PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
SCHOOL OF COMPUTER SCIENCE
POSTGRADUATE PROGRAMME IN COMPUTER SCIENCE

LANDMARK-BASED APPROACHES FOR PLAN RECOGNITION TASKS

RAMON FRAGA PEREIRA

Dissertation presented as partial requirement for obtaining the degree of Masters in Computer Science at Pontifical Catholic University of Rio Grande do Sul.

Advisor: Prof. Dr. Felipe Meneguzzi

Porto Alegre
2016

Dados Internacionais de Catalogação na Publicação (CIP)

P436L Pereira, Ramon Fraga

Landmark-based approaches for plan recognition tasks / Ramon Fraga Pereira. – 2016.

76 f.

Dissertação (Mestrado) – Faculdade de Informática, Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Dr. Felipe Meneguzzi

1. Heurística (Informática). 2. Planejamento. 3. Cognição. 4. Informática. I. Meneguzzi, Felipe. II. Título.

CDD 006.3

Ficha Catalográfica elaborada por Loiva Duarte Novak – CRB10/2079

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada *"Landmark-Based Approaches for plan Recognition Tasks"* apresentada por Ramon Fraga Pereira como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, aprovada em 15 de março de 2016 pela Comissão Examinadora:

[illegible]

Prof. Dr. Felipe Rech Meneguzzi – Orientador — PPGCC/PUCRS

[illegible]

Prof. Dr. Rodrigo Coelho Barros – — PPGCC/PUCRS

MOSER FAGUNDES

Prof. Dr. Moser Silva Fagundes - — IFSUL - Campus Feliz

Homologada em 19/05/2016, conforme Ata No. Ø10 pela Comissão Coordenadora.

[illegible]

Prof. Dr. Luiz Gustavo Leão Fernandes
Coordenador.

TAREFAS PARA RECONHECIMENTO DE PLANOS BASEDAS EM PONTOS DE REFERENCIA

RESUMO

Técnicas de planejamento automático são eficientes no reconhecimento de objetivos e planos a partir da execução de ações e evidências incompletas. Para muitas aplicações é importante reconhecer objetivos e planos não somente acuradamente, mas também de maneira rápida e precisa. Assim, para lidar com esse desafio, desenvolvemos uma abordagem a qual utiliza uma heurística baseada em técnicas de planejamento automático, guiando-se por pontos-de-referência, que filtra possíveis objetivos e planos a partir de observações. Em planejamento automático, pontos-de-referência são propriedades (ou ações), em que todo o plano precisa alcançar (ou executar), em alguma determinada parte da execução do plano a fim de atingir um objetivo estipulado. Neste trabalho, formalizamos a tarefa de reconhecimento de objetivos e planos sem a utilização de biblioteca de planos, ou seja, utilizamos uma definição de domínio para planejamento automático. Sendo assim, estabelecemos o problema e o comportamento do agente a ser observado (ações e objetivos) utilizando uma linguagem de planejamento automático. A partir disso, mostramos a aplicabilidade da nossa abordagem baseada em técnicas de planejamento de três formas: (1) desenvolvendo uma heurística baseada em pontos-de-referencia para reconhecer objetivos e planos; (2) refinando uma abordagem existente para reconhecimento de planos; e for fim, (3) desenvolvendo uma abordagem para reconhecer abandono de planos. A abordagem para reconhecimento de abandono de planos desenvolvida tem como objetivo analisar uma seqüência de observações (ações), afim de detectar quais não contribuem para alcançar o objetivo o qual está sendo monitorado. Para fins de avaliação e experimentação, utilizou-se vários domínios de planejamento automático, e com isso, foi possível mostrar que nossa abordagem para reconhecimento de planos comporta-se acuradamente e rapidamente quando comparada com o estado-da-arte. Ainda, demonstramos que a nossa abordagem para detectar abandono de planos comporta-se com precisão e com baixo custo computacional, detectando precisamente ações que não contribuem para alcançar um determinado objetivo monitorado.

Palavras Chave: Abandono de Planos, Reconhecimento de Planos, Planejamento, Plano, Objetivo.

LANDMARK-BASED APPROACHES FOR PLAN RECOGNITION TASKS

ABSTRACT

Recognition of goals and plans using incomplete evidence from action execution can be done efficiently by using automated planning techniques. In many applications it is important to recognize goals and plans not only accurately, but also quickly. In order to address this challenge, we develop recognition approaches based on planning techniques that rely on planning landmarks to filter candidate goals and plans from observations. In automated planning, landmarks are properties or actions that cannot be avoided to achieve a goal. We address the task of recognizing goals and plans without pre-defined static plan libraries, and instead we use a planning domain definition to represent the problem and the expected agent behavior. In this work, we show the applicability of planning techniques for recognition tasks in three settings: first, we use planning landmarks to develop a heuristic-based plan recognition approach; second, we refine an existing planning-based plan recognition approach; and finally, we use planning techniques to develop an approach for detecting plan abandonment. The plan abandonment detection approach we develop aims to analyze a sequence of observations and a monitored goal to determine if an observed agent is still pursuing, or has no intention to complete such monitored goal. These recognition approaches are evaluated in experiments over several planning domains. We show that our plan recognition approach yields not only accuracy comparable to other state-of-the-art techniques, but also substantially lower recognition time over such techniques. Furthermore, our plan abandonment detection approach yields high accuracy at low computational cost to detect which actions do not contribute for achieving a particular monitored goal.

Keywords: Plan Abandonment, Plan Recognition, Planning, Plan, Goal.

LIST OF ALGORITHMS

LIST OF FIGURES

LIST OF SYMBOLS

LIST OF TABLES

CONTENTS

1. INTRODUCTION

As more computer systems require reasoning about what agents (both human and artificial) other than themselves are doing, the ability to accurately and efficiently recognize plans and goals from agent behavior becomes increasingly important. Plan recognition is the task of recognizing plans and goals based on often incomplete observations that include actions executed by agents and properties of agent behavior in an environment [SGG$^+$14]. Most plan recognition approaches [GG05, AK05] employ plan libraries to represent agent behavior (*i.e*, a library with all plans for achieving goals), resulting in approaches to recognize plans that are analogous to parsing. Recent work [RG09, RG10, PL10] use a planning domain definition (a domain theory) to represent potential agent behavior, bringing plan recognition closer to planning algorithms. These approaches allow techniques used in planning algorithms to be employed in plan recognition. Meanwhile, the related recognition task of detecting when an agent abandons plans has received comparatively little attention. Anticipating when an agent is deviating from its current plan can be an important mechanism for monitoring applications. Both plan recognition and plan abandonment detection are important in applications to monitor and anticipate agent behavior, including crime detection and prevention, monitoring activities in elderly-care, among others.

In this work we develop two recognition approaches that are based on planning techniques: a landmark-based plan recognition approach; and a planning-based plan abandonment detection approach. For recognizing plans, we develop an approach that relies on planning landmarks to filter candidate goals and plans from observations. In automated planning, landmarks are properties (or actions) that every plan must satisfy (or execute) at some point in every plan execution to achieve a goal [HPS04]. Whereas in planning algorithms these landmarks are used to focus search, in this work, they allow our plan recognition approach to rule out goals whose landmarks have been missed from observations. Thus, based on landmarks, we develop a filtering algorithm that filters candidate goals by estimating how many landmarks required by every goal in the set of candidate goals have been reached within a sequence of observed actions. Since computing a subset of landmarks for a set of goals can be done very quickly, our approach results in substantial runtime gains. In this way, we use this filtering algorithm alongside two settings. First, we build a plan recognition heuristic that analyzes the amount of achieved landmarks to estimate the percentage of completion of each candidate goal. Second, we show that this filter can also be applied to other planning-based plan recognition approaches, such as [RG09].

Plan abandonment occurs when an agent interrupts the current plan execution for any reason by executing a sufficient number of actions that do not contribute towards a monitored goal. To accomplish this task, we develop a planning-based approach that unlike prior approaches [Gei02, GG03] that only deal with pre-defined static plan libraries, we propose an approach that formalizes agent plans and goals over a planning domain definition. More specifically, our approach is composed of two stages. In the first stage, our approach computes useful information by analyzing the domain definition, as well as landmark extraction. The second stage is the behavior analysis, wherein our approach uses two methods that can work independently or together for detecting plan abandonment. The first method takes as input the result of an observed action for estimating the distance (using domain-independent heuristics) to the monitored goal, and therefore, it analyzes possible variations over earlier observations in the plan execution. The second method aims to predict which actions might be "good" to be executed (*i.e*, to reduce the distance to the goal) in the next observation by looking at the closest (estimated plan length) landmarks of the current monitored goal. Thus, our approach detects which actions in the execution of an agent plan do not contribute to the plan for achieving the goal being monitored.

We evaluate our recognition approaches using domains from the International Planning Competition (IPC) and a dataset for plan recognition proposed by Ramírez and Geffner in [RG09]. We show that our plan recognition approach yields promising results regarding time and accuracy when compared to the work of Ramírez and Geffner [RG09]. In addition, we show that our filtering method provides substantial improvements in recognition time for existing planning-based plan recognition approaches. With regard to detecting plan abandonment, experiments over several domains show that our approach provides high accuracy at low computational cost to detect abandonment for both optimal and sub-optimal execution of agent plans.

The main contribution of this work is the use of planning techniques and landmarks for recognition tasks. This work provides the first steps to bridge the gap between planning and plan abandonment detection by exploiting planning techniques, for instance, the extraction and exploitation of landmark information, and domain-independent heuristics. Our plan recognition approach is novel and original due to the use of landmarks for recognizing goals and plans, and therefore, this approach often improves time and accuracy when compared and applied to other state-of-the-art approaches.

1.1 Outline of the Dissertation

The outline of this dissertation is structured as follows. Chapter 2 provides background on classical planning, domain-independent heuristics, landmarks, plan recognition, and plan abandonment detection. In Chapter 3 we describe how we extract useful information from planning domain definition. In Chapter 4 we present our landmark-based plan recognition approach. In Chapter 5 we present our planning-based approach for detecting plan abandonment. Chapter 6 presents the results of experiments of our recognition approaches. Chapter 7 surveys related work. Finally, in Chapter 8 we conclude this work by discussing limitations and future directions of our recognition approaches.

2. BACKGROUND

In this chapter we review concepts and terminologies for this dissertation. First, in Section 2.1 we review classical planning and its terminology. In Section 2.2 we cover the domain-independent heuristics that we use to estimate goal distance from a particular state. In Section 2.3 we describe the concept of landmarks in planning, the core of our recognition approaches. Finally, in Sections 2.4 and 2.5 we describe the formulation of plan recognition and plan abandonment detection over planning terminologies we adopt in this work.

2.1 Planning

According to Ghallab *et al.* [GNT04, Chapter 1 – page 1], planning is the reasoning side of acting, more specifically, planning is a deliberation process that aims to choose and organize (as best as possible) a sequence of actions to achieve a particular goal. In automated planning, the task of finding this sequence of actions depends on a variety of properties, for example: the environment can be deterministic or non-deterministic, and fully or partially observable; and actions can be performed instantly or with a time duration. Thus, some of these properties imply a high computational complexity to find this sequence of actions, which can grow according to the amount of detail used to model the actions and the environment. However, in this work we formalize planning problems for deterministic and fully observable domains, and therefore, no metric resources or temporal aspects are present in a domain formalization.

To represent planning domains and problems, we adapt the terminology from Ghallab *et al.* [GNT04, Chapter 1 – page 17]. First, we define a state in the environment, as follows in Definition 1.

Definition 1 (State) *A state is a finite set of facts that represent logical values (true or false) according to some interpretation. Facts are divided into two types: positive and negated facts, as well as constants for truth ($\top$) and falsehood ($\bot$). A predicate is denoted by an n-ary predicate symbol p applied to a sequence of zero or more terms (τ_1, τ_2, ..., τ_n). Terms are either constants or variables.*

Definition 2 (Operator) *An operator is represented by a triple $a = \langle$name(a), pre(a), eff(a)$\rangle$: in which name(a) represents the description or signature of a; pre(a) describes the preconditions of a, a set of facts or predicates that must exist in the current state for a be executed; and eff(a) represents the effects of a in the current state. These effects are divided into eff(a)$^+$, an add-list of positive facts or predicates, and eff(a)$^-$, a delete-list of negative facts or predicates.*

Operator definitions are used in the construction of a planning domain, which represents the environment dynamics that guide an agent's search for plans to achieve its goals. States are modified when state transitioning actions are executed by the following Definition 3.

Definition 3 (Action) *An action is a ground instantiated*[1] *operator over free variables. Thus, if all operator free variables are substituted by objects when instantiating an operator, we have a ground action.*

Definition 4 (Planning Domain) *A planning domain definition Ξ is represented by a pair $\langle\Sigma, \mathcal{A}\rangle$, which specifies the knowledge of the domain, and consists of a finite set of facts Σ and a finite set of actions $\mathcal{A}$.*

A planning instance comprises both a planning domain and the elements of a planning problem, by describing the initial state of the environment and the goal state which an agent wishes to achieve. This is formalized in Definition 5.

Definition 5 (Planning Instance) *A planning instance is represented by a triple $\Pi = \langle\Xi, \mathcal{I}, G\rangle$, in which $\Xi = \langle\Sigma, \mathcal{A}\rangle$ is the domain definition; $\mathcal{I} \subseteq \Sigma$ is the initial state specification, which is defined by specifying the value for each fact or predicate in the initial state; and $G \subseteq \Sigma$ is the goal state specification, which represents a desired state to be reached.*

Classical planning representations often separate the definition of $\mathcal{I}$ and G as part of a planning problem in order to be used together with a domain Ξ. According to Erol *et al.*, planning in this formalism is a problem EXPSPACE-Complete [ENS95].

Finally, a sequence of actions that achieves the goal state of a planning problem is the solution to a planning instance, as formalized in Definition 6.

Definition 6 (Plan) *A plan is a sequence of actions $\pi = \langle a_1, a_2, ..., a_n\rangle$ that modifies the initial state $\mathcal{I}$ into one in which the goal state G holds by the successive execution of actions in a plan π. The length of a plan π is the number of actions in a plan. Thus, a plan π is considered optimal if its plan length is the shortest one.*

To solve planning problems (*i.e*, the task of finding a plan), it is necessary a planner or planning algorithm. Planners are often based on search algorithms and heuristics, and take as input a problem specification and a knowledge about a certain domain, *i.e*, $\Pi = \langle\Xi, \mathcal{I}, G\rangle$. As output, a planner returns a successful plan or failure. Over the years, the planning community has developed automated planners with increasing computational capabilities, including:

[1]Grounding is a concrete instantiation of a planning domain definition, and it is generated from a finite set of objects as defined in the problem definition. An instantiation occurs if the arguments of a predicate or operator are all consistent with the finite set of objects.

- 1995 – GraphPlan [BF97];
- 1998 – BlackBox [KS98];
- 2001 – HSP (Planning as Heuristic Search) [BG01];
- 2001 – FF (Fast Plan Generation Through Heuristic Search) [HN01];
- 2006 – Fast Downward [Hel06]; and
- 2008 – LAMA [RW10].

For representing domains and problems (*i.e*, input for planners) in planning, we introduce the most commonly used planning languages, as follows: STRIPS (Stanford Research Institute Problem Solver) and PDDL (Planning Domain Definition Language). STRIPS [FN71] is a planner created in 1971, that became important to the planning community due to its formalization for describing domains and problems. PDDL [MGH$^+$98] was developed in order to define a standard formalization for planning problems in order to be used when comparing the performance of planning algorithms. Moreover, PDDL provides more expressivity (*e.g*, types, quantifiers, and others) than other formalizations, such as STRIPS. Usually, both languages are composed of a domain and a problem, and each one of them is represented in different files.

2.2 Domain-Independent Heuristics

Heuristics are used to estimate the cost to achieve a particular goal [GNT04, Chapter 1 – page 193]. With regard to classical planning, this estimate is often the number of actions to achieve the goal state from a particular state. In this work, we consider that the action cost is $c(a) = 1$ for all $a \subseteq A$. Thus, the cost for a plan $\pi = \langle a_1, a_2, ..., a_n\rangle$ is $c(\pi) = \Sigma\ c(a_i)$. Estimating the number of actions exactly is a problem NP-hard [Byl94]. In automated planning, heuristics can be domain-dependent or domain-independent. If the heuristic is domain-dependent, then such heuristic only estimates for a specific domain. However, if the heuristic is domain-independent, then such heuristic can estimate for a variety of domains. In general, the use of heuristics can result in a huge reduction of search time for estimating how many actions are necessary to achieve a particular goal.

In automated planning, heuristics make no guarantees of the optimality of their estimations. Heuristics that never overestimate the cost to achieve a goal are called admissible. An heuristic $h(s)$ is admissible if $h(s) \leq h*(s)$ for all states, where $h*(s)$ is the optimal cost to the goal from state s. Clearly, heuristics that overestimate the cost are called inadmissible. In this work, we consider both admissible and inadmissible domain-independent heuristics for approximating the estimated value to achieve a particular goal.

2.2.1 Max Heuristic

The max heuristic h_{max} was proposed by Bonet and Geffner [BG01], and it is based on the delete-list relaxation, in which delete-effects of actions are ignored during calculation of the heuristic cost to a goal. This calculation is the cost of a conjunctive goal, which represents the maximum cost to achieve each of the individual facts. For representing the delete-list effects relaxation, we use a Relaxed Planning Graph (RPG) [BK07], as formalized in Definition 7.

Definition 7 (Relaxed Planning Graph – RPG) *Based on the planning graph proposed by Blum and Furst in GraphPlan [BF97], the RPG is a leveled graph, but the RPG structure ignores the delete-list effects of all actions, and in this way, there are no mutex relation in this graph. Thus, as a leveled graph, the graph levels are structured as follows: F_0, A_0, F_1, A_1, ..., F_{m-1}, A_{m-1}, F_m of fact sets F_i (fact levels) and action sets A_i (action levels). More specifically, the fact level F_0 contains the facts that are true in the initial state, the action level A_0 contains those actions whose preconditions are reached from F_0, F_1 contains F_0 plus the add effects of the actions in A_0, and so on. In a nutshell, the RPG is composed by $F_i \subseteq F_{i+1}$ and $A_i \subseteq A_{i+1}$ for all i.*

By using an RPG structure it is possible to verify the solvability of a planning problem. If the relaxed plan is unreachable, then the RPG reaches a fixpoint before reaching the goal facts, thereby proving unsolvability. If the goal is solvable, then eventually the last fact level contains the goal facts. Deciding about the solvability of planning problems ignoring delete-lists can be done in polynomial time by building the RPG. The task of building a full RPG structure is a variation of the algorithm proposed by Bylander [Byl94], which proves that the plan existence is polynomial in the absence of delete-list effects. In this work, we use the RPG structure for building heuristic methods to estimate the goal distance from a particular state, filtering facts and actions by reachability analysis, and extracting landmarks.

