UNIVERSIDAD DE ALCALÁ
DEPARTAMENTO DE AUTOMÁTICA
PROGRAMA DE DOCTORADO EN INVESTIGACIÓN ESPACIAL

IMPROVING COST AND PROBABILITY ESTIMATES USING INTERACTION

Dissertation written by
Yolanda Escudero Martín

Under the supervision of
Dr. María Dolores Rodríguez Moreno

Dissertation submitted to the School of Computing of
the Universidad de Alcalá, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Improving Cost and Probability Estimates using Interaction
2016

To Casimiro, María del Carmen, and Marieta

Acknowledgements

This thesis would not exist without the help and support of many people. Among them Luis Usero who always encouraged me to keep going beyond what I believed I was capable of. Bonifacio Castaño who for some reason always trusted me without even knowing me. Last and most importantly, my advisor María Dolores Rodríguez Moreno. Malola, I wish I had the words to express how important it was for me to meet you. I would not have considered a research career without your motivation and encouragement. Your generosity got me where I am today, and let me be part of your GSI family where I met those who became my dear colleagues including David Fernández and Pablo Muñoz who were always there to help me. I cannot thank you enough for letting me enjoy this time.

During my time in Mountain View I had a wonderful experience thanks to the Planning and Scheduling Group at NASA Ames Research Center. A special mention to Jeremy Frank for making it possible in the first place, Robert Morris for all his support, encouragement, and friendship, Paul Morris and John Bresina for their kindness, and Christian Plaunt for his kindly help. This experience would have been completely different without my little American family Carolina Padilla, Guillaume Brat, Jaqueline Campos, Jason Weber, Jim Murdoch, Jocelyn Blackwood, Joyce Kellner, Kristoffer Svarre, Maya Peterson, and Nidhi Seth, who welcomed me with open arms and made me feel at home.

I thank my dear friends in Madrid: Ana Martínez, Carmen Somolinos, Luis Sanz, Patricia Álvarez, Paula Benito, Raquel de Sande, and Tamara Fernández for their understanding and encouragement during these years.

A big thanks goes to my parents Casimiro and María del Carmen. Papá, mamá, soy lo que soy gracias a vuestro amor, comprensión, cariño y apoyo. Siempre habéis estado ahí, siempre estáis y siempre estaréis. Gracias por vuestra gratitud ilimitada y sobre todo gracias por creer en mí.

I would also like to specially thank my sister Marieta whose fairy tale letters made this time magical; I will never get tired of reading them. Thanks to my lovely nieces Jimena and María for their beautiful drawings that always made me smile.

Above all, my deepest gratitude to David E. Smith. You made this possible with so much generosity, caring, kindness, and advice. Your patience$^{\infty}$ in guiding my research, correcting and understanding my writing and speaking, and answering my millions of questions helped me to grow enormously as a person and professional. You bent over backwards to extend yourself, were always available, and I can never hope to repay you.

Abstract

Automated Planning is concerned with finding a program of actions that given an initial state reaches a desired goal. This program can be a simple sequence or a more complex partially ordered collection of actions. When action outcomes are deterministic and information is perfect, it is called deterministic planning. Significant advances have been made in recent years in the size and complexity of deterministic problems that can be solved with these planners. However, in real problems, unexpected events may occur, actions may have unexpected outcomes, and the state of the world may not be known with certainty. If a deterministic planner is used to solve such problems, the execution of the plan may fail because the plan does not take into account the possible contingencies. Traditional approaches to planning under uncertainty involve Markov Decision Processes, which generates robust plans, but have high computational overhead. Other approaches to planning under uncertainty use contingency planning, translation into deterministic planning, or determinization and replanning.

In recent years, planning-based solutions have been also used to solve goal recognition problems. Goal recognition may be seen as the inverse of planning since it is concerned with inferring an agent's goals given some or all agent's performed actions. There are few planning-based approaches for goal recognition, but this paradigm is still in its infancy.

The previous paradigms all have one thing in common: the dominant techniques to solve their problems involve heuristic functions to guide the planning search. For this reason, in this thesis, we investigate classical heuristics that deal with action costs and introduce a novel domain-independent heuristic function that computes more accurate estimates of cost and estimates of probability. The approach involves a propagation of cost or probability plus Interaction information through a plan graph. This heuristic guides a classical planner to low-cost solutions, guides a probabilistic planner to high probability of success solutions, and rapidly solves goal recognition problems.

Resumen Ampliado

La planificación automática consiste en producir una colección de acciones o plan que lleven a un agente desde un estado inicial a un objetivo. Esta colección puede ser una secuencia simple o una secuencia más compleja parcialmente ordenada de acciones. Se denomina planificación clásica cuando se cuenta con información completa del problema y las acciones son deterministas. Durante los últimos años se han conseguido significativos avances en la planificación automática, siendo capaz de resolver problemas de considerable tamaño y complejidad. Sin embargo, este enfoque no es efectivo a la hora de resolver problemas reales ya que en este tipo de problemas pueden suceder eventos inesperados, la respuesta tras realizar una acción no se puede predecir y como consecuencia el estado actual del mundo no se conoce con certeza. Por lo tanto, la ejecución de un plan generado por un planificador clásico ante un problema de la vida real podría fallar al no tener en cuenta dichas contingencias. Cuando se cuenta con información incompleta del problema y/o las acciones no son deterministas se denomina planificación bajo incertidumbre. Tradicionalmente, estos enfoques hacen uso de Procesos de Markov para generar planes robustos, pero de alta sobrecarga computacional. Otros enfoques de planificación bajo incertidumbre hacen uso de planificación de contingencias, traducción del problema con incertidumbre a un problema determinista o *determinization* y replanificación para resolver problemas.

En los últimos años, la planificación automática se ha sumado al área de estudio del reconocimiento de metas, el cual se puede interpretar como la operación inversa a la planificación ya que tiene como objetivo inferir la(s) meta(s) de un agente tras observar parcial o completamente las acciones llevadas a cabo por el mismo. Recientemente, se han aplicado técnicas de planificación para resolver problemas de reconocimiento de metas, pero este enfoque está todavía en sus comienzos.

Problemas de planificación clásica, de planificación bajo incertidumbre y de reconocimiento de metas se pueden resolver mediante búsqueda heurística, una de las técnicas que más éxito ha tenido resolviendo estos problemas. Las funciones heurísticas más comunes en planificación automática calculan estimaciones de distancia en forma de coste o probabilidad de alcanzar el estado meta desde un estado actual particular. Se denominan heurísticas admisibles a aquellas que guían la búsqueda hacia soluciones óptimas. Es decir, que minimizan el coste o maximizan la probabilidad. Estas heurísticas, a pesar de producir una solución óptima, pueden no ser suficientemente informativas o ser de alto coste computacional. Por otro lado, se denominan heurísticas no admisibles a aquellas que generan soluciones subóptimas. Estas heurísticas pueden o no producir la solución óptima, pero son más informativas que las heurísticas admisibles y han demostrado tener un buen rendimiento en cuanto a tiempo y calidad de la solución. En esta tesis, se investiga sobre aquellas heurísticas en el estado del arte que consideran acciones con coste o acciones probabilísticas para calcular estimaciones de coste y estimaciones de probabilidad más precisas.

Para mejorar la precisión de las estimaciones de coste, se desarrolla una función heurística que lleva a cabo propagación de costes en un grafo de planificación. Estas estimaciones son más exactas gracias al uso de *Interaction*, término que permite calcular la relación de independencia, de sinergia o de exclusión mutua entre pares de elementos. Estas estimaciones de coste se utilizan para (1) guiar a un planificador clásico hacia soluciones que minimizan el coste, (2) guiar a un planificador probabilístico hacia soluciones que maximizan la probabilidad y (3) resolver eficientemente problemas de reconocimiento de metas.

Para mejorar la precisión de las estimaciones de probabilidad, se desarrolla un novedoso enfoque que lleva a cabo propagación de probabilidades en un grafo de planificación. Esta propagación de probabilidades es más avanzada que la previa ya que considera (1) la probabilidad global de cada proposición entre los posibles efectos de cada acción probabilística y (2) la dependencia de pares de proposiciones entre los posibles efectos de una acción probabilística. La unión de ambas técnicas permite calcular estimaciones de probabilidad más exactas y así generar soluciones de alta probabilidad de éxito.

Como resultado de este estudio se obtiene una familia de heurísticas que calculan aproximaciones de coste y aproximaciones de probabilidad más exactas y constantes que otras heurísticas del estado del arte.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter presents an overview of the theoretical framework needed to understand the motivation of this dissertation. It starts with an introduction of the research area and presents the motivation of this work. Next, it presents the main contributions and describes the structure and contents of this dissertation.

