*, and *FN*), the balanced accuracy (*BACC*) is an average of the true positive and the true negative rates, given below.

$$BACC = \frac{1}{2} \times \left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP} \right) \tag{4.2}$$

4.3.3 Experiments

We evaluated the performance of the standard PM-based GR system and the three adaptive GR strategies (see Section 4.2) over the synthetic and real-world instances of the adaptive GR problems. We denote the evaluated GR systems that follow different adaptive strategies as follows: SYS_{std} is the standard PM-based GR system; SYS_{ol} is the adaptive PM-based GR system that implements the open-loop adaptive GR strategy; SYS_{cla} is the adaptive PM-based GR system that implements the closed-loop adaptive GR strategy based on average accuracy; and SYS_{clt} is the adaptive PM-based GR system that implements the closed-loop adaptive GR strategy based on accuracy trend.

We used the performance of SYS_{std} as the baseline to compare it with the performance of the three adaptive GR systems. In SYS_{ol}, we set the system to relearn after every ten GR inferences. In SYS_{cla}, we set the relearn threshold of the highest average accuracy over ten consecutive GR inferences, denoted by ACC_{h10}, to 80%. Hence, the system would relearn if the average accuracy over the past ten GR inferences is below $0.8 \times ACC_{h10}$. In SYS_{clt}, if the predicted accuracy of the following ten GR inferences is below $0.8 \times ACC_{h10}$, then the GR

system relearns the process models. We simulated partial observability of the agent in each GR task by only including 50% of the actions from each trace towards a goal supplied in the datasets.

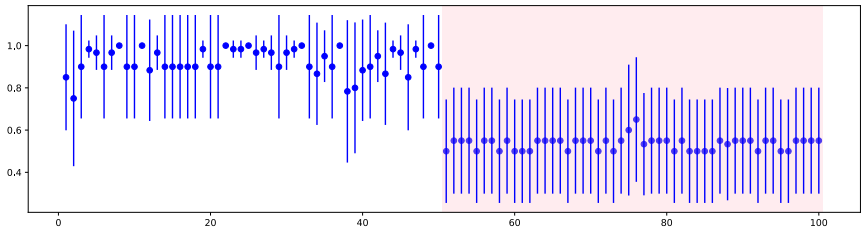
All the adaptive GR problem instances from the datasets were solved using our open-source implementation of the PM-based GR system and the proposed adaptive GR strategies on a single core of an Intel Xeon Processor (Skylake, IBRS) @ 2.0GHz with 16GB of RAM.[10]

4.3.4 Results

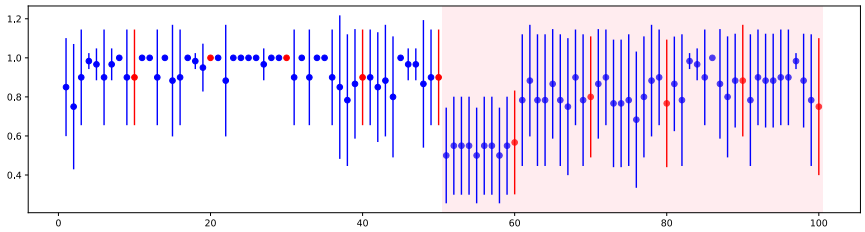
The results of the experiments confirmed that, compared to the standard PM-based GR system, the adaptive GR systems achieve better recognition accuracies. Figure 4.8 summarizes the performance of the four evaluated GR systems (SYS_{std}, SYS_{ol}, SYS_{cla}, and SYS_{clt}) on an adaptive GR problem instance from the synthetic zeno-travel domain with a sudden drift from the original to the modified environment. Specifically, the first 50 GR problems are defined for the original environment, while problems 51 to 100 are defined for the modified environment. In each plot in Figure 4.8, the X-axis shows a sequence of problems tackled by the GR system, while the Y-axis shows the average balanced accuracy of the recognized goals. Note that each adaptive GR problem instance in the dataset is defined for multiple (n) candidate goals (G_1 to G_n). Recall that for each GR problem at position j in the sequence on the X-axis, we have n observations (i.e., traces of actions), one trace the agent has followed for each goal. By $BACC^{s}_{G_i}$, we denote the balanced accuracy of performing goal recognition using GR system s on the trace towards goal G_i, $i \in [1..n]$, $s \in \{SYS_{std}, SYS_{ol}, SYS_{cla}, SYS_{clt}\}$. Hence, each data point in the plots stands for $\frac{\left(\sum_{i=1}^{n} BACC^{s}_{G_i}\right)}{n}$ and is denoted by $ABACC^{s}_{j}$. The line segment associated with each data point represents the corresponding standard deviation range, while data points in red, in addition, represent that at that position in the sequence, the system decided to update its knowledge base and relearned the models it uses for solving GR problems.

Figure 4.8a shows that the average recognition accuracy drops from 0.927 to 0.670 after the environment changes if the GR system does not update the process models. In contrast, for the GR systems with relearn mechanisms, the recognition accuracy can recover to some degree after the environment changes. The open-loop adaptive GR system SYS_{ol} (Figure 4.8b) relearns ten times, and the average accuracy after environment change is 0.789. The closed-loop adaptive GR system based on average accuracy SYS_{cla} (Figure 4.8c) relearns

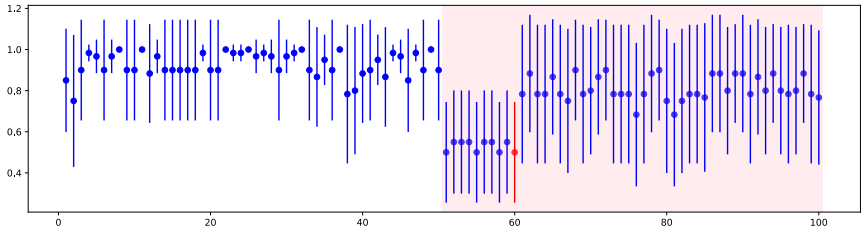
[10]The code to run the experiments and our implementations of the GR systems can be accessed here: https://doi.org/10.26188/25329490.



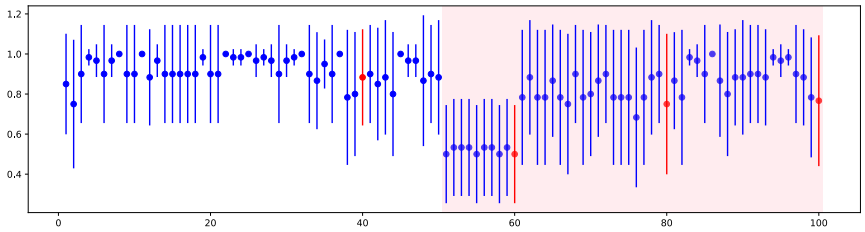
(a) The standard PM-based GR system SYS_{std}; average accuracy in cases 1–50 and 51–100 is 0.927 and 0.670, respectively, with no updates of the process models.



(b) The open-loop adaptive PM-based GR system SYS_{ol}; average accuracy in cases 1–50 and 51–100 is 0.943 and 0.789, respectively, with ten updates of the process models.



(c) The closed-loop adaptive PM-based GR system based on average accuracy SYS_{cla}; average accuracy in cases 1–50 and 51–100 is 0.927 and 0.761, respectively, with one update of the process models.



(d) The closed-loop adaptive PM-based GR system based on accuracy trend SYS_{clt}; average accuracy in cases 1–50 and 51–100 is 0.921 and 0.792, respectively, with four updates of the process models.

Figure 4.8 Performance of four GR systems on a single adaptive GR problem instance with a sudden drift.

once, and the average accuracy after environment change is 0.761. Finally, the closed-loop adaptive GR system based on accuracy trend SYS_{clt} (Figure 4.8d) relearns four times, and the average accuracy after environment change is 0.792. The detailed results for each adaptive GR system, performing in different problem instances under three types of environmental drift, are available online.[11]

We computed the average accuracies of the goal recognition for the four evaluated GR systems in the original and modified environments over all the problem instances and all the drift types in the datasets. The average accuracy achieved by GR system s in the original environment is denoted by $ABACC0^s$ and equals $\frac{(\sum_{j\in E0} ABACC_j^s)}{|E0|}$, while $ABACC1^s$ stands for the average accuracy in the modified environment and equals $\frac{(\sum_{j\in E1} ABACC_j^s)}{|E1|}$; $E0$ and $E1$ are the sets of all positions in the sequence of GR problems defined over the original environment and the modified environment, respectively. The difference between the average accuracies in the original and modified environments for the SYS_{std} system shown in Figure 4.8a (i.e., the system without relearn capabilities) is the drop in accuracy for the baseline system, see below.

$$ACCDROP = ABACC0^{SYS_{std}} - ABACC1^{SYS_{std}} \tag{4.3}$$

The accuracy improvement is the difference between the average accuracies in the modified environments computed for adaptive GR system s and the standard GR system SYS_{std}.

$$Improvement^s = ABACC1^s - ABACC1^{SYS_{std}}, \quad s \in \{SYS_{ol}, SYS_{cla}, SYS_{clt}\} \tag{4.4}$$

Next, the improvement ratio for GR system s is defined as follows.

$$ImprovementRatio^s = \frac{Improvement^s}{ACCDROP}, \quad s \in \{SYS_{ol}, SYS_{cla}, SYS_{clt}\} \tag{4.5}$$

Table 4.5 shows accuracy improvement ratios computed per drift type for the three adaptive GR systems and the average number of times (per problem instance) the system triggered relearning of the knowledge base. The detailed results of all the experiments based on which the data in the table was computed are publicly available.[12] For instance, the table suggests that the open-loop system on a problem with a sudden drift relearns (on average) ten times and demonstrates the improvement ratio of 0.62. That is, it recovers (on average) 62% of the drop in recognition accuracy due to the change of the environment. The last column in Table 4.5 shows the absolute accuracy drop for each type of drift over all problem instances. We conclude from the results that the proposed adaptive GR systems can improve

[11] https://doi.org/10.26188/21901296
[12] https://doi.org/10.26188/21901296

recognition accuracy compared to the standard PM-based GR system. In general, the more times the system relearned, the better improvement in accuracy was observed. However, good improvements in accuracy can already be obtained by retraining the system a small number of times. Hence, it is evident that a closed-loop GR system can achieve good performance over a wide range of domains and drift types while keeping the number of relearn episodes low.

Drift	Open-loop	Closed-loop (Avg)	Closed-loop (Trend)	*ACCDROP*
Sudden	62.0% / 10	33.5% / 1.4	51.6% / 2.3	0.170
Gradual	106.2% / 15	43.6% / 1.7	87.6% / 3.2	0.170
Reoccuring	62.3% / 20	36.1% / 3.4	51.5% / 5.4	0.167

Table 4.5 Accuracy improvement ratio and the average number of times the process models were relearned (improvement ratio/avg. number of relearn episodes per problem instance) for three adaptive GR systems over 218 problem instances.

The improvement ratios in GR accuracy achieved by using adaptive GR systems, along with the number of times the system triggered relearning of process models for the real-world business domains—BPIC 2011, Sepsis Cases, and Hospital Billing—are displayed in Table 4.6. The column labeled "*ACCDROP*" represents the absolute accuracy drop after the environmental change (concept drift) when using SYS_{std}. Nevertheless, these performance drops can be mitigated using the adaptive GR systems: SYS_{ol}, SYS_{cla}, and SYS_{clt}. For instance, for the BPIC 2011 event log, the average accuracy drops by 0.116 when using SYS_{std} due to the environmental change, and SYS_{ol} can recover 122.4% of the accuracy drop by relearning the process models ten times over the entire goal recognition period.

Domain	Open-loop	Closed-loop (Avg)	Closed-loop (Trend)	*ACCDROP*
BPIC 2011	122.4% / 10	95.7% / 6	152.6% / 7	0.116 (from 0.540 to 0.424)
Sepsis Cases	98.4% / 10	173.8% / 2	123.0% / 1	0.061 (from 0.713 to 0.652)
Hospital Billing	108.3% / 10	96.4% / 4	137.0% / 3	0.084 (from 0.720 to 0.636)

Table 4.6 Accuracy improvement ratio and the average number of times the process models were relearned (improvement ratio/avg. number of relearn episodes per problem instance) for three adaptive GR systems over three real-world domains.

Chapter 5

Evidence-Based Goal Recognition for Powered Transhumeral Prostheses

This chapter focuses on *RQ3: Are evidence-based goal recognition techniques practically useful?* We claim that the evidence-based goal recognition (GR) framework introduced in Chapter 3 is applicable in real-world scenarios. To empirically verify our claim, this chapter conducts a case study that applies process mining (PM)-based GR techniques to powered transhumeral prostheses scenarios. This study aims to develop a powered transhumeral prosthesis, guided by PM-based GR techniques, to enhance the efficiency of prosthetic use for individuals with upper limb disabilities [75, 142, 141] We conduct both offline and human-in-the-loop (HITL) experiments to evaluate our proposed target pose recognition[1] approaches using process mining techniques. The results confirm that our approaches outperform the existing state-of-the-art baseline approaches [141, 57] in the field of transhumeral prostheses.[2]

Concretely, this chapter makes the following contributions:

- It applies the PM-based GR framework to target pose recognition for powered transhumeral prostheses, aiming to assist the daily activities of people with disabilities. The proposed techniques adapt the PM-based GR approach to operate with multi-dimensional, real-valued, continuous measurements that characterize the observed behavior of interest;
- It evaluates the two proposed PM-based target pose recognition techniques by comparing them with state-of-the-art baselines, including static LDA, dynamic LDA, and LSTM, through an offline experiment;

[1] In this chapter, we use the terms "goal recognition" and "target pose recognition" interchangeably.

[2] This chapter is an adaptation of our previously published works: "Data-driven goal recognition in transhumeral prostheses using process mining techniques." In 2023 5th IEEE International Conference on Process Mining (ICPM), pages 25-32, 2023.

- It conducts a human-in-the-loop (HITL) experiment to assess the two best-performing approaches identified in the offline setting, the PM-based GR with a classifier and the dynamic LDA baseline. The experiment results verify that the PM-based approach outperforms the baseline, thereby supporting the claim that the evidence-based GR framework is practically useful in real-world scenarios.

5.1 Study Description

Transhumeral prostheses, designed to restore upper limb function by replacing missing limb segments below the shoulder, facilitate tasks such as reaching or grasping, thereby enhancing users' ability to perform activities of daily living. This study aims to develop a technique for guiding actuator-driven powered transhumeral prostheses towards a target that the human user knows, while the prosthesis does not. Accurately identifying this goal is crucial, as failure to achieve it can result in inefficient task execution, user frustration, and potential device abandonment [49]. The prosthetic joint movements are usually controlled through the user's volitional movement comprising surface electromyography (sEMG) signals of the above-elbow muscles, the movement of the residual limb (upper arm) and body [5, 42]. We aim to develop an algorithm that identifies a user's intended target from continuous, real-valued sensor measurements, similar to goal recognition techniques. However, developing effective GR algorithms in transhumeral prostheses is challenging due to the lack of direct inputs linked to the joint movements—muscles anatomically connected to the wrist are not accessible—and the significant variability in kinematic and muscle activity signals of individuals [110].

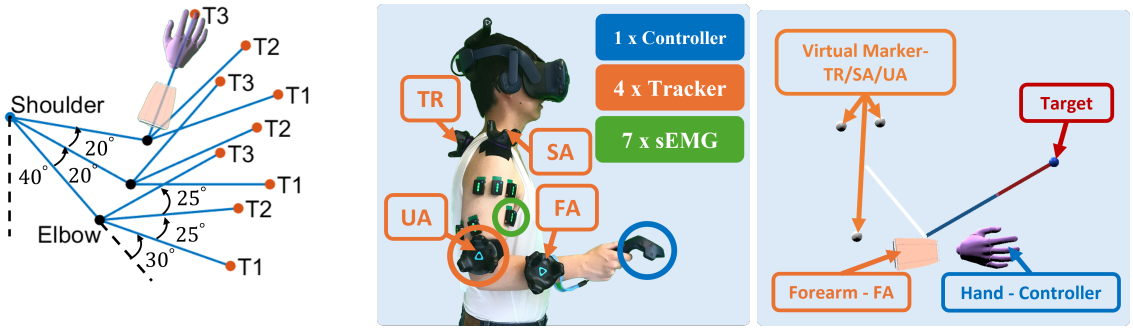
We conduct a two-step study that consists of (i) offline experiments for algorithm development and (ii) human-in-the-loop (HITL) prosthesis movement control experiments employing the developed algorithm. This two-step approach is commonly employed in the literature of powered prostheses [49]. We first use pre-collected body movement and muscle activity datasets from non-disabled human subjects to develop an accurate and robust goal recognition algorithm based on offline experiments. Then, we deploy the algorithm to control the prosthetic device in real-time based on the inputs generated by human users that are impacted by the HITL control mechanisms.