To estimate the relaxed distance to the goal, h_{max} takes as input an initial and a goal state. First, an approximation of reachability from the initial state to goal state is built using the RPG structure. Second, for deciding about the goal solvability, h_{max} verifies the first level in the RPG that contains all facts of the goal, and if such level exists, the goal is solvable. Thus, the relaxed distance to the goal is the distance between the first level and the level that contains all facts of the goal, that is the relaxed minimal number of actions required to achieve the goal, considering that the goal is conjunctive. In other words, the relaxed minimal number of actions represent the number of levels that the RPG had to expand to achieve the goal. We use this approximation of reachability as a heuristic because deciding the solvability of a planning problem using an RPG structure can be done in polynomial time.

Although h_{max} is both fast and admissible to compute the estimated distance, this heuristic will often heavily underestimate the cost remaining for many planning problems. For example, consider the BLOCKS-WORLD[2] planning problem example in Figure 2.1. For this example, the optimal cost[3] is $h^*(G) = 6$, and the estimated cost by the max heuristic is $h_{max}(G) = 3$.

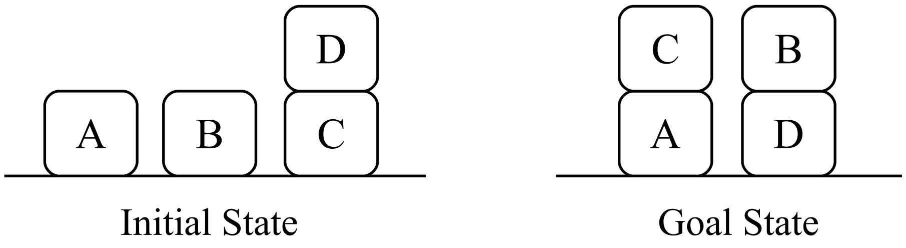

Figure 2.1 – BLOCKS-WORLD problem example.

2.2.2 Fast Forward Heuristic

The Fast Forward (FF) heuristic is one of the best-known heuristics in the planning community [HN01], and the heuristic itself has also been a component of multiple successful planners, such as Fast Downward (FD) [Hel11], metric-FF [Hof11], and SGPlan [CWwH06]. We denote the FF heuristic as h_{ff}.

To estimate the relaxed distance for planning problems, h_{ff} also uses an RPG structure to represent the delete-list effects relaxation, but unlike h_{max}, this RPG structure contains no-ops in the action levels. No-ops are stub actions with one precondition and one effect, and these stub actions are the facts being maintained from the previous fact level. In this way, to estimate the relaxed distance, an RPG structure containing no-ops is built from a given planning problem (*i.e*, the initial and goal states). Afterwards, a set of facts G_{ff} is created to store all facts of the goal state. Then, the relaxed distance is regressively extracted from the first RPG level that contains all facts of the goal state. Thus, starting from this level, for each fact g in G_{ff}, the first action that achieves g in the previous action level is selected to be added to the relaxed distance. If this action level contains no-ops, then these will always be chosen over normal actions. The effects of selected actions are removed from the set of facts G_{ff}, and the union of all preconditions are added. This process iterates until the set of facts G_{ff} becomes a subset of the initial state.

[2]BLOCKS-WORLD is a classical planning domain where a set of stackable blocks must be re-assembled on a table [GNT04].

[3]A possible optimal plan for the BLOCKS-WORLD problem example (in Figure 2.1) is: `(unstack D C)` `(put-down D)` `(pick-up B)` `(stack B D)` `(pick-up C)` `(stack C A)`, *i.e*, 6 action steps.

Similar to h_{max}, h_{ff} is fast and can be computed in polynomial time. But, unlike h_{max}, h_{ff} is inadmissible because it occasionally overestimates the distance for some planning problems. However, h_{ff} usually underestimates the distance to the goal. For example, considering the same example used by the max heuristic, Figure 2.1. By using FF heuristic, the estimated cost is $h_{ff}(G)$ $= 5$. In fact, for this example the FF heuristic does not overestimate the cost, but occasionally this heuristic may overestimate the cost to achieve a goal, depending on the planning problem.

2.3 Landmarks

In the planning literature, landmarks [J. 01, HPS04] are defined as necessary features that must be true at some point in every valid plan to achieve a particular goal. Landmarks are often partially ordered following the sequence in which they must be achieved. The concept of landmarks was first introduced by Porteous *et al.* [J. 01], and later studied in more detail by other authors, for instance Hoffman *et al.* [HPS04]. In this work we adopt the terminology defined by Hoffman *et al.* [HPS04], as follows.

Definition 8 (Fact Landmark) *Given a planning instance $\Pi = \langle \Xi, \mathcal{I}, G \rangle$, a fact L is a landmark in Π iff L is true at some point along all valid plans that achieve G from $\mathcal{I}$. In other words, fact landmarks are formulas over a set of facts that must be satisfied at some point along all valid plan.*

From the concept of fact landmarks, Hoffman *et al.* [HPS04] introduce two types of landmarks as formulas: conjunctive and disjunctive landmarks. A conjunctive landmark is a set of facts that must be true together at some point in every valid plan to achieve a goal. A disjunctive landmark is a set of facts in which at least one of facts must be true at some point in every valid plan to achieve a goal. For example, as a conjunctive landmark formula: `(at PLACE-1)` $\wedge$ `(holding BLOCK-A)`; and as a disjunctive landmark formula: `((at PLACE-1)` $\wedge$ `(holding BLOCK-A))` $\vee$ `((at PLACE-1)` $\wedge$ `(holding BLOCK-B))`.

Definition 9 (Action Landmark) *Given a planning instance $\Pi = \langle \Xi, \mathcal{I}, G \rangle$, an action A is a landmark in Π iff A is a necessary action to be executed at some point along all valid plans that achieve G from $\mathcal{I}$.*

To exemplify the concept of landmarks in planning, consider the BLOCKS-WORLD problem example in Figure 2.1. This example shows an initial state and a goal state, and trivially, all facts in both states are landmarks because they cannot be avoided to achieve the goal. By looking to the goal state, we can intuitively infer some fact and action landmarks. For example, in the goal state block `C` must be clear and block `D` must be on the table. From the initial state, such actions can be performed to achieve these properties: unstack `D` from `C`, and put down `D` on the table.

Consequently, the following actions must be landmarks: `(unstack D C)` and `(put-down D)`. The result of performing such actions provides the following fact landmarks: `(clear C)` and `(ontable D)`. From this, we also know that to stack `C` on `A`, and stack `B` on `D`, block `A` must be clear and block `C` must be holding, likewise for stacking `B` on `D`, and so on. This is a simple planning example, and the task of extracting fact and action landmarks looks a simple task, however, for more complex problems this task becomes harder.

Hoffman *et al.* [HPS04] show that the task of deciding whether any given formula or action is a landmark and deciding orderings between landmarks are known to be PSPACE-complete [HPS04]. Therefore, many approaches that comprise the task of extracting landmarks focus on extracting just a subset of landmarks for a given planning problem, because extracting all landmarks is a problem as complex as planning.

Whereas in planning the concept of landmarks is used in algorithms to focus on heuristics and search, *e.g*, LAMA [RW10], in this work, landmarks allow recognition algorithms to rule out goals whose landmarks have been missed or pursued from observations. So, by using the concept of landmarks, we consider that fact and action landmarks are way-points for achieving a set of candidate goals, and thus, we can monitor these way-points to determine which goal(s) an observed plan execution is likely going to achieve.

2.4 Plan Recognition

Plan recognition is the task that aims to identify how agents achieve their goals by observing their interactions in an environment [Car01, SGG$^+$14]. These interactions can be observed events performed by an agent in an environment, as well as actions (*e.g*, a simple movement, cook, drive), and changing properties in an environment (*e.g*, at home, at work, resting). In the plan recognition context, agent observations are defined as available evidence that can be used for recognizing plans [AA07]. As most plan recognition approaches, knowledge of the agent's possible plans is required for representing its typical behavior, in other words, this knowledge provides the recipes (*i.e*, know-how) for achieving goals, and we call these recipes as agent behavior models. Generally, these recipes are often called plan libraries and are used as input for many plan recognition approaches [GG05, AK05]. Thus, the task of recognizing agent plans consists of matching observations with the agent behavior model in order to determine the hypotheses plans that better explain the observations. Plan recognition approaches are classified according to the role that the observed agent performs during the task of recognizing plans [AA07], as follows.

- In *intended plan recognition*, the observed agent is aware of the task of plan recognition. In this kind of recognition process the observed agent usually cooperates with the process by notifying the plan recognizer of its actions;

- In *keyhole plan recognition*, the observed agent is unaware of the task of plan recognition, that is, the observed agent provides partially observable input to the plan recognition process, which requires to infer unobserved events; and

- In *obstructed plan recognition*, the observed agent is aware of the task of plan recognition and purposely obstructs the process. In other words, the agent does not cooperate with the plan recognition process.

Regarding the task of recognizing plans, we highlight three important aspects. First, the task of recognizing agent plans often uses an agent behavior model, *i.e*, domain dependent information, but the task of recognizing agent plans is domain independent information [AA07]. Second, plan recognition is closely related to intent/goal recognition [HP13], because plan recognizers can also infer what goal an observed agent is pursuing. Finally, plan recognition is distinct from planning, while in plan recognition we search for goals and plans that better explain the observed actions, in planning we search for a sequence of actions (*i.e*, a plan) to achieve a particular goal [RG09].

To illustrate the task of recognizing plans, we present an example adapted from [SSG78]. This example considers two possible goals that an agent called Steve can achieve by performing a sequence of actions, as follows.

- G_1: Steve wants to prepare his breakfast.

- G_2: Steve wants to listen to a record.

The observed actions performed by Steve are:

- O_1: Steve walks to the cabinet.

- O_2: Steve opens the cabinet.

- O_3: Steve takes a record out of the cabinet.

- O_4: Steve takes the record out of the record jacket.

From the observations, we can infer that Steve wants to achieve the goal G_2, however, in order to infer the same conclusion, the task of recognizing plans needs:

1. A model to formalize Steve's possible behavior to establish a match between actions and goals; and

2. A reasoning mechanism (*e.g*, plan recognizer) that takes as input a model containing candidate goals and observations, in order to conclude which goal(s) from observed actions is likely going to be achieved.

In this work we focus on single-agent keyhole plan recognition, and as a model of agent possible behavior, we use a planning domain definition, more specifically, we use the STRIPS [FN71] fragment of PDDL [MGH$^+$98]. In this way, we follow Ramírez and Geffner [RG09] to formally define a plan recognition problem over a planning domain definition, as follows in Definition 10.

Definition 10 (Plan Recognition Problem) *A plan recognition problem is a four-tuple* $T_{PR} = \langle \Xi, \mathcal{I}, \mathcal{G}, O \rangle$*, in which* $\Xi = \langle \Sigma, \mathcal{A} \rangle$ *is the domain definition, and consists of a finite set of facts* Σ *and a finite set of actions* $\mathcal{A}$*;* $\mathcal{I}$ *represents the initial state;* $\mathcal{G}$ *is the set of possible goals of a hidden goal* G*, such that* $G \in \mathcal{G}$*; and* $O = \langle o_1, o_2, ..., o_n \rangle$ *is an observation sequence of a plan execution with each observation* $o_i \in O$ *being an action in the finite set of actions* $\mathcal{A}$ *from the domain definition* Ξ*. This observation sequence can be full or partial, which means that for a full observation sequence we observe all actions during the execution of an agent plan, and for a partial observation sequence, only a sub-sequence of actions of the execution of an agent plan is observed. The solution for this problem is to find a hidden goal* G *in the set of possible goals* $\mathcal{G}$ *that the observation sequence of a plan execution achieves.*

2.5 Plan Abandonment Detection

We define plan abandonment as a situation in which an agent switches from executing the actions of one plan to executing actions from another plan due to some reason. The reasons why an agent abandons its plans may include:

- concurrent plans in daily activities, that is, when an agent has to perform multiple concurrent activities at the same time; or
- dealing with conflicting plans, in this case, the agent decides which activity is more important given the current situation.

However, in this work we are not concerned with the reason for abandonment.

We understand that the ability to detect plan abandonment is important for any application of plan recognition. Most plan recognition approaches assume that when an agent interrupts a plan to perform another one, this agent will eventually resume executing the original plan to completion [GG03]. In real-world scenarios, such assumption is too strong and does not reflect how people actually behave, so plan abandonment detection can provide helpful reminders to avoid this situation.

Unlike previous approaches [Gei02, GG03] that only deal with pre-defined static plan libraries, we develop a domain-independent approach for detecting plan abandonment that formalizes agent plans and goals over a planning domain definition, as follows in Definition 11.

Definition 11 (Plan Abandonment Problem) *A plan abandonment detection problem is a four-tuple $T_{PA} = \langle \Xi, \mathcal{I}, G, O \rangle$, in which $\Xi = \langle \Sigma, \mathcal{A} \rangle$ is the domain definition, and consists of a finite set of facts Σ and a finite set of actions $\mathcal{A}$; $\mathcal{I}$ represents the initial state; G is the monitored goal; and $O = \langle o_1, o_2, ..., o_n \rangle$ is an observation sequence of the plan execution with each observation $o_i \in O$ being an action in the finite set of actions $\mathcal{A}$ from the domain definition Ξ. To detect plan abandonment, we consider that this observation sequence is a full observation, which means that for a full observation sequence we observe all actions during the execution of an agent plan.*

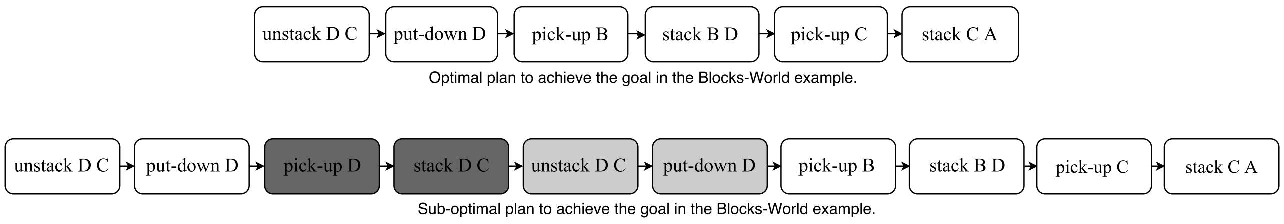

Figure 2.2 – Blocks-World – Practical example of plan abandonment.

As an example of what we consider plan abandonment, consider the Blocks-World planning problem example in Figure 2.1. From this planning problem, Figure 2.2 shows two possible plan executions that achieve the goal: an optimal and a sub-optimal plan. In the sub-optimal plan execution, boxes in gray represent actions that do not contribute to the goal, *i.e*, actions that indicate plan abandonment. Boxes in light-gray represent actions that must be taken to resume the plan execution to achieve the goal.

3. EXTRACTING RECOGNITION INFORMATION FROM DOMAIN ANALYSIS

In this chapter we review techniques from the literature that we adapt and use for extracting recognition information from domain analysis. Section 3.1 describes a reachability analysis technique that we use to reduce the amount of information of planning problems. In Section 3.2 we extend a landmark extraction algorithm for our recognition methods. In Section 3.3 we describe a method that classifies particular types of facts into partitions by analyzing preconditions and effects in operator definition. Finally, in Section 3.4 we discuss the usefulness from extracting recognition information.

3.1 Reachability Analysis

In automated planning, reachability analysis techniques can reduce the amount of facts and actions for a given planning instance. For some planning instances, this analysis can result in a significant reduction of the amount of facts and actions required to generate a plan for a planning instance, while in others, this analysis does not make any difference. To obtain an approximation of reachability from an initial state to a goal state, and reduce the amount of facts and actions for a planning instance, we use the Relaxed Planning Graph (RPG) structure. Using the RPG structure, it is possible to reason for finding the facts and actions that are really needed to achieve the goal. Usually, a large amount of unreachable actions is generated during the grounding process, and thereby the reachability analysis is able to prune such unreachable actions. With regard to the plan abandonment detection, we use the reachability analysis to prune actions that the observed agent does not have to perform to achieve a monitored goal. In this way, reachability analysis can reduce the agent's possible behaviors that can achieve a monitored goal. For example, in a DEPOTS[1] problem instance, the naive amount of actions is $|\mathcal{A}| = 29232$ because there are many objects to be considered (*e.g*, packages, pallets, trucks, and etc). However, after a simple reachability analysis the amount of actions becomes $|\mathcal{A}| = 4182$. The reachability analysis algorithm we implemented is shown in Algorithm 2. In this algorithm, the call BUILDFULLRPG (Line 2) builds a full RPG structure (Definition 7) given an initial state and a goal state, as shown in Algorithm 1.

Figures 3.1 and 3.2 show the reduction over the original amount of actions for two complex planning domains: DEPOTS and DOCK-WORKER-ROBOTS[2]. Regarding DEPOTS domain, the reachability analysis algorithm provides a huge reduction, reducing at least 50% of the original amount. However, for DOCK-WORKER-ROBOTS domain, the reduction is a bit less significant, reducing at most 15% of the original amount. For the BLOCKS-WORLD domain, the reachability analysis algorithm does not reduce the amount of actions for any planning problem.

[1]The DEPOTS domain combines transportation and stacking between locations. This domain was proposed by the International Planning Competition in 2002.

[2]The DOCK-WORKER-ROBOTS domain involves a number of cranes, locations, robots, containers, and piles, in which goals involve transporting containers to a final destination according to a desired order [GNT04].

Algorithm 1 Algorithm for Building a Relaxed Planning Graph.