1.1 Motivation

Automated Planning is the problem of finding a program of actions that given an initial state reaches a desired goal. This program can be a simple sequence or a more complex partially ordered collection of actions. A classical planning problem may be represented as a state-transition graph whose nodes represent different states, and whose arcs represent operators, which designate the transition from a state s to a state s' (Ghallab et al., 2004). A domain-independent planner searches for a path in the space of states that connects the initial state to a goal state.

Planning approaches can be divided into two categories based on the solution they seek: optimal planning, in which plans are guarantee to be optimal, and satisficing planning, in which plans are not guarantee to be optimal. There are a number of techniques to solve classical planning problems, but heuristic search is the dominant technique in classical planning. In order to guide the search for a solution, it makes use of a heuristic function to estimate the cost of reaching a goal state from a particular current state. The STRIPS heuristic (Fikes and Nilsson, 1971) is one of the first heuristic functions introduced in planning. Modern heuristics use delete relaxation to consider a less constrained version of the planning problem. McDermott (1996) and Bonet (1997) are examples of delete relaxation heuristics, which improved significantly the scalability of domain-independent planners. Current state-of-the-art approaches continue using this technique in both optimal (Haslum and Geffner, 2000; Bonet and Geffner, 2001; Helmert and Domshlak, 2009) and satisficing (Hoffmann and Nebel, 2001; Keyder and Geffner, 2007; Richter and Westphal, 2010) planning. Improvements in the performance of heuristic search come from two factors: improvements in search algorithms and improvements in heuristic functions. This thesis focuses on the later to seek a domain-independent heuristic that quickly computes estimates of cost and estimates of probability, and improves the accuracy of those estimates to (1) guide the search towards low-cost plans in classical planning, (2) guide the search towards high probability of success plans in probabilistic planning, and (3) rapidly solve goal recognition problems.

1.2 Outline of contributions

The contributions of this thesis can be summarized as follows:

1. The h^I heuristic: a more informative and stable heuristic that steps further from the independent assumption by considering the cost and Interaction information computed in a plan graph, obtaining closer approximations to the optimal cost. Additionally, we propose three additive alternatives to the admissible heuristic h^2 (Haslum and Geffner, 2000): h_g^{2+}, h_c^{2+}, and h_m^{2+}. We evaluate and demonstrate in various contexts the difference between the h^2 and the h^I heuristics, and perform an accuracy evaluation of the state-of-the-art heuristics (in Chapter 3).

2. The *Fast Goal Recognition* (FGR) approach: a goal recognition approach that uses the h^I heuristic to quickly infer probability estimates for the possible goals for a goal recognition problem with certain observations (in Chapter 4).

3. The characterization of the ISS Crew Activities Domain: a real time goal recognition problem concerned with maintenance tasks that astronauts must conduct for the Envriomental Control and Life Support System aboard the International Space Station. We demonstrate that FGR is practical for this real time goal recognition problem (in Chapter 4).

4. The *uncertain Fast Goal Recognition* (uFGR) framework: a goal recognition solution that combines a Bayesian network and a plan graph to solve goal recognition problems with uncertain observations (in Chapter 4).

5. The use of the cost and Interaction information to make better choices for actions in computing relaxed plans for probabilistic planning, which goes beyond what Bryce (2006) did. We show that this technique generates high probability plans (in Chapter 5).

6. The simple Incremental Contingency Planning framework that goes beyond what Foss (2007) did by computing a high-probability seed plan, and a *Gain* value that evaluates which outcomes will improve the overall seed plan probability. In addition, we have included the *Confrontation* technique to repair unrecoverable outcomes, which consists of adding a conformant solution that achieves the goal by using a different path (in Chapter 6).

7. The characterization of a new technique to compute probability estimates without determinization. This technique, called *Probability Estimates without Determinization* (PEWD), computes more accurate probability estimates because it considers the overall probability of a proposition across all the action's outcomes and the dependence between propositions in outcomes (in Chapter 7).

1.3 Structure and contents

This dissertation is divided into eight chapters. Chapter 1 describes the motivation, contributions, and structure of this dissertation. Chapter 2 presents a state of the art overview of the Automated Planning field. Chapter 3 and 4 deal with the computation of estimates of cost and how we apply them for classical planning and goal recognition. Chapters 5, 6, and 7 deal with the computation of estimates of probability and how we apply them for probabilistic planning. In particular:

- **Chapter 2: Automated Planning**, presents an overview of the three main problems addressed in this dissertation: classical planning, planning-based goal recognition, and probabilistic planning. It first presents a description of the classical planning problem, describes the planning algorithms, and revises some of the planning techniques in the state-of-the-art. Next, a brief description of the goal recognition problem is presented along with some planning-based techniques to solve it. Finally, it describes the probabilistic planning problem and enumerates the most significant approaches in the state-of-the-art.

- **Chapter 3: Cost Estimates in a Plan Graph using Interaction**, presents a novel technique to propagate cost through a plan graph that makes use of

Interaction information. It first describes the simple cost propagation in a plan graph. Then, it introduces the term Interaction. Next, it presents the new heuristic estimator h^I, and the new h^{2+} family of heuristics, along with an accuracy study of them and other heuristics in the state-of-the-art. Finally, it shows the use of the h^I heuristic in planning along with an experimental evaluation.

- **Chapter 4: The h^I Family of Heuristics in Goal Recognition**, presents the use of the h^I heuristic in goal recognition. It first describes *FGR*, a heuristic approach for goal recognition based on h^I, and shows an experimental evaluation. Next, it presents a real-time goal recognition problem for the International Space Station crew activities, and shows an experimental evaluation. Finally, it outlines a framework for goal recognition problems with uncertain observations.

- **Chapter 5: The h^I Family of Heuristics in Probabilistic Planning**, presents the use of the h^I heuristic in probabilistic planning, in particular, through the *Determinization and Replanning* technique. Then, it presents an experimental evaluation.

- **Chapter 6: Incremental Contingency Planning for Recovering from Uncertain Outcomes**, presents an approach to improve the overall probability of a non-branching seed plan by incrementally generating contingency branches to deal with the most critical outcomes of the actions of the plan. Then, it presents an experimental evaluation.

- **Chapter 7: Probability Estimates without Determinization**, presents a new technique to compute estimates of probability without determinization. It first describes the problematic of the Determinization technique. Then, it defines Interaction in terms of probability. Next, it presents a search algorithm in the space of probabilistic states, and continues defining a novel heuristic function based on probability propagation in a plan graph. Finally, it presents an experimental evaluation.

- **Chapter 8: Conclusions and Future Work**, presents conclusions and discusses some relevant future research directions.

- **Appendix A: Extended Experimental Results for Goal Recognition**, presents more details on the experimental evaluation performed in Chapter 4.

- **Appendix B: ISS Crew Activities Domain Definition**, presents the PDDL description of the ISS Crew Activities Domain characterized in Chapter 4.

1.4 Publications

Some of the work presented in this thesis has been previously published in the following articles:

1. Yolanda E-Martín, María D. R-Moreno, and David E. Smith. *Probabilistic Plan Graph Heuristic for Probabilistic Planning*. In Proceedings of the 25th National Conference on Artificial Intelligence (AAAI), 2011.

2. Yolanda E-Martín, María D. R-Moreno, and David E. Smith. *Using a Plan Graph with Interaction Estimates for Probabilistic Planning*. In Proceedings of the 31st SGAI International Conference on Artificial Intelligence, 2011. (One of the best six refereed papers.)

3. Bonifacio Castaño, Yolanda E-Martín, María D. R-Moreno, and Luis Usero. *Sistema Inteligente de Detección y Orientación de Usuarios en Bibliotecas*. In Revista Española de Documentación Científica 36(1), pages 1–9, 2013.

4. Yolanda E-Martín, María D. R-Moreno, and David E. Smith. *Progressive Heuristic Search based on Interaction Estimates*. In Expert Systems 31(5), pages 421–436, 2014.

5. Yolanda E-Martín, María D. R-Moreno, and David E. Smith. *A Fast Goal Recognition Technique based on Interaction Estimates*. In Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI), 2015.

6. Yolanda E-Martín, María D. R-Moreno, and David E. Smith. *A Heuristic Estimator based on Cost Interaction*. In Proceedings of the 7th ICAPS-15 Workshop on Heuristics and Search for Domain-Independent Planning (HSDIP), 2015.

7. Yolanda E-Martín, María D.R-Moreno, and David E. Smith. *Practical Goal Recognition for ISS Crew Activities*. In Proceedings of the 9th International Workshop on Planning and Scheduling for Space (IWPSS), 2015.

Chapter 2

Automated Planning

This chapter contains the background definitions mentioned in subsequent chapters. It first presents a classical planning problem. Next, it defines a goal recognition problem in the context of automated planning. Finally, it describes a probabilistic planning problem.