The distinction between offline and HITL experiments is whether the subjects interact with the powered prosthesis. In offline experiments, measurements are taken as non-disabled subjects reach goals with their intact limb without interacting with the prosthesis. This dataset is collected to inform the development of the GR system. In HITL experiments, real-time measurements are used by the developed GR system to recognize the goal. The

motors then drive the prosthesis to the recognized goal (pose), which closely aligns with the intended real-life use of the prosthetic device. In a HITL experiment, the subject interacts with the prosthesis by experiencing its movement and adjusting their joint and muscle movements accordingly. These "interactions" between the prosthesis and the user can lead to inputs to the GR system that are substantially different from those captured in the offline dataset. HITL experiments thus support testing the robustness of the developed GR system. Additionally, doubts exist in the literature regarding the correlation between offline and HITL performance [92, 81, 52], highlighting the need for HITL experiments to justify the efficacy of the developed powered transhumeral prostheses.

5.1.1 Offline Experiments

The dataset for offline experiments was collected as non-disabled subjects extended their intact upper limbs forward to reach three goal elbow poses repeated at three shoulder flexion/extension poses. The goals (target elbow poses) are denoted as *T1*, *T2*, and *T3* in Figure 5.1a, which illustrates the side-view schematic of the upper limb. The dataset captures the above-elbow joint movements and muscle activities through motion trackers and surface electromyography (sEMG) sensors, respectively, with 12 joint kinematic movement features and 35 sEMG features extracted at a rate of 10 Hz (the measurements were taken every 0.1 seconds). The process of extracting the sEMG and joint movement features has been described in detail in previous work [142].



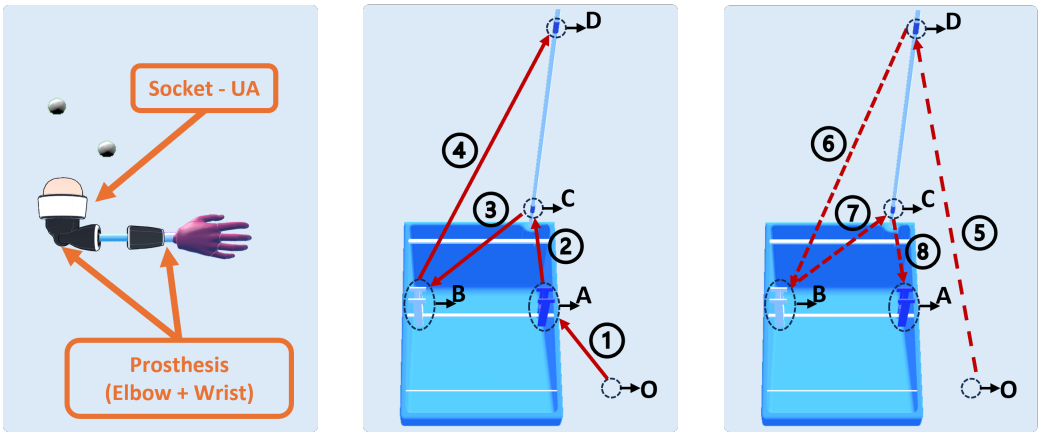
(a) (b) (c)

Figure 5.1 Offline dataset collection setup: (a) Target shoulder and elbow poses; T1–T3 denote three goals, (b) Experimental setup and the placement of VIVE trackers and sEMG electrodes, and (c) VR avatar showing target example (side view).

The sensor placements and the virtual avatar in VR are shown in Figures 5.1b and 5.1c, respectively. To capture the residual limb joint kinematics, three HTC VIVE trackers were strategically positioned at the upper arm (UA), shoulder acromion (SA), and trunk (TR). The

displacement and velocity of the six degrees of freedom (DoF) shoulder and trunk movements were extracted as features. An additional tracker was placed on the forearm (FA) to acquire the elbow joint kinematics. A controller was held in the hand to move the hand avatar shown in Figure 5.1c. For monitoring the muscle activity, seven Trigno™ wireless sEMG electrodes by Delsys® were attached to the upper-arm muscles of the dominant arm. Each electrode produces five features.

In the experiment, the goal is considered reached when the middle finger of the virtual avatar hits the target sphere, which is generated to elicit the target shoulder and elbow poses. For each goal, the subjects were tasked with 30 iterations of the forward-reaching. They were instructed to keep their final upper limb pose for one second upon reaching the goal. The data spanned from the initiation of the movement to the end of the holding period were reserved for feature extraction.



(a) (b) (c)

Figure 5.2 Human-in-the-loop experiment setup: (a) virtual 3-DoF prosthesis avatar; (b) forward stage of the RCRT task, and (c) backward stage of the RCRT task; numbers indicate goals (intended movements) with red solid and dashed arrows demonstrating the movement paths, letters A to D show the relocation positions, and O represents the arm resting position (upper-arm pointing downwards).

5.1.2 Human-In-The-Loop Experiments

In the human-in-the-loop (HITL) experiments, a real-world task is assessed in the HMD VR environment, where subjects are tasked with picking up and relocating the clothespins by using a virtual prosthesis attached to their dominant side, as depicted in Figure 5.2a. Such a task involves three prosthetic DoFs: elbow flexion/extension, wrist pronation/supination, and hand open/pinch. It is worth noting that this task involves different target poses (goals) compared to the offline experiment. Therefore, a different training dataset is first collected

from the subject executing the clothespin relocation task to construct the GR system, followed by the HITL experiment.

For real-time goal recognition, features are extracted at a rate of 10 Hz and streamed as inputs to the GR system for controlling the prosthetic elbow and wrist movement. The hand open/pinch function typically requires a dedicated control algorithm or GR system separate from joint control due to the temporal sequence of gross arm movement and hand manipulation [5]. Thus, a switching mechanism is needed to transition between the two control algorithms. To isolate the effects of switching between control algorithms, the hand open/pinch function is controlled through a button held in the non-dominant hand, as described in [75]. The socket of the prosthesis tracks the movement of the UA tracker and connects the prosthesis to the subject's residual limb. The sensor setup mirrors the one used in the offline evaluation. Twelve more kinematic features are investigated, comprising the acceleration of the six DoF movements used in the offline experiments and the kinematics of two additional DoFs, which are essential to extend the workspace from a plane (Figure 5.1a) to a 3-dimensional space.

The HITL experiments adhere to the widely used Refined Clothespin Relocation Task (RCRT) documented in [58, 75], with a dedicated one-to-one scale virtual setup. The task comprises eight kinds of movements, representing eight distinct goals, accomplished in two stages. In the forward stage, see Figure 5.2b, the subjects begin from the upper-arm resting and pointing downwards (denoted as "O" in Figure 5.2b). They then sequentially pick up the two clothespins placed vertically on the horizontal rod (positions A and B) and transfer them to the vertical rod (positions C and D). Subsequently, in the backward stage, they start from the resting position and then return the clothespins at positions C and D to the original positions A and B, see Figure 5.2c. The desired movements and goal categories are marked using red arrow lines and corresponding numbers; solid lines are used to depict desired movement trajectories in the forward stage, while dashed lines capture the desired movement trajectories for the backward stage of the experiment. From a GR perspective, which typically focuses on distinguishing different goals from the same initial state, the eight movement trajectories present three GR challenges: identifying whether the subject aims for A or D from initial position O; identifying whether the arm is moving toward C or D from position B; and identifying whether the movement is toward A or B from position C.

5.2 Target Pose Recognition Using Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a trainable classification function that linearly separates multi-dimensional, real-valued, continuous measurements (data points) into a specified number of clusters. LDA is a lightweight classifier which can perform well with small amounts of training data and its fast reaction time, making it suitable for real-time tasks. It is commonly used in various applications analyzing myoelectric signals, such as prosthetic control [49] and gesture recognition [59]. In recent studies of powered transhumeral prostheses, LDA has been used to analyze sEMG and kinematic signals collected from the residual body parts of disabled patients to predict intended movements [141, 142], thereby guiding powered transhumeral prostheses to reach the desired poses of the patients.

Suppose the sensors capture a sequence of 15 data point signals for a subject moving their arm to reach a pose; see Figure 5.3. Each data point contains multi-dimensional features. The sequence is divided into two phases. Data points 1–10 are captured while the arm is moving towards the target pose. The last five data points, data points 11–15, are collected when the arm is fixed after reaching the target position. Yu et al. [141, 142] assume that the instantaneous sEMG and kinematic signals at different target poses exhibit distinct patterns. Therefore, they train the LDA classifier using sets of signals collected while the arm is held at specific static poses, we refer to these approaches as *static LDA*. When testing the recognition performance of static LDA, as the prosthesis moves closer and closer to a specific target pose, the last observed instantaneous signals become increasingly recognizable to the trained LDA classifier. Due to the characteristic of the LDA classifier being trained with static poses (where subjects hold their arms at the target pose), it may not effectively predict the target pose when receiving signals while the arm is in motion towards the target pose. To address this limitation and ensure that the LDA classifier reflects signal patterns during motion, we modify the training phase to incorporate signals collected *during* arm movements towards the target pose, referred to as *dynamic LDA*. In the example illustrated in Figure 5.3, the static LDA classifier is trained using the last five data points, whereas the dynamic classifier is trained using data points 1–10. However, note that both static and dynamic LDAs are applied to classify single data points, such as the sEMG and kinematic signals at a specific timestamp, rather than a sequence of signals.

Note that there are no customizable parameters in this approach. It is trained using all data points along the trajectory, and once the subject reaches a target, any remaining data points are discarded. Both methods, *static LDA* and *dynamic LDA*, are used as baselines for comparison with the PM-based GR techniques demonstrated in Section 5.3.

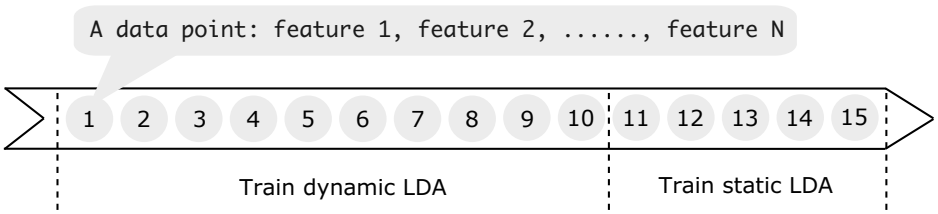


Figure 5.3 A sequence of data points captured by sensors.

5.3 Target Pose Recognition Using Process Mining

We propose target pose recognition approaches that leverage the PM-based GR framework. For a detailed understanding of our approach, please refer back to the PM-based GR framework discussed in Chapter 3 for concepts such as *traces*, *event log*, *process discovery techniques* (e.g., Directly Follows Miner [72]), *Petri nets*, *conformance checking*, *optimal alignment*, and the formulas for computing alignment weight and probability distribution. The PM-based GR technique is designed to recognize the intentions of agents by analyzing sequences of discrete events (i.e., actions). However, this project aims to recognize the intended poses of subjects based on multi-dimensional real-valued features collected by sensors. Therefore, to effectively utilize the PM-based GR framework, we propose two data conversion approaches to transform the original multi-dimensional real values into discrete events. We use two running examples to illustrate the step-by-step process of the data conversion approaches, followed by an explanation of how to employ the PM-based GR technique to recognize the intended poses by analyzing the converted data. The first approach, based on hierarchical clustering and K-means clustering algorithm, is presented in Section 5.3.1, while the second approach, involving training an LDA classifier leveraging the target pose information in the training data, is discussed in Section 5.3.2.

5.3.1 Event Identification Using Clustering

In this section, we use a running example to illustrate how the PM-based GR framework combines feature selection and event discretization techniques to tackle the target pose prediction problem. In the example, we instructed a subject to perform six iterations of reaching tasks, three times to reach target T1 and three times to reach target T2. The GR system observed six sequences of signals, each comprising 30 continuous real-valued features (including sEMG signals and kinematic signals), denoted as f_1 to f_{30}. Traces 1 to 3 represent signal sequences recorded during movements towards target pose T1, while traces 4 to 6 represent sequences for movements towards T2. The dataset of input signal sequences and

tools for reproducing the results are publicly available.[3] Table 5.1 presents a portion of the example dataset, with each row containing collected feature values ordered by their respective timestamps of data collection.

Trace	Goal	f_1	f_2	f_3	...	f_{29}	f_{30}
1	T1	5.19727337	7.02395793	0.00254431	...	5.39759498	-0.3722619
1	T1	7.76278776	8.08816201	0.00472689	...	1.01557531	1.37592798
1	T1	13.4185557	8.87159453	0.00821896	...	-4.0004147	1.65328609
1	T1	22.0916619	9.04377674	0.01015369	...	-5.5399488	-1.7805512
1	T1	31.3641039	9.3586209	0.009165	...	-3.5156837	1.36367015
1	T1	38.2312577	10.139119	0.00616715	...	-1.4720033	5.87820456
1	T1	42.0592085	10.8827908	0.00315491	...	-0.3338844	4.29640897
2	T1	7.39110795	6.07336937	0.00064332	...	2.92403705	1.46698529
2	T1	10.5229866	7.44734189	0.00194998	...	1.60034347	2.94734496
2	T1	17.6705947	8.62902577	0.00393832	...	1.41373702	2.84419105
...	...	...	...	...	...	...	...
6	T2	53.1712171	19.394227	0.00270796	...	-3.5619104	0.78601719
6	T2	60.1200614	22.6060167	0.00091891	...	-2.6239834	1.7355157
6	T2	64.1830578	25.2975943	-0.0003433	...	-1.1970367	0.92412363
6	T2	66.8916142	27.5304609	-0.0017204	...	0.31022101	0.95595258

Table 5.1 Extract of the running example dataset.

The PM-based GR approach, using clustering algorithms for feature selection and event discretization, comprises five steps:

Step 1: The first step involves reducing the dimensionality of the input data by selecting features that have a substantial predictive power. Specifically, we exclude highly correlated features. First, we compute correlations between each pair of features in the dataset. Figure 5.4 summarizes the absolute values of the Pearson correlation coefficients for all pairs of features. Then, we use agglomerative hierarchical clustering [26] to group features into N_f clusters using correlation coefficients to define distances between the features. The resulting dendrogram, refer to Figure 5.5, provides a visual representation of the hierarchical structure of the computed clusters based on the correlations from Figure 5.4. One can use a threshold value to determine the number of clusters they want to extract from the dendrogram. The similarity threshold specifies the desired distance between formed clusters. As shown in Figure 5.5, in our example, setting the threshold to 1.23 (see the red dashed horizontal line in the figure) allows us to extract 15 clusters; groups of features connected below the threshold are considered as one cluster, while groups of features above the threshold are separate clusters. From each extracted cluster of features, we then select one representative feature with the largest correlation with all the other features in the cluster. The selected features constitute a lower-dimensional representation of the dataset. In the running example, we

[3] https://doi.org/10.26188/25487290

reduced the feature space to N_f = 15 dimensions; the selected 15 features are highlighted in red in Figure 5.5.