Input: $\Xi = \langle \Sigma, \mathcal{A} \rangle$ *planning domain*, in which Σ is a finite set of facts and $\mathcal{A}$ is a finite set of actions, $\mathcal{I}$ *initial state*, and G *goal state*.
Output: RPG *relaxed planning graph*.

```
1: function BuildFullRPG($\Xi$, $\mathcal{I}$, $G$)
2:     $i := 0$
3:     $\mathrm{RPG.FactLevel}_0 := \mathcal{I}$
4:     while $G \not\subseteq \mathrm{RPG.FactLevel}_i$ do
5:         $\mathrm{RPG.ActionLevel}_i := \{a \in \mathcal{A} \mid pre(a) \in \mathrm{RPG.FactLevel}_i\}$
6:         $\mathrm{RPG.FactLevel}_{i+1} := \mathrm{RPG.FactLevel}_i \cup eff(a)^+, \forall a \in \mathrm{RPG.ActionLevel}_i$
7:         if $\mathrm{RPG.FactLevel}_{i+1} \equiv \mathrm{RPG.FactLevel}_i$ then
8:             return Failure To Build RPG    ▷ The algorithm fails whether at some point
   before reaching the facts of the goal no new fact level is added in the graph.
9:         end if
10:        $i := i + 1$
11:    end while
12:    return RPG
13: end function
```

Algorithm 2 Reachability Analysis Algorithm.

Input: $\Xi = \langle \Sigma, \mathcal{A} \rangle$ *planning domain*, in which Σ is a finite set of facts and $\mathcal{A}$ is a finite set of actions, $\mathcal{I}$ *initial state*, and G *goal state*.
Output: Σ' *set of reachable facts*, and $\mathcal{A}'$ *set of reachable actions*.

```
1: function FilterReachableFactsAndActions($\Xi$, $\mathcal{I}$, $G$)
2:     RPG := BuildFullRPG($\Xi$, $\mathcal{I}$, $G$)
3:     if $G \not\subseteq \mathrm{RPG.LastFactLevel}$ then
4:         return $G$ is not reachable
5:     end if
6:     $\Sigma', \mathcal{A}' := \langle\ \rangle$    ▷ $\Sigma'$ is a finite set of reachable ground facts, and $\mathcal{A}'$ is a finite set of
   reachable ground actions.
7:     for each level $i$ in RPG do
8:         $\Sigma' := \Sigma' \cup \mathrm{RPG.FactLevel}_i$
9:         $\mathcal{A}' := \mathcal{A}' \cup \mathrm{RPG.ActionLevel}_i$
10:    end for
11:    return $\Sigma', \mathcal{A}'$
12: end function
```

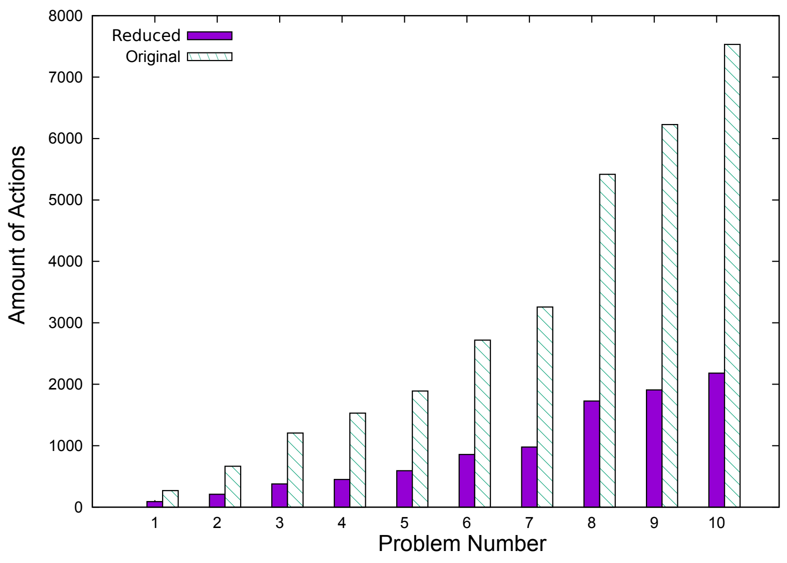

Figure 3.1 – DEPOTS amount of actions using Reachability Analysis. The problem number represents the problem size.

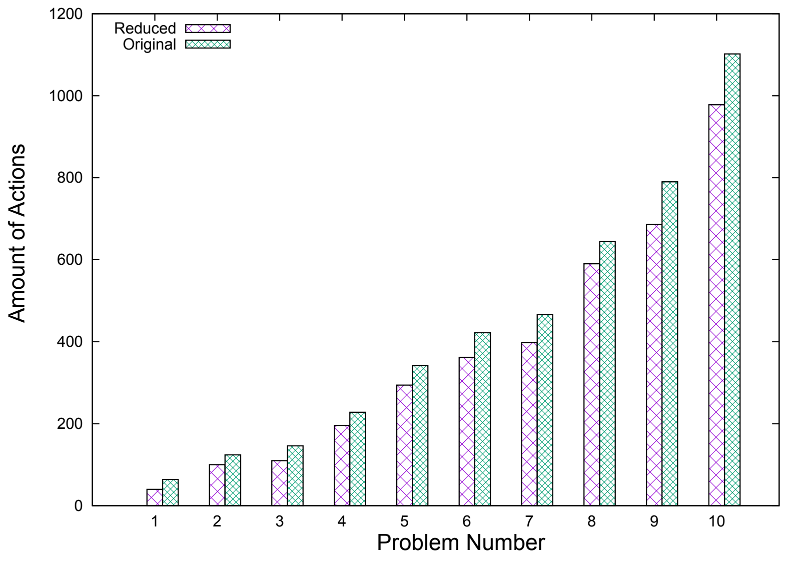

Figure 3.2 – DOCK-WORKER-ROBOTS amount of actions using Reachability Analysis. The problem number represents the problem size.

3.2 Extracting Landmarks

Using the concept of landmarks [HPS04] it is possible to monitor way-points in plan execution in order to determine whether a plan is going to be abandoned or be carried out until goal achievement. In other words, if some way-points are not achievable or they are too far, we may determine that the observed agent has no intention to complete the monitored plan. Landmarks allow us to infer whether a sequence of observations cannot possibly lead to a certain goal. In order to compute a set of landmarks of a planning problem, we adapt an existing landmark extraction algorithm from Hoffman et al. [HPS04]. To represent landmarks and their ordering, this algorithm uses a tree where nodes represent landmarks and edges represent necessary prerequisites between landmarks. Each node in the tree represents a conjunction of facts that must be true simultaneously at some point during plan execution, and the root node is a landmark representing the goal state. The landmark extraction algorithm we develop is shown in Algorithm 3. As input, the algorithm uses a ground planning problem, *i.e*, a finite set of ground actions and facts, an initial state, and a goal state. Initially, the algorithm builds a full RPG structure from an initial state to goal state (Line 2). If the goal state is not reachable in the RPG, then landmarks are not extracted, whereas if the goal state is reachable, a set of landmark candidates C is initialized with the facts from the goal state (Line 7). Afterwards, the algorithm iterates over the set of landmark candidates in C (Line 8), and, for each landmark l, the algorithm identifies fact landmarks that must be true immediately before l. This iteration ends when the set of landmark candidates C is empty. Then, in Line 10, from landmark candidate l the set of actions A' is extracted from the RPG. The set A' comprises all actions in the RPG at the action level immediately before l. In other words, A' represents those actions that achieve the level of facts in which l is. In Line 11, a for iteration filters the set A' for those actions such that can achieve l, *i.e*, actions that contain l in their effects. From these filtered actions, the algorithm takes as new landmark candidates those facts that every action that requires l as a precondition in these filtered actions (Line 13). In Line 14, the algorithm checks if the landmark candidate F is a landmark using a function called IsLandmark. This function evaluates whether a landmark candidate is a necessary landmark [HPS04]. For example, consider an RPG′ structure built from a planning problem $\mathcal{I}$ and G in which every action level in this RPG′ does not contain actions that achieve the landmark candidate l'. Given this modified RPG′ structure, we test the solvability of the planning problem $\mathcal{I}$ and G. So, if this problem is unsolvable, then the landmark candidate l' is a necessary landmark. More specifically, it means that such actions that achieve facts l' must exist in the RPG′ to solve this planning problem, that is, actions which contain the facts l' in their effects are necessary to solve the problem. Thus, in Line 15, facts which are necessary landmarks must be selected to extract other landmarks. Moreover, the algorithm stores facts that must be true together (Line 16). In Lines 19 and 21, the algorithm stores fact landmarks in their necessary order. Finally, from extracted fact landmarks in L, the algorithm extracts action landmarks that require these facts as preconditions (Lines 24 and 25).

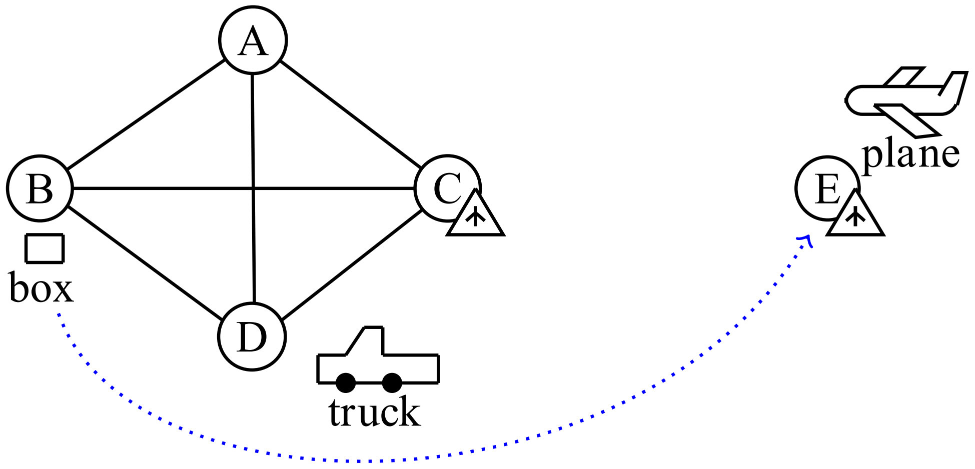

Figure 3.3 – Logistics problem example.

Algorithm 3 Landmark Extraction Algorithm.

Input: $\Xi = \langle \Sigma, \mathcal{A} \rangle$ *planning domain*, $\mathcal{I}$ *initial state*, and G *goal state*.
Output: L *set of fact landmarks*, and A *set of action landmarks*.

```
 1: function $\textsc{ExtractLandmarks}(\Xi, \mathcal{I}, G)$
 2:     $\text{RPG} := \textsc{BuildFullRPG}(\mathcal{I}, G)$
 3:     if $G \nsubseteq \text{RPG}.\textsc{LastFactLevel}$ then
 4:         return $\textsc{Failure To Extract Landmarks}$
 5:     end if
 6:     $L := \langle\ \rangle$                                    ▷ Set of extracted facts landmarks.
 7:     $C := G$                                                  ▷ Set of fact landmark candidates.
 8:     while $C \neq \emptyset$ do
 9:         $l := C.\textsc{pop}$
10:         $A' := \text{RPG}.\textsc{ActionLevel}_{(\text{RPG}.\textsc{level}(l)-1)}$
11:         for each action $a$ in $A'$ such that $l \in \mathit{eff}(a)^{+}$ do
12:             $L_{Together} := \langle\ \rangle$               ▷ Set of fact landmarks that must be true together.
13:             for each fact $F$ in $\mathit{pre}(a)$ do
14:                 if $\textsc{IsLandmark}(F, \mathcal{I}, G)$ then
15:                     $C := C \cup F$
16:                     $L_{Together} := L_{Together} \cup F$
17:                 end if
18:             end for
19:             $L := L \cup L_{Together}$
20:         end for
21:         $L := L \cup l$
22:     end while
23:     $A := \langle\ \rangle$                                    ▷ Set of extracted action landmarks.
24:     for each fact $l$ in $L$ do
25:         $A :=$ all action $a'$ in $\mathcal{A}'$ such that $l \in \mathit{pre}(a')$
26:     end for
27:     return $L, A$
28: end function
```

To exemplify the process of extracting landmarks, we use a well-known and realistic planning domain, LOGISTICS[3], as illustrated in Figure 3.3. This example shows two cities: the city on the left that contains four locations A through D; and the city on the right that contains only location E. The locations C and E are airports. The goal of this example is to transport the box located at location B to location E. For this example, the algorithm we adapt extract a set of fact and action landmarks, as shown respectively in Figure 3.4 and Listing 3.1. Note that Figure 3.4 shows such landmarks ordered by facts that must be true together (bottom-up). The algorithm we adapt extracts just a subset of landmarks for a given planning instance. Thus, using landmarks, we can monitor way-points during plan execution to determine which goals this plan is going to achieve. More specifically, we can discard candidate goals if some landmarks are not achievable or they do not appear as precondition or effect of actions in the observations.

[3]The LOGISTICS domain consists of airplanes and trucks transporting packages between locations (*e.g*, airports and cities).

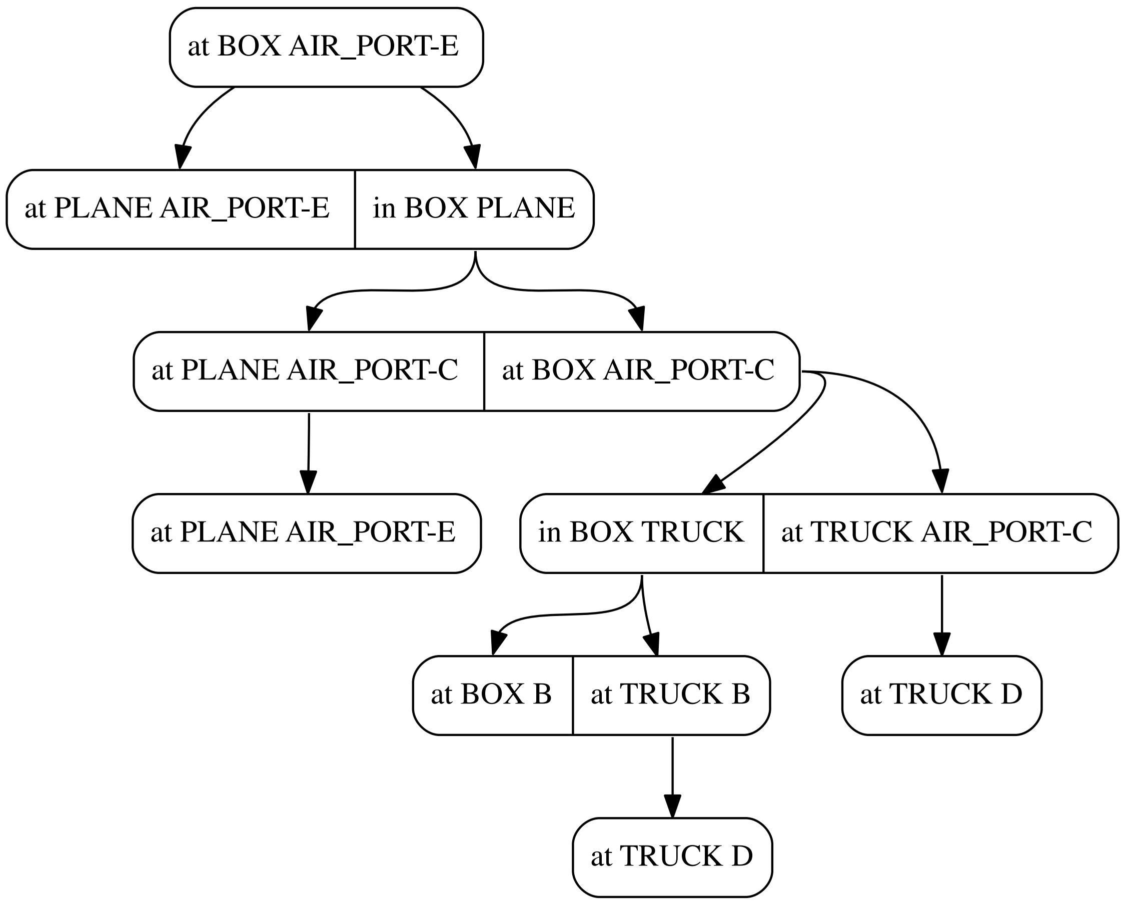

Figure 3.4 – Ordered fact landmarks extracted from LOGISTICS problem example shown in Figure 3.3. Fact landmarks that must be true together are represented by connected boxes, which are conjunctive facts, *i.e*, representing conjunctive landmarks.

```
(and (at BOX AIR_PORT-E))
(and (at PLANE AIR_PORT-E) (in BOX PLANE))
                => (unload-airplane BOX plane1 AIR_PORT-E)
(and (at PLANE AIR_PORT-E))
                => (fly-airplane PLANE AIR_PORT-E AIR_PORT-C)
(and (at PLANE AIR_PORT-C) (at BOX AIR_PORT-C))
                => (load-airplane BOX PLANE AIR_PORT-C)
(and (at BOX B) (at TRUCK B))
                => (load-truck BOX TRUCK B)
(and (at TRUCK D))
                => (drive-truck TRUCK D B CITY-1)
(and (in BOX TRUCK) (at TRUCK AIR_PORT-C))
                => (unload-truck BOX TRUCK AIR_PORT-C)
```

Listing 3.1 – Fact and Action landmarks extracted from LOGISTICS problem example shown in Figure 3.3.

3.3 Fact Partitioning

Pattison and Long [PL10] classify facts into mutually exclusive partitions in order to reason about whether certain observations are likely to be goals for their goal recognition approach. Their classification relies on the fact that, in some planning domains, there are facts that may provide additional information which can be extracted by analyzing preconditions and effects in operator definition. We use this classification to infer if certain observations are consistent with a particular goal, and if not, we can determine that this goal is not achievable. The fact partitions we use are defined as follows.

Definition 12 (Strictly Activating) *A fact f is strictly activating if $f \in \mathcal{I}$ and $\forall a \in \mathcal{A}$, such that $f \notin$ (eff(a)$^+ \cup$ eff(a)$^-$) and $\exists a \in \mathcal{A}$, such that $f \in$ pre(a).*

Strictly Activating facts appear as a precondition, and do not appear as add or delete effect in an operator definition. This means that unless defined in the initial state, this type of fact can never be added or deleted by an operator.