2.1 Classical Planning

Classical planning is the problem of choosing and organizing a sequence of actions that when applied in a given initial state results in a goal state. Formally, a planning problem model can be described as a tuple $\Pi = < S, s_0, s_G, O, t >$ where:

- S is a set of states.
- $s_0 \in S$ is a set that represents an initial state.
- $s_G \in S$ is a set that represents goal states.
- O is a set of actions.
- t is a transition function where $t : S \times O \mapsto S'$.

A classical planning problem may be represented as a state-transition graph whose nodes represent different states in S, and whose arcs represent actions in O, which designate the transition from a state s to a state s' (Ghallab et al., 2004). The solution of a classical planning problem is a sequence of actions $\pi = \{o_1, ..., o_n\}$, or *plan*, which can be understood as a path from the initial node state to the goal node state. If the transition function has unit cost, the cost of the plan solution is the number of actions, or *length*. In contrast, if the transition function deals with costs, the cost of the plan solution is the sum of the cost of the actions in the plan, i.e.:

$$Cost(\pi) = \sum_{o_i \in \pi} cost(o_i)$$

For large planning problems it is generally impractical to explicit represent all the possible states and transition states. For this reason, planning languages based on propositional logic emerged. These languages consist of a set of variables that represents those predicates that are true or false about the world. The most common and first representation is known as and was used in the Stanford Research Institute Problem Solver (STRIPS) (Fikes and Nilsson, 1971). STRIPS defines a planning problem as a tuple $\Pi = < P, O, I, G >$ where:

- P is a set of predicates.
- O is a set of actions, each having the form $< prec(O), add(O), del(O) >$ where:
  - $prec(O) \subseteq P$ is the set of predicates that must be satisfied (true) before the action can be executed.
  - $add(O) \subseteq P$ is the set of positive predicates that are true when the action is applied.
  - $del(O) \subseteq P$ is the set of negative predicates that are false when the action is applied.
- $I \subseteq P$ is a set of initial state predicates.
- $G \subseteq P$ is a set of goal state predicates.

A plan solution is a sequence of actions $\pi = \{o_1, ..., o_n\}$, which represents the path to reach G starting from I. A plan solution is *optimal* when $Cost(\pi)$ is the minimum among all the possible plans.

Since STRIPS, different planning languages have been developed. The most common is the Planning Domain Description Language (PDDL) (McDermott, 1998), with PDDL 3.1 (Helmert, 2008) being the latest version. PDDL has been adapted as the standard language in planning competitions. A PDDL planning problem consists of a *domain* that defines predicates, constants, functions, and actions, and a *problem* that defines objects, an initial state, and a goal state. The objects described in the problem definition are used to instantiate predicates, functions, and actions defined in the domain description. This process is known as *grounding* and consists of performing all possible combinations of objects to created ground predicates or *propositions* from predicates, and *ground actions* from actions. Actions described in the domain definition may have an associated function *total-cost*, which represents the cost of execute them.

```
(define (domain logistics)
 (:requirements :strips :typing :action-costs)
 (:types truck package location)
 (:predicates (connected ?a1 ?a2 - location)
              (at ?l - location ?pkg - package)
              (at ?l - location ?t - truck)
              (in ?pkg - package ?t - truck)
              (scanned ?pkg - package ?t - truck)
              (verified ?pkg - package ?t - truck))
 (:functions (total-cost))

 (:action drive
  :parameters (?t - truck ?from - location ?to - location ?pkg - package)
  :precondition (and (connected ?from ?to) (at ?from ?t) (in ?pkg ?t) (verified ?pkg ?t))
  :effect (and (not (at ?from ?t)) (at ?to ?t) (increase (total-cost) 1))

 (:action load
  :parameters (?pkg - package ?l - location ?t - truck)
  :precondition (and (at ?l ?t) (at ?l ?pkg))
  :effect (and (in ?pkg ?t) (not (at ?l ?pkg)) (increase (total-cost) 1))

 (:action verify
  :parameters (?pkg - package ?t - truck ?l - location)
  :precondition (and (in ?pkg ?t) (at ?l ?t))
  :effect (and (verified ?pkg ?t) (increase (total-cost) 1))

 (:action scan
  :parameters (?pkg - package ?t - truck)
  :precondition (and (in ?pkg ?t) (verified ?pkg ?t))
  :effect (and (scanned ?pkg ?t) (increase (total-cost) 1))

 (:action unload
  :parameters (?pkg - package ?t - truck ?l - location)
  :precondition (and (at ?l ?t) (scanned ?pkg ?t))
  :effect (and (not (in ?pkg ?t)) (at ?l ?pkg) (increase (total-cost) 1))
```

```
(define (problem logistics-p01)
 (:domain logistics)
 (:objects a b - location trk - truck pkg - package
 (:init (connected a b) (at a trk) (at a pkg) (= (total-cost) 0))
 (:goal (at b pkg))
 (:metric minimize (total-cost)))
```

Figure 2.1: A PDDL domain and problem description in the Logistics Domain.

A description in PDDL of a simple Logistics domain and problem is shown in Figure 2.1, where there is a package and a truck at location *a* and the package needs to be delivered to location *b*. The package can be loaded on the truck. The truck can drive between locations, but before it can do it must be verified that the package is on the truck. In addition, the package needs to be scanned before it can be unloaded from the truck. Therefore, this domain lists five operators for driving between connected locations, and loading, unloading, scanning, and verifying a package. The *drive* operator can be instantiated with a truck object *trk*, location objects *a* and *b*, and a package object *pkg* to generate, for instance, the ground action (drive trk a b) whose preconditions are {(connected a b), (at a trk), (in pkg trk), (verified pkg trk)}, and effects are {(at b trk), (¬at a trk)}. Each action in the domain description has an associated cost of 1 given by the function (increase (total-cost) 1). The problem instance in Figure 2.2 indicates that the initial state is:

$$I = \{\text{(connected a b), (at a trk), (at a pkg)}\}$$

and the goal state is:

$$G = \{\text{(at b pkg)}\}$$

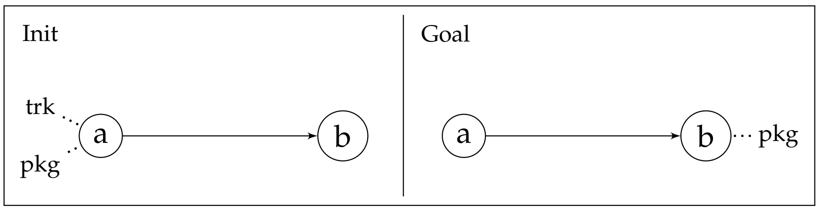


Figure 2.2: Initial and goal states for a simple Logistics domain.

A valid plan π for this problem is:

$$\pi = \begin{pmatrix} \text{(load pkg a trk),} \\ \text{(verify pkg trk a),} \\ \text{(drive trk a b),} \\ \text{(scan pkg trk)} \\ \text{(unload pkg trk b)} \end{pmatrix} \quad \text{with} \quad Cost(\pi) = 5$$

2.1.1 Planning algorithms and techniques

Classical planning models assume that the planning problem Π is *finite*, i.e., it has a finite set of states; is *fully observable*, i.e., the represented knowledge in Π is always known; is *deterministic*, i.e., the resulting state after the application of an action may be predicted exactly. Therefore, classical planning may be performed as a deterministic state model, which may be solved by search algorithms. A search space may be in the *space of states*, which defines the search space as a tree whose nodes represent a state in the world, and whose edges represent a transition action; or in the *space of plans*, which defines the search space as a tree whose nodes represent a partial plan, and whose edges represent an action that may be added to the plan.

A search algorithm may be *progressive*, which searches forward from the initial state considering sequence of actions until it finds a sequence π that reaches the goal state. Initially, π is empty. Each step in the search involves the selection of an action. In particular, given a state description s, it first checks if s is a goal state. If so, the search process finishes. Otherwise, it finds the set of applicable ground actions to s, and computes the successor state $s\prime$ resulting from applying a ground action a in s, that is:

$$s\prime = \{s - del(a)\} \cup add(a)$$

A search algorithm may be also *regressive*, which searches backward from the goal state considering the inverse of ground operators to produce subgoals, and stopping if it produces a set of subgoals satisfied by the initial state. Given a goal description g and a ground action a, the regression from g over a gives a state g' such as:

$$g' = \{g - add(a)\} \cup prec(a)$$

Progression and regression search algorithms return a plan solution (if there is one) that corresponds to a path from the initial state to the goal state, or a *failure* if a path is not found.