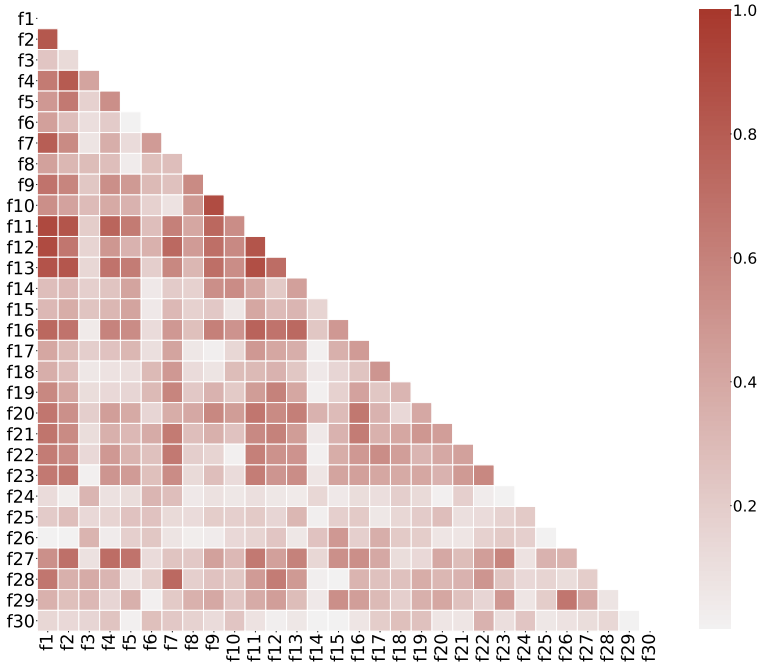


Figure 5.4 Correlation matrix.

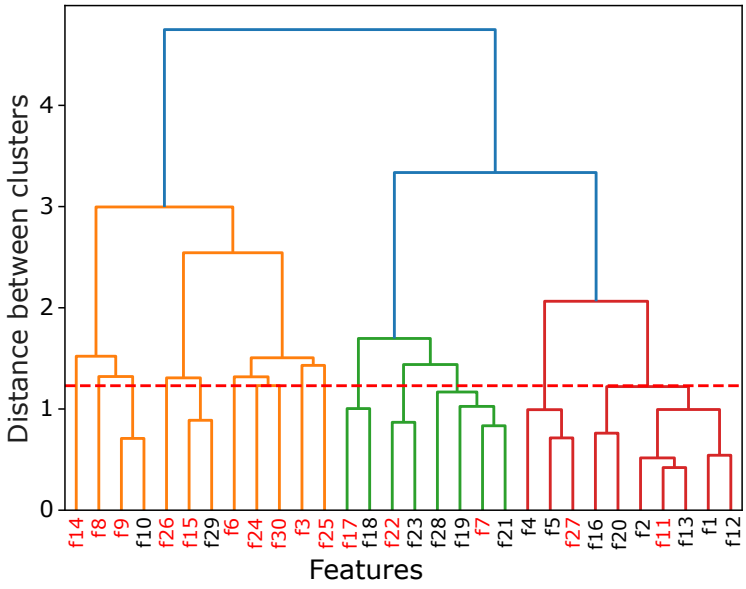


Figure 5.5 Dendrogram and selection of clusters. The red dotted line represents the threshold used to cut the dendrogram to form 15 clusters.

An extract of the reduced dataset is displayed in Table 5.2.

Trace	Goal	f_3	f_6	...	f_{27}	f_{30}	Event
1	T1	0.002544311	0.121301538	...	9.057075924	-0.372261852	e_0
1	T1	0.004726894	0.210557727	...	10.55266625	1.375927982	e_6
1	T1	0.008218956	0.391712037	...	4.416704571	1.653286092	e_5
1	T1	0.010153689	0.327711654	...	1.372325318	-1.780551234	e_2
1	T1	0.009165	0.311548058	...	5.526959903	1.363670147	e_5
1	T1	0.00616715	0.734098175	...	8.869853729	5.87820456	e_6
1	T1	0.003154906	1.227944362	...	4.916643199	4.296408971	e_8
2	T1	0.000643317	0.103164637	...	12.86040763	1.466985286	e_8
2	T1	0.001949978	0.336630569	...	13.45204634	2.94734496	e_6
2	T1	2.00E-05	2.91E-05	...	0.65070719	88.8458582	e_2
...	...	...	...	...	...	...	...
6	T2	3.47E-05	1.25E-05	...	0.16984547	90.8573749	e_1
6	T2	3.85E-05	1.06E-05	...	0.30483842	50.9656092	e_1
6	T2	-0.000343322	0.410761081	...	23.78469914	0.924123631	e_4
6	T2	-0.001720367	0.504684583	...	21.08908217	0.955952575	e_3

Table 5.2 Extract of the reduced running example dataset.

Step 2: In the next step, we convert the reduced dataset into event traces. To this end, we perform K-means clustering [50] over N_f-dimensional data points to obtain N_c clusters. The K-means algorithm groups similar data points together to minimize the variance within each cluster and maximize the distance between different clusters. Within each cluster, the data points are considered as instances of the same event.

The K-means algorithm takes the number of clusters it constructs as input. The approach we use to determine the appropriate number of clusters is elaborated in Section 5.4. For the purpose of demonstration, in the running example, we use $N_c = 10$. Consequently, the 15-dimensional data points are grouped into 10 clusters, represented by events e_0 to e_9, as shown in the "Event" column in Table 5.2. Then, we split the obtained traces of events into event logs, where each event log contains all the traces toward a specific candidate goal. As there are two target poses in the running example, we split the traces into two event logs: L_1 and L_2. The traces in the event logs L_1 and L_2 aim towards target poses *T1* and *T2*, respectively.

Steps 1 and 2 illustrate the event identification method using clustering algorithms. To present a complete running example, steps 3 to 5 demonstrate how to use the proposed PM-based GR framework (Chapter 3 to recognize the target pose given the converted event logs.

Step 3: We use process discovery techniques to construct process models from the event logs obtained in the previous step. Figures 5.6 and 5.7 depict the Petri nets M_1 and M_2, which are constructed by the Directly Follows Miner from the event logs L_1 and L_2 obtained in the previous step. The models M_1 and M_2 describe the processes for reaching the corresponding

target poses, T1 and T2, respectively. They are considered as "knowledge" learned from historical experiences and stored in our GR system.

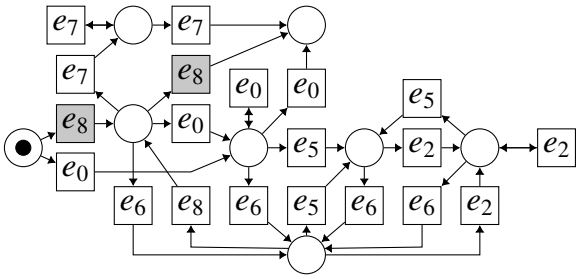


Figure 5.6 Process model M_1 discovered from even log L_1.

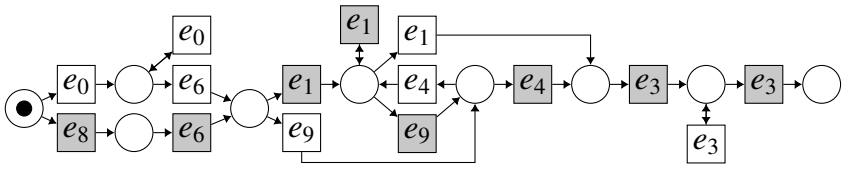


Figure 5.7 Process model M_2 discovered from even log L_2.

Step 4: Next, we use conformance checking techniques [126] to assess the commonalities and discrepancies between the discovered models and newly observed traces, leveraging this information to infer the intended target poses. In the running example, suppose we collect a new sequence of signals as shown in Table 5.3. As with data collected for training, each signal is a data point containing multi-dimensional, real-valued measurements representing different features, including sEMG and kinematic features. The example observation is partial—it comprises six first measurements during movement toward a target pose.

Timestamps	f_1	f_2	f_3	...	f_{29}	f_{30}
$t1$	6.003909937	7.491469679	-0.000426667	...	3.003725052	1.044722093
$t2$	10.4000259	8.661748533	0.002236012	...	-1.678683223	1.119218147
$t3$	18.34086353	9.394745911	0.005143733	...	-4.419588229	0.414732289
$t4$	31.05712269	10.02530957	0.006663839	...	-5.512023759	-0.470670561
$t5$	45.26542356	11.12347556	0.005400478	...	-5.503936393	-0.967709384
$t6$	56.95525497	13.56039176	0.002532783	...	-4.995283183	1.20365095

Table 5.3 A partial sequence of signals observed by the GR system.

The GR system then extracts the features selected in step 1 and converts the multi-dimensional data points into discrete events by relating data points to clustering obtained in step 2. These converted events are ordered according to the timestamp. In the running example, this procedure results in trace τ shown below.

$$\tau = \langle e_8, e_6, e_2, e_1, e_1, e_9 \rangle .$$

Given trace τ and process models M_1 and M_2 discovered in the previous step, the GR system computes optimal alignments σ_1 and σ_2 between the trace and the models. The constructed optimal alignments are shown below. The transitions of M_1 and M_2 involved in the alignments are shaded in gray in Figures 5.6 and 5.7.

$$\sigma_1 = \begin{array}{|c|c|c|c|c|c|c|c|} \hline \tau & e_8 & \gg & e_6 & e_2 & e_1 & e_1 & e_9 \\ \hline M_1 & e_8 & e_8 & \gg & \gg & \gg & \gg & \gg \\ \hline \end{array}$$

$$\sigma_2 = \begin{array}{|c|c|c|c|c|c|c|c|c|c|} \hline \tau & e_8 & e_6 & e_2 & e_1 & e_1 & e_9 & \gg & \gg & \gg \\ \hline M_2 & e_8 & e_6 & \gg & e_1 & e_1 & e_9 & e_4 & e_3 & e_3 \\ \hline \end{array}$$

Step 5: Finally, in the last step, the PM-based GR system leverages information on moves in the constructed optimal alignments to calculate the probability distribution over the candidate target poses. The probabilities associated with each target pose indicate the likelihood of the subject aiming to reach that pose. To compute the probability distribution, the system first computes alignment weights between trace τ and models M_1 and M_2 according to Equation 3.5. Then, it uses Equations (3.3) and (3.4) to compute the probability of reaching every target pose. In our running example, given the two alignments σ_1 and σ_2, the probabilities of the subject that induced trace τ aiming to reach target poses *T1* and *T2* are 0.06 and 0.94, respectively. Consequently, the PM-based GR system infers that the subject aims to reach *T2*.

5.3.2 Event Identification Using Linear Discriminant Analysis

This section describes another labeling technique utilizing a trained LDA classifier to assign labels to signals. An LDA classifier can map multi-dimensional values to particular categories. To train an LDA classifier, we first manually label the signals in the observed traces and then proceed to train the classifier using these labeled signals. We assume that the data points in the trajectories can be divided into sub-groups based on their timestamps. For example, data points collected early in the trajectory may differ significantly from those collected later. Therefore, we aim to divide the data points into sub-groups according to their time order and then train an LDA classifier to identify which sub-group a data point belongs to. This approach allows the classifier to distinguish whether a data point was collected in the early or late stage of the trajectory. Once trained, the classifier is applied to convert multi-dimensional real-valued signals into discrete labels. The subsequent steps follow the same logic as steps 3-5 outlined in Section 5.3.1. We create event logs and use the process discovery technique to construct process models. With these process models, we conduct conformance checking to construct optimal alignments between the process models and a newly observed converted

trace. Finally, we utilize these optimal alignments to compute probabilities and infer the intended pose accordingly.

We use another running example to illustrate the PM-based GR approach with an LDA classifier. In this example, we observed 20 traces towards target poses T1 and T2, with 10 traces per target pose.[4] The approach consists of four steps, explained below:

Step 1: In this step, signals of measurements are converted into discrete events, which is achieved by training an LDA classifier. During the training phase, we label the data points according to two aspects: the part of the trajectory the data points relate to and the final target pose reached by the corresponding movement. For instance, consider a sequence of signals illustrated in Figure 5.8, which eventually reaches target pose *T2*. During the movement, 10 data points are collected. We divide the sequence into five segments based on the timestamps of the collected data points. Note that the number of segments to use is a parameter, and for illustration purposes, we use five segments in our example. We then assign each data point a label of the format *TxPy*, where *Tx* is the target pose that is reached by the sequence of signals the data point belongs to and *Py* is part of the trace the data point is located at. For example, the first part of the sequence of signals from Figure 5.8 capturing the initial phase of the movement contains two data points. Consequently, these data points are assigned label *T2P1*.

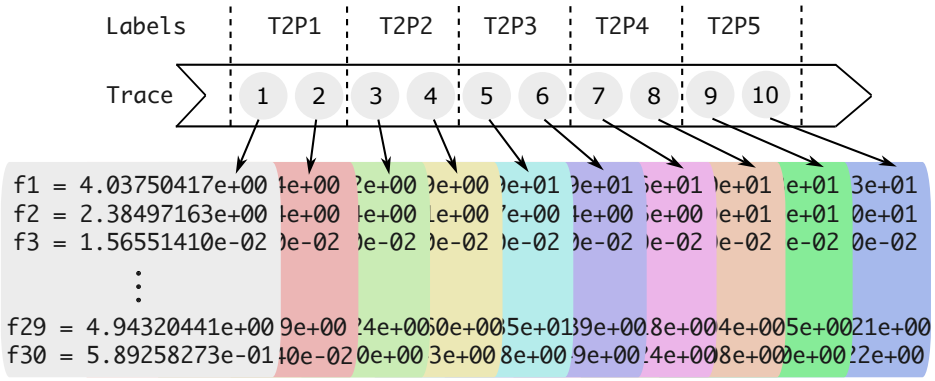


Figure 5.8 LDA partition.

Once all the data points are labeled, we use the obtained labels to train an LDA classifier. Once we obtain this classifier, we use it to convert the original multi-dimensional signals into discrete labels representing events. We then group the obtained events into event traces, where each trace captures all events from one movement toward a target pose. The traces are then grouped into event logs, where each event log contains all the constructed traces to a

[4]The data and source code required to replicate the running example: https://doi.org/10.26188/25487290

particular goal. In the running example, 20 sequences of signals are converted into two event logs L_1' and L_2', as shown below:

Event log 1 ($L1'$)
$< T2P1, T1P1, T1P2, T1P2, T1P3, T1P4, T1P5, T1P5 >$
$< T2P1, T1P1, T1P1, T2P1, T1P2, T1P2, T1P3, T1P3, T1P4, T1P4, T1P5, T1P5 >$
$< T1P1, T1P2, T1P2, T1P2, T1P3, T1P3, T1P4, T1P4, T1P5 >$
$< T2P1, T1P2, T1P2, T1P3, T1P4, T1P4, T1P5 >$
$< T1P1, T2P1, T2P2, T1P2, T2P3, T1P3, T1P4, T1P4, T1P5, T1P5 >$
$< T1P1, T1P1, T1P2, T1P2, T1P3, T1P3, T1P3, T1P4, T1P5 >$
$< T1P1, T1P1, T1P2, T1P2, T1P3, T1P3, T1P4, T1P4, T1P5 >$
$< T1P1, T1P1, T1P1, T1P2, T1P2, T1P3, T1P3, T1P3, T1P4, T1P5, T1P5 >$
$< T1P1, T1P1, T1P2, T1P2, T1P3, T1P3, T1P4, T1P4, T1P5 >$
$< T1P1, T1P1, T1P2, T1P2, T1P3, T1P3, T1P4, T1P4 >$
Event log 2 ($L2'$)
$< T2P1, T2P1, T2P2, T2P2, T2P3, T2P3, T2P4, T2P4, T2P5, T2P5 >$
$< T2P1, T2P1, T2P2, T2P2, T2P3, T2P4, T2P4, T2P5 >$
$< T2P1, T2P1, T2P1, T2P1, T2P1, T2P2, T2P2, T2P3, T2P3, T2P4, T2P4, T2P5 >$
$< T2P1, T2P1, T1P1, T1P2, T2P2, T2P3, T2P3, T2P4, T2P4, T2P5, T2P5 >$
$< T2P1, T2P1, T2P1, T2P2, T2P3, T2P3, T2P4, T2P4, T2P4 >$
$< T2P1, T2P1, T2P2, T2P2, T2P3, T2P3, T2P4, T2P4, T2P5, T2P5 >$
$< T2P1, T1P1, T2P2, T2P2, T2P3, T2P3, T2P4, T2P5, T2P5, T2P5 >$
$< T2P1, T2P2, T2P2, T2P3, T2P4, T2P4, T2P5, T2P5 >$
$< T2P1, T2P1, T1P2, T2P2, T2P3, T2P3, T1P4, T2P4, T2P5, T2P5, T2P5 >$
$< T1P1, T1P1, T1P2, T1P2, T2P3, T2P3, T2P3, T2P4, T2P4, T2P5 >$

Table 5.4 Event logs

Step 1 illustrates the event identification method through training an LDA classifier. Similarly, steps 2 to 4 then demonstrate the complete example of using the PM-based GR framework to recognize the intended target pose.