Definition 13 (Unstable Activating) *A fact f is unstable activating if $f \in \mathcal{I}$ and $\forall a \in \mathcal{A}$, $f \notin$ eff(a)$^+$ and $\exists a \in \mathcal{A}, f \in$ pre(a) and $\exists a \in \mathcal{A}$, $f \in$ eff(a)$^-$.*

Unstable Activating facts appear as both a precondition and a delete effect in two operator definitions, so once deleted, this type of fact cannot be re-achieved. The deletion of an unstable activating fact may prevent a plan execution to achieve a goal.

Definition 14 (Strictly Terminal) *A fact f is strictly terminal if $\exists a \in \mathcal{A}$, such that $f \in$ eff(a)$^+$ and $\forall a \in \mathcal{A}$, $f \notin$ pre(a) and $f \notin$ eff(a)$^-$.*

Strictly Terminal facts do not appear as a precondition of any operator definition, and once added, cannot be deleted. For some planning domains, this type of fact is the most likely to be in the set of goal facts, because once added in the current state, it cannot be deleted, this means that this fact will remain until the final state.

The fact partitions that we can extract depend on the planning domain definition. For example, from the BLOCKS-WORLD domain, it is not possible to extract any fact partition. However, it is possible to extract fact partitions from the DEPOTS domain, such as *Strictly Activating* facts. In this work, we use fact partitions to obtain more information on fact landmarks. For example, consider a fact landmark L_{ua} classified as *Unstable Activating*. If L_{ua} is deleted from the current state, then this landmark cannot be re-achieved by any action. Thus, we can trivially determine that goals for which this fact is a landmark are unreachable, because there is no available action that achieves the landmark L_{ua} again.

3.4 Chapter Remarks

In this chapter we have described how we extract recognition information for our recognition approaches. By using reachability analysis technique we can reduce substantially the amount of information that is necessary for solving a planning problem. Although for some planning domains this technique does not reduce the amount of information (*i.e*, a set of facts and actions), for most planning domains it does. We must clarify that the landmark extraction algorithm we have presented does not extract all landmarks for a given planning problem. In Section 6.4 we show the impact of extracting/using landmarks and fact partitions over the set of planning domains we use in this work.

4. LANDMARK-BASED PLAN RECOGNITION

In this chapter we present in detail our landmark-based approach for recognizing plans. Our plan recognition approach is employed in two methods. First, in Section 4.1 we describe a method for filtering candidate goals based on observations. This method can also be used by any planning-based plan recognition approach. Second, in Section 4.2 we show a heuristic method based on landmarks that we formulate for recognizing plans. Finally, in Section 4.3 we draw some conclusions about our landmark-based approach for plan recognition.

4.1 Filtering Candidate Goals From Landmarks in Observations

As the core of our approach to plan recognition, we develop a method for filtering candidate goals based on the evidence of fact landmarks and partitioned facts in preconditions and effects of observed actions in a plan execution. Thus, by analyzing fact landmarks in preconditions and effects of observed actions, we develop a filtering process that analyzes a set of candidate goals by selecting those goals that have achieved most of their associated landmarks.

This filtering process is detailed in function $\textsc{FilterCandidateGoals}$ of Algorithm 4, which takes as input a plan recognition problem (Definition 10), which is composed of a planning domain definition Ξ, an initial state $\mathcal{I}$, a set of candidate goals $\mathcal{G}$, and a set of observed actions O. Our algorithm iterates over the set of candidate goals $\mathcal{G}$, and, for each goal g in $\mathcal{G}$, it extracts and classifies fact landmarks and partitions for g from the initial state $\mathcal{I}$ (Lines 4 and 5). We then check whether the observed actions O contain fact landmarks or partitioned facts in either their preconditions or effects. At this point, if any *Strictly Activating* facts for the candidate goal g are not in initial state $\mathcal{I}$, then the candidate goal g is no longer achievable and we discard it (Line 11). Subsequently, we check for *Unstable Activating* and *Strictly Terminal* facts of goal g in the preconditions and effects of the observed actions O, and if we find any (Line 11), we discard the candidate goal g (Lines 12 and 19). If no facts from partitions are observed as evidence from the actions in O, we move on to checking landmarks of g within the actions in O. If we observe any landmarks in the preconditions and effects of the observed actions (Line 15), we compute the percentage of achieved landmarks for goal g. As we deal with partially observed actions in a plan execution, whenever a fact landmark is identified, we also infer that its predecessors have been achieved. For example, let us consider that the set of fact landmarks to achieve a goal from a state is represented by the following ordered facts: `(at A)` $\prec$ `(at B)` $\prec$ `(at C)` $\prec$ `(at D)`, and we observe just one action during a plan execution, and this observed action contains the fact landmark `(at C)` as an effect. Based on this partially observed action, we can infer that the predecessors of `(at C)` have been achieved before the observations, and thus, we also include them as achieved landmarks. Given the number of achieved fact landmarks of g, we estimate the percentage of fact landmarks that the observed actions O have achieved according to the ratio between the amount of achieved

Algorithm 4 Algorithm for Filtering Candidate Goals.

Input: $\Xi = \langle \Sigma, \mathcal{A} \rangle$ *planning domain*, $\mathcal{I}$ *initial state*, $\mathcal{G}$ *set of candidate goals*, and O *observations*.
Output: *A set of candidate goals with the highest percentage of achieved landmarks.*

```
 1: function FilterCandidateGoals($\Xi$, $\mathcal{I}$, $\mathcal{G}$, $O$)
 2:     $\Lambda_{\mathcal{G}} := \langle\rangle$                                   ▷ Map goals to % of landmarks achieved .
 3:     for each goal $G$ in $\mathcal{G}$ do
 4:         $\mathcal{L}_G :=$ ExtractLandmarks($\Xi, \mathcal{I}, G$)
 5:         $\langle F_{sa}, F_{ua}, F_{st} \rangle \leftarrow$ PartitionFacts($\mathcal{L}_g$)      ▷ $F_{sa}$: Strictly Activating, $F_{ua}$: Unstable
    Activating, $F_{st}$: Strictly Terminal.
 6:         if $F_{sa} \cap \mathcal{I} = \emptyset$ then
 7:             continue                                          ▷ Goal $G$ is no longer possible.
 8:         end if
 9:         $\mathcal{AL}_G := \langle\ \rangle$                                      ▷ Achieved landmarks for $G$.
10:         for each observed action $o$ in $O$ do
11:             if $(F_{ua} \cup F_{st}) \subseteq (pre(o) \cup eff(o)^+ \cup eff(o)^-)$ then
12:                 $discardG = $ true
13:                 break
14:             else
15:                 $L :=$ select all fact landmarks $l$ in $\mathcal{L}_G$ such that $l \in pre(o) \cup eff(o)^+$
16:                 $\mathcal{AL}_G := \mathcal{AL}_G \cup L$
17:             end if
18:         end for
19:         if $discardG$ then break                     ▷ Avoid computing achieved landmarks for $G$.
20:         end if
21:         $\Lambda_{\mathcal{G}} := \Lambda_{\mathcal{G}} \cup \langle G, \left(\frac{|\mathcal{AL}_G|}{|\mathcal{L}_G|}\right)\rangle$                    ▷ Percentage of achieved landmarks for $G$.
22:     end for
23:     return all $G$ s.t $\langle G, v \rangle \in \Lambda_{\mathcal{G}}$ and $v = \max_{v_i} \langle G', v_i \rangle \in \Lambda_{\mathcal{G}}$
24: end function
```

fact landmarks and the total amount of landmarks (Line 21). Finally, after analyzing all candidate goals in $\mathcal{G}$, we return the goals with the highest percentage of achieved landmarks (Line 23).

As an example of how Algorithm 4 filters a set of candidate goals, let us consider the Blocks-World problem example shown in Figure 4.1, which represents an initial configuration of stackable blocks, as well as set of candidate goals. The candidate goals consist of the following stackable words: `BED`, `DEB`, `EAR`, and `RED`. Now consider that the following actions have been observed in the plan execution: `(stack E D)` and `(pick-up S)`. After filtering the set of candidate goals, the algorithm returns the following filtered goals: `BED` and `RED`. The algorithm returns these goals because the observed action `(stack E D)` has in its preconditions the fact landmarks `(and (clear D) (holding E))`, and its effects contain `(on E D)`. Consequently, from these landmarks, it is possible to infer the evidence for another fact landmark, that is: `(and (on E A) (clear E) (handempty))`. This fact landmark is inferred because it must be true before the landmark `(and (clear D) (holding E)`. The observed action `(pick-up S)` does not provide any evidence for filtering the set the candidate goals. Thus, the estimated percentage of achieved fact landmarks of the filtered candidate goals `BED` and `RED` is 75%. The algorithm estimates 75% because both of these goals have 8 fact landmarks, and based on the evidence in the observed actions, we infer 6 fact

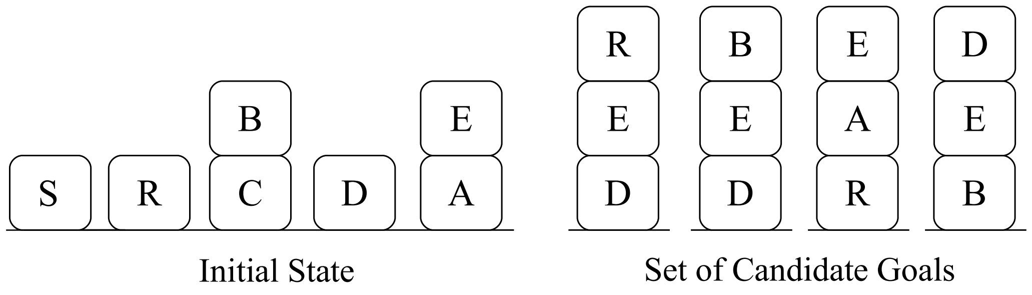

Figure 4.1 – BLOCKS-WORLD example containing a set of candidate goals.

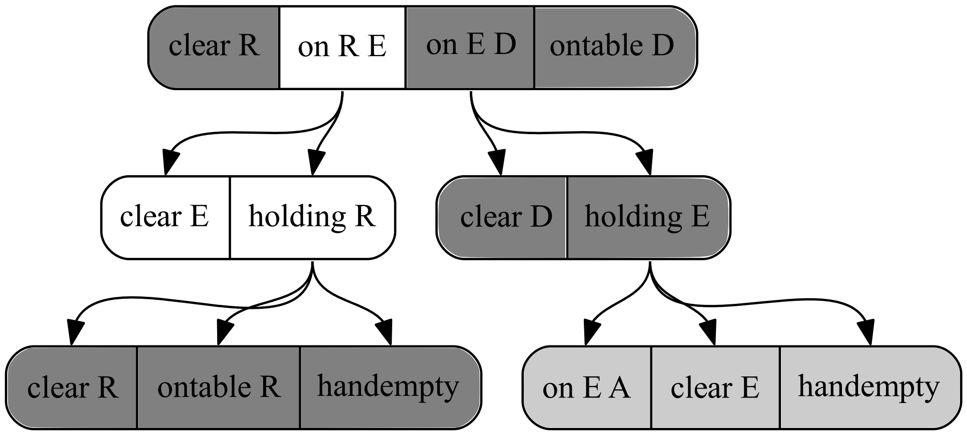

Figure 4.2 – Fact Landmarks for the word RED.

landmarks, *i.e*, $\frac{6}{8} = 0.75$. Regarding the goals `EAR` and `DEB`, the observations allow us to conclude that, respectively, 3 and 2 out of 7 and 9 fact landmarks were achieved. Figures 4.3 and 4.2 show the ordered fact landmarks for the filtered candidate goals `BED` and `RED`. Boxes in dark gray show achieved fact landmarks for these goals while boxes in light gray show inferred fact landmarks.

This filtering process shows that it can efficiently prune candidate goals that do not correspond to the observed plan execution. However, for observed actions that do not provide any evidence of landmarks this filter can be considered too greedy. This happens because the filter may eliminate the real goal of a set candidate goals due to the absence of evidence in observations.

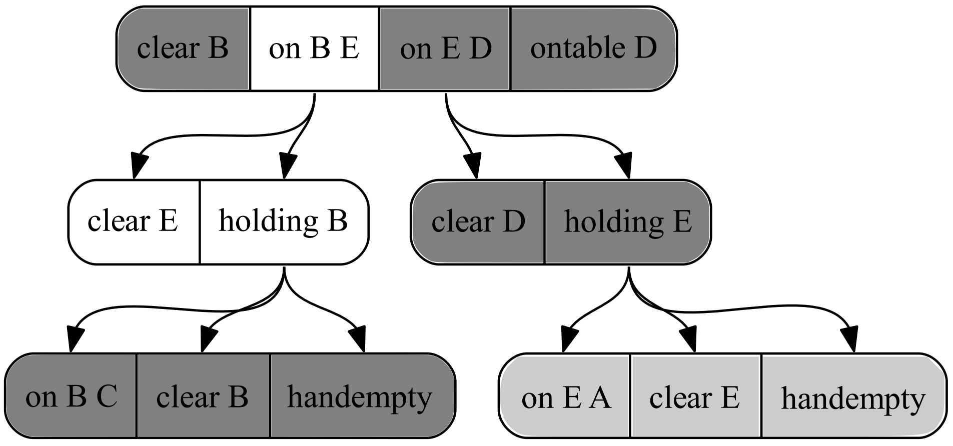

Figure 4.3 – Fact Landmarks for the word BED.

Moreover, we argue that this filter does not execute any planning algorithm during the filtering process, and it is thus fundamentally more efficient, we filter out goals by considering only available evidence provided by the observed actions (*i.e*, landmarks). In addition, this filtering process can be used by any planning-based plan recognition approach, and we show in detail its applicability to a state-of-the-art plan recognition approach in Chapter 6.

4.2 Heuristic Plan Recognition Using Landmarks

We now present a landmark-based plan recognition heuristic that estimates the goal completion of every goal in the set of filtered goals by analyzing the number of achieved landmarks for each goal provided by the filtering process. Each candidate goal is composed of a set of sub-goals: atomic facts that are part of a conjunction of facts. This estimate represents the percentage of sub-goals in a goal that have been accomplished based on the evidence of achieved fact landmarks in observations.

In order to estimate the completion of a goal, our heuristic method computes the percentage of completion completion towards a goal by using the set of achieved fact landmarks provided by the filtering process, as shown in Algorithm 4 (Line 15). We aggregate the percentage of completion of each sub-goal into an overall percentage of completion for all facts in a candidate goal. This heuristic, denoted as h_{prl}, is computed by the following formula, where $\mathcal{AL}_g$ is the number of achieved landmarks from observations of a candidate goal G, and $\mathcal{L}_g$ represents the number of necessary landmarks to achieve every sub-goal g of G:

$$h_{prl}(G) = \left(\frac{\sum_{g \in G} \frac{|\mathcal{AL}_g|}{|\mathcal{L}_g|}}{|G|} \right) \tag{4.1}$$

In this way, heuristic $h_{prl}(G)$ estimates the completion of a goal G by calculating the ratio between the sum of the percentage of completion for every sub-goal $g \in G$, denoted as $\sum_{g \in G} \frac{|\mathcal{AL}_g|}{|\mathcal{L}_g|}$, and the number of sub-goals in G.

To exemplify how heuristic h_{prl} estimates a goal completion, recall the BLOCKS-WORLD example in Figure 4.1. Considering the goal `DEB` and, as shown in Figure 4.4, has the following sub-goals: `(clear D)`, `(on D E)`, `(on E B)`, and `(ontable B)`. Furthermore, consider the observed actions: `(unstack B C)`, `(pick-up R)`, and `(stack D E)`. Initially, we conclude that the sub-goal `(clear D)` has been already achieved because it is in the initial state, and the observed actions do not delete this fact. Although fact `(clear D)` in the initial state does not correspond to the final configuration of goal `DEB`, because block D must also be on block E, we take this fact in the heuristic calculation since we consider all observed evidence. At this point, our heuristic computes that 25% of the goal `DEB` has been accomplished. However, for this goal, there is even more information to

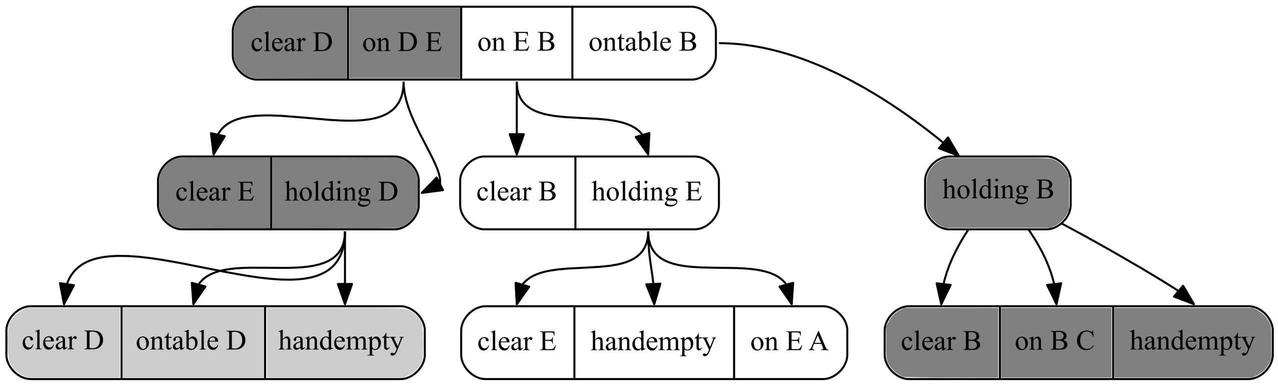

Figure 4.4 – Fact landmarks for the word DEB.

be considered in order to calculate the percentage of the goal completion for the work `DEB`. By analyzing the amount of achieved fact landmarks for the others sub-goals, we have the following achieved landmarks for them.

- **`(clear D)`**: is in the initial state, and this sub-goal achieved 1 out of its 1 landmark;
- **`(on D E)`**: for this sub-goal, the observed action `(stack D E)` provides the landmarks `(and (clear E) (holding D)` and `(and (clear D) (on D E)`, as preconditions and effects respectively. Therefore, from fact landmark `(and (clear E) (holding D)`, we infer another fact landmark, `(and (on E A) (clear E) (handempty))`. This fact landmark is inferred because it must be true before fact landmark `(and (clear D) (holding E))`. This sub-goal achieved 3 out of its 3 landmarks ;
- **`(on E B)`**: this sub-goal does not have any achieved fact landmark; and
- **`(ontable B)`**: for this sub-goal, the observed action `(unstack B C)` provides the landmarks `(and (on B C) (clear B) (handempty))` and `(holding B)`, as preconditions and effects respectively. This sub-goal achieved 2 out of its 3 landmarks.