There are several techniques that may be used in progression and regression algorithms and may perform a search in a space of states or a space of plans. In this dissertation, we focus on planning graph techniques (Blum and Furst, 1997) and heuristic search planning (Bonet and Geffner, 2001; Hoffmann and Nebel, 2001), which are described in Sections 2.1.2 and 2.1.3 respectively. Other well-known techniques are:

- Total order planning: generates totally ordered plans, which means that each action in the plan is ordered with respect to every other action. Total order planners are commonly associated with a states-space search (Fikes and Nilsson, 1971; Vidal and Pierre, 1999). However, there are several well-known total order planners have been plan-based (Veloso et al., 1995; Tate, 1974; Warren, 1974).

- Partial order planning: generates partially ordered plans, which means that each action in the plan is partially ordered with respect to every other action. Partial order planners are commonly plans-space (Coles et al., 2010; Penberthy and Weld, 1994, 1992; Bonet and Geffner, 2003b; Joslin and Pollack, 1994), but it is possible to have a state-based planner (Godefroid and Kabanza, 1991)

- SAS$^+$ planning: the SAS$^+$ formalism (Bäckström and Nebel, 1995) represents a planning problem using multi-valued state variables instead of the propositional facts in STRIPS. It has been used to derive heuristics (Helmert, 2006; Helmert et al., 2008), landmarks (Richter et al., 2008), search models (Chen et al., 2008), and strong mutual exclusion constraints (Chen et al., 2009).

- Propositional satisfiability planning: converts the planning problem into a propositional satisfiability, that is, as the problem of determining whether a propositional formula is satisfiable. Then, it extracts a plan. Planning as satisfiability was first proposed by Kautz and Selman in their SATPLAN system (Kautz and Selman, 1992), and it has been improving continuously over the years (Kautz and Selman, 1996; Rintanen, 2004, 2006; Chen et al., 2009; Wehrle and Rintanen, 2007; Robinson et al., 2009; Rintanen, 2011, 2012; Huang et al., 2012).

2.1.2 Planning reachability

As mentioned previously, classical planning is the problem of finding a path from an initial state to a goal state in a state-transition graph. A significant drawback in planning is the state-space representation size, which may be impractical. Given the difficulty in solving large planning problems, reachability analysis has been studied. The aim of reachability analysis is to detect in advance unreachable propositions and actions from the initial state. By discovering those actions, the search space may be reduced, resulting in a simpler search space for planning. A common method of doing reachability analysis makes use of *plan graphs*.

A plan graph was first used in the GRAPHPLAN planner (Blum and Furst, 1997). It is a powerful and inexpensive structure that provides an efficient method to estimate the set of achievable propositions, and the set of feasible actions, starting in a particular state. This structure does not detect all the propositions and actions that may be excluded. However, any proposition and any action that is *pruned*, i.e., does not appear in the structure, may be dismissed for planning purposes.

A plan graph is a directed leveled graph where each level l is composed of two layers: P_l, the set of achievable propositions, and A_l, the set of feasible actions. Arcs represent relations between propositions and actions, and between propositions. An outgoing arc from a proposition node in P_l to an action node in A_l is a *precondition* arc. An outgoing arc from an action node in A_l to a proposition node in P_{l+1} is an *effect* arc, which may be positive or negative. An outgoing arc from a proposition node in P_l to a proposition node in P_{l+1} is a *no-operation* arc, which is a *doing nothing* action. A plan graph is extended forward until all goals are reached. Starting at level 0, the first layer consists of the set P_0 of propositions defining the initial state of the problem. The subsequent actions' layer consists of the set A_l of actions, which contains all the actions whose preconditions are true in P_0. The union of all positive and negative effects of A_0, along with the propositions in P_0, form the next propositions' layer which consists of the set P_1 of preconditions. Because no-operations (noop) are allowed, a proposition that appears in the propositions' layer P_l also appears in the propositions' layer P_{l+1}.

As a plan graph represents parallel propositions and actions that can be mutually exclusive, during the construction phase the relations that may exist between pairs of propositions and pairs of actions are considered in each level. Specifically, the algorithm defines the following mutual exclusion relations:

- Between pairs of operators:
    - *Inconsistent effects*: the effect of one operator deletes a proposition that is added by the other.
    - *Effects clobbering preconditions*: one action's effect deletes a precondition of the other.
    - *Competing needs*: the two operators have mutex preconditions.
- Between pairs of propositions:
    - *Contradiction*: one proposition is the negation of the other.
    - *Inconsistent support*: all the ways for obtaining both propositions are mutex.

Figure 2.3 shows a graphical representation of mutex relations. Circles refer to propositions and squares refer to actions. Red lines define mutex relation between two elements.

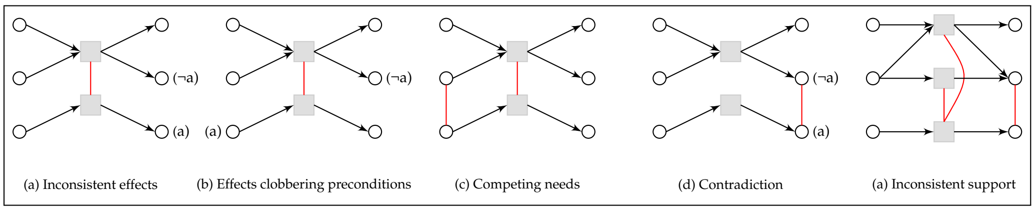


Figure 2.3: Graphical description of mutex relations in a plan graph.

The plan graph construction process is repeated until (1) a set P_l of propositions that contains all goal propositions is achieved, and goal propositions are not mutually exclusive between them, or (2) the plan graph reaches quiescence, i.e., two consecutive sets of propositions P_{l-1} and P_l are identical. If the goal is not achieved when the plan graph reaches quiescence, then the goal is not reachable, which means that there is no plan solution. This process is illustrated in Figure 2.4 for the simple logistics problem shown in Figure 2.1. Table 2.1 lists the mutex relations for the example presented in Figure 2.4.

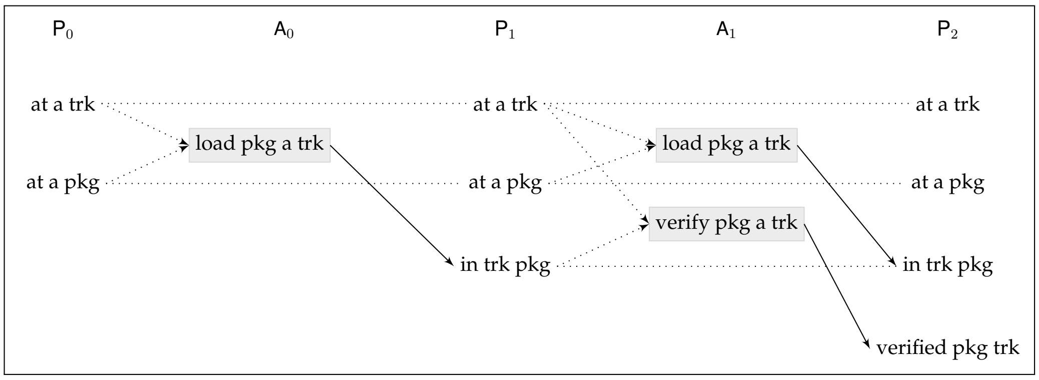


Figure 2.4: Example of a plan graph.

Table 2.1: List of mutex relations.

Layer	Mutex Pair	Mutex Relation
A_0	{load pkg a trk} × {noop-at a pkg}	Inconsistent effects
P_1	{at a pkg} × {in trk pkg}	Inconsistent support
	{at a pkg} × {¬at a pkg}	Contradiction
A_1	{load pkg a trk} × {noop-at a pkg}	Inconsistent effect
	{load pkg a trk} × {noop-in trk pkg}	Competing needs
	{load pkg a trk} × {verify pkg a trk}	Inconsistent effect
P_2	{at a pkg} × {in trk pkg}	Inconsistent support
	{at a pkg} × {verified pkg trk}	Inconsistent support

2.1.3 Planning as heuristics search

A classical planner aims to automatically solve a deterministic planning problem by exploring the state-space of the problem to find a path from I to G. One optimal way to find an optimal solution to a planning problem is to perform an exhaustive search over the state-space and get the best (optimal) solution. However, this is impractical for a large size states-spaces. As mentioned previously, reachability analysis may help to reduce the state-space, although it is insufficient. A more effective way to solve deterministic planning problems is to use heuristic search algorithms (Hart et al., 1968; Nilsson, 1980; Hoffmann and Nebel, 2001; Hansen and Zilberstein, 2001), which have been used in most recent classical planners (Bonet and Geffner, 2001; Hoffmann and Nebel, 2001; Hoffmann, 2003; Gerevini et al., 2003; Do and Kambhampati, 2003; Chen et al., 2006; Keyder and Geffner, 2007, 2008a; Keyder et al., 2010; Richter and Westphal, 2010; Keyder et al., 2012).