Step 2: In this step, we use Directly Follows Miner to discover process models from all event log obtained in step 1. For instance, using event logs L_1' and L_2', we construct process models M_1' and M_2' shown in Figures 5.9 and 5.10, respectively, representing the "knowledge" about how target poses *T1* and *T2* can be reached.

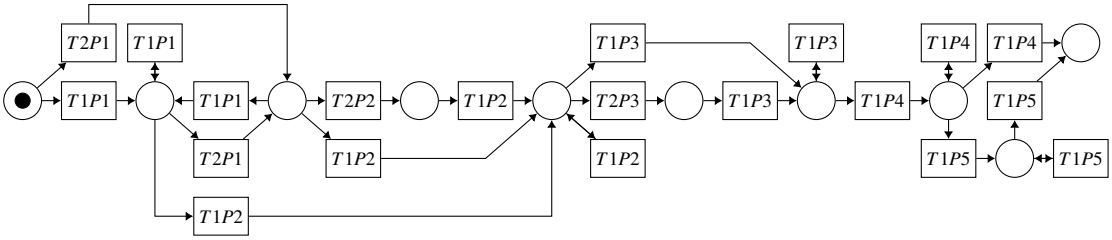


Figure 5.9 Process model M_1' discovered from event log L_1'.

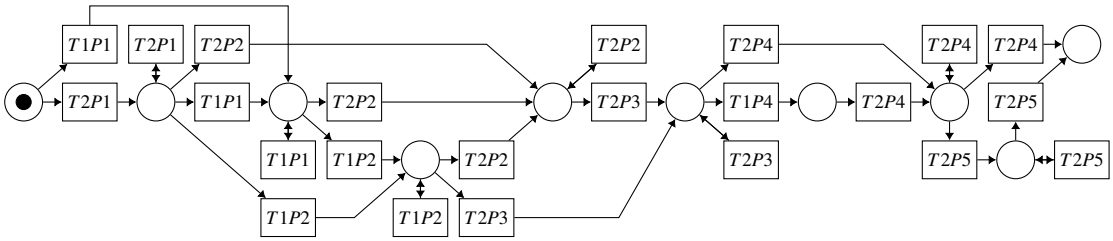


Figure 5.10 Process model M_2' discovered from event log L_2'.

Step 3: In this step, we conduct conformance checking to identify discrepancies between a newly observed trace and the discovered models. Once the GR system receives a new sequence of sensor data points recorded during a subject's movement, we use the trained classifier from step 1 to map the data points to events, resulting in a trace of events. For illustration purposes, consider we obtain event trace τ' shown below.

$$\tau' = \langle T1P1, T1P1, T2P1, T2P3, T2P3, T2P3, T2P4 \rangle .$$

Note that τ' is a partially observed trace, representing only a prefix of a full trace. The subject finally reached target pose T2 after completing that full trace. The optimal alignments between τ' and the two discovered models $M1'$ and $M2'$ are represented by symbols σ_2' and σ_2', as depicted below.

$$\sigma_1' = \begin{array}{|c|c|c|c|c|c|c|c|c|c|} \hline \tau' & T1P1 & T1P1 & T2P1 & \gg & T2P3 & \gg & \gg & T2P3 & T2P3 & T2P4 \\ \hline M_1' & T1P1 & T1P1 & T2P1 & T1P2 & T2P3 & T1P3 & T1P4 & \gg & \gg & \gg \\ \hline \end{array}$$

$$\sigma_2' = \begin{array}{|c|c|c|c|c|c|c|c|c|} \hline \tau' & T1P1 & T1P1 & \gg & T2P1 & T2P3 & T2P3 & T2P3 & T2P4 \\ \hline M_2' & T1P1 & T1P1 & T2P2 & \gg & T2P3 & T2P3 & T2P3 & T2P4 \\ \hline \end{array}$$

Step 4: Finally, in the last step, we compute the probabilities that the subject aims to reach target poses *T1* and *T2* using the diagnoses of the synchronous and asynchronous moves in the optimal alignments σ_1' and σ_2'. Assuming the asynchronous moves with skips in the model have the cost of one, and using the default parameters for the GR system, the alignment weights computed following Equation 3.5 are 110.75 and 53, respectively. Then, we use Equations (3.3) and (3.4) to compute the probabilities of the subject reaching *T1* and *T2* after observing trace τ', which are equal to 0.26 and 0.74, respectively. These probabilities indicate that, based on the sensor data, the subject is more likely to reach target pose *T2*.

5.4 Evaluation

In this section, we compare the performance of our PM-based GR approaches for recognizing target poses with three baselines. The experiments are conducted in two settings: offline experiments, as described in Section 5.1.1, and human-in-the-loop (HITL) experiments, as detailed in Section 5.1.2. In the offline experiments, we evaluate two PM-based GR approaches presented in Section 5.3 by comparing their performance with three baselines, namely the LSTM-based approach [57], the static LDA classifier [142], and the dynamic LDA classifier (an improved version of the static LDA approach described in Section 5.2). In the HITL experiments, we compare the two best-performing approaches from the offline experiments: the PM-based approach that uses an LDA classifier to convert sensor data into events and the dynamic LDA classifier. When compared, the different GR approaches were trained using (possibly different parts of) data collected during the same arm movements toward target poses.

For convenience, we denote the five evaluated GR approaches as follows:

1. $PM_{clustering}$ denotes the PM-based GR approach that uses clustering to convert sensor data into events;
2. $PM_{classifier}$ indicates the PM-based GR approach that uses an LDA classifier to convert sensor data into events;
3. *LSTM* signifies the LSTM-based approach for target pose recognition;
4. *sLDA* represents the static LDA classifier for target pose recognition; and
5. *dLDA* denotes the dynamic LDA classifier for target pose recognition.

5.4.1 Performance Measures

To assess the quality of goal inferences by the evaluated techniques, we use the F_1 score and balanced accuracy. These measures are computed based on four terms: True Positive, True Negative, False Positive, and False Negative. The True Positive (TP) term denotes the number of correct goals inferred by the GR system. The True Negative (TN) component is the number of incorrect goals that were not inferred. The False Positive (FP) term represents the number of incorrect goals inferred by a GR system. Finally, the False Negative (FN) component refers to the number of correct goals that were not recognized by the system.

Given the four terms, the F_1 score is computed as follows:

$$F_1 = \frac{2 \times TP}{2 \times TP + FP + FN}.$$

Balanced accuracy (*bacc*) is computed as follows:

$$bacc = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right).$$

The F_1 score is also known as the harmonic mean of precision and recall, where precision is the fraction of the correctly inferred target poses among the total number of poses that were inferred, and recall is the fraction of the correctly inferred target poses among the relevant poses. Balanced accuracy is well suited for measuring performance in imbalanced scenarios, like the case when there is only one true target pose.

Both offline and HITL experiments include multiple GR problem instances, where a problem instance comprises one attempt to recognize the target pose at some stage during the movement toward that pose. For each problem instance, we compute F_1 score and balanced accuracy, subsequently calculating the averages across all problem instances for each individual subject.

5.4.2 Baselines

We compare our PM-based GR approaches with three baselines: the LSTM-based approach, the static LDA approach, and the dynamic LDA approach.

LSTM neural networks, tailored to recognize dependencies and patterns in sequential data, have proven exceptionally adept at classifying multi-dimensional, continuous, real-value measurements, such as sEMG and kinematic sensor data. In our experiments, we adopt configurations and hyperparameters outlined in [57] for implementing the LSTM-based GR baseline.

LDA functions are trainable classifiers that utilize linear decision boundaries to categorize multi-dimensional, continuous, real-valued data points into pre-defined clusters. They are effective in analyzing individual data points, such as sEMG and kinematic signals, captured at specific moments. For our experiments, we implemented the static LDA classifier baseline detailed in [141] and the dynamic LDA classifier baseline discussed in Section 5.2. The static and dynamic LDA classifiers differ in the data that is used for training. The static LDA classifier is trained on data collected from the arm being held in target poses, using ten data points from each pose. The dynamic LDA classifier, however, is trained on data collected during the movement of the arm toward the target poses, capturing the arm's motion dynamics.

5.4.3 Offline Experiments

The offline experiments involve ten subjects (subject IDs from 1 to 10) completing three tasks, each task requiring the subject to move their hands to reach one of the three target positions, T1, T2, or T3 (the tasks are described in Section 5.1.1). Each subject is required to complete each reaching task 30 times, resulting in the collection of 90 trajectories for each subject in total. We cross-validate the recognition performance at an individual level by splitting the collected trajectories into training and testing sets. For each subject, we conduct experiments over 30 iterations. In each iteration, we select three traces, with one target pose per trace, to serve as testing instances. The remaining 87 traces, calculated as $3 \times (30-1)$ traces, are used for training the GR systems.

As $PM_{clustering}$ (Section 5.3.1) contains two key parameters, the number of selected features N_f and the number of clusters N_c, we use a brute force search methodology to determine the optimal combination of these parameters. In the experimental dataset, which comprises 47 distinct features, we set the selection range for the number of features (N_f) to any integer from 1 to 47, inclusive. For the number of discrete event clusters (N_c), we chose values from the set $\{10, 20, \ldots, 200\}$, in increments of 10. To identify the optimal combination of N_f and N_c that maximizes F_1 score, we systematically evaluate each pair against the F_1 score. We evaluate and compare all approaches, including $PM_{clustering}$, $PM_{classifier}$, *LSTM*, *sLDA*, and *dLDA*. Each approach is implemented and running on a cloud server, using a single core of an Intel® Xeon Processor at 2.0GHz, and uses the same set of selected features. As GR techniques aim to identify goals before the full sequences of signals are observed, we evaluate the approaches using prefixes of the full traces that are cut off at the first 10%, 30%, 50%, and 70% of the full sequences.

Table 5.5 displays the average F_1 score and balanced accuracy (*bacc*) achieved by each individual subject in recognizing target poses. These five distinct target pose recognition approaches are evaluated at different prefix lengths (10%, 30%, 50%, and 70% of the full trace). The two columns "Features" and "Clusters" show the number of selected features, N_f, and the number of discrete event clusters, N_c, for each subject. The last row shows the average across all subjects and all levels of observation.

subject ID	features	clusters	obs	$PM_{clustering}$		*LSTM*		*sLDA*		*dLDA*		$PM_{classifier}$	
				F_1 score	*bacc*	F_1 score	*bacc*	F_1 score	*bacc*	F_1 score	*bacc*	F_1 score	*bacc*
1	29	70	10%	0.519	0.525	0.356	0.517	0.333	0.500	**0.544**	**0.658**	0.533	0.547
			30%	0.515	0.567	0.444	0.583	0.344	0.508	**0.633**	**0.725**	0.604	0.700
			50%	0.570	0.656	0.389	0.542	0.367	0.525	0.689	0.767	**0.744**	**0.808**
			70%	0.691	0.758	0.544	0.658	0.467	0.600	0.756	0.817	**0.796**	**0.847**
2	1	10	10%	**0.489**	0.497	0.311	0.483	0.333	**0.500**	0.311	0.483	0.333	0.486
			30%	**0.504**	0.533	0.333	0.500	0.333	0.500	0.444	**0.583**	0.422	0.561
			50%	**0.544**	0.558	0.333	0.500	0.378	0.533	0.511	**0.633**	0.519	0.631
			70%	**0.572**	0.589	0.333	0.500	0.422	0.567	0.522	0.642	0.530	**0.647**
3	2	150	10%	0.491	0.494	0.322	0.492	0.267	0.450	0.344	**0.508**	**0.494**	0.506
			30%	**0.563**	0.597	0.356	0.517	0.322	0.492	0.456	0.592	0.528	**0.628**
			50%	**0.604**	0.678	0.311	0.483	0.367	0.525	0.600	**0.700**	0.552	0.650
			70%	**0.652**	**0.731**	0.333	0.500	0.422	0.567	0.567	0.675	0.633	0.714
4	34	50	10%	0.498	0.508	0.444	0.583	0.389	0.542	**0.511**	**0.633**	0.467	0.492
			30%	0.500	0.533	0.467	0.600	0.344	0.508	0.556	0.667	**0.589**	**0.681**
			50%	0.519	0.572	0.489	0.617	0.400	0.550	0.500	0.625	**0.541**	**0.639**
			70%	0.589	0.653	0.511	0.633	0.567	0.675	0.578	0.683	**0.622**	**0.708**
5	32	90	10%	0.511	0.531	0.356	0.517	0.244	0.433	0.544	**0.658**	**0.556**	0.614
			30%	0.617	0.686	0.478	0.608	0.289	0.467	0.522	0.642	**0.619**	**0.700**
			50%	**0.630**	**0.708**	0.400	0.550	0.300	0.475	0.578	0.683	0.604	0.700
			70%	**0.811**	**0.858**	0.522	0.642	0.556	0.667	0.544	0.658	0.622	0.714
6	28	160	10%	**0.483**	0.497	0.367	0.525	0.367	0.525	0.400	**0.550**	0.478	0.528
			30%	0.496	0.567	0.378	0.533	0.344	0.508	0.456	0.592	**0.511**	**0.617**
			50%	0.496	0.592	0.422	0.567	0.378	0.533	0.567	0.675	**0.574**	**0.675**
			70%	0.600	0.681	0.489	0.617	0.622	0.717	0.589	0.692	**0.633**	**0.725**
7	22	80	10%	0.522	0.542	0.367	0.525	0.322	0.492	0.467	**0.600**	**0.528**	0.597
			30%	0.493	0.550	0.422	0.567	0.344	0.508	0.522	0.642	**0.619**	**0.694**
			50%	0.520	0.608	0.400	0.550	0.456	0.592	**0.733**	**0.800**	0.696	0.769
			70%	0.554	0.631	0.400	0.550	0.522	0.642	0.600	0.700	**0.704**	**0.769**
8	34	100	10%	0.456	0.489	0.433	0.575	0.300	0.475	0.433	0.575	**0.507**	**0.586**
			30%	0.494	0.597	0.411	0.558	0.311	0.483	**0.644**	**0.733**	0.630	0.717
			50%	0.572	0.667	0.511	0.633	0.378	0.533	0.633	0.725	**0.719**	**0.786**
			70%	0.707	0.781	0.522	0.642	0.589	0.692	**0.756**	**0.817**	0.719	0.789
9	23	100	10%	0.498	0.503	0.411	0.558	0.400	0.550	0.500	**0.625**	**0.506**	0.519
			30%	0.528	0.564	0.356	0.517	0.344	0.508	**0.589**	**0.692**	0.557	0.658
			50%	0.457	0.542	0.344	0.508	0.367	0.525	**0.611**	**0.708**	0.607	0.700
			70%	0.467	0.558	0.411	0.558	0.567	0.675	0.678	0.758	**0.704**	**0.778**
10	28	170	10%	**0.500**	0.500	0.467	0.600	0.322	0.492	0.489	**0.617**	0.467	0.481
			30%	0.648	0.714	0.511	0.633	0.378	0.533	0.589	0.692	**0.657**	**0.733**
			50%	0.733	**0.794**	0.578	0.683	0.311	0.483	0.700	0.775	**0.733**	0.792
			70%	**0.867**	**0.900**	0.656	0.742	0.533	0.650	0.678	0.758	0.711	0.783
average				0.562	0.613	0.422	0.567	0.390	0.543	0.559	**0.669**	**0.589**	0.667
standard deviation				0.030	0.033	0.026	0.019	0.030	0.023	0.033	0.025	0.032	0.031

Table 5.5 Average F_1 score and balanced accuracy (*bacc*) for individual subjects at different levels of observation (highest in bold).