The observed action `(pick-up R)` does not show any evidence of landmarks that contribute to the h_{prl} heuristic. Thus, to conclude this example, heuristic h_{prl} estimates that from the evidence of achieved landmarks in the observed actions, the percentage of completion for the goal `DEB` is 66%, as follows.

$$h_{prl}(G) = \frac{1 + \frac{3}{3} + 0 + \frac{2}{3}}{4} = 0.66 \tag{4.2}$$

4.3 Chapter Remarks

We have shown that our landmark-based plan recognition heuristic can estimate goal completion from observations by computing the ratio of achieved landmarks for each fact (sub-goal) in a goal. Nevertheless, this heuristic in its current shape has limitations when eliminating goal hypotheses. Assume that the goal completion of two candidate goals is too close, *e.g*, 49% and 50%, and the goal completion of the real goal is 49%, in this way, our heuristic may eliminate a real goal erroneously. Thus, to improve the estimate of goal completion for eliminating goal hypotheses, we aim to use a parameterizable threshold for eliminating candidate goals whose goal completion are too close (*e.g*, 49% and 50%).

5. PLANNING-BASED PLAN ABANDONMENT DETECTION

In this chapter we present a planning-based approach for detecting plan abandonment. In Section 5.1 we describe a pre-processing stage that computes key information from a planning domain definition. Section 5.2 defines behavior analysis, a method that checks which observed actions in a plan execution do not contribute to achieve a monitored goal. In Section 5.3 we develop our approach to detect plan abandonment that brings together the domain analysis and behavior analysis methods. Finally, in Section 5.4 we conclude this chapter with a discussion regarding our approach for detecting plan abandonment.

5.1 Pre-processing Key Information From Domain Analysis

We now present a pre-processing stage that computes key information given a plan abandonment detection problem by using techniques presented in Chapter 3. More specifically, this pre-processing stage is composed of three phases, as follows.

- In the first phase, this pre-processing stage filters facts and actions performing a planning graph-based reachability analysis technique – Algorithm 2. We filter facts and actions of a planning problem in order to reduce the amount of information of the problem, and thus, we monitor only necessary information (*e.g*, facts and actions) for achieving a monitored goal;
- Second, after the reachability analysis phase, this phase extracts fact and action landmarks – Algorithm 3. Fact and action landmarks are stored according to the order in which they must be achieved and executed, and thus, we monitor way-points that are necessary for achieving a monitored goal; and
- Lastly, this method attempts to classify particular types of predicates by analyzing preconditions and effects in the domain operators definition. We classify predicates according to partitions defined in Section 3.3. This classification allows our approach to monitor particular landmarks, for instance, a landmark that once deleted, cannot be achieved again.

5.2 Behavior Analysis

Using the information obtained from the pre-processing stage, we now present two methods that analyze the agent behavior during a plan execution. The first method uses a domain-independent heuristic to estimate the distance to the monitored goal for every observed action in the plan execution, and thus analyze if there is some variation between them. The second method aims to anticipate what action the agent has to perform in the next observation in order to reduce the estimated distance to the monitored goal.

5.2.1 Analyzing Plan Execution Variation

To analyze possible plan execution variation, we compute the estimated distance to the monitored goal for every observation in the plan execution. Since admissible heuristics are often too expensive to estimate the distance to a goal. Therefore, for this method, we use an inadmissible domain-independent heuristic, the Fast Forward (FF) heuristic h_{ff}, a well-known heuristic in the planning community [HN01], as introduced in Subsection 2.2.2. Using the estimated distance to the monitored goal, we analyze possible variation between every observations in a plan execution. We consider a variation between observations when such test is satisfied: $h(o_{i-1}) < h(o_i)$. This test checks if the estimated distance to the state resulting from the action observed at time i is greater than the state resulting from the action observed at time $i-1$. When such variation happens during plan execution, it may indicate that the agent behavior is unstable to achieve the goal, or the agent is performing concurrent plans. Figure 5.1 illustrates graphically how we use the h_{ff} heuristic to analyze variation for plan execution. For this, we use two plan executions: an optimal and a sub-optimal one, as shown in the plan abandonment example (Figure 2.2). Note that during the execution of the sub-optimal plan (red) there are variations between observation times: 1 and 2; 2 and 3. By analyzing this plan variation, we conclude that actions `(pick-up D)` and `(stack D C)` do not contribute to achieve the goal because they increase the distance to the goal.

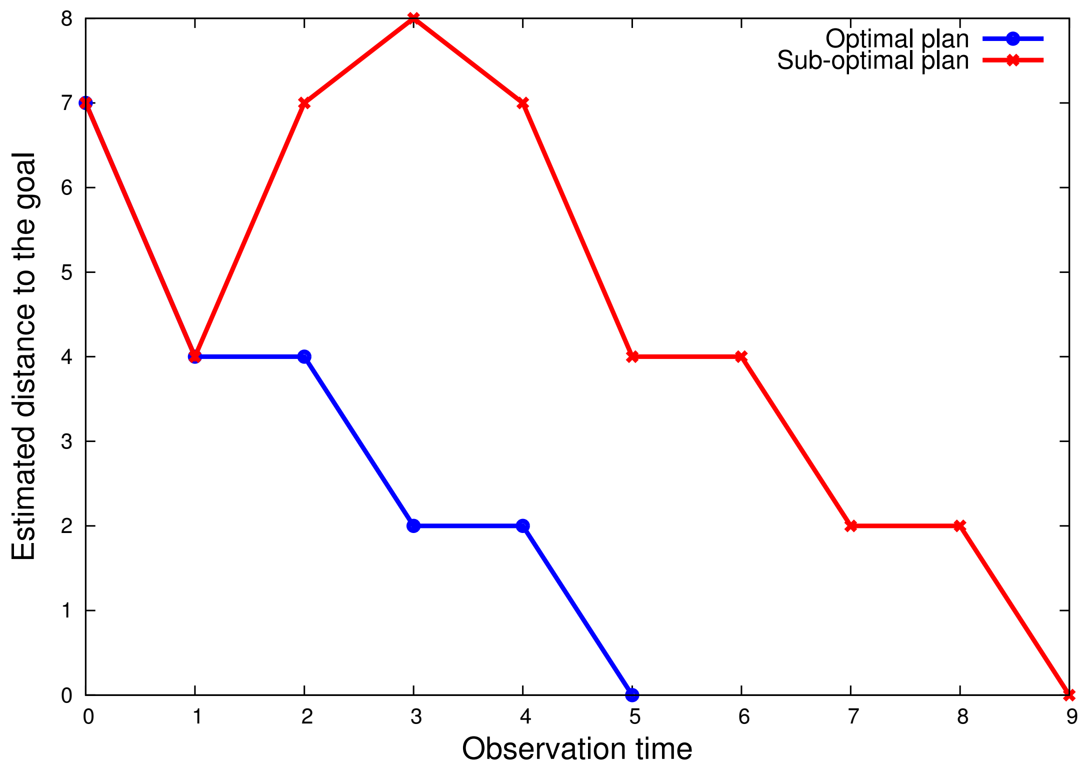

Figure 5.1 – Plan Execution example using heuristic h_{ff} for the BLOCKS-WORLD problem example shown in Figure 2.1.

5.2.2 Predicting Upcoming Actions from Landmarks

Ordered landmarks (facts and actions) provide what we consider to be "way-points" towards the monitored goal. We extract landmarks to obtain key information about what cannot be avoided to achieve a particular monitored goal. Unlike prior heuristic-based landmark extractors, we compute the initial state as part of the fact landmarks, and thus we compute the landmarks from the initial state to the goal state, such as the landmarks shown in Figure 5.2. From this, we propose an approach that exploits fact landmarks of a monitored goal for predicting which actions might be executed in the next observation in order to reduce the distance for both upcoming landmarks and monitored goal. This approach uses such prediction to check the set of observed actions of a plan execution to determine which actions do not contribute to the monitored goal. Algorithm 5 formalizes this approach.

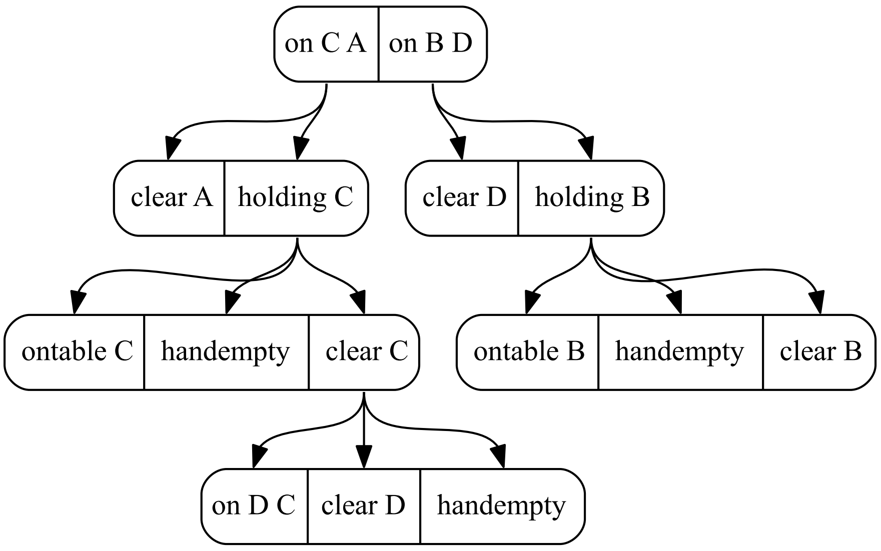

Figure 5.2 – Fact Landmarks for the Blocks-World problem example shown in Figure 2.1.

In order to predict which actions might be "good" to be executed in the next observation, our algorithm analyzes the closest landmarks by estimating the distance to the landmarks from the current state. For estimating the distance to landmarks, our algorithm uses an admissible domain-independent heuristic, the max heuristic, which is denoted as h_{max}, as introduced in Subsection 2.2.1. For this algorithm, we decide to use an admissible domain-independent heuristic because it never overestimates the distance to the monitored goal. We consider that the closest fact landmarks are those that return estimated distance $h_{max}(l) = 0$ and $h_{max}(l) = 1$. In this way, the algorithm iterates over a set of ordered fact landmarks $\mathcal{L}$ (Line 3), and, for each landmark l, the max heuristic estimates the distance from the current state to l. If the estimated distance to landmark l is $h_{max}(l)$ = 0 (Line 5), this means that this landmark is in the current state, and the algorithm selects those actions that contain such landmark as a precondition, because these can be immediately executed (Line 6). Otherwise, if the estimated distance to landmark l is $h_{max}(l) = 1$ (Line 7), this means that

```
- (on C A)                                     = 3
 (and (clear A) (holding C))                   = 2
 (and (ontable C) (handempty) (clear C))       = 1
 (and (on D C) (clear D) (handempty))          = 0
     * To achieve this landmark an applicable action is:
                (unstack D C)
- (on B D)                                     = 2
 (and (clear D) (holding B))                   = 1
 (and (ontable B) (handempty) (clear B))       = 0
     * To achieve this landmark an applicable action is:
                (pick-up B)
```

Listing 5.1 – Predicting upcoming actions from the BLOCKS-WORLD. example.

this landmark can be reached by executing a single action, and the algorithm selects those actions that are applicable in the current state and contain such landmark as an effect (Line 8). These actions are selected because they reduce the distance to the next landmark, and consequently to the monitored goal. Thus, we can estimate, using the observed plan execution, which actions do not contribute to achieve a goal.

Algorithm 5 Algorithm for Predicting Upcoming Actions from Landmarks.

Input: $\Xi = \langle \Sigma, \mathcal{A} \rangle$ *planning domain*, δ *current state*, and $\mathcal{L}$ *ordered fact landmarks*.

Output: $\eta_{UpcomingActions}$ *set of possible upcoming actions*.

```
 1: function PREDICTUPCOMINGACTIONS($\Xi$, $\delta$, $\mathcal{L}$)
 2:     $\eta_{UpcomingActions} := \langle\ \rangle$
 3:     for each fact landmark $l$ in $\mathcal{L}$ do
 4:         $Al := \langle\ \rangle$
 5:         if $h_{max}(l) = 0$ then                        ▷ $h_{ff}(l)$ estimates $l$ from $\delta$.
 6:             $Al :=$ all action $a$ in $\mathcal{A}$ such that $l \in pre(a)$
 7:         else if $h_{max}(l) = 1$ then
 8:             $Al :=$ all action $a$ in $\mathcal{A}$ such that $pre(a) \in \delta \wedge l \in eff(a)^+$
 9:         end if
10:         $\eta_{UpcomingActions} := \eta_{UpcomingActions} \cup Al$
11:     end for
12:     return $\eta_{UpcomingActions}$
13: end function
```

To exemplify how our algorithm predicts upcoming actions, recall the BLOCKS-WORLD problem example shown in Figure 2.1. If the current state is the initial state, then the algorithm predicts upcoming actions that might be executed as the first observation in the plan execution. As output for this example, Listing 5.1 shows fact landmarks (on the left), the estimated distance from the initial state to fact landmarks (after the symbol =), and on the bottom of the fact landmarks, which applicable actions our method predicts to be the first observation. Note that, although there are fact landmarks for which the estimated distance is $h_{max}(l) = 1$, there is no applicable action in the initial state to achieve these fact landmarks.

5.3 Detecting Plan Abandonment

We now describe our planning-based approach to detect plan abandonment, which is formalized in Algorithm 6. As a planning-based approach, this algorithm takes as input a planning domain, an initial state, a monitored goal, and a set of observed actions as the execution of an agent plan, following Definition 11. At first, for detecting plan abandonment, the algorithm computes key information using the pre-processing methods described in Section 5.1. Afterward, the algorithm analyzes the plan execution by iterating over the set observed actions, and, for each observed action, it checks whether that action does not contribute to the monitored goal. We call those actions that do not contribute to a goal as non-optimal actions. To check observed actions, we use the behavior analysis methods described in Section 5.2. For analyzing plan execution variation we use of the h_{ff}, and to predict upcoming actions, we use the h_{max}. We use fact partitions to extract additional information about fact landmarks of a monitored goal. If we identify such partitions during a plan execution we can immediately determine that the analyzed plan execution is not going to achieve the monitored goal.

Algorithm 6 Algorithm for Detecting Plan Abandonment.

```
Input: $\Xi = \langle \Sigma, \mathcal{A} \rangle$ planning domain, $\mathcal{I}$ initial state, $G$ monitored goal, and $O$ observed actions.
Output: $A_{NonOptimalActions}$ as actions.
 1: function DetectPlanAbandonment($\Xi$,$\mathcal{I}$,$G$,$O$)
 2:     $\Xi' = \langle \Sigma', \mathcal{A}' \rangle$ := FilterReachableFactsAndActions($\Xi$, $\mathcal{I}$, $\mathcal{G}$)   ▷ Filtering the set of facts $\Sigma$ and set of actions $\mathcal{A}$ in $\Xi$ using Reachability Analysis.
 3:     $L$ := ExtractLandmarks($\Xi'$, $\mathcal{I}$, $G$)
 4:     $F_{Partitions}$ := PartitionFacts($L$)
 5:     $\delta := \mathcal{I}$   ▷ $\delta$ is the current state.
 6:     $\eta_{UpcomingActions}$ := PredictUpcomingActions($\Xi'$, $\delta$, $L$)
 7:     $D_G$ := EstimateGoalDistance($\delta$, $G$)   ▷ $h_{max}$ estimating goal $G$ from $\delta$.
 8:     $A_{NonOptimalActions} := \langle \rangle$ ▷ Actions that do not contribute to achieve the monitored goal $G$.
 9:     for each observed action $o$ in $O$ do
10:         $\delta := \delta$.Apply($o$)
11:         if Facts in $F_{Partitions}$ are in $\delta$ then
12:             return Monitored goal $G$ is no longer possible.
13:         end if
14:         $D'_G$ := EstimateGoalDistance($\delta$, $G$)
15:         if $o \notin \eta_{UpcomingActions} \wedge (D'_G > D_G)$ then
16:             $A_{NonOptimalActions} := A_{NonOptimalActions} \cup o$
17:         end if
18:         $\eta_{UpcomingActions}$ := PredictUpcomingActions($\Xi'$, $\delta$, $L$)
19:         $D_G := D'_G$
20:     end for
21:     return $A_{NonOptimalActions}$
22: end function
```

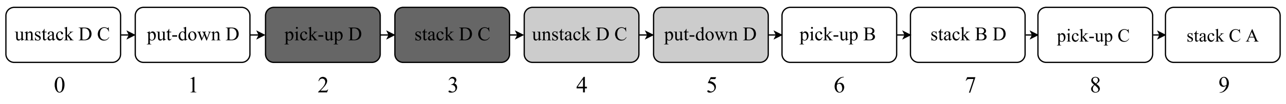

Figure 5.3 – Sub-optimal Plan Execution that achieves the goal in the BLOCKS-WORLD problem example shown in Figure 2.1.

Algorithm 6 estimates which actions do not contribute for achieving a monitored goal by returning non-optimal actions of the plan execution. This algorithm can also determine that a monitored goal is no longer possible to be achieved for a given plan abandonment detection problem, since the evidence of classified landmarks as fact partitions in the plan execution.