A heuristic search algorithm is a technique that uses a heuristic to find the sequence of actions that reach the goal from the initial state. The search space consists of a set of nodes, which represent a description of a state s in form of propositions. There is a unique node s_0 known as the *initial state node*. Nodes are connected by arcs, which represent an action a that was used to reach a state s' from state s. s' is known as the *successor* node of s. A basic heuristic search procedure explores the search space by ranking all the current reachable nodes according to the following heuristic function:

$$f(n) = cost(s) \; + \; h(c(s, g)) \tag{2.1}$$

where $cost(s)$ is the cost of the explored part, and $h(c(s, g))$ is the estimated cost of reaching a goal state g from a given state s (Pearl, 1984). Therefore, $f(n)$ is used to guide the search towards the most promising nodes. The algorithm starts from s_0 by expanding and computing $f(n)$ for each successor of s_0. These successors are then ranked according to their $f(n)$ in an *open list*. At each step, the most promising successor is removed from the open list to be inserted in a *closed list*, while its successors are added to the open list, ordered by the value of $f(n)$. In order to avoid repeated states, it is verified if the current successor is present in the open list or closed list, which means that it has already been generated. If it is present in the closed list or in the open list with a worse $f(n)$, the value and the action associated with the node are updated, and if it was present in the closed list, the node is moved to the open list.

Pearl (1984) defines a *heuristic* as a method to decide among alternative courses of action in order to find the most effective way to achieve a goal. A heuristic needs to be simple to compute and effective to discriminate among the different alternatives. Although it does not always guarantee optimality, it has been an efficient method in combination with heuristic search algorithms to explore the state-space even when it is extremely large.

In heuristic planning techniques, the search space is assumed to be an OR graph where a path from an initial state to a goal state needs to be found. A graph consists of a set of nodes, which in the planning context represent the description of a state s in the form of propositions. Nodes are connected by arcs, which in the planning context represent an action a that was used to reach a state s' from state s. In the search of this path using heuristic planning techniques, a heuristic takes the form of *heuristic estimator* (h), which is the estimated cost of reaching a goal state g from a given state s through the course of action beginning with a, i.e., $h(s) = c(s, a)$. The perfect heuristic estimator, h^*, provides the optimal (or cheapest) cost from a state s to a goal state g. That is, $h^*(s) = \min\{c(s, a)\}$. This means that, there is always an optimal path to reach g from s. Essentially, in order to guarantee an optimal solution, h should be optimistic and never overestimate the cost of reaching the goal. Such heuristics are called *admissible heuristics*. A heuristic estimator h is admissible when:

$$h(s) \leq h^*(s) \ \forall s \tag{2.2}$$

Admissible heuristic estimators in combination with heuristic search algorithms produce optimal solutions, but may take a long time for large state-spaces. A drawback of an admissible heuristic estimator is that it may spend a significant amount of time discriminating among the different courses of action for some problems. In those cases where admissibility is harmful to the search process, non-admissible heuristic estimators can be used, which may lead the problem towards suboptimal solutions. Non-admissible heuristic estimators can overestimate the cost to reach the goal, which means that potential courses of action that actually have a lower cost may not be explored. For this reason, non-admissible heuristics can produce suboptimal solutions, but make often the search more tractable.

Heuristic functions are commonly derived from simplified models of the original problem, which are generated by removing constraints and by making assumptions (Pearl, 1984). This is known as *relaxation*. A common relaxation-based heuristic in planning is the *delete relaxation*, which consists of ignoring delete effects. In other words, in an original problem, any applicable action a in a state s produces a new state s' where $s' = \{s - (add(a) \cup del(a))\}$. If delete effects are ignored, $s' = \{s + add(a)\}$. This means that a proposition holds after the first level at which it can be achieved. Relaxation-based heuristics are admissible if the cost of the relaxation is lower than the optimal cost. The following heuristic functions are based on this relaxation idea.

The max heuristic

The max heuristic h^{max} (Bonet and Geffner, 2001) computes an estimation of cost, which is defined by the following recursive equation:

$$h^{max}(p,s) \overset{def}{=} \begin{cases} 0 & \text{if } p \in s \\ \underset{a \in O(p)}{\text{argmin}} \left\{ cost(a) + \underset{q \in prec(a)}{\max} h^{max}(q,s) \right\} & \text{otherwise} \end{cases} \tag{2.3}$$

where $h^{max}(p,s)$ refers to an estimate of the cost of achieving a proposition p from state s. This estimate is the minimum cost among all the actions a that add p, which, in turn, is defined as the maximum cost among all preconditions of a. The max heuristic is admissible since the cost of achieving a set of propositions cannot be lower than the cost of achieving any proposition of the set.

The additive heuristic

The additive heuristic h^{+} (Bonet and Geffner, 2001) computes a cost approximation, which is defined by the following recursive equation:

$$h^{+}(p,s) \overset{def}{=} \begin{cases} 0 & \text{if } p \in s \\ \underset{a \in O(p)}{\text{argmin}} \left\{ cost(a) + \sum_{q \in prec(a)} h^{+}(q,s) \right\} & \text{otherwise} \end{cases} \tag{2.4}$$

where $h^{+}(p,s)$ refers to an estimate of the cost of achieving a proposition p from state s. This estimate is the minimum cost among all the actions a that add p, which, in turn, is defined as the sum of the cost of the individual preconditions of a. This estimation of cost assumes that *subgoals* are *independent*. When this assumption is true, that is, when preconditions of an action do not interfere with each other, the additive heuristic is admissible in the delete relaxation. On the other hand, when this assumption is not true, that is, when preconditions of an action are synergistic with each other, it may overestimate the cost. For this reason, the additive heuristic is not admissible. Therefore, the estimated cost computed is an *approximation* of the optimal cost.

The relaxed plan heuristic

The relaxed plan heuristic h^{FF} (Hoffmann and Nebel, 2001) computes a cost approximation by computing a plan, which is not guaranteed to be optimal. This is done using a construction plan graph procedure like is described in Section 2.1.2. Since delete relaxation is assumed, a relaxed plan graph with no mutex is built. The relaxed plan graph is used to extract a plan π starting from state s, where the length of π gives an approximation of the estimated cost of achieving the goal from s.

Function RELAXEDPLANEXTRACTION (G_l)

l	$\equiv$	number of levels in the relaxed plan graph
G_l	$\equiv$	the set of goals at level l in the relaxed plan graph
g	$\equiv$	a goal proposition
S	$\equiv$	the set of propositions of G_l already achieved
A_l	$\equiv$	the set of actions at level l for an specific goal
a	$\equiv$	an action
l_a	$\equiv$	the level in which action a is first reached
π	$\equiv$	a relaxed plan
L_π	$\equiv$	the length in number of actions of π

```
1. $\pi \leftarrow \emptyset$
2. while $l > 0$ do
3.     $S \leftarrow \emptyset$
4.     $G_{l-1} \leftarrow \emptyset$
5.     while $G_l \neq \emptyset$ do
6.         $g \leftarrow g \in G_l$
7.         if $g \notin S$ then do
8.             $A_l \leftarrow \{a : g \in add(a) \wedge l_a < l\}$
9.             $a \leftarrow \underset{a \in A_l}{\text{argmin}} \{ \mid prec(a) \mid \}$
10.            $S \leftarrow S \cup add(a)$
11.            $G_{l-1} \leftarrow G_{l-1} \cup prec(a)$
12.            $\pi \leftarrow \pi \cup \{a\}$
13.        $G_l \leftarrow G_l - \{g\}$
14.    $l \leftarrow l - 1$
15. return $L_\pi$
```

Figure 2.5: The relaxed plan extraction pseudo-algorithm.

Figure 2.5 shows the high-level algorithm used to extract a relaxed plan. The algorithm starts with the propositions corresponding to the goals at level l. For each goal proposition g, the algorithm selects an action from level $l-1$, that supports g. If there is more than one supporter, the process selects the one with the lowest number of preconditions. Each time an action is selected, all its positive effects are marked as *achieved* at level l, to prevent double check goals from level

l, and its preconditions are added as goals for the $l - 1$ level. The algorithm repeats the same process on the $l - 1$ level of the plan graph, with preconditions of the actions selected at actions' layer $l - 1$ as the goals for the $l - 1$ level. The search succeeds when level 0 is reached, that is, the initial state. The sequence of selected actions forms relaxed plan. The heuristic value is the length, or number of actions, of the relaxed plan.