We conducted t-tests to assess the statistical significance of differences in average F_1 score and *bacc* between the two PM-based GR approaches and the three other baselines. The null hypothesis of the t-tests is that there is no significant difference between the average F_1 score and *bacc*. The t-tests comparing the average F_1 score and *bacc* across all the five approaches yield pairwise p-values, which are presented in Table 5.6 and Table 5.7 in a matrix format. The bottom-left corner of the matrix shows the p-values between each pair of approaches among the total five approaches, while the upper-right corner of the matrix indicates whether the two approaches are significantly different from each other. If the

p-value is less than 0.05,[5] we use the symbol "diff" to represent that the two approaches are significantly different (rejecting the null hypothesis); otherwise, we use "—" to represent that they are not significantly different (cannot reject the null hypothesis).

	$PM_{clustering}$	*LSTM*	*sLDA*	*dLDA*	$PM_{classifier}$
$PM_{clustering}$		diff	diff	diff	—
LSTM	3.767e-07		diff	diff	diff
sLDA	6.244e-19	5.593e-14		diff	diff
dLDA	2.600e-03	6.741e-04	1.093e-19		—
$PM_{classifier}$	1.022e-01	3.437e-05	1.018e-18	1.358e-01	

Table 5.6 The pairwise t-test results for comparing the average F_1 score across the five approaches.

	$PM_{clustering}$	*LSTM*	*sLDA*	*dLDA*	$PM_{classifier}$
$PM_{clustering}$		diff	diff	—	—
LSTM	1.458e-02		diff	diff	—
sLDA	2.442e-10	5.593e-14		diff	diff
dLDA	4.579e-01	6.741e-04	1.093e-19		—
$PM_{classifier}$	3.397e-01	1.433e-01	1.426e-09	6.183e-01	

Table 5.7 The pairwise t-test results for comparing the average balanced accuracies (*bacc*) across the five approaches.

The $PM_{classifier}$ achieves the highest F_1 score, while the *dLDA* approach achieves the highest *bacc* based on the average performance across all subjects. However, according to the results of the t-test, these two approaches are not statistically significantly different from each other in the offline experiment settings. In the next section, we use these two approaches to conduct HITL experiments to compare their performance further. Note that the number of selected features varies significantly between individual subjects. This may be due to unique patterns of muscle and kinematic activity exhibited by each subject when moving their arms, potential difficulties encountered by the sensors during data collection, or limitations in the brute-force search method used to find the optimal combination of features and clusters. While the exact cause remains uncertain, we acknowledge this as an area for further exploration.

5.4.4 Human-In-The-Loop Experiments

The human-in-the-loop (HITL) experiments include six additional subjects (subject IDs 11 to 16, none of whom had prior experience with our designed experiment), performing

[5]The t-tests use the Šidák correction at a 95% confidence level.

the Refined Clothespin Relocation Task (RCRT) as detailed in Section 5.1.2. The HITL experiment contains two phases, the training phase and the testing phase.
Training Phase: During this phase, all sensors are activated to collect signals with 59 features, while the virtual avatar (Figure 5.1c) visible in the VR environment is designed to mirror the real arm's movements. The wrist of the subject is constrained using a brace, only allowing forearm rotation movement. Subjects are instructed to complete all tasks across ten iterations, with each iteration consisting of eight distinct goals performed only once. All signals collected during this phase are used to train the two best-performing approaches identified in the offline experiment—$PM_{classifier}$ and *dLDA*—as we aim to compare these approaches in the HITL experiment settings
Testing Phase: During this phase, subjects are required to use the two trained approaches to control the virtual prosthesis, see Figure 5.2a. For each approach, subjects are instructed to complete five iterations of the same task as they performed during the training phase. However, in the testing phase, kinematic sensors below the elbow joint are deactivated to simulate scenarios of patients with disability. We observe and collect data to assess whether the "simulated disabled patients" can successfully complete the tasks using our approaches. Note that the subjects test the $PM_{classifier}$ and *dLDA* approaches in random order; half of the cohort began with $PM_{classifier}$, while the rest started with *dLDA*. This is intended to minimize any learning effects where experience gained from the first approach could improve performance in the second, ensuring a fair comparison. We use the same performance measures, F_1 score and *bacc*. Additionally, we measure the average time spent by subjects to pick up each clothespin and relocate it using the provided approaches. Note that, during HITL experiments, we conduct parallel alignment weight calculations to enhance reaction speeds. While the offline experiments were run on a cloud server, the HITL experiments were conducted on a lab PC with an Intel® Core™ i7-8700K processor at 3.7GHz, for easy interaction with the subjects.

Table 5.8 shows the performance of the $PM_{classifier}$ approach and the *dLDA* approach in the HITL experiment. We present the F_1 score, *bacc*, and average task completion time for the six subjects (subject IDs from 11 to 16) using these two approaches for executing each task. The third-to-last row, labeled 'average,' represents the average F_1 score, *bacc*, and completion time across the six subjects for each approach, $PM_{classifier}$ and *dLDA*. The second-to-last row, labeled 'std.,' displays the standard deviation for all measures across the subjects. The final row, labeled 'p-value,' presents the p-value from a t-test comparing the average performance of the two approaches, measured by F_1 score, *bacc*, and completion time, to determine if the differences are statistically significant. Table 5.8 demonstrates that, on average across all subjects, $PM_{classifier}$ outperforms the *dLDA* baseline significantly in

subject ID	F_1 score		*bacc*		Time	
	$PM_{classifier}$	*dLDA*	$PM_{classifier}$	*dLDA*	$PM_{classifier}$	*dLDA*
11	0.353	0.246	0.630	0.569	9.348	10.793
12	0.287	0.252	0.590	0.573	8.872	8.078
13	0.433	0.286	0.678	0.592	10.948	11.316
14	0.490	0.375	0.709	0.643	8.577	8.778
15	0.559	0.254	0.751	0.573	11.729	25.084
16	0.404	0.271	0.662	0.583	6.986	7.616
average	0.421	0.281	0.670	0.589	9.410	11.944
std.	0.097	0.049	0.057	0.028	3.656	21.114
p-value	0.010		0.011		0.196	

Table 5.8 Average F_1 score, balanced accuracy (*bacc*), and task completion time for subjects using the $PM_{classifier}$ and *dLDA* approaches in the HITL experiments.

terms of the F_1 score and *bacc*. Furthermore, subjects utilizing the prosthesis controlled by the $PM_{classifier}$ approach complete tasks faster, on average, with a smaller standard deviation. This suggests that the $PM_{classifier}$ approach is not only quicker but also less sensitive to human variations compared to the *dLDA* baseline. These findings underscore the potential of the proposed method $PM_{classifier}$ in the application of prosthetic devices.

Chapter 6

Conclusion

This thesis introduces the evidence-based GR framework that uses process mining techniques to learn the knowledge model and infer the goals. We instantiated several GR systems from this framework, including the GR system to solve single-shot GR problems and the systems addressing adaptive GR challenges over extended periods. Through synthetic and real-world experiments, we verified that the evidence-based GR framework successfully addresses the challenges faced by existing GR techniques, including reliance on hand-crafted models, model generalization, and model explainability. Furthermore, extensive testing and case studies, particularly in the context of powered transhumeral prostheses, validate the framework's potential benefits in real-world scenarios. Therefore, we claim that our proposed GR approach, which uses process mining to automatically learn explainable models, provides a novel contribution to the GR research field. The main contributions of this thesis are presented in Chapters 3-5. In this conclusion chapter, we summarize the core content from these chapters, address frequently asked questions raised by readers, and acknowledge the potential limitations of our findings. Finally, we highlight directions for future work.

6.1 Summary

In Chapter 3, we introduced a solution to the GR problem that does *not* rely on pre-defined models of behavior, such as plan libraries or domain dynamic descriptions (e.g., planning domains). Instead, our approach leverages recorded past behaviors, represented as collections of event traces, to automatically learn skill models using process discovery techniques. The agent's goal is then inferred by checking conformance, or aligning, observations against these skill models. To represent skill models compactly as sequences of actions, we used Petri nets, which facilitate efficient alignment and conformance checking. This approach capitalizes on the availability of logs of past behaviors, which are often accessible or can

be collected with reasonable effort, unlike plan libraries or domain descriptions that may be costly or unavailable in many real-world domains. Our experiments used the Directly Follows Miner (DFM) [72] as the primary process discovery technique. While DFM excels in simplicity and speed, it can produce overly complex models when applied to logs with high behavioral variability. Nonetheless, in datasets with clear sequential patterns, DFM provided accurate models. We demonstrated that our approach achieved recognition accuracy comparable to state-of-the-art GR methods that rely on full domain knowledge, with superior recognition speed. Additionally, the method was validated on real-world datasets where domain knowledge was unavailable, underscoring its practicality and scalability. Finally, we performed sensitivity analysis on the four parameters of our system, using the Patient Rule Induction Method (PRIM) to identify configurations yielding high GR accuracy. This analysis confirmed that the method's performance is robust, even with limited past behavioral data.

In Chapter 4, we examined the GR problem over extended periods, where the environment in which the observed agent operates may change. This eliminates the assumption made in RQ1 (Chapter 3) that the environment is stationary. We first define the *adaptive* GR problem in an abstract manner that suits, among others, standard definitions of probabilistic goal recognition. Roughly speaking, the *adaptive* GR problem amounts to solving the current GR instance task *in the context of previously solved problem instances*; the various instances may (slightly) differ in the underlying environment as well as in the true agent's goal. We then presented three strategies for adaptive GR systems to address this challenging problem. To accurately recognize the goals in changing environments, the proposed GR systems were designed as both open- and closed-loop control systems that measure their recognition accuracy over time and react by relearning their knowledge base if the recognition accuracy becomes unsatisfactory. We conducted an experimental evaluation to validate the effectiveness of our proposed GR systems in addressing *adaptive* GR problems and showed that, indeed, they achieve better overall recognition accuracies across the target time intervals. Furthermore, we compared our (adaptive) GR systems based on the *effort* they require to achieve their performance, as the number of times they update their knowledge base. For the experiments conducted in synthetic domains, we use a collection of adaptive GR problem instances generated by the GRACE tool. Additionally, we assessed the performance of the proposed adaptive GR systems in real-world domains where the problem instances are modified from existing business logs. The evaluation results demonstrate a trade-off between the GR performance over time and the effort invested in adaptations of the GR mechanisms of the system, showing that few well-planned adaptations can lead to a consistently high GR performance.

In Chapter 5, we explore the applicability of the evidence-based GR framework in identifying target poses for users employing powered transhumeral prosthetics, aimed at restoring missing anatomical segments below the shoulder, including the hand, using continuous, real-valued data from surface electromyography electrodes and kinematic sensors. We propose methods to transform sensor data into discrete events and incorporate these methods with evidence-based GR techniques to develop target pose recognition approaches. Two data transforming approaches–clustering, which extracts features without reliance on target poses, and linear discriminant analysis (LDA) classification, which is based on target poses–are proposed. In addition, we propose a dynamic LDA approach, which uses the entire time series of measurements to identify classes, as an enhancement of the existing static LDA approach. These methods are evaluated through offline datasets and online human-in-the-loop experiments, comparing them with established techniques such as static LDA and approaches based on neural networks. Real-time human-in-the-loop experiments further validate the effectiveness of the proposed methods, demonstrating that the GR system with an LDA classifier achieves superior harmonic mean of precision and recall (the F_1 score) and balanced accuracy compared to state-of-the-art baselines.

6.2 Discussion

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. The research on cognitive architectures [8, 68, 43, 35] attempts to model the core capabilities of humans, including, but not limited to, perception, attention mechanisms, action selection, learning, memory, reasoning, and metareasoning [65]. Our GR framework can be seen as a goal-intention recognition module of a cognitive architecture, such as ACT-R [8] and Soar [68]. However, although our proposed evidence-based GR framework is inspired by observational learning theory, we do not aim to introduce a novel cognitive architecture like SOAR or ACT-R, which is beyond the scope of the thesis.

Process discovery resembles action model learning [41, 109, 70, 87]. 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, discovery techniques are also designed to work with human-driven processes and, thus, are made to be robust towards 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 [13, 54], while Petri nets have been compiled into planning models to facilitate process analysis using planning technology [32, 27]. 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.

Although the framework is primarily designed for goal prediction, it can also be used for plan recognition by identifying the remaining steps to achieve a goal through optimal alignment techniques. Instead of inferring goals, the GR system can return skill models and remove misaligned branches, from which plans can be inferred. However, learning from noisy or incomplete observations may lead to two issues: (i) missing actions result in asynchronous moves, increasing alignment weight and reducing accuracy, and (ii) incorrectly included actions do not impact accuracy, as they are not penalized in the trace.

The evidence-based GR approach requires historically observed traces as input to learn skill models. These traces are pre-classified based on the final goal state they achieve. The GR system then learns and stores multiple skill models, each describing the actions required to achieve a specific goal. However, it could be more useful and robust if the GR system does not require pre-classified traces. In this case, traces with different goals would be mixed in a single event log, and the GR system would discover a general model that formulates the transitions between states. This means that, in any given state, only a few valid actions would be available to transition to the next state. This is commonly observed in business process management systems, where the process model could be augmented with AI techniques to automatically unfold traces and generate meaningful processes to accomplish specific tasks. This challenge is referred to as *process framing* [31]. One relevant AI technique is planning algorithms, which aim to find optimal or near-optimal plans for task completion. Planning algorithms often rely on heuristic functions to guide action selection. Similarly, in process framing tasks, incorporating a goal-oriented perspective with heuristics could provide valuable insights to guide the unfolding algorithm in finding desirable processes from the process model.

GR in multi-agent systems is another hot research area focused on inferring agents' goals from their observable actions and interactions. In the field of process mining, there is also research focused on discovering multi-agent systems, such as the work on *agent miner* [119],

which aims to extract models from environments where multiple agents collaborate. The algorithm generates a high-level process model that encapsulates the entire system, with each agent's behavior represented as a low-level process model. Each of these low-level models corresponds to the actions of a single agent, and together, they form a comprehensive view of how agents interact and collaborate to accomplish tasks. The resulting high-level model provides an integrated description of the entire system, illustrating how individual agents' actions contribute to the collective achievement of goals within the system.

6.3 Limitations and Future Work

The main process discovery technique we used is the Directly Follow Miner (DFM) [72]. While DFM excels in simplicity and the directness of model generation, its major limitation lies in its potential to generate overly complex models in cases of logs with high variability in behavior. However, for logs with clear and strong sequential patterns, DFM offers a quick and accurate snapshot of process flows. We have not tested with a wider range of process discovery techniques, including learning and using stochastic process models [4]. Stochastic process mining can discover models that encode information about behavior frequencies. For instance, if we observe a certain action sequence repeatedly, the model will reflect these traces as high-frequency occurrences with corresponding high probabilities. Additionally, exploring conformance checking techniques that leverage frequency information to compute probability distributions could yield valuable insights [73]. Therefore, applying stochastic process mining techniques to implement evidence-based GR systems can be an interesting direction for future study.

In Chapter 3, we introduce four key parameters in our approach. To assess their impact on GR accuracy, we conducted a sensitivity analysis and used the Patient Rule Induction Method (PRIM) to identify the parameter settings that yield the highest accuracy. Our experiments demonstrate that, despite relying on a limited number of past behaviors, our approach achieves accuracy comparable to state-of-the-art GR methods that have full access to domain knowledge, while also offering faster recognition speeds. While the PRIM algorithm, a result-oriented method, improves performance through iterative experimentation, we believe an ideal parameter optimization method should not rely on experimental results. Thus, another interesting direction for future research could involve exploring methods for optimizing system parameters or investigating the development of parameter-free GR models.

Our GR approach, like most existing methods, assumes pre-defined goal candidates, limiting its ability to identify new or previously unseen goals. The reliance on pre-defined target poses for training, as discussed in Chapter 5, limits the system's ability to generalize

to arbitrary or unseen target poses. Extending the framework to recognize and incorporate unseen goals would enhance its flexibility and applicability in dynamic environments. Identifying new goals that agents aim to achieve, particularly in scenarios where the goals have not been observed before, presents an important direction for future research.