To exemplify how our approach detects plan abandonment, let us consider the sub-optimal plan execution of Figure 5.3 for the BLOCKS-WORLD problem example shown in Figure 2.1. This plan execution contains 10 actions, in which they have been executed subsequently from observation time 0 to 9. Before observation time 0, as shown in Listing 5.1, our algorithm predicts that the actions that can reduce the distance to the next landmarks are: `(unstack D C)` or `(pick-up B)`, and as shown in the plan execution one of these actions has been performed, and consequently this action reduces the distance to the the goal and next landmarks. At observation time 1, our algorithm predicts that the next action must be: `(put-down D)`, and the predicted action has been performed. After that, at observation time 2, our algorithm predicts that next actions can be: `(pick-up B)` or `(pick-up C)`, and as shown in the plan execution, the performed action is `(pick-up D)`. Although `(pick-up D)` does not correspond to our prediction, this action deviates for achieving the goal, *i.e*, the estimate distance to goal increases at this observation time, as shown in Figure 5.1 – at observation time 2. So, this action is added as non-optimal action. Subsequently, at observation time 3, the action `(stack D C)` is also considered as a non-optimal action, since it does not correspond to our prediction (which is `(put-down D)`), and such action also increases the distance to the goal. Afterward, at observation time 4, as our algorithm predicts (`(unstack D C)` or `(pick-up B)`), the observed action is the expected action, *i.e*, `(unstack D C)`, and thereby from this observation until the last one the plan execution remains correct as predicted for achieving the monitored goal. Note that the actions performed at observation 4 and 5 represent expected actions that advance the plan execution for achieving the monitored goal.

5.4 Chapter Remarks

We have shown that our approach detects the observation time when an action deviates for achieving the monitored goal. For this example, our approach detects efficiently what actions do not contribute to achieve the monitored goal, but there are planning problem examples that our approach does not detect non-optimal actions efficiently (for more details see Section 6.6). Our approach can also be employed in at least four additional applications, as follows:

1. to provide helpful reminders for an observed agent in order to resume the plan execution for achieving the monitored goal, as shown in Listing 5.1;

2. for monitoring plan optimality;

3. to analyze possible concurrent plans; and

4. to detect interleaving plans in a plan execution.

6. EXPERIMENTS AND EVALUATION

In this chapter we present empirical results of experiments over our recognition approaches. In Section 6.1 we describe planning domains that we use to experiment and evaluate our recognition approaches. Section 6.2 describes the evaluation metrics we use to evaluate our recognition approaches. Sections 6.3 and 6.4 show the impact of using domain-independent heuristics and landmarks of the selected planning domains over our recognition approaches. In Section 6.5 we show experimental results of our plan recognition approach against state-of-the-art planning-based plan recognition approach. In Section 6.6 we show experimental results of our planning-based approach for detecting plan abandonment. Finally, in Section 6.7 we conclude this chapter by discussing the results of using our recognition approaches over complex planning domains.

6.1 Domains

For experiments we select a set of planning domains from different editions of the International Planning Competition (IPC) domains[1], more specifically, from IPC–1 to IPC–5. Most of these domains model real-world scenarios [FL11]. In addition, we use a dataset provided by Ramírez and Geffner[2], comprising hundreds of plan recognition problems.

- BLOCKS-WORLD domain consists of a set of blocks, a table, and a robot hand. Blocks can be on top of other blocks or on the table. A block that has nothing on it is clear. The robot hand can hold one block or be empty. The goal is to find a sequence of actions that achieves a final configuration of blocks;

- DEPOTS domain combines transportation and stacking. For transportation, packages can be moved between depots by loading them on trucks. For stacking, hoists can stack packages on palettes or other packages. The goal is to move and stack packages by using trucks and hoists between depots;

- DOCK-WORKER-ROBOTS is a domain that involves a number of cranes, locations, robots, containers, and piles. Goals involve transporting containers to a final destination according to a desired order;

- DRIVER-LOG is a domain that contains drivers that can walk between locations and trucks that can drive between locations. Walking from locations requires traversal of different paths. Trucks can be loaded with or unloaded of packages. The goal of this domain is to transport packages between locations, ending up with a subset of the packages, and trucks and drivers at specific destinations;

[1]http://www.icaps-conference.org/index.php/Main/Competitions
[2]https://sites.google.com/site/prasplanning/file-cabinet

- Easy-IPC-Grid domain consists of an agent that moves in a grid from connected cells to others by transporting keys to open locked locations;

- Intrusion-Detection represents a domain where a hacker tries to access, vandalize, steal information, or perform a combination of these attacks on a set of servers;

- Ferry is a domain that consists of set of cars that must be moved to desired locations using a ferry that can carry only one car at a time. A location is accessible from each other location. Cars can be debarked or boarded. A ferry can carry only one car at a time;

- Logistics is a domain which models cities, and each city contains locations. These locations are airports. For transporting packages between locations, there are trucks and airplanes. Trucks can drive between cities. Airplanes can fly between airports. The goal is to get and transport packages from locations to other locations;

- Miconic is a domain that involves transporting a number of passengers using an elevator to reach destination floors;

- Rovers is a domain that models a set of rovers that navigate in a planet surface. Rovers must navigate on this planet's surface to collect samples, take pictures of objectives, and communicate the results of these tasks to a lander;

- Satellite is a domain that involves using one or more satellites to make observations, by collecting data and down-linking the data to a desired ground station; and

- Zeno-Travel is a domain where passengers can embark and disembark onto aircraft that can fly at two alternative speeds between locations.

As an example of input that we use for our recognition approaches, we show a domain and problem formalization in PDDL 2.1, which is equivalent to STRIPS representation. Listing 6.1 illustrates a PDDL domain that formalizes a set of operators and predicates of the Easy-IPC-Grid domain. For example, the action `move` defines that a robot can perform a movement to any place inside the grid, since that robot is at a current position `curpos`, this current position must be connected to the next position `netxpos` (*e.g*, `place_0_0` is connected to `place_0_1`), and the next position must be open, *e.g*, `(open place_0_1)`. Listing 6.2 illustrates a PDDL problem instance within this domain, showing a grid that contains 9 places (3 x 3), 1 key, and a locked place. The goal of this problem instance represents a robot that want to achieve the `place_2_2`, *i.e*, `(at-robot place_2_2)`. We use the symbol [...] to represent a code fragmentation, which means that we summarize some parts in problem formalization.

```
(define (domain grid)
(:requirements :strips :typing)
(:types place shape key)
(:predicates
  (conn ?x ?y - place)
  (key-shape ?k - key ?s - shape)
  (lock-shape ?x - place ?s - shape)
  (at ?r - key ?x - place )
  (at-robot ?x - place)
  (locked ?x - place)
  (carrying ?k - key)
  (open ?x - place)
)
(:action unlock
  :parameters (?curpos ?lockpos - place ?key - key ?shape - shape)
  :precondition (and (conn ?curpos ?lockpos) (key-shape ?key ?shape)
                     (lock-shape ?lockpos ?shape) (at-robot ?curpos)
                     (locked ?lockpos) (carrying ?key))
  :effect (and (open ?lockpos) (not (locked ?lockpos)))
)
(:action move
  :parameters (?curpos ?nextpos - place)
  :precondition (and (at-robot ?curpos)
                     (conn ?curpos ?nextpos)
                     (open ?nextpos))
  :effect (and (at-robot ?nextpos) (not (at-robot ?curpos)))
)
(:action pickup
  :parameters (?curpos - place ?key - key)
  :precondition (and (at-robot ?curpos) (at ?key ?curpos))
  :effect (and (carrying ?key) (not (at ?key ?curpos)))
))
```

Listing 6.1 – EASY-IPC-GRID domain formalization in PDDL.

6.2 Evaluation Metrics

To evaluate our recognition approaches, we use the following metrics: accuracy, precision, recall, and F1-score. For evaluation of our approach for detecting plan abandonment, we now clarify what represents each used metric. Precision is the ratio between true positive results, and the sum of true positive and false positive results. True positive results represent the number of correct abandoned plans that our approach has detected. False positive results represent the number of successful plans that our approach has detected as abandonment. Recall is the ratio between true positive results, and the sum of true positive and false negative results. False negative results represent the number of abandoned plans that our approach has not detected as abandonment. Precision provides the percentage of positive predictions that is correct. Recall provides the percentage of positive cases that our approach has detected. F1-score is a measure of accuracy that aims to provide a trade-off between precision and recall. For evaluation of our plan recognition approach, we use the accuracy metric, which represents how well a hidden goal is recognized from a set of possible goals for a given plan recognition problem. Formally, these metrics are defined as follows.

```
(define (problem grid-problem-example)
(:domain grid)
(:objects
  place_0_0 place_0_1 place_0_2
  place_1_0 place_1_1 place_1_2
  place_2_0 place_2_1 place_2_2 - place
  key_0 - key
  shape_0 - shape
)
(:init
  (at-robot place_1_1)
  (key-shape key_0 shape_0)
  (at key_0 place_0_2)
  (conn place_0_0 place_0_1) (conn place_0_0 place_1_0)
  (conn place_1_0 place_0_0) (conn place_1_0 place_1_1)
  (conn place_1_0 place_2_0)
  ...
  (locked place_2_2) (lock-shape place_2_2 shape_0)
)
(:goal
  (and (at-robot place_2_2))
))
```

Listing 6.2 – EASY-IPC-GRID problem formalization in PDDL.

$$\textsc{Precision} = \frac{\sum True\ \ positive}{\sum True\ \ positive + \sum False\ \ positive} \tag{6.1}$$

$$\textsc{Recall} = \frac{\sum True\ \ positive}{\sum True\ \ positive + \sum False\ \ negative} \tag{6.2}$$

$$\text{F1-}\textsc{score} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{6.3}$$

$$\textsc{Accuracy} = \frac{\sum True\ \ positive + \sum True\ \ negative}{\sum Total\ \ population} \tag{6.4}$$

6.3 Impact of using Domain-Independent Heuristics

We now show how the used domain-independent heuristics perform over some of the selected planning domains. As we reported before, we use two domain-independent heuristics: h_{max}, an admissible heuristic; and h_{ff}, an inadmissible heuristic. To show how these heuristics perform to estimate the goal distance, we select 3 complex and realistic domains from the set of planning domains presented before, as follows: DEPOTS, DRIVER-LOG, and SATELLITE. For these domains, we select 10 planning problems varying their complexity and size according to the problem number. Table 6.1 shows a comparison between heuristic estimates of h_{max} and h_{ff} against the optimal plan ($|\pi|$) for 10 planning problems for the planning domains Depots, Satellite, and Driver-Log.

	Depots			Satellite			Driver-Log		
Problem	h_{max}	h_{ff}	$\|\pi\|$	h_{max}	h_{ff}	$\|\pi\|$	h_{max}	h_{ff}	$\|\pi\|$
1	4	10	11	3	8	9	6	6	6
2	5	14	17	3	15	13	4	16	19
3	5	24	27	3	12	13	4	11	17
4	4	20	31	3	22	20	4	15	18
5	5	25	32	3	25	22	4	16	23
6	5	20	34	3	23	26	3	11	10
7	4	28	38	3	32	28	4	17	18
8	6	40	67	3	35	34	4	18	25
9	9	46	75	3	39	35	6	25	22
10	8	45	111	3	45	40	4	19	20

Table 6.1 – Comparison between heuristic estimates of h_{max} and h_{ff} against the optimal plan $|\pi|$ for the planning domains Depots, Satellite, and Driver-Log.

From this table, it is possible to see that the estimated cost provided by h_{max} is too low, and h_{max} underestimates the cost to the goal for all planning problems of the selected domains, excepting for the problem 1 of Driver-Log domain, in which h_{max} estimates the same cost compared to the optimal cost. Yet, the cost that h_{ff} estimates is quite close compared to the optimal cost. For Satellite domain, h_{ff} overestimates the cost for most planning problems, but such overestimate exceeds at most 5 actions compared to the optimal plan. For Driver-Log, h_{ff} overestimates the cost for 2 planning problems, exceeding at most 3 actions.

6.4 Impact of using Landmarks

We now show averages regarding the amount of extracted landmarks (*i.e*, facts and actions) for all selected planning domains. The objective of this section is to convey how important the amount of extracted landmarks can be for our recognition approaches. As we mentioned before, we use landmarks as way-points during the process of recognizing plans and detecting plan abandonment. Thus, as many landmarks we can extract, they may improve the accuracy of our recognition approaches. Table 6.2 shows averages for fact and action landmarks extracted for all planning domains. Recall that we extract just conjunctive landmarks.

From extracted fact landmarks, we classify landmarks according to a set of partitions: *Strictly Activating*, *Unstable Activating*, and *Strictly Terminal*, as we presented in Section 3.3. Table 6.3 shows the occurrence of such partitions in the selected planning domains. Some planning domains do not have any partitions, such as Blocks-World, Driver-Log, Logistics, among others.

Domain	**Fact Landmarks**	**Action Landmarks**
BLOCKS-WORLD (50)	16.6	7.9
DEPOTS (15)	51.3	26.5
DRIVER-LOG (20)	13.7	6.1
DOCK-WORKER-ROBOTS (20)	39.4	19.6
EASY-IPC-GRID (50)	19.4	9.8
FERRY (30)	17.5	8.4
INTRUSION-DETECTION (50)	16.8	10.2
LOGISTICS (50)	12.8	8.0
MICONIC (30)	15.5	9.4
ROVERS (20)	20.1	8.9
SATELLITE (20)	18.1	5.4
ZENO-TRAVEL (15)	11.1	5.6

Table 6.2 – Averages of extracted landmarks (*i.e*, facts and actions) from the set of planning domains.

Domain	***Strictly Activating***	***Unstable Activating***	***Strictly Terminal***
DEPOTS	✓	✗	✗
EASY-IPC-GRID	✓	✓	✗
INTRUSION-DETECTION	✗	✗	✓
MICONIC	✗	✗	✓
ROVERS	✗	✓	✓
SATELLITE	✗	✗	✓

Table 6.3 – Planning domains that contain Fact Landmarks as Fact Partitions. The symbol ✓ denotes occurrence, and the symbol ✗ denotes absence. The other planning domains we use do not contain any Fact Partitions.

6.5 Plan Recognition Experimental Results

We now present plan recognition experimental results by comparing our approach to two other approaches: the approach of Ramírez and Geffner [RG09]; as well as a combination of their approach and our filter. We emphasize that we selected the plan recognition approach from Ramírez and Geffner that yields best results regarding both recognition time and accuracy. Thus, we present a comparison of two plan recognition approaches over four domains and hundreds of problems. These domains contain a domain description as well as an initial state, a set of candidate goals $\mathcal{G}$, a hidden goal G in $\mathcal{G}$, and an observation sequence O. An observation sequence contains actions that represent an optimal plan for a hidden goal G, and this observation sequence can be full or partial. A full observation sequence represents the whole plan for a hidden goal G, *i.e*, 100% of the actions having been observed. A partial observation sequence represents a plan for a hidden goal G with 10%, 30%, 50%, or 70% of its actions having been observed.

For evaluation, we use the accuracy metric, which represents how well a hidden goal is recognized from a set of possible goals for a given plan recognition problem; as well as recognition time, which represents how long it takes for a hidden goal to be recognized given a plan recognition problem. In the BLOCKS-WORLD domain, the accuracy metric measures how well these approaches recognize, from observations, the word that is being assembled. For the EASY-IPC-GRID domain, how accurate these approaches recognize the cell where keys are being to transported by the observed agent. With regard to INTRUSION-DETECTION domain, how accurate these approaches recognize

				h_{prl}		R&G		R&G + Filter	
Domain	$\|\mathcal{G}\|$	**% Obs**	$\|O\|$	**Time**	**Accuracy**	**Time**	**Accuracy**	**Time**	**Accuracy**
Blocks-World (780)	20	10	1.1	0.192	36.1%	1.656	83.8%	0.452	52.7%
		30	2.9	0.218	54.4%	1.735	90.0%	0.458	77.7%
		50	4.2	0.238	63.8%	1.836	97.2%	0.462	94.4%
		70	6.5	0.239	81.6%	2.056	98.8%	0.483	96.1%
		100	8.5	0.262	100.0%	2.378	100.0%	0.494	100.0%
Easy-IPC-Grid (390)	7.5	10	1.8	0.670	82.2%	1.206	97.7%	0.770	87.7%
		30	4.3	0.705	86.6%	1.291	98.8%	0.790	90.0%
		50	6.9	0.763	94.4%	1.306	98.8%	0.860	88.8%
		70	9.8	0.797	95.5%	1.715	100.0%	0.932	93.3%
		100	13.3	0.832	100.0%	2.263	100.0%	1.091	97.7%
Intrusion-Detection (390)	15	10	1.9	0.336	61.1%	1.130	98.8%	0.506	85.5%
		30	4.5	0.360	76.6%	1.142	100.0%	0.521	92.0%
		50	6.7	0.376	97.7%	1.203	100.0%	0.531	94.2%
		70	9.5	0.397	100.0%	1.482	100.0%	0.568	100.0%
		100	13.1	0.412	100.0%	1.567	100.0%	0.566	100.0%
Logistics (390)	10	10	2	0.524	66.6%	1.125	100.0%	0.615	84.2%
		30	5.9	0.553	81.1%	1.195	100.0%	0.663	87.7%
		50	9.5	0.581	83.3%	1.248	98.8%	0.712	92.2%
		70	13.4	0.634	92.2%	1.507	100.0%	0.786	98.8%
		100	18.7	0.672	100.0%	1.984	100.0%	0.918	100.0%

Table 6.4 – Comparison and experimental results of our landmark-based approach h_{prl} against Ramirez and Geffner [RG09]. R&G denotes their plan recognition approach and R&G + Filter denotes the same approach but using our filtering method.

the type of attack and servers that are being hacked observations. For Logistics domain, how accurate these approaches recognize the location where the packages are being transported from observations.

Table 6.4 compares the results for the three plan recognition approaches, showing the total number of plan recognition problems used under each domain name. For each approach we show the number of candidate goals $|\mathcal{G}|$, the percentage of the plan that is actually observed, the average number of observed actions per problem $|O|$, and, for each approach, the time to recognize the hidden goal, given the observations, and the accuracy with which the approaches correctly infer the goal. Each row of this table expresses averages about the set of candidate goals $|\mathcal{G}|$, observability (% Obs), the number of observed actions $|O|$, recognition time, and accuracy. From this table, it is possible to see that our plan recognition approach is faster than Ramírez and Geffner [RG09], and, even their approach combined with our filter affords a bigger speedup, as shown in Figures 6.1, 6.2, 6.3, and 6.4. Nevertheless, our approach yields poorer accuracy results when dealing with partial observations, primarily for 10% and 30% observability. Indeed, dealing with partial observations is currently a limitation of our plan recognition approach because both filter and heuristic work over inferred information (*i.e*, fact landmarks) from observed actions.