To illustrate this algorithm, considerer the simple Logistics problem described in Section 2.1. Figure 2.6 shows the plan graph for the problem. The goal set at level 4 contains the single proposition (at b pkg), which supporter is action (unload pkg trk b) at level 3. Propositions (scanned pkg trk) and (at b trk), which are the preconditions of (unload pkg trk b), are the goals at level 3. The action (scan pkg trk) supports the goal (scanned pkg trk), and adds (in trk pkg) and (verified pkg trk) as goals at level 2. The action (drive trk a b) supports the goal (at b trk), and adds (at a trk), as well as (in trk pkg) and (verified pkg trk) that were already added by (scan pkg trk). The goal (at a trk) and (in trk pkg) at level 2 have the (noop) action as supported, while (verified pkg trk) has action (verify pkg a trk), which adds propositions (at a trk) and (in trk pkg) as goals at level 1. (These two propositions were already added by the (noop) action.) Again, (at a trk) is supported by the (noop).The proposition (in trk pkg) is supported by action (load pkg a trk), which adds propositions (at a trk) and (at a pkg) goals at level 0. Finally, propositions (at a trk) and (at a pkg) are true at level 0, which means there is a valid solution. The relaxed plan consists of actions (load pkg a trk), (verify pkg a trk), (scan pkg trk), (drive trk a b), and (unload pkg trk b). Therefore, for this example the heuristic value is 5.

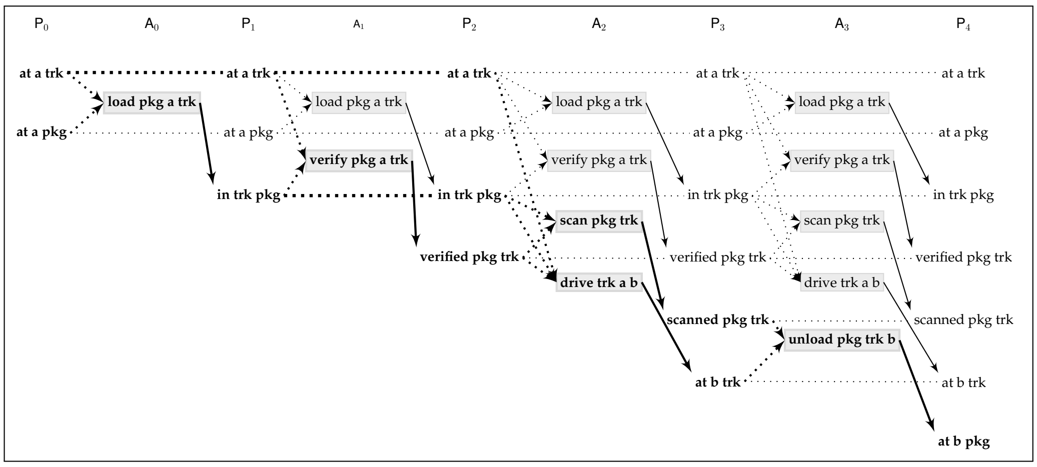


Figure 2.6: A relaxed plan graph example.

Unlike the additive heuristic and the max heuristic, the original relaxed plan heuristic does not take cost information into account, and considers unit action costs. Nevertheless, it is possible to easily extend this to action costs by summing the cost of the actions in the relaxed plan.

The higher-order heuristic

The higher-order heuristic h^m (Haslum and Geffner, 2000) computes an admissible cost approximation for regression search. It is a generalization of the max heuristic that computes estimates of cost for sets of m propositions from the initial state s_0. It is defined by the following equation:

$$h^m(A) \overset{def}{=} \begin{cases} 0 & \text{if } A \in s_0 \\ \min\limits_{\langle B,a\rangle \,\in\, R(A)} cost(a) + h^m(B) & \text{if } |A| \leq m \text{ and } A \nsubseteq s_0 \\ \max\limits_{B \,\subset\, A,\, |B|\,=\,m} h^m(B) & \text{if } |A| > m \end{cases} \tag{2.5}$$

where $h^m(A)$ refers to the cost of achieving a state s where a set of propositions A of size m is true, s_0 refers to the initial state definition, and $R(A)$ refers to the set of pairs $\langle B, a\rangle$ where B is the resulting state of regressing A through a, defined as $R(B, a) = \{A - add(a)\} \cup prec(a)$.

The h^m heuristic is admissible because $h^m(B) \leq h^*(B)$ for every B, and it is true for size m of the set. However, it presents a trade-off between accuracy and efficiency. Higher values of m are more accurate, but more expensive to compute. The computation of h^m is a low-order polynomial in N^m where N is the number of propositions in the problem, and m the size of the considered sets. For practical purposes, it is limited to $m = 2$ or $m = 3$. When the size of m is one, the heuristic h^m is equivalent to the max heuristic. When the size of m is two, the heuristic h^m computes the estimated cost between two propositions p and q as follows:

$$h^2(\{p, q\}) = \min \begin{cases} \min\limits_{a \,\in\, O(p \wedge q)} & cost(a) + h^2(prec(a)); \\ \min\limits_{a \,\in\, O(p|q)} & cost(a) + h^2(prec(a) \cup \{q\}); \\ \min\limits_{a \,\in\, O(q|p)} & cost(a) + h^2(prec(a) \cup \{p\}) \end{cases} \tag{2.6}$$

where $h^2(\{p\}) = \min_{a \in O(p)} cost(a) + h^2(prec(a))$, $O(p \wedge q)$ refers to the set of actions that achieves p and q simultaneously, $O(p|q)$ refers to the set of actions that adds p and does not add or delete q, and $O(q|p)$ refers to the set of actions that adds q and does not add or delete p.

The set-additive heuristic

The set-additive heuristic h_a^s (Keyder and Geffner, 2007) is a modification of the additive heuristic that produces relaxed plans taking the cost information into consideration. It can be defined by the following recursive equation:

$$\begin{aligned}
h_a^s &= \pi_a(p,s) \overset{def}{=} \begin{cases} \{\} & \text{if } p \in s \\ \pi_a(a_p, s) & \text{otherwise} \end{cases} \\
a_p &= \underset{a \in O(p)}{\text{argmin}} \left\{ Cost(\pi_a(a,s)) \right\} \\
\pi_a(a,s) &= \{a\} \bigcup \left\{ \bigcup_{q \in prec(a)} \pi(q,s) \right\} \\
Cost(\pi_a(a,s)) &= \sum_{a' \in \pi(a,s)} cost(a')
\end{aligned} \tag{2.7}$$

where $\pi_a(p,s)$ refers to a relaxed plan for p from s. During the construction of a relaxed plan graph for a state s, the additive heuristic is used to propagate the cost and keep track of the actions that minimize each proposition in the relaxed plan graph. During the construction of a relaxed plan $\pi(s)$ in s, actions that minimize the cost of each subgoal are selected. The estimated relaxed plan for a set of propositions in s is the union of the relaxed plans for each proposition in s. The cost approximation is the sum of the costs of the actions in $\pi(s)$. That is:

$$h_a^s(s) = \sum_{a \in \pi(s)} cost(a) \tag{2.8}$$

2.2 Goal Recognition

In literature, plan recognition and goal recognition concepts are used interchangeable. However, there is a subtle difference between them. Specifically, plan recognition aims to infer an actor's *plan* and *goals* from some or all of the actor's observed (performed) actions, while goal recognition aims to infer an actor's *goals* from some or all of the actor's observed actions. Therefore, plan recognition can solve goal recognition problems, but not vice versa. Another related problem to goal recognition is Goal Recognition Design (Keren et al., 2014, 2015), which performs off-line analysis of a goal recognition problems given the domain description and the set of possible goals to (1) evaluate the ability to perform goal recognition within a problem, and (2) find efficient ways to compute and optimize such problem.

Plan recognition and goal recognition disciplines have captured the attention of several computer science communities and have been useful in several areas such as intelligent personal assistants (Weber and Pollack, 2008), smart environments (Wu et al., 2007), monitoring user's needs (Pollack et al., 2003; Kautz et al., 2002), and for intelligent tutoring systems (Brown et al., 1977). Several different techniques have been used to solve plan recognition problems: parsing algorithms (Geib and Goldman, 2009), Bayesian inference (Albrecht et al., 1997; Bui, 2003), hand-coded action taxonomies (Kautz and Allen, 1986), consistency graphs (Lesh and Etzioni, 1995), and probabilistic belief networks (Huber et al., 1994). In early approaches to goal recognition, the actor's observed actions are compared against a hand generated library of plans. To overcome this limitation, some authors have proposed the use of machine learning techniques (Bauer, 1998), or more recently automated planning.