In Section 6.2, we discussed how business process management systems can be augmented with AI techniques like planning to frame processes. However, planning typically requires a domain model, such as a PDDL model, which is highly abstract and difficult to apply to real-world scenarios. In contrast, using process mining techniques to derive models from event logs can result in a Petri net, which represents only a partial view of the domain (not covering the entire state space described by a PDDL model). The advantage of Petri nets is that they can be generated automatically using process discovery algorithms. This stresses the importance of two research directions: (1) how to mine process models with state information, and (2) how to define heuristic functions to guide the search algorithm in finding a path from the current state to the goal state within the process model.

The adaptive GR framework we present in Chapter 4 uses feedback on GR performance to detect changes in the environment. Exploring and developing more effective concept drift detection techniques could be an interesting direction for future work, offering potential benefits for solving adaptive GR problems. Exploring and applying process model repair techniques for updating process models could be another interesting direction. This approach may improve overall GR performance while minimizing the costs associated with updating the process models. Moreover, the three adaptive GR systems presented in Section 4.2 can be further refined. For example, when the systems update the learned process models, they relearn new models from scratch. An alternative, arguably more efficient, way forward is to use process model repair techniques [33, 97] to update the models incrementally.

For the adapttive GR problem, We assume that the feedback on the solution to each GR problem can be obtained (in the form of the true goal the agent eventually achieved) before the next GR problem is to be solved. In future work, one can generalize our setup to account for situations when certain true goals cannot be attained, or the next GR problem must be solved before the solutions or true goals for the previous problems are available.

In the case study presented in Chapter 5, the current feature selection and event discretization approaches are arguably simple and we would like to explore elaborate feature selection and event discretization. The brute-force search approach, used to determine which features to select and how many clusters are needed for grouping the data points to convert original signals into events, could be replaced by a more efficient method. For the PM-based GR approach using an LDA classifier, exploring alternative machine learning-based classifiers could be interesting. Additionally, the method for partitioning the trajectory and labeling data

points could be revised; there may be other innovative ways to label data points for training classifiers. Furthermore, it would be interesting to identify feature conditions capturing *meaningful* prosthetic postures/configurations, and test whether sequences of these yield good predictors for the goal being pursued. Doing so will also allow us to take advantage of one of the key features of our PM-based GR approach, namely, the possibility of explaining the outcome of the system based on the process model and misalignment of the observed behavior.

Yet another area for further exploration is the testing of alternative alignment approaches. While process discovery demands linear time [72], alignment techniques—including the one we used in this work [126]—are often exponential in the worst case. For HITL experiments, as our approach requires alignments to be computed at recognition time, fast extraction is crucial for real-time applicability. However, worst-case scenarios often do not occur in practice, and by computing alignments in parallel, we effectively improve reaction times. Our HITL experiments demonstrate that our approaches can respond within the necessary timeframe, thus performing faster and more accurately than the baselines. Nonetheless, we would like to experiment with techniques that are specifically designed for *online* conformance checking [129] as well as those that seek *approximate* alignments, but *fast* [128]. Indeed, we conjecture online conformance checking approaches may be a good fit for our GR setting, since observations are incrementally extended and such approaches extract alignments incrementally by re-using previously computed ones. Another potential challenge arises with our system learning numerous process models, as the recognition phase requires substantial computational resources for parallel alignment weight computations, where insufficient resources may slow reaction speeds. To address this, we could explore smarter and more efficient methods for parallelizing these computations.

The datasets for both offline and HITL experiments are arguably small, with the offline experiment using an existing dataset of 10 subjects and the HITL experiments including 6 subjects. The positive outcomes achieved so far encourage expanding the study to include a larger number of subjects to confirm the results' generalizability. Another aspect worth further exploration is the significant performance gap between the offline experiments and the HITL experiments. This discrepancy might be due to the dataset capturing features from forward-reaching tasks using *sound* limbs, rather than from real-time control of a prosthesis. The difference between real-time prosthetic states and sound limbs introduces variations in visual feedback, which could affect the feature patterns collected during movement. The feature patterns can have an impact on the real-time recognition accuracy of machine-learning-based techniques [139, 92]. Therefore, collecting data from real-time control experiments and pre-defining standard traces towards the goal with distinguishable feature patterns for

training the PM-based GR system has the potential to improve accuracy. The adaptive GR framework from Chapter 4 shows that if GR performance is below expectations, an adaptive GR system can automatically update its model to improve performance in current scenarios. Using the adaptive GR technique, we could start by learning GR models from the movements of sound limbs. Then, as subjects collaborate with the prostheses to perform tasks, the adaptive GR technique can adjust the initially learned models to improve human-in-the-loop recognition performance.

Bibliography

[1] Arya Adriansyah, Natalia Sidorova, and Boudewijn F. van Dongen. Cost-based fitness in conformance checking. In *Proceedings of the International Conference on Application of Concurrency to System Design (ACSD)*, pages 57–66, 2011.

[2] Simone Agostinelli, Francesco Chiariello, Fabrizio Maria Maggi, Andrea Marrella, and Fabio Patrizi. Process mining meets model learning: Discovering deterministic finite state automata from event logs for business process analysis. *Information Systems*, 114:102180, 2023.

[3] Hanan Alkhammash, Artem Polyvyanyy, Alistair Moffat, and Luciano García-Bañuelos. Entropic relevance: A mechanism for measuring stochastic process models discovered from event data. *Information Systems*, 107:101922, 2022.

[4] Hanan Alkhammash, Artem Polyvyanyy, and Alistair Moffat. Stochastic directly-follows process discovery using grammatical inference. In *Proceedings of the International Conference on Advanced Information Systems Engineering (CAiSE)*, volume 14663 of *Lecture Notes in Computer Science*, pages 87–103, 2024.

[5] Nasser A. Alshammary, Daniel A. Bennett, and Michael Goldfarb. Synergistic Elbow Control for a Myoelectric Transhumeral Prosthesis. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 26:468–476, 2018.

[6] Leonardo Amado, Ramon Fraga Pereira, João Aires, Mauricio Magnaguagno, Roger Granada, and Felipe Meneguzzi. Goal recognition in latent space. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2018.

[7] Leonardo Amado, Reuth Mirsky, and Felipe Meneguzzi. Goal recognition as reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 36:9644–9651, 6 2022.

[8] John R. Anderson. *How Can the Human Mind Occur in the Physical Universe?* Oxford University Press, 10 2007.

[9] Adriano Augusto, Raffaele Conforti, Marlon Dumas, Marcello La Rosa, and Artem Polyvyanyy. Split miner: Automated discovery of accurate and simple business process models from event logs. *Knowledge and Information Systems*, 59(2):251–284, 2019.

[10] Dorit Avrahami-Zilberbrand and Gal A. Kaminka. Fast and complete symbolic plan recognition. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 653–658, 2005.

[11] Chris L. Baker, Rebecca Saxe, and Joshua B. Tenenbaum. Action understanding as inverse planning. *Cognition*, 113(3):329–349, 2009.

[12] Albert Bandura. Observational learning. In *The International Encyclopedia of Communication*, 2008.

[13] Blai Bonet, Patrik Haslum, Sarah L. Hickmott, and Sylvie Thiébaux. Directed unfolding of Petri nets. *Transactions on Petri Nets and Other Models of Concurrency*, 1:172–198, 2008.

[14] Diana Borsa, Nicolas Heess, Bilal Piot, Siqi Liu, Leonard Hasenclever, Rémi Munos, and Olivier Pietquin. Observational learning by reinforcement learning. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 1117–1124, 2019.

[15] Michael E. Bratman. *Intentions, Plans, and Practical Reason*. Harvard University Press, 1987.

[16] Benjamin P. Bryant and Robert J. Lempert. Thinking inside the box: A participatory, computer-assisted approach to scenario discovery. *Technological Forecasting and Social Change*, 77(1):34–49, 2010.

[17] Daniel Bryce, J. Benton, and Michael W. Boldt. Maintaining evolving domain models. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2016.

[18] Sandra Carberry. *Plan Recognition in Natural Language Dialogue*. MIT Press, 1990.

[19] Sandra Carberry. Techniques for plan recognition. *User Modeling and User-Adapted Interaction*, 11:31–48, 2001.

[20] Josep Carmona, Boudewijn F. van Dongen, Andreas Solti, and Matthias Weidlich. *Conformance Checking — Relating Processes and Models*. Springer, 2018.

[21] Tathagata Chakraborti, Yu Zhang, David E. Smith, and Subbarao Kambhampati. Planning with resource conflicts in human-robot cohabitation. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 1069–1077, 2016.

[22] Tathagata Chakraborti, Sarath Sreedharan, Yu Zhang, and Subbarao Kambhampati. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.

[23] Eugene Charniak and Robert P. Goldman. A bayesian model of plan recognition. *Artificial Intelligence*, 64:53–79, 1993.

[24] Mattia Chiari, Alfonso Emilio Gerevini, Francesco Percassi, Luca Putelli, Ivan Serina, and Matteo Olivato. Goal recognition as a deep learning task: The grnet approach. *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 33:560–568, 7 2023.

[25] Alexandra Coman and Hector Muñoz-Avila. Generating diverse plans using quantitative and qualitative plan distance metrics. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2011.

[26] William H. E. Day and Herbert Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification*, 1:7–24, 1984.

[27] Massimiliano de Leoni, Giacomo Lanciano, and Andrea Marrella. Aligning partially-ordered process-execution traces and models using automated planning. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 321–329, 2018.

[28] Robert Demolombe and Erwan Hamon. What does it mean that an agent is performing a typical procedure? A formal definition in the situation calculus. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 905–911, 2002.

[29] Vadim Denisov, Elena Belkina, and Dirk Fahland. BPIC 2018: Mining concept drift in performance spectra of processes. 2018.

[30] Daniel Dennett. *The Intentional Stance*. MIT Press, 1987.

[31] Marlon Dumas, Fabiana Fournier, Lior Limonad, Andrea Marrella, Marco Montali, Jana-Rebecca Rehse, Rafael Accorsi, Diego Calvanese, Giuseppe De Giacomo, Dirk Fahland, Avigdor Gal, Marcello La Rosa, Hagen Völzer, and Ingo Weber. Ai-augmented business process management systems: A research manifesto. *ACM Transactions on Management Information Systems*, 14(1):11:1–11:19, 2023.

[32] Stefan Edelkamp and Shahid Jabbar. Action planning for directed model checking of Petri nets. *Electronic Notes in Theoretical Computer Science*, 149(2):3–18, 2006.

[33] Dirk Fahland and Wil M. P. van der Aalst. Model repair - Aligning process models to reality. *Information Systems*, 47:220–243, 2015.

[34] Richard E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[35] R. James Firby, Roger E. Kahn, Peter N. Prokopowicz, and Michael J. Swain. An architecture for vision and action. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 72–81, 1995.

[36] Jonas Firl and Quan Tran. Probabilistic maneuver prediction in traffic scenarios. In *Proceedings of the European Conference on Mobile Robots (ECMR)*, pages 89–94, 2011.

[37] Maria Fox, Alfonso Gerevini, Derek Long, and Ivan Serina. Plan stability: Replanning versus plan repair. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 212–221, 2006.

[38] Guillem Francés, Miquel Ramirez, and Collaborators. Tarski: An AI planning modeling framework. https://github.com/aig-upf/tarski, 2018.

[39] Chiara Di Francescomarino, Marlon Dumas, Fabrizio Maria Maggi, and Irene Teinemaa. Clustering-based predictive process monitoring. *IEEE Transactions on Services Computing*, 12(6):896–909, 2019.

[40] Jerome H. Friedman and Nicholas I. Fisher. Bump hunting in high-dimensional data. *Statistics and Computing*, 9(2):123–143, 1999.

[41] Ramón García-Martínez and Daniel Borrajo. An integrated approach of learning, planning, and execution. *Journal of Intelligent & Robotic Systems*, 29(1):47–78, 2000.

[42] Ricardo Garcia-Rosas, Tianshi Yu, Denny Oetomo, Chris Manzie, Ying Tan, and Peter Choong. Exploiting inherent human motor behaviour in the online personalisation of human-prosthetic interfaces. *IEEE Robotics and Automation Letters*, 6(2):1973–1980, 2021.

[43] Erann Gat. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 809–815, 1992.

[44] Purushothaman Geethanjali. Myoelectric control of prosthetic hands: State-of-the-art review. *Medical Devices: Evidence and Research*, 9:247–255, 2016.

[45] Christopher W Geib and Robert P Goldman. Partial observability and probabilistic plan/goal recognition. In *Proceedings of the International workshop on modeling other agents from observations (MOO)*, volume 8, pages 1–6, 2005.

[46] Christopher W. Geib and Robert P. Goldman. A probabilistic plan recognition algorithm based on plan tree grammars. *Artificial Intelligence*, 173:1101–1132, 2009.

[47] Malik Ghallab, Dana S. Nau, and Paolo Traverso. *Automated planning — Theory and practice*. Elsevier, 2004.

[48] Eun Y. Ha, Jonathan P. Rowe, Bradford W. Mott, and James C. Lester. Goal recognition with Markov logic networks for player-adaptive games. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 32–39, 2011.

[49] Levi J. Hargrove, Erik J. Scheme, Kevin B. Englehart, and Bernard S. Hudgins. Multiple binary classifications via linear discriminant analysis for improved controllability of a powered prosthesis. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 18(1):49–57, 2010.

[50] John A. Hartigan and Manchek A. Wong. Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1): 100–108, 1979.

[51] Patrik Haslum, Nir Lipovetzky, Daniele Magazzeni, and Christian Muise. *An Introduction to the Planning Domain Definition Language*. Morgan & Claypool Publishers, 2019.

[52] Jiayuan He, Xinjun Sheng, Xiangyang Zhu, Chaozhe Jiang, and Ning Jiang. Spatial Information Enhances Myoelectric Control Performance with only Two Channels. *IEEE Transactions on Industrial Informatics*, 15(2):1226–1233, 2019.

[53] J.C. Helton and F.J. Davis. Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems. *Reliability Engineering & System Safety*, 81 (1):23–69, 2003.

[54] Sarah L. Hickmott and Sebastian Sardiña. Optimality properties of planning via Petri net unfolding: A formal analysis. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2009.

[55] Bart Hompes, Joos C. A. M. Buijs, Wil M. P. van der Aalst, Prabhakar M. Dixit, and Hans Buurman. Detecting change in processes using comparative trace clustering. In *Proceedings of the International Symposium on Data-driven Process Discovery and Analysis*, volume 1527 of *CEUR Workshop Proceedings*, 2015.

[56] Jun Hong. Goal recognition through goal graph analysis. *Journal of Artificial Intelligence Research*, 15:1–30, 7 2001.

[57] Jin Huang, Guoxin Li, Hang Su, and Zhijun Li. Development and continuous control of an intelligent upper-limb neuroprosthesis for reach and grasp motions using biological signals. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(6): 3431–3441, 2021.

[58] Ali Hussaini and Peter Kyberd. Refined clothespin relocation test and assessment of motion. *Prosthetics and Orthotics International*, 41(3):294–302, 2017.

[59] Andres G. Jaramillo-Yanez, Marco E. Benalcázar, Sebastian Sardiña, and Fabio Zambetta. Towards discriminant analysis classifiers using online active learning via myoelectric interfaces. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 6996–7004, 2022.

[60] Yuqian Jiang, Harel Yedidsion, Shiqi Zhang, Guni Sharon, and Peter Stone. Multi-robot planning with conflicts and synergies. *Autonomous Robots*, 43(8):2011–2032, 2019.

[61] Michael Katz and Shirin Sohrabi. Reshaping diverse planning. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 34:9892–9899, 4 2020.

[62] Michael Katz, Shirin Sohrabi, Octavian Udrea, and Dominik Winterer. A novel iterative approach to top-k planning. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 132–140, 2018.

[63] Henry A. Kautz and James F. Allen. Generalized plan recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 32–37, 1986.

[64] Julian Francisco Pieter Kooij, Nicolas Schneider, Fabian Flohr, and Dariu M. Gavrila. Context-based pedestrian path prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 8694 of *LNCS*, pages 618–633, 2014.