Table 6.5 shows the speedup and the accuracy ratio between our approach and Ramírez and Geffner's [RG09] approach. The quality of recognizing plans of our approach is measured by accuracy ratio, which is defined as the ratio between our approach and Ramírez and Geffner's approach. Thus, the closer the accuracy ratio is to 1, then, the greater agreement of accuracy for recognizing plans exists between our approach and Ramírez and Geffner's approach. Although the accuracy ratio is low for very few observability (10% and 30%), a substantial gain in recognition time is shown in the speedup column. Note that we lose accuracy only when our approach deals with very few observations, but for substantial speedups.

Domain	% Obs	Speedup	Accuracy Ratio
Blocks-World	10	8.625	0.430
	30	7.958	0.604
	50	7.771	0.656
	70	8.602	0.825
	100	9.076	1
Average		**8.408**	**0.714**
Easy-IPC-Grid	10	1.800	0.841
	30	1.831	0.876
	50	1.711	0.955
	70	2.151	0.955
	100	2.719	1
Average		**2.065**	**0.926**
Intrusion-Detection	10	3.363	0.618
	30	3.172	0.766
	50	3.199	0.977
	70	3.732	1
	100	3.803	1
Average		**3.468**	**0.872**
Logistics	10	2.146	0.666
	30	2.160	0.811
	50	2.148	0.843
	70	2.376	0.922
	100	2.952	1
Average		**2.381**	**0.848**

Table 6.5 – Speedup and Accuracy Ratio between our landmark-based heuristic h_{prl} and Ramirez and Geffner [RG09] approach.

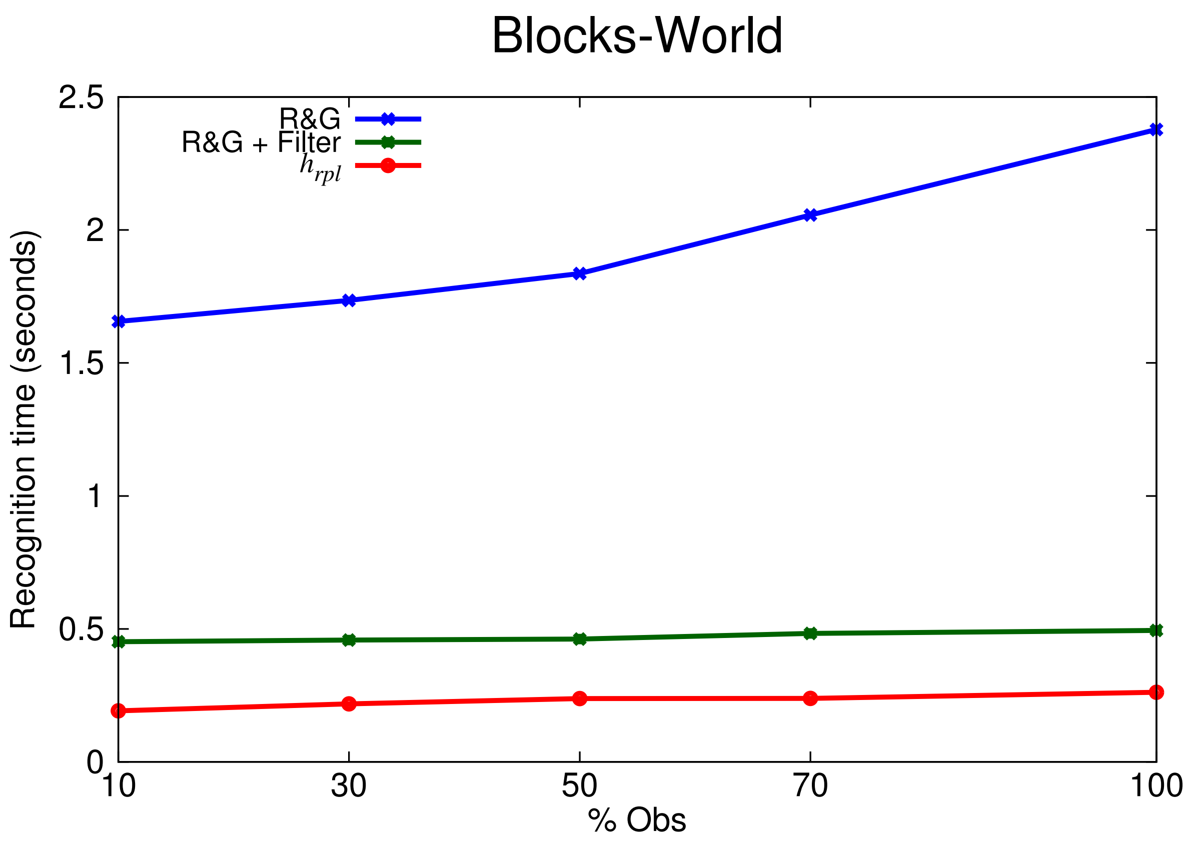

Figure 6.1 – Blocks-World – Comparison of Plan Recognition Time.

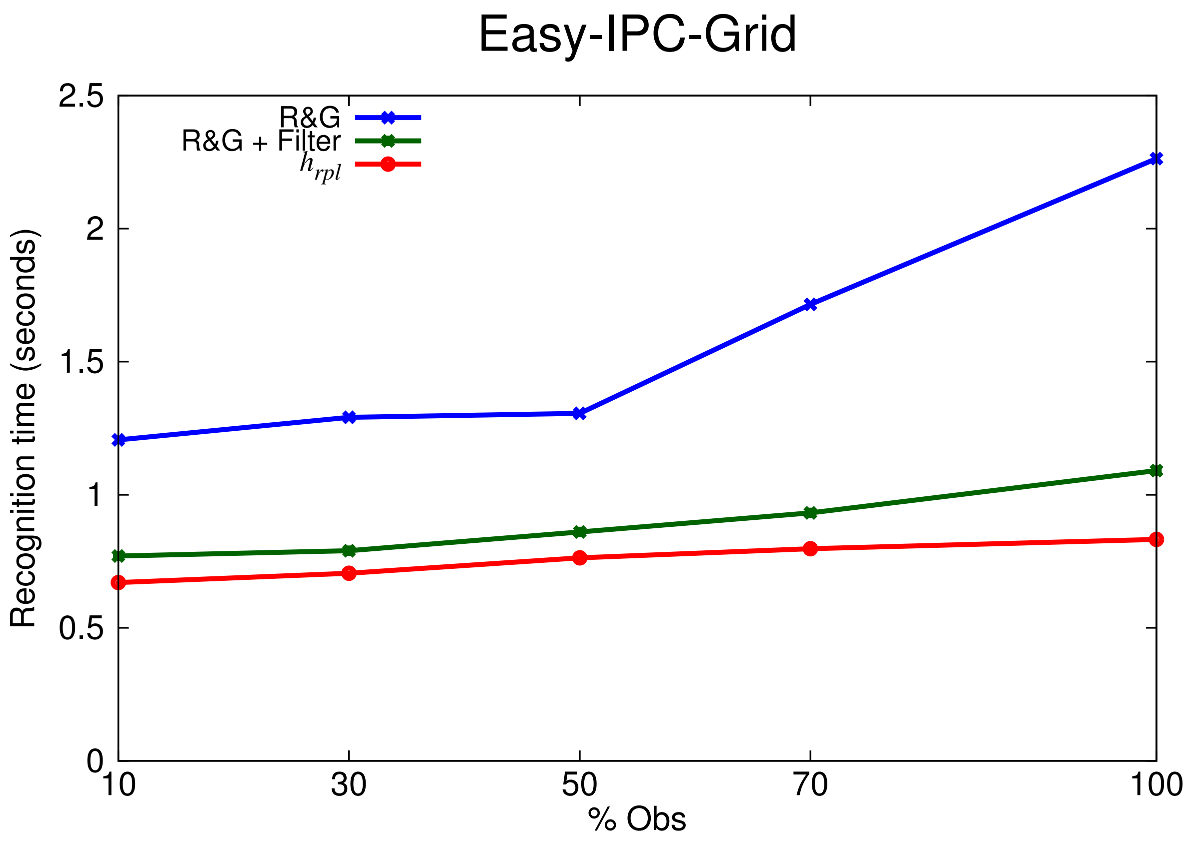

Figure 6.2 – EASY-IPC-GRID – Comparison of Plan Recognition Time.

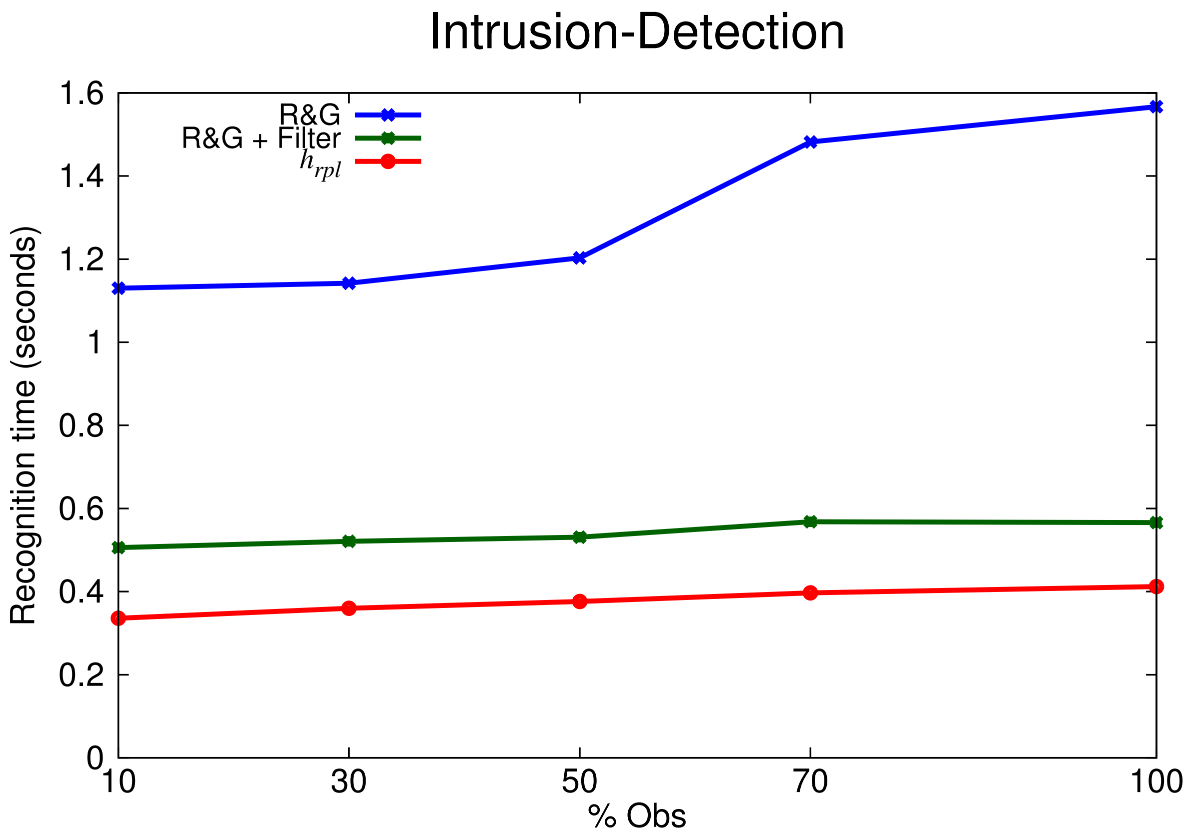

Figure 6.3 – INTRUSION-DETECTION – Comparison of Plan Recognition Time.

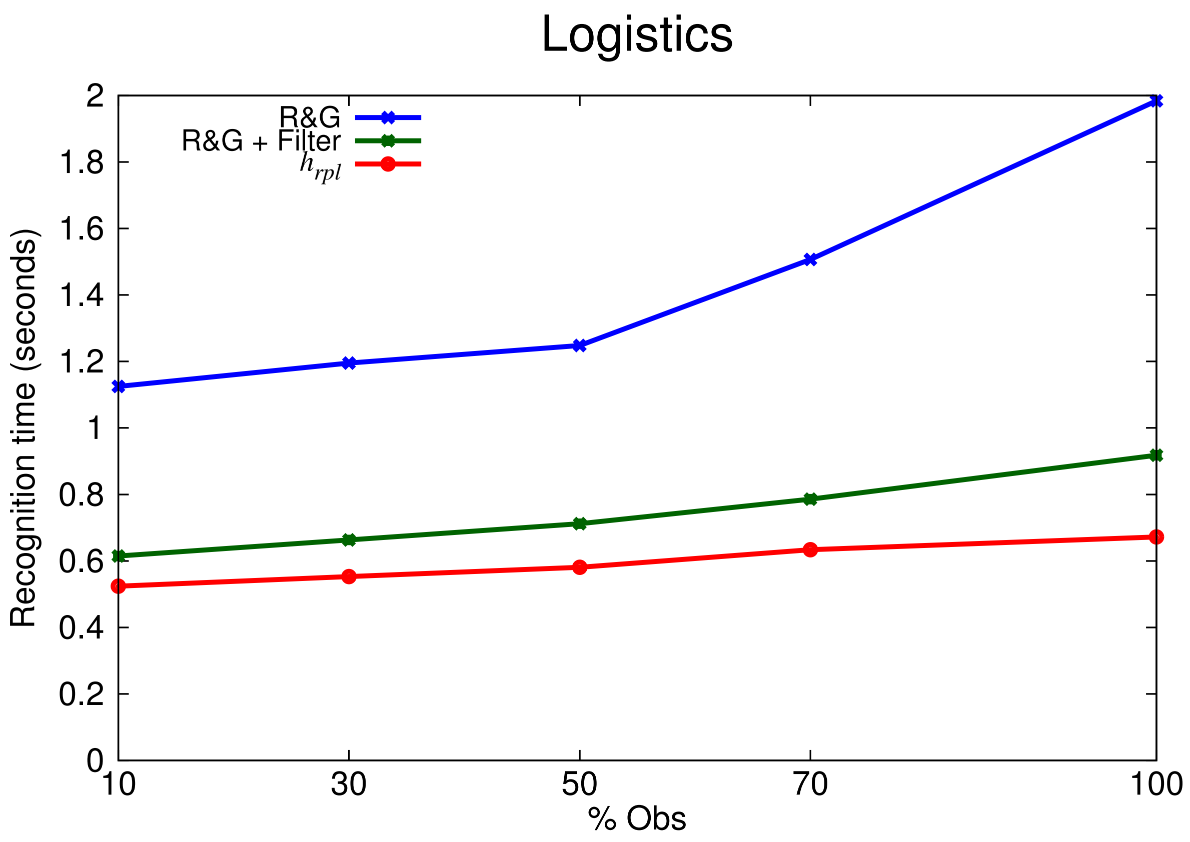

Figure 6.4 – LOGISTICS – Comparison of Plan Recognition Time.

6.6 Plan Abandonment Detection Experimental Results

We now present the experiments and evaluation of our approach for detecting plan abandonment. For experiments we use 11 planning domains, and for these domains we select 15-30 associated non-trivial problem instances. Each problem instance is also associated to a set of observations (*i.e*, plan execution). This set of observations can represent an optimal or a sub-optimal plan execution. We generate plans (optimal and sub-optimal) using open-source planners, such as BLACKBOX[3] and FF[4]. Moreover, generating plans for some problem instances is time-consuming. For this reason, we have evaluated our approach using 15-30 problem instances with associated plans. Thus, these experiments aim to evaluate how accurate our approach detects actions that do not contribute to achieve a correspond monitored goal.

Table 6.6 shows the experimental results of our approach over the selected domains. Each row of this table expresses a set of experiments over a domain by representing averages about the observations $|O|$ (*i.e*, the number of actions in a plan execution), monitoring time, and metrics (precision, recall, and the F1-score). From this table, it is possible to see that for most domains our approach yields good results regarding correct positive predictions (precision), positive cases (recall), and monitoring time. As shown in column $|O|$, we analyze non-trivial plans, in which a number

[3]https://goo.gl/qbbJRu
[4]https://goo.gl/mJzUs4

Domain	$\lvert O\rvert$	Time	Precision	Recall	F1-score
Blocks-World (30)	13.9	1.05	87.5%	80.0%	83.5%
Depots (15)	29.6	5.11	76.0%	100.0%	86.3%
Driver-Log (20)	20.1	1.57	61.5%	88.8%	72.6%
Dock-Worker-Robots (20)	33.0	5.19	77.2%	70.8%	73.8%
Easy-IPC-Grid (30)	14.1	1.16	100.0%	83.3%	90.7%
Ferry (30)	13.8	0.74	80.0%	71.4%	75.4%
Logistics (30)	20.8	1.83	75.8%	91.6%	82.9%
Miconic (30)	18.0	0.91	90.9%	83.3%	86.9%
Rovers (20)	24.5	4.72	52.1%	75.0%	61.4%
Satellite (20)	25.7	3.78	68.7%	73.3%	70.9%
Zeno-Travel (15)	11.4	1.63	77.7%	100%	87.4%

Table 6.6 – Plan Abandonment Detecting experimental results.

of actions differ between 11.4 and 33. Apart from the Rovers domains, that under-performs for correct positive predictions, some domains show near-perfect precision, such as Miconic and Easy-IPC-Grid. Regarding recall, that measures how accurate our approach detects actions that do not contribute to the monitored goal, our approach yields better results, by yielding at least 70.8% for all domains. The accuracy-rate (F1-score) for all domains is over 70.9%. This table shows that we obtain good results for detecting plan abandonment observing in deterministic planning domains.

6.7 Chapter Remarks

In this chapter we have shown experimentally that our recognition approaches yield high accuracy at low computational cost for experiments over several complex planning domains. We also compare our plan recognition approach against the state-of-the-art planning-based plan recognition approach, in which our approach loses in accuracy when deals with very few observability (10% and 30%). Our plan abandonment detection approach yields good results in detecting plan abandonment by dealing with realistic well-known deterministic planning domains. To refine experiments and evaluation over our plan abandonment detection approach, we plan to deal with partial plan execution against more planning domains and problems.

7. RELATED WORK

In this chapter we survey some of the most important work concerning goal/plan recognition and plan abandonment detection over the past years. In Section 7.1 we review some works on plan recognition, by surveying approaches based on both plan libraries and planning techniques. In Section 7.2 we survey the literature by describing the only approach that deals with the problem of detecting plan abandonment. Finally, in Section 7.3 we situate differences and similarities of our recognition approaches compared to the presented related work.