Goal recognition may be seen as the inverse of planning since it is concerned with inferring an agent's goals given some or all agent's performed actions. There are few planning-based approaches for goal recognition, but this paradigm is still in its infancy. The classical planning techniques and algorithms presented in Section 2.1 are the foundations for these approaches. In particular, Pattison (2010) developed AUTOGRAPH that uses domains encoded in PDDL and then applies Helmert's translation (2009) to translate the problem into a SAS$^+$ formalism. This translation produces a Domain Transition Graph and a Causal Graph that are used to estimated distance to each fact after action observations and thus determine potential goal facts. Jigui and Minghao (2007) developed Incremental Planning Recognition (IPR from now on) that is based on reachability in a plan graph. Ramírez and Geffner (2009; 2010; 2012) developed a more principled approach for estimating the probability of each possible goal, given the observations that is based on heuristic search planners. In this dissertation, we focus on the two latter approaches so they are reviewed in detail in the next two subsections.

2.2.1 Goal recognition through heuristic planning

Ramirez and Geffner (2009) developed an approach for identifying the goal set $G \in \mathcal{G}$ where some optimal plan for G is compatible with the sequence of observed actions. In this work, a goal recognition problem is defined as a tuple $T = \langle P, \mathcal{G}, O \rangle$ where P is a planning domain in the form of $P = \langle F, A, I \rangle$ where F is the set of predicates, A is the set of actions, and I is the set of initial conditions; $\mathcal{G}$ is a set of possible goal sets or hypotheses; and O is an observed action sequence $O = (o_1, ..., o_n)$ where o_n is an action in the sequence. The observation sequence O may be incomplete, but is sequentially ordered. However, the time step in which the actions have been observed is unknown. The solution to this goal recognition problem is a set $\mathcal{G}^*$, which consists of all goals subsets $G \in \mathcal{G}$ having an optimal plan π consistent with O.

In order to solve a goal recognition problem using planning, the goal recognition problem $T = \langle P, \mathcal{G}, O \rangle$ is transformed into a problem $T' = \langle P', \mathcal{G}', O' \rangle$, where O' is empty and for each of the observed actions $o_i \in O$ a new set of predicates F_o and a new set of actions A_o are created such that $F_o = \{p_o | o \in O\}$ and $A_o = \{a_o | o \in O\}$. Each new action a_o has the same preconditions and effects as its counterpart a except for the new predicate p_o, which is added to $add(a_o)$, and $p_{o_{i-1}}$, which refers to the observed action that precedes action $a \in O$ and is, therefore, added to $prec(a_o)$. Hence, in the transformed goals recognition problem T', the planning domain is $P' = \langle F', A', I' \rangle$ where $F' = F \cup F_o$, $A' = A \cup A_o$, and $I' = I$, and the set of possible goal sets $\mathcal{G}' = \{G \cup F_o \,|\, G \in \mathcal{G}\}$. The extra goals F_o can be only achieved by the new actions A_o, which because of the added preconditions can be only applied after all the corresponding preceding $o_i \in O$ are executed. Therefore, the transformed problem P' along with each $G' \in \mathcal{G}'$ can be executed in a planner to find a plan π that is consistent with O. The set solution $\mathcal{G}^*$ consists of each goal set $G' \in \mathcal{G}'$ that gets a consistent plan.

In particular, this approach uses the optimal planner HSP$_f^*$ (Haslum and Geffner, 2000) to exactly compute $\mathcal{G}^*$. Although, it gives the optimal solution, it is computationally expensive. In addition, if there is a non-optimal plan consistent with the observations, this method fails. To remove both limitations, this work also considers approximate methods. The first method uses the suboptimal planner FF (Hoffmann and Nebel, 2001) to approximate the set, while the second is based on the h_a^s heuristic, which approximates the solution by computing a relaxed plan avoiding planning altogether.

The limitation of this work is the assumption that agents act optimally, which limits the theory to the space of optimal plans. Hence, suboptimal plans compatible with the observations are not considered. To overcome this limitation, Ramirez and Geffner (2010) developed a more principled approach for estimating the probability of each possible goal, based on how much the observed actions contribute to achieving that goal. This theory is rooted in the space of all possible plans that might be used to achieve each goal, and defines likelihood of a goal given a sequence of observations as the cost difference between achieving the goal complying with the observations, and not complying with them. This cost is computed by means of two calls to a planner for each possible goal. This approach defines a goal recognition problem as a tuple $T = \langle P, \mathcal{G}, O, Pr\rangle$ where P is a planning domain and initial conditions, $\mathcal{G}$ is a set of possible goal set or hypotheses, O is the observed action sequence $O = (o_1, ..., o_n)$, and Pr is the prior probability distribution over the goal sets $G \in \mathcal{G}$. The solution to a goal recognition problem is a probability distribution over each goal set $G \in \mathcal{G}$ giving the relative likelihood of each goal set. These posterior goal probabilities $P(G|O)$ can be characterized using Bayes Rule as:

$$Pr(G|O) = Pr(O|G)\, Pr(G) \tag{2.9}$$

where $Pr(G)$ is the prior distribution over $G \in \mathcal{G}$, and $Pr(O|G)$ is the likelihood of observing O when the goal is G. Ramirez goes on to characterize the likelihood $Pr(O|G)$ in terms of cost differences for achieving G under two conditions: complying with the observations O, and not complying with the observations O. More precisely, Ramirez characterizes the likelihood, $Pr(O|G)$, in terms of a Boltzman distribution:

$$Pr(O|G) = \alpha \frac{\exp\{-\beta\, \Delta(G,O)\}}{1 + \exp\{-\beta\, \Delta(G,O)\}} \tag{2.10}$$

where α is a normalizing constant, β is a positive constant, and $\Delta(G, O)$ is the cost difference between achieving the goal with and without the observations:

$$\Delta(G,O) = Cost(G|O) - Cost(G|\overline{O}) \tag{2.11}$$

Putting Equations 2.9 and 2.10 together yields:

$$Pr(G|O) = \alpha \frac{\exp\{-\beta\, \Delta(G,O)\}}{1 + \exp\{-\beta\, \Delta(G,O)\}} Pr(G) \tag{2.12}$$

By computing $\Delta(G, O)$ for each possible goal, Equation 2.12 can be used to compute a probability distribution over those goals. The two costs necessary to compute $\Delta(G, O)$ can be found by optimally solving the two planning problems $G|O$ and $G|\overline{O}$. Ramírez shows how the constraints O and $\overline{O}$ can be compiled into the goals, conditions, and effects of the planning problem so that a standard planner can be used to find plans for $G|O$ and $G|\overline{O}$.

To illustrate the Ramírez approach, consider the example shown in Figure 2.7, where an agent can move *up*, *left*, and *right* at cost 1. It has two possible goals, G_1 and G_2, and $O = (o_1)$ as the observed sequence. For goal G_1, $Cost(G_1|O) = 4$, and $Cost(G_1|\overline{O}) = 4$. (The costs are the same since o_1 is on an optimal path to G_1 and there is another optimal path that reaches G_1 but does not include o_1.) Hence, $\Delta(G_1, O) = 0$, and $Pr(G_1|O) = \alpha(0.5)$. In contrast, $Cost(G_2|O) = 3$ and $Cost(G_2|\overline{O}) = 3$. This results in $\Delta(G_2, O) = 0$, and $Pr(G_2|O) = \alpha(0.5)$. This means that G_1 and G_2 are the same likely to occur given $O = (o_1)$.

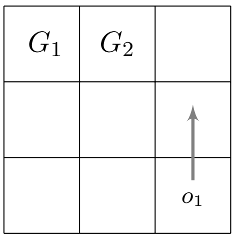


Figure 2.7: A 2x3 plan network for goals G_1 and G_2.

A major drawback to the Ramirez's approach is the computational expense of finding two plans for every possible goal. Moreover, the constraints O and $\overline{O}$ make the planning problems more difficult to solve. As a result, even for relatively simple problems it can take a significant amount of time to find all the plans necessary for this computation. This makes the approach impractical to use for any sort of real-time goal recognition problem.

2.2.2 Incremental plan recognition

Jigui and Minghao (2007) developed IPR, a framework for plan recognition that narrows the set of possible goal sets by incrementally pruning a plan graph as actions are observed. (As a plan recognition approach, it can also solves goal recognition problems.) In this work, a plan recognition problem is defined as a tuple $T = \langle P, \mathcal{G}, O \rangle$ where P is a planning domain and initial conditions, $\mathcal{G}$ is a set of possible goal sets or hypotheses, and O is an observed action sequence $O = (o_1, ..., o_n)$ where each o_n is a tuple $\langle a_n, t_n \rangle$ where a_n is an observed action and t_n is the time step in which a_n has been observed. O may be incomplete, but the time step is known. Therefore, the observed sequence is essentially ordered. The solution to a plan recognition problem is any plan $\pi = (a_i, ..., a_n)$ consistent with the sequence of actions O. A plan solution $\pi = (a_1, ..., a_n)$ is consistent with a sequence of observed actions $O = \langle a_1, t_1 \rangle, ..., \langle a_n, t_n \rangle$, if the time steps of the observed actions coincide with the time steps in which the actions in the plan solution occur. For instance, assuming the following sequence of observed actions $O = (\langle a, 0 \rangle, \langle b, 2 \rangle \ \langle c, 3 \rangle)$, $\pi_1 = (a, c, b, c)$ and $\pi_2 = (a, d, b, c)$ are consistent with O, but not $\pi_3 = (a, b, d, c)$.