[65] Iuliia Kotseruba and John K. Tsotsos. 40 years of cognitive architectures: Core cognitive abilities and practical applications. *Artificial Intelligence Review*, 53(1): 17–94, 2020.

[66] Alexander Kott and William McEneaney. *Adversarial reasoning: Computational approaches to reading the opponent's mind*. CRC Press, 2006.

[67] Jan H. Kwakkel. The exploratory modeling workbench: An open source toolkit for exploratory modeling, scenario discovery, and (multi-objective) robust decision making. *Environmental Modelling & Software*, 96:239–250, 2017.

[68] John E. Laird. Extending the soar cognitive architecture. In *Proceedings of the Artificial General Intelligence Conference*, volume 171 of *Frontiers in Artificial Intelligence and Applications*, pages 224–235, 2008.

[69] Geetika T. Lakshmanan, Songyun Duan, Paul T. Keyser, Francisco Curbera, and Rania Khalaf. Predictive analytics for semi-structured case oriented business processes. In *Proceedings of the BPM Workshops*, volume 66 of *LNBIP*, pages 640–651, 2010.

[70] Leonardo Lamanna, Alessandro Saetti, Luciano Serafini, Alfonso Gerevini, and Paolo Traverso. Online learning of action models for PDDL planning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4112–4118, 2021.

[71] Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. Process and deviation exploration with inductive visual miner. In *Proceedings of the BPM Demo Sessions*, volume 1295 of *CEUR Workshop Proceedings*, page 46, 2014.

[72] Sander J. J. Leemans, Erik Poppe, and Moe Thandar Wynn. Directly follows-based process mining: Exploration & a case study. In *Proceedings of the International Conference on Process Mining (ICPM)*, pages 25–32, 2019.

[73] Sander J. J. Leemans, Wil M. P. van der Aalst, Tobias Brockhoff, and Artem Polyvyanyy. Stochastic process mining: Earth movers' stochastic conformance. *Information Systems*, 102:101724, 2021.

[74] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. A survey on motion prediction and risk assessment for intelligent vehicles. *Robomech Journal*, 1(1):1–14, 2014.

[75] Mathilde Legrand, Charlotte Marchand, Florian Richer, Amélie Touillet, Noël Martinet, Jean Paysant, Guillaume Morel, and Nathanaël Jarrassé. Simultaneous control of 2DOF upper-limb prosthesis with body compensations-based control: A multiple cases study. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 30:1745–1754, 2022.

[76] Neal Lesh. Adaptive goal recognition. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1208–1214, 1997.

[77] Neal Lesh, Charles Rich, and Candace L. Sidner. Using plan recognition in human-computer collaboration. In *Proceedings of the CISM International Centre for Mechanical Sciences*, pages 23–32, 1999.

[78] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics. Doklady*, 10:707–710, 1965.

[79] Nir Lipovetzky and Hector Geffner. Best-first width search: Exploration and exploitation in classical planning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 3590–3596, 2017.

[80] Jianwei Liu, Xinjun Sheng, Dingguo Zhang, Jiayuan He, and Xiangyang Zhu. Reduced daily recalibration of myoelectric prosthesis classifiers based on domain adaptation. *IEEE Journal of Biomedical Health Informatics*, 20(1):166–176, 2016.

[81] Bo Lv, Guohong Chai, Xinjun Sheng, Han Ding, and Xiangyang Zhu. Evaluating User and Machine Learning in Short- And Long-Term Pattern Recognition-Based Myoelectric Control. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 29:777–785, 2021.

[82] Fabrizio Maria Maggi, Chiara Di Francescomarino, Marlon Dumas, and Chiara Ghidini. Predictive monitoring of business processes. In *Proceedings of the International Conference on Advanced Information Systems Engineering (CAiSE)*, volume 8484 of *LNCS*, pages 457–472, 2014.

[83] Peta Masters and Sebastian Sardiñ. Expecting the unexpected: Goal recognition for rational and irrational agents. *Artificial Intelligence*, 297:103490, 2021.

[84] Peta Masters and Sebastian Sardiña. Cost-based goal recognition for path-planning. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 750–758, 2017.

[85] Peta Masters and Sebastian Sardiña. Goal recognition for rational and irrational agents. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 440–448, 2019.

[86] Peta Masters and Sebastian Sardiña. Cost-based goal recognition in navigational domains. *Journal of Artificial Intelligence Research*, 64:197–242, 2019.

[87] Viraj Mehta, Biswajit Paria, Jeff Schneider, Stefano Ermon, and Willie Neiswanger. An experimental design perspective on model-based reinforcement learning. In *Proceedings of the International Conference on Learning Representations*, 2022.

[88] Wookhee Min, Bradford Mott, Jonathan Rowe, Barry Liu, and James Lester. Player goal recognition in open-world digital games with long short-term memory networks. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2590–2596, 2016.

[89] Reuth Mirsky, Sarah Keren, and Christopher Geib. *Introduction to Symbolic Plan and Goal Recognition*, pages 1–24. Springer International Publishing, 2021.

[90] Jeffrey P Mower. PREP-mt: predictive RNA editor for plant mitochondrial genes. *BMC Bioinformatics*, 6(1), apr 2005.

[91] Allen Newell, John C. Shaw, and Herbert A. Simon. Report on a general problem-solving program. *Proceedings of the International Conference on Information Processing*, pages 256–264, 1959.

[92] Max Ortiz-Catalan, Faezeh Rouhani, Rickard Brånemark, and Bo Håkansson. Offline accuracy: A potentially misleading metric in myoelectric pattern recognition for prosthetic control. In *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 1140–1143, 2015.

[93] Nawadita Parajuli, Neethu Sreenivasan, Paolo Bifulco, Mario Cesarelli, Sergio Savino, Vincenzo Niola, Daniele Esposito, Tara J. Hamilton, Ganesh R. Naik, Upul Gunawardana, and Gaetano D. Gargiulo. Real-time EMG based pattern recognition control for

hand prostheses: A review on existing methods, challenges and future implementation. *Sensors*, 19(20):4596, 2019.

[94] David Pattison and Derek Long. Accurately determining intermediate and terminal plan states using Bayesian goal recognition. In *Proceedings of the First Workshop on Goal, Activity and Plan Recognition (GAPRec)*, pages 32–37. June 2011.

[95] Ramon Fraga Pereira, Nir Oren, and Felipe Meneguzzi. Landmark-based heuristics for goal recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 31, 2 2017.

[96] Ramon Fraga Pereira, Nir Oren, and Felipe Meneguzzi. Landmark-based approaches for goal recognition as planning. *Artificial Intelligence*, 279:103217, 2020.

[97] Artem Polyvyanyy, Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and Moe Thandar Wynn. Impact-driven process model repair. *ACM Transactions on Software Engineering and Methodology*, 25(4):28:1–28:60, 2017.

[98] Artem Polyvyanyy, Andreas Solti, Matthias Weidlich, Claudio Di Ciccio, and Jan Mendling. Monotone precision and recall measures for comparing executions and specifications of dynamic systems. *ACM Transactions on Software Engineering and Methodology*, 29(3):17:1–17:41, 2020.

[99] Alberto Pozanco, Yolanda E-Martín, Susana Fernández, and Daniel Borrajo. Counter-planning using goal recognition and landmarks. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4808–4814, 4 2018.

[100] David V. Pynadath and Michael P. Wellman. Probabilistic state-dependent grammars for plan recognition. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 507–514, 2000.

[101] Miquel Ramírez and Hector Geffner. Plan recognition as planning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1778–1783, 2009.

[102] Miquel Ramírez and Hector Geffner. Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 1121–1126, 2010.

[103] Miquel Ramírez, Nir Lipovetzky, and Christian Muise. Lightweight Automated Planning ToolKiT. http://lapkt.org/, 2015.

[104] Wolfgang Reisig. *Understanding Petri Nets - Modeling Techniques, Analysis Methods, Case Studies*. Springer, 2013.

[105] Silvia Richter, Malte Helmert, and Matthias Westphal. Landmarks revisited. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 975–982, 2008.

[106] Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global sensitivity analysis: The primer*. John Wiley & Sons, 2008.

[107] Luísa R A Santos, Felipe Meneguzzi, Ramon Fraga Pereira, and André Grahl Pereira. An lp-based approach for goal recognition as planning. *Artificial Intelligence*, 35: 11939–11946, 5 2021.

[108] C. F. Schmidt, N. S. Sridharan, and J. L. Goodson. The plan recognition problem: An intersection of psychology and artificial intelligence. *Artificial Intelligence*, 11:45–83, 1978.

[109] Dafna Shahaf and Eyal Amir. Learning partially observable action schemas. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 913–919, 2006.

[110] Ahmed W. Shehata, Heather E. Williams, Jacqueline S. Hebert, and Patrick M. Pilarski. Machine learning for the control of prosthetic arms: Using electromyographic signals for improved performance. *IEEE Signal Processing Magazine*, 38(4):46–53, 2021.

[111] Maayan Shvo, Andrew C. Li, Rodrigo Toro Icarte, and Sheila A. McIlraith. Interpretable sequence classification via discrete optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 9647–9656, 2021.

[112] Dhirendra Singh, Sebastian Sardiña, Lin Padgham, and Geoff James. Integrating learning into a BDI agent for environments with changing dynamics. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2525–2530, 2011.

[113] Ilya M. Sobol. Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates. *Mathematics and Computers in Simulation*, 55(1):271–280, 2001.

[114] Shirin Sohrabi, Anton V. Riabov, and Octavian Udrea. Plan recognition as planning revisited. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3258–3264, 2016.

[115] David Speck, Robert Mattmüller, and Bernhard Nebel. Symbolic top-k planning. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 34:9967–9974, 4 2020.

[116] Gita Sukthankar and Katia Sycara. A cost minimization approach to human behavior recognition. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 1067–1074, 2005.

[117] Gita Sukthankar, Christopher Geib, Hung Hai Bui, David Pynadath, and Robert P. Goldman. *Plan, activity, and intent recognition: Theory and practice*. Newnes, 2014.

[118] Irene Teinemaa, Marlon Dumas, Marcello La Rosa, and Fabrizio Maria Maggi. Outcome-oriented predictive process monitoring: Review and benchmark. *ACM Transactions on Knowledge Discovery from Data*, 13(2):17:1–17:57, 2019.

[119] Andrei Tour, Artem Polyvyanyy, Anna A. Kalenkova, and Arik Senderovich. Agent miner: An algorithm for discovering agent systems from event data. In *Proceedings*

of the International Conference on Business Process Management (BPM), volume 14159 of *Lecture Notes in Computer Science*, pages 284–302, 2023.

[120] Wil M. P. van der Aalst. The application of Petri nets to workflow management. *Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.

[121] Wil M. P. van der Aalst. Process design by discovery: Harvesting workflow knowledge from ad-hoc executions. In *Knowledge Management: An Interdisciplinary Approach, Dagstuhl Seminar Report*, 2000.

[122] Wil M. P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer Berlin, Heidelberg, 2011.

[123] Wil M. P. van der Aalst. *Process Mining: Data Science in Action, Second Edition*. Springer Berlin, Heidelberg, 2016.

[124] Wil M. P. van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.

[125] Wil M. P. van der Aalst, Vladimir A. Rubin, H. M. W. Verbeek, Boudewijn F. van Dongen, Ekkart Kindler, and Christian W. Günther. Process mining: A two-step approach to balance between underfitting and overfitting. *Software and Systems Modeling*, 9(1):87–111, 2010.

[126] Wil M. P. van der Aalst, Arya Adriansyah, and Boudewijn F. van Dongen. Replaying history on process models for conformance checking and performance analysis. *WIREs Data Mining and Knowledge Discovery*, 2:182–192, 2012.

[127] Jan Martijn E. M. van der Werf, Artem Polyvyanyy, Bart R. van Wensveen, Matthieu J. S. Brinkhuis, and Hajo A. Reijers. All that glitters is not gold: Four maturity stages of process discovery algorithms. *Information Systems*, 114:102155, 2023.

[128] Boudewijn F. van Dongen, Josep Carmona, Thomas Chatain, and Farbod Taymouri. Aligning modeled and observed behavior: A compromise between computation complexity and quality. In *Proceedings of the International Conference on Advanced Information Systems Engineering (CAiSE)*, pages 94–109, 2017.

[129] Sebastiaan van Zelst, Alfredo Bolt, Marwan Hassani, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. Online conformance checking: Relating event streams to process models using prefix-alignments. *International Journal of Data Science and Analytics*, 8:269–284, 2019.

[130] Mor Vered and Gal A. Kaminka. Heuristic online goal recognition in continuous domains. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4447–4454, 2017.

[131] Mor Vered, Gal A. Kaminka, and Sivan Biham. Online goal recognition through mirroring: Humans and agents. In *Proceedings of the Annual Conference on Advances in Cognitive Systems*, 2016.

[132] Mor Vered, Ramon Fraga Pereira, Mauricio C. Magnaguagno, Gal A. Kaminka, and Felipe Meneguzzi. Towards online goal recognition combining goal mirroring and landmarks. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 2112–2114, 2018.

[133] Ilya Verenich, Marlon Dumas, Marcello La Rosa, Fabrizio Maria Maggi, and Chiara Di Francescomarino. Complex symbolic sequence clustering and multiple classifiers for predictive process monitoring. In *Proceedings of the BPM Workshops*, volume 256 of *LNBIP*, pages 218–229, 2015.

[134] Marc Vilain. Getting serious about parsing plans: a grammatical analysis of plan recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 190–197, 1990.

[135] Matthias Weidlich, Artem Polyvyanyy, Nirmit Desai, Jan Mendling, and Mathias Weske. Process compliance analysis based on behavioural profiles. *Information Systems*, 36(7):1009–1025, 2011.

[136] A. J. M. M. Weijters, Wil M. P. van der Aalst, and Ana K. Alves De Medeiros. *Process mining with the HeuristicsMiner algorithm*. Technische Universiteit Eindhoven, 2006.

[137] A.J.M.M. Weijters and J.T.S. Ribeiro. Flexible heuristics miner (fhm). In *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 310–317, 2011.

[138] Nils Wilken and Heiner Stuckenschmidt. Combining symbolic and statistical knowledge for goal recognition in smart home environments. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events*, pages 26–31, 2021.

[139] Richard B. Woodward and Levi J. Hargrove. Adapting myoelectric control in real-time using a virtual environment. *Journal of NeuroEngineering and Rehabilitation*, 16(11): 1–12, 2019.

[140] Anton Yeshchenko, Claudio Di Ciccio, Jan Mendling, and Artem Polyvyanyy. Visual drift detection for event sequence data of business processes. *IEEE Transactions on Visualization and Computer Graphics*, 28(8), 2022.

[141] Tianshi Yu, Ricardo Garcia-Rosas, Alireza Mohammadi, Ying Tan, Peter Choong, and Denny Oetomo. Comparing the outcomes of population-averaged and personalised input feature selection for transhumeral prosthetic interfaces. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 417–422, 2021.

[142] Tianshi Yu, Alireza Mohammadi, Ying Tan, Peter Choong, and Denny Oetomo. Sensor selection with composite features in identifying user-intended poses for human-prosthetic interfaces. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 31:1732–1742, 2023.

[143] X. Y. Zhang, M. N. Trame, L. J. Lesko, and S. Schmidt. Sobol sensitivity analysis: A tool to guide the development and evaluation of systems pharmacology models. *CPT: Pharmacometrics & Systems Pharmacology*, 4(2):69–79, 2015.