7.1 Goal and Plan Recognition

Most work on plan recognition are based on plan libraries, where the space of possible plans for achieving goals is static and must be encoded beforehand for every agent. In this way, we now survey both single and multi-agent approaches based on plan libraries for recognizing goals and plans through the years. In [GG05], Geib proposes a probabilistic single-agent approach to plan recognition that selects candidate goals from a plan library with the higher probability based on a set of observations at each observation time. Namely, this approach selects those plans that are a priori more likely than others given a set of observations. Avrahami-Zilberbrand and Kaminka propose a fast and complete single-agent plan recognition approach [AK05], in which it generates a Feature Decision Tree (FDT) to efficiently match a set of observations with a plan library, and then it provides an implicit representation of the goal and plan hypotheses. In [Avr09][SGG$^+$14, Chapter 4 – page 87], Avrahami-Zilberbrand proposes a hybrid single-agent keyhole plan recognition that uses a symbolic approach to eliminate inconsistent hypothesis by matching a set of observations with a plan library, and from this, this approach uses a probabilistic approach that attempts to recognize the most likely goal/plan hypothesis. In [ZL11], Zhuo *et al.* propose a multi-agent plan recognition approach that uses partial plan libraries for representing goals and plans for a group of people (i.e, team). In this work, their multi-agent plan recognition approach deals with partial team observation trace, and given a partial team observation trace, this approach builds a set of soft/hard constraints, takes as input into a weighted MAX-SAT solver, and then recognizes the team plan that achieves their goal(s). In [BK11], Banerjee and Kraemer propose a multi-agent plan recognition approach that can deal with interleaving plans. This approach uses plan libraries to represent team plans, and the process of recognizing plans is based on the method of branch and price, which matches the team observation trace with the team plan library.

Regarding goal and plan recognition without plan libraries, we now review some single and multi-agent approaches based on planning techniques. As the first planning-based approach for recognizing goals and plans, Hong [Hon01] extends the concept of planning graph [BF97], proposing a similar structure that represents every possible path (e.g, state transitions that connect facts and actions) from an initial state to a goal state, by calling this structure as goal graph. Basically, as actions are observed during a plan execution, a goal graph is constructed, in which facts that represent recognized goals are linked into a goal level. Later, in [RG09], Ramirez and Geffner propose the use of heuristic estimation to eliminate goals from a candidate set, due to an increasing heuristic distance from the current state. In this work the authors present two approaches, they consider optimal or sub-optimal plans, in which goals that have become impossible being removed from the candidate set. Once candidate goals have been eliminated, they are never reconsidered. Moreover, Ramirez and Geffner work with an assumption of partial observability, it means that only a sub-sequence of the plan is observed. Follow-up work, in [RG10], Ramirez and Geffner propose a probabilistic plan recognition approach by using off-the-shelf planners. In [PL10] Pattison and Long propose AUTOGRAPH (AUTOmatic Goal Recognition with A Planning Heuristic), a probabilistic heuristic-based goal recognition over planning domains. AUTOGRAPH makes use of heuristic estimation and domain analysis to determine which goals a plan execution of an observed agent is pursuing. As multi-agent plan recognition using planning domain definition, in [ZYK], Zhuo *et al.* present an approach in which the team behavior model is defined as a planning domain definition (i.e, every agent behavior is based on the planning domain definition), and the task of plan recognition analyzes a partial team observation trace for recognizing team plans. In this work the authors also show a comparative between their previous plan library based approach and the planning domain definition based. Most recently, in [KGK14], Keren *et al.* present an alternate view regarding the goal and plan recognition problem. This work uses planning techniques to assist in the design of goal and plan recognition problems. Instead of generating plan libraries for a given planning problem, such as Blaylock and Allen in [BA05], Keren *et al.* attempt to reduce the number of non-unique plans for each candidate goal in a set of candidate goals, and therefore, simplifying the process of goal recognition.

7.2 Plan Abandonment Detection

In several real-world scenarios the ability of detecting plan abandonment can be useful, such as in human assistive living, as well as providing an specific task assistance, e.g, elderly care [Gei02], in which a person receives assistance of carers or family members. Normally, in this scenario there are many strict activities to aid elderly people, where it is important to detect when the assistant fails to do something, for example, providing some medication. In task assistance, such as driver assistant systems [PW13], it is important to detect when the driver is going away from the intended destination. However, recent research focuses on activity and plan recognition [DLP11, PW13, CG13, HP13, AG13], but the task of detecting when an agent has abandoned plans has received little attention.

To the best of our knowledge, the only prior work on goal and plan abandonment detection is an explicit model of the problem proposed by Geib and Goldman [GG03]. Based on PHATT (Probabilistic Hostile Agent Task Tracker) model [GGM99], in [GG03], Geib and Goldman propose a formal model to recognize goal and plan abandonment in the plan recognition context. This formal model estimates the probability that a sequence of observed actions contributes to the goal being monitored. Furthermore, in [Gei02], Geib addresses some issues and requirements for recognizing goals in the elderly-care domain, as well as plan abandonment detection. Commonly, plan recognition approaches do not deal with plan abandonment, which means that when an agent has no intention to complete or finish a plan, these approaches will continuously attempt to recognize what the agent is doing. In this case, these approaches will pursue an increasingly large set of uncompleted plans for the observed agent. According to Geib and Goldman [GG03], the ability of detecting plan abandonment is an operational requirement for any plan recognition approach. Namely, if any plan recognition approach can deal with plan abandonment, it will probably improve the efficiency of the process of recognizing plans, by filtering out abandoned plans.

7.3 Chapter Remarks

In this chapter we have surveyed some related work to our recognition approaches. The planning-based plan recognition approaches we have reviewed, such as [Hon01, RG09, RG10, PL10, ZYK], are close related to our plan recognition approach regarding the use of planning domain definition to model agent behavior (or teams behavior). For instance, such approaches make use of planning languages, e.g, STRIPS or PDDL, as ours. However, the only planning-based plan recognition approach that we use to perform a direct comparison is Ramirez and Geffner approach [RG09]. In Chapter 6 we show experiments and evaluation that yields a direct comparison against Ramirez and Geffner approach [RG09] by using a dataset with a set of complex planning domains, as provided by themselves in [RG09]. With regard to related approaches to detect plan abandonment, the only work we have found in the literature is [GG03], a probabilistic approach for detecting plan abandonment that makes use of plan libraries to model agent behavior, and due to this feature, a direct comparison becomes difficult, because we use planning domain definition to model agent behavior. To accomplish such comparison against Geib and Goldman approach [GG03], our definition of plan abandonment detection problem must be converted to plan library, which is a problem that we do not address in this work.

8. CONCLUSION

In this work we have presented recognition approaches that use landmarks to monitor plan execution for plan recognition and plan abandonment detection. Landmarks provide key information about what cannot be avoided to achieve a goal, and therefore, we demonstrate that landmarks can be used efficiently for both plan recognition and plan abandonment detection. We also show that the problem of detecting plan abandonment can be modeled using planning domain definition. We consider that modeling agent goals and plans using planning domain definition is more advantageous than using plan libraries, because it requires less design effort and makes no assumption about the particular strategy used by the agent to generate behavior. We argue that, in many non-trivial domains, encoding a plan library that contains results for all possible goals and plans is a difficult task that may, in the worst case, be impossible or intractable.

We have shown experimentally that our plan recognition approach yields good accuracy results against, but more importantly, superior recognition time over the state-of-the-art plan recognition approach [RG09]. Nevertheless, our plan recognition approach is limited when dealing with highly incomplete observations (e.g, 10% and 30% of plan observability) or when such observations do not pass through landmarks. However, we argue whenever the actions of an observation sequence correspond to a plan that achieves a goal in the set of candidates, then there is a set of fact landmarks that this plan cannot avoid to achieve this goal. Regarding plan abandonment, we have shown in experiments that by using planning techniques (landmarks and domain-independent heuristics) it is possible to monitor and anticipate which actions in a plan execution do not contribute to achieve a monitored goal.

There are multiple avenues for future work. First, we intend to explore other planning techniques with our recognition approaches, such as heuristics and symmetries in classical planning [SKH$^+$]. Second, we plan to improve the recognition approaches we have developed for this work by exploring landmarks in a different way, such as disjunctive landmarks and temporal landmarks [KWWH15]. Third, to obtain better accuracy in plan recognition, we intend to explore goal orderings in order to know what is the ordering that a set fact of a conjunctive goal must be achieved [KH00]. This information can be used within our landmark-based plan recognition heuristic. Finally, as another potential area of study and application, we consider that besides detecting plan abandonment, our approach can also be used to repair sub-optimal plans, e.g, by detecting which parts of the plan is non-optimal, and then improve it.

BIBLIOGRAPHY

[AA07] Armentano, M. G.; Amandi, A. “Plan Recognition for Interface Agents.”, *Journal of Artificial Intelligence Research (JAIR)*, vol. 28–2, 2007, pp. 131–162.

[AG13] Amir, O.; Gal, Y. K. “Plan Recognition and Visualization in Exploratory Learning Environments”, *ACM Transactions on Interactive Intelligent Systems*, vol. 3–3, 2013, pp. 16:1–16:23.

[AK05] Avrahami-Zilberbrand, D.; Kaminka, G. A. “Fast and Complete Symbolic Plan Recognition”. In: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI), 2005, pp. 653–658.

[Avr09] Avrahami-Zilberbrand, D. “Efficient Hybrid Algorithms for Plan Recognition and Detection of Suspicious and Anomalous Behavior”, Ph.D. Thesis, Bar Ilan University, Israel, 2009.

[BA05] Blaylock, N.; Allen, J. F. “Generating Artificial Corpora for Plan Recognition”. In: Proceedings of the 10th International Conference on User Modeling (UM), Edinburgh, Scotland, UK, July 24-29, 2005, pp. 179–188.

[BF97] Blum, A. L.; Furst, M. L. “Fast Planning Through Planning Graph Analysis”, *Journal of Artificial Intelligence Research (JAIR)*, vol. 90–1-2, 1997, pp. 281–300.

[BG01] Bonet, B.; Geffner, H. “Planning as Heuristic Search”, *Journal of Artificial Intelligence Research (JAIR)*, vol. 129, 2001, pp. 5–33.

[BK07] Bryce, D.; Kambhampati, S. “A Tutorial on Planning Graph Based Reachability Heuristics”, *AI Magazine*, vol. 28–1, 2007, pp. 47–83.

[BK11] Banerjee, B.; Kraemer, L. “Branch and Price for Multi-Agent Plan Recognition”. In: Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, San Francisco, California, USA, August 7-11, 2011.

[Byl94] Bylander, T. “The Computational Complexity of Propositional STRIPS Planning”, *Journal of Artificial Intelligence Research (JAIR)*, vol. 69, 1994, pp. 165–204.

[Car01] Carberry, S. “Techniques for Plan Recognition”, *User Modeling and User-Adapted Interaction*, vol. 11–1-2, 2001, pp. 31–48.

[CG13] Charniak, E.; Goldman, R. P. “Plan Recognition in Stories and in Life”, *Computing Research Repository (CoRR)*, vol. abs/1304.1497, 2013.

[CWwH06] Chen, Y.; Wah, B. W.; wei Hsu, C. “Temporal Planning using subgoal partitioning and resolution in SGPlan”, *Journal of Artificial Intelligence Research (JAIR)*, vol. 26, 2006, pp. 369.

[DLP11] Dong, W.; Lepri, B.; Pentland, A. S. “Modeling the Co-evolution of Behaviors and Social Relationships Using Mobile Phone Data”. In: Proceedings of the 10th International Conference on Mobile and Ubiquitous Multimedia, 2011, pp. 134–143.

[ENS95] Erol, K.; Nau, D. S.; Subrahmanian, V. S. “Complexity, Decidability and Undecidability Results for Domain-Independent Planning.”, *Artificial Intelligence*, vol. 76–1-2, 1995, pp. 75–88.

[FL11] Fox, M.; Long, D. “The 3rd International Planning Competition (IPC-3): Results and Analysis”, *Computing Research Repository (CoRR)*, vol. abs/1106.5998, 2011.

[FN71] Fikes, R. E.; Nilsson, N. J. “STRIPS: A new approach to the application of theorem proving to problem solving”, *Journal of Artificial Intelligence Research (JAIR)*, vol. 2–3, 1971, pp. 189–208.

[Gei02] Geib, C. W. “Problems with Intent Recognition for Elder Care”. In: Proceedings of the Eighteenth AAAI Conference on Artificial Intelligence, 2002, pp. 13–17.

[GG03] Geib, C. W.; Goldman, R. P. “Recognizing Plan/Goal Abandonment”. In: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI), 2003, pp. 1515–1517.

[GG05] Geib, C. W.; Goldman, R. P. “Partial Observability and Probabilistic Plan/Goal Recognition”. In: Proceedings of the International Workshop on Modeling Others from Observations (MOO-2005), 2005.

[GGM99] Goldman, R. P.; Geib, C. W.; Miller, C. A. “A New Model of Plan Recognition”, *Computing Research Repository (CoRR)*, vol. abs/1301.6700, 1999.

[GNT04] Ghallab, M.; Nau, D. S.; Traverso, P. “Automated Planning - Theory and Practice.” Elsevier, 2004.

[Hel06] Helmert, M. “The Fast Downward Planning System”, *Journal of Artificial Intelligence Research (JAIR)*, vol. 26, 2006, pp. 191–246.

[Hel11] Helmert, M. “The Fast Downward Planning System”, *Computing Research Repository (CoRR)*, vol. abs/1109.6051, 2011.

[HN01] Hoffmann, J.; Nebel, B. “The FF Planning System: Fast Plan Generation Through Heuristic Search”, *Journal of Artificial Intelligence Research (JAIR)*, vol. 14–1, May 2001, pp. 253–302.

[Hof11] Hoffmann, J. “The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables”, *Computing Research Repository (CoRR)*, vol. abs/1106.5271, 2011.

[Hon01] Hong, J. “Goal Recognition through Goal Graph Analysis”, *Journal of Artificial Intelligence Research (JAIR)*, vol. 15, 2001, pp. 1–30.

[HP13] Han, T. A.; Pereira, L. M. “State-of-the-art of Intention Recognition and Its Use in Decision Making”, *AI Commun.*, vol. 26–2, 2013, pp. 237–246.

[HPS04] Hoffmann, J.; Porteous, J.; Sebastia, L. “Ordered Landmarks in Planning”, *Journal of Artificial Intelligence Research (JAIR)*, vol. 22–1, 2004, pp. 215–278.

[J. 01] J. Porteous, L. Sebastia, J. H. “On the Extraction, Ordering and Usage of Landmarks in Planning”. In: European Conference of Planning (ECP’01), 2001, pp. 37–48.

[KGK14] Keren, S.; Gal, A.; Karpas, E. “Goal Recognition Design”. In: Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling (ICAPS), Portsmouth, New Hampshire, USA, June 21-26, 2014.

[KH00] Koehler, J.; Hoffmann, J. “On Reasonable and Forced Goal Orderings and their Use in an Agenda-Driven Planning Algorithm”, *Journal of Artificial Intelligence Research (JAIR)*, vol. 12, 2000, pp. 338–386.

[KS98] Kautz, H.; Selman, B. “BLACKBOX: A New Approach to the Application of Theorem Proving to Problem Solving”. In: Proceedings of Planning as Combinatorial Search (AIPS-98), Pittsburgh, Pennsylvania, USA, AAAI Press, 1998, pp. 58–60.

[KWWH15] Karpas, E.; Wang, D.; Williams, B. C.; Haslum, P. “Temporal Landmarks: What Must Happen, and When”. In: Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling (ICAPS), Jerusalem, Israel, June 7-11, 2015, pp. 138–146.

[MGH$^+$98] McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; Wilkins, D. “PDDL – The Planning Domain Definition Language”, *Technical Report – Yale Center for Computational Vision and Control*, 1998.

[PL10] Pattison, D.; Long, D. “Domain Independent Goal Recognition.” In: Proceedings of the Fifth European Starting AI Researcher Symposium (STAIRS), Ågotnes, T. (Editor), 2010, pp. 238–250.

[PW13] Pynadath, D. V.; Wellman, M. P. “Accounting for Context in Plan Recognition, with Application to Traffic Monitoring”, *Computing Research Repository (CoRR)*, vol. abs/1302.4980, 2013.

[RG09] Ramírez, M.; Geffner, H. “Plan Recognition as Planning.” In: Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI), Boutilier, C. (Editor), 2009, pp. 1778–1783.

[RG10] Ramírez, M.; Geffner, H. "Probabilistic Plan Recognition Using Off-the-Shelf Classical Planners". In: Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, Atlanta, Georgia, USA, July 11-15, 2010.

[RW10] Richter, S.; Westphal, M. "The LAMA Planner: Guiding Cost-based Anytime Planning with Landmarks", *Journal of Artificial Intelligence Research (JAIR)*, vol. 39–1, 2010, pp. 127–177.

[SGG$^+$14] Sukthankar, G.; Goldman, R. P.; Geib, C.; Pynadath, D. V.; Bui, H. H. "Plan, Activity, and Intent Recognition: Theory and Practice" . Elsevier, 2014.

[SKH$^+$] Shleyfman, A.; Katz, M.; Helmert, M.; Sievers, S.; Wehrle, M. "Heuristics and Symmetries in Classical Planning". In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA, pp. 3371–3377.

[SSG78] Schmidt, C. F.; Sridharan, N. S.; Goodson, J. L. "The Plan Recognition Problem: An Intersection of Psychology and Artificial Intelligence.", *Journal of Artificial Intelligence Research (JAIR)*, vol. 11–1-2, 1978, pp. 45–83.

[ZL11] Zhuo, H. H.; Li, L. "Multi-Agent Plan Recognition with Partial Team Traces and Plan Libraries". In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI), Barcelona, Catalonia, Spain, July 16-22, 2011, pp. 484–489.

[ZYK] Zhuo, H. H.; Yang, Q.; Kambhampati, S. "Action-Model Based Multi-agent Plan Recognition". In: Proceedings of the 26th Annual Conference on Neural Information Processing Systems, December 3-6, 2012, Lake Tahoe, Nevada, USA, pp. 377–385.