[144] Tan Zhi-Xuan, Jordyn L. Mann, Tom Silver, Josh Tenenbaum, and Vikash Mansinghka. Online Bayesian goal inference for boundedly rational planning agents. In *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

Appendix A

Additional Results and Statistics

A.1 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.1. 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.2, 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

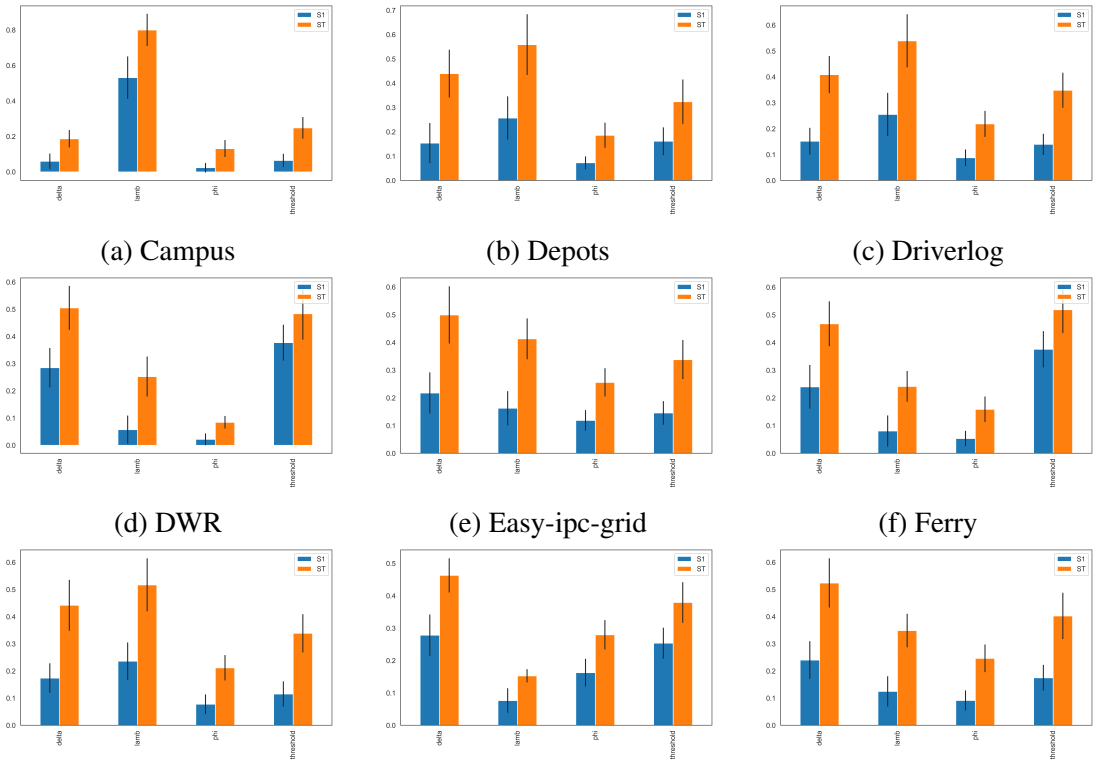
Table A.1 The number of skipped problems in synthetic domains for the top-k planner and the diverse planner.

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

A.2 Sobol Sensitivity Analysis

The results of the Sobol sensitivity analysis on the synthetic domains (excluding the domain of Blocks-world discussed in Section 3.5.2) with the cost-optimal traces generated by the top-k planner are shown in Figure A.1.

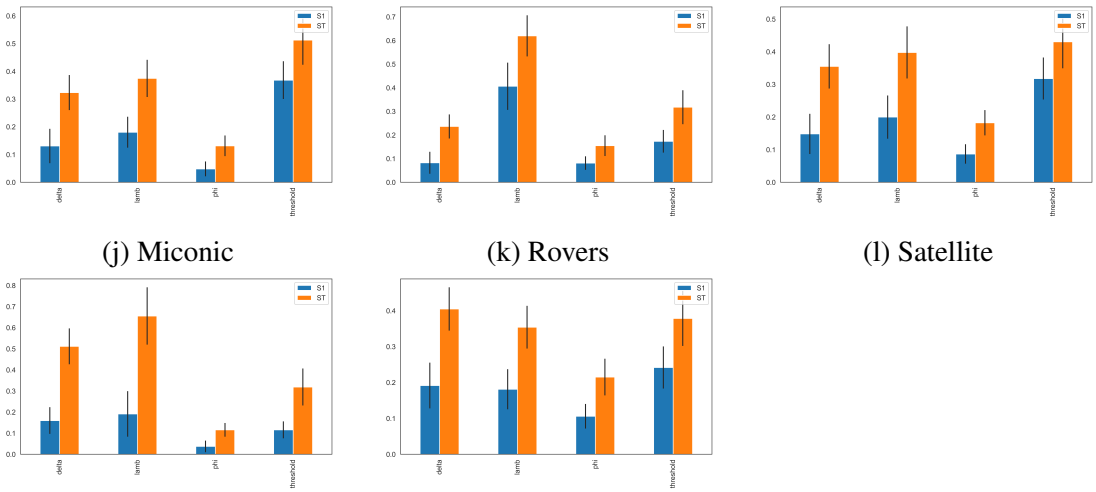


(a) Campus (b) Depots (c) Driverlog

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

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

– Continued on the next page. . .

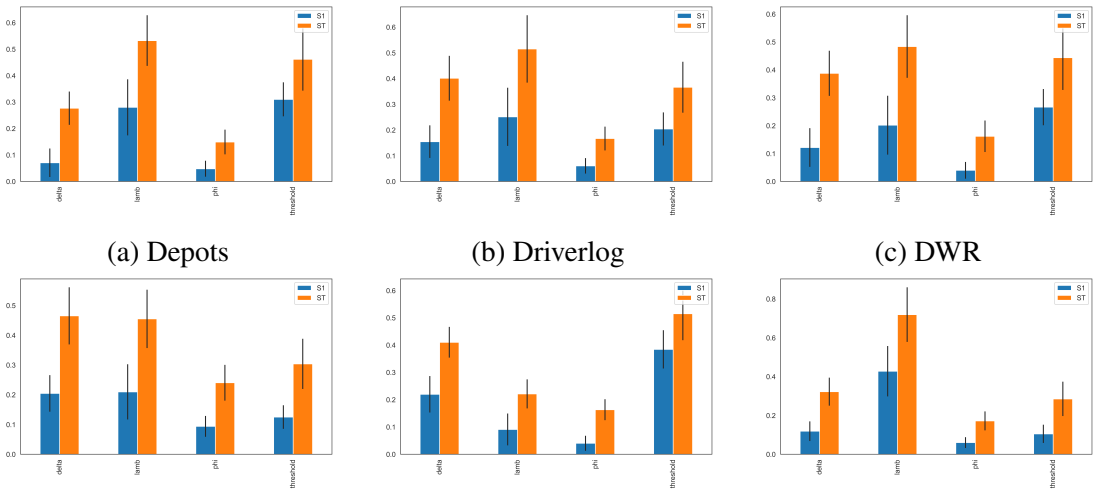


(j) Miconic (k) Rovers (l) Satellite

(m) Sokoban (n) Zeno-travel

Figure A.1 Sobol sensitivity analysis (the GR system was trained with the cost-optimal traces).

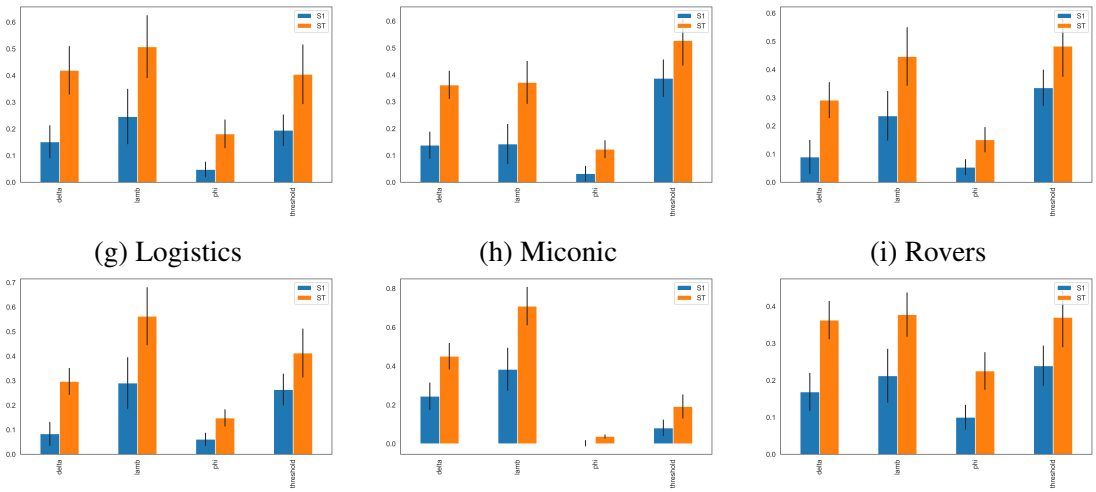
The results of the Sobol sensitivity analysis on the synthetic domains (excluding the domain of Blocks-world discussed in Section 3.5.2) with the divergent traces generated by the diverse planner are shown in Figure A.2.



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

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

– Continued on the next page. . .



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

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

Figure A.2 Sobol sensitivity analysis (the GR system was trained with the divergent traces).

A.3 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

Table A.3 – continued on the next page. . .

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
	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
	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
intrusion-detection	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
	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	—	—	—	—	—	—	—	—
kitchen	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	—	—	—	—	—	—	—	—
	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
logistics	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
	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
miconic	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
	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
rovers	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
	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
satellite	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
	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
sokoban	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
	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
zeno-travel	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 A.3 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).

A.4 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
	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
blocks-world	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
	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
campus	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
	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
depots	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

Table A.4 – continued on the next page...

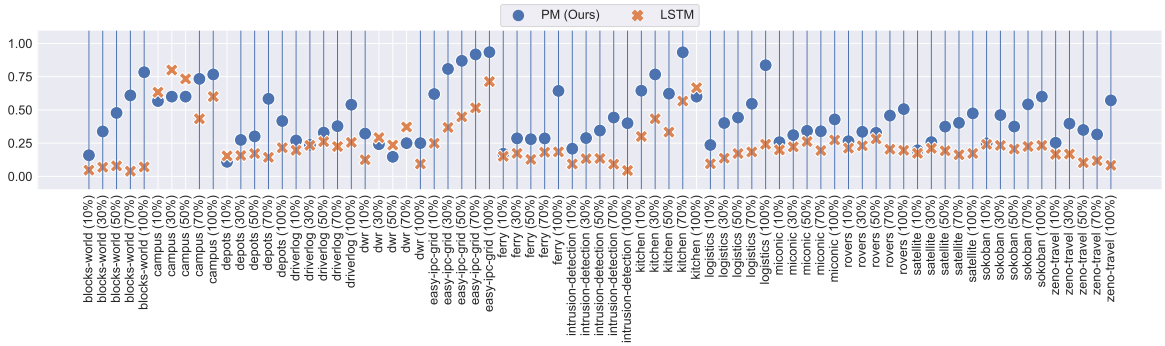
Domain	%O	PM-based (ours)				Landmark-based				R&G (DUAL-BFWS)				R&G (Greedy LAMA)				LP-based			
		p	r	a	t	p	r	a	t	p	r	a	t	p	r	a	t	p	r	a	t
	100	0.54	0.67	0.90	0.93	0.79	1.00	0.95	0.95	0.65	0.92	0.75	214.83	0.83	0.83	0.95	367.32	1.00	1.00	1.00	1.67
	10	0.29	0.68	0.58	0.65	0.24	0.80	0.48	0.83	0.39	0.45	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
driverlog	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
	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
dwr	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
	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
easy-ipc-grid	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
	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
ferry	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
	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
intrusion-detection	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
	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
kitchen	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
	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
logistics	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
	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
miconic	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
	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
rovers	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
	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
satellite	50	0.41	0.60	0.75	0.30	0.59	0.92	0.77	1.25	0.34	0.69	0.55	27.00	0.64	0.80	0.84	10.57	0.82	0.96	0.94	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
	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
sokoban	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
	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
zeno-travel	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 A.4 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.

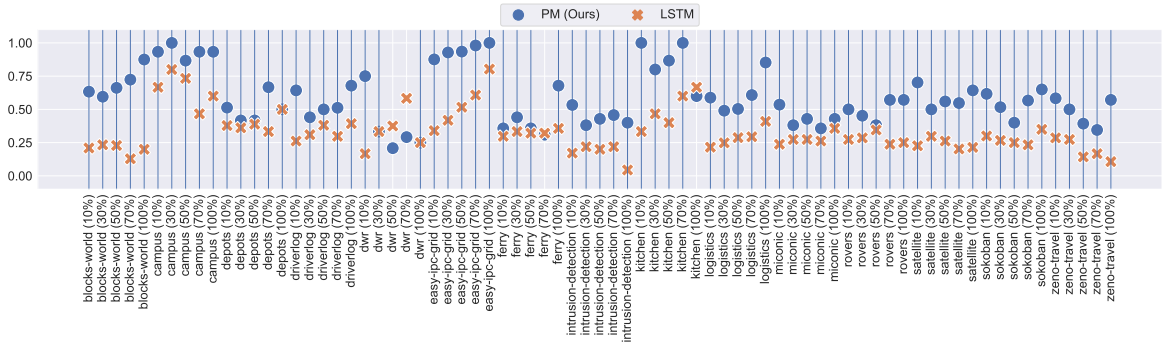
A.5 Performance Comparison with the LSTM-Based GR Approach

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

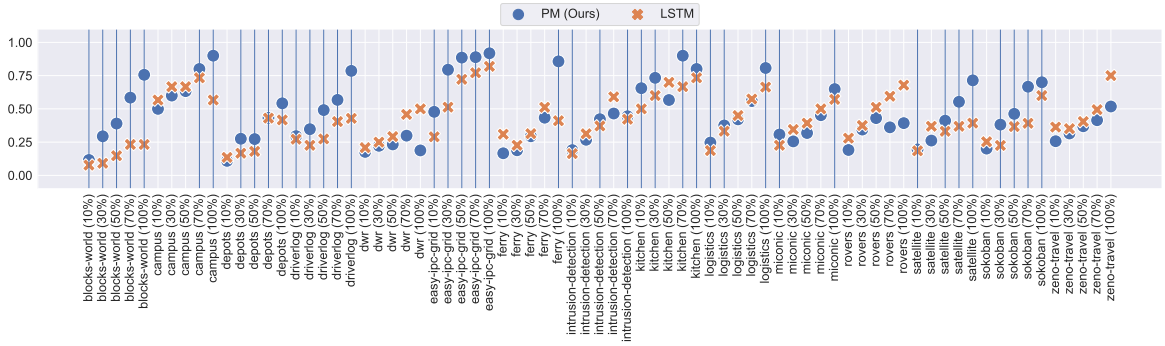
Table A.5 Performance of the PM-based GR approach (ours) and the LSTM-based approach; (10): trained with 10 traces per goal, (100): trained with 100 traces per goal, %O: the level of observation, p: precision, r: recall, a: accuracy. Both approaches are trained with the divergent traces or with the cost-optimal traces if the divergent traces are not available. Our approach is configured with the PRIM parameters.



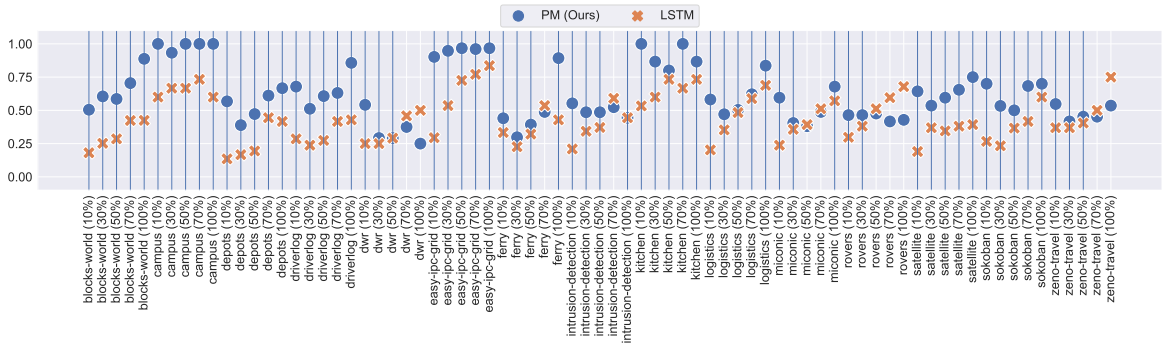
(a) Precision (trained with 10 traces per goal).



(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 A.3 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.