In general, IPR consists of building a plan graph to determine which actions and which propositions are true (1), false (-1), or unknown (0) given the observations. The process starts at level zero where it is assumed that the initial state is true. As a consequence, every proposition has value 1. In addition, when an action is observed at a level it gets value 1. The process incrementally builds a plan graph and updates it level by level. The values of propositions and actions are updated according to the following rules:

1. An action in the plan graph gets value -1 when any of its preconditions or any of its effects is -1.

2. An action in the plan graph gets value 1 when it is the sole producer of an effect that has value 1.

3. A proposition in the plan graph gets value -1 when all of its producers are -1, (noop) included.

4. A proposition in the plan graph gets value 1 when any of its consumers or any of its producers is 1.

5. An action or proposition in the plan graph gets value -1 when it is mutually exclusive with an action or proposition that has value 1.

The process results in a plan graph where each proposition and each action is labeled as 1, -1, or 0. Those propositions and actions identified as -1 can be ignored for plan recognition purposes, meaning that these are pruned from the resulting plan graph. The approach backward searches for a plan consistent with O in the pruned plan graph.

This propagation and pruning technique is also useful to identify potential goal sets in goal recognition problems using the set of goals remainder in the plan graph. Each propositions labeled as 1 is considered a known goal and each propositions labeled as 0 is considered a possible goal, while each propositions labeled as -1 cannot be a goal.

To illustrate this propagation and pruning technique, consider our simple logistics problem described in Figure 2.1. Suppose that the sequence of observed

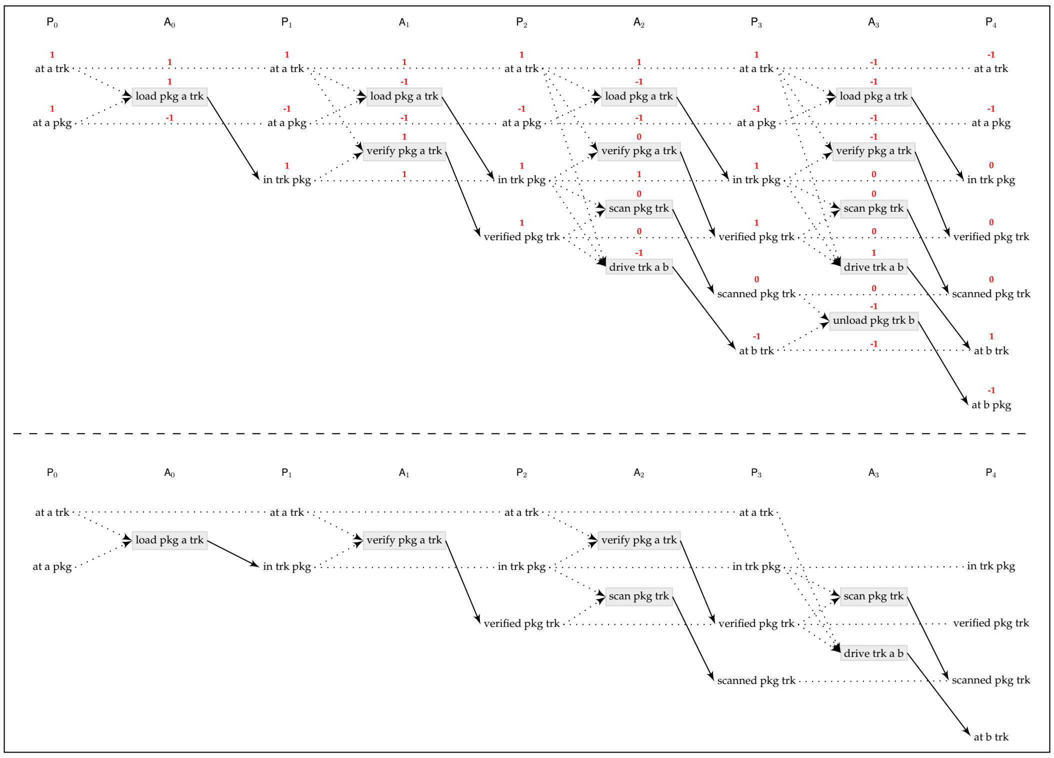


Figure 2.8: A partial plan graph with status values of propositions and actions.

actions is (verify pkg trk a) at level 1 and (drive trk a b) at level 3. Figure 2.8 shows the plan graph for this problem in the upper half. The numbers above the propositions and actions are the values for each of them, computed using the above propagation rules. The lower image shows the resulting pruned plan graph. As a result of the propagation, (load pkg a trk) must be true (value 1) at level 0 because its preconditions (at a trk) and (at a pkg) are true (value 1) at level 0. As a consequence of (load pkg a trk) being true, its effect (in trk pkg) is also true (value 1) at level 1. At level 2, (verified pkg trk) must be true because (verify pkg a trk) was observed. Since (verify pkg a trk) and (noop-at a pkg) are mutually exclusive, action (noop-at a pkg) and its effect (at a pkg) are false (value -1). As a consequence of (at a pkg) being false, (load pkg a trk) is false at level 2. Action (scan pkg trk) is unknown (value 0) at level 2 since there is not enough information to determine whether it is true or false. Action (drive trk a b) is true at level 3 since it was observed. As a result, (at b trk) is true at level 4. Actions (noop-at a trk) and (unload pkg trk b) are false at level 3 since they are mutually exclusive with the observed action (drive trk a b). As a consequence of (unload pkg trk b) being false, (at b trk) is also false at level 3, which makes (drive trk a b) false at level 2. Likewise, the consequence of (noop-at a trk) being false results in (at a trk) being false too. Proposition (scanned pkg trk) is unknown at every level since there is not enough information to label it.

The propagation technique results in a plan graph where at level 5 (at a trk) and (at a pkg) are false (value -1), and (in pkg trk), (verified pkg trk), (scanned pkg trk), (at b trk), and (at b pkg) are unknown (value 0). In terms of goal recognition, this means that propositions (at a trk) and (at a pkg) are dismissed as a solution, while (in pkg trk), (verified pkg trk), (scanned pkg trk), (at b trk), and (at b pkg) are possible goals solutions.

2.3 Probabilistic Planning

Classical planning is based on the assumption of complete knowledge of the initial conditions and the effects of actions. Significant advances have been made in recent years in the size and complexity of deterministic problems that can be solved with these planners. However, in real planning problems (Currie and Tate, 1991; Kingston et al., 1996; Drabble et al., 1997; Bernard et al., 1998; Tate et al., 2000; Khan et al., 2003; Meuleau et al., 2009; Fox et al., 2011; Castaño et al., 2014; Bresina, 2015), unexpected events may occur, actions may have unexpected outcomes and the state of the world may not be known with certainty. If a deterministic planner is used to solve such problems, the execution of the plan may fail because the plan does not take into account the possible contingencies. In order to address this problem, it is important to develop planners capable of handling uncertainty in the domain (Bonet and Geffner, 2005; Little and Thiébaux, 2006; Yoon et al., 2007, 2008, 2010; Keyder and Geffner, 2008b; Kalyaman and Givan, 2008; Buffet and Aberdeen, 2009; Teichteil-Königsbuch et al., 2010; Kolobov et al., 2012; Keller and Helmert, 2013).

A line of research dealing with planning problems under uncertainty is probabilistic planning, which describes the uncertainty using probability distributions. Probabilistic planning techniques have been developed to address problems where the uncertainty is *fully observable*, *partial observable*, or *not observable*. Probabilistic PDDL (PPDDL) (Younes et al., 2005) is an extension of the PDDL language capable of representing probabilistic planning problems by allowing probabilities on initial conditions and on action outcomes. The semantic of a PPDDL planning problem is given in terms of a discrete-time Markov Decision Processes (Puterman, 1994). To define probabilistic planning problems, PPDDL describes probabilistic actions as in PDDL with the addition of probabilistic outcomes on action definitions:

$$(\texttt{probabilistic}\ \ p_1\, e_1\ \ \ldots\ \ p_k\, e_k)$$

where each e_i is a PDDL effect that occurs with probability p_i. A probabilistic effect represents a distribution over deterministic effects where $\sum_{i=1}^{k} p_i = 1$. A description in PPDDL of a simple probabilistic Logistics domain and problem is shown in Figure 2.9. The source of uncertainty in this domain is in action *drive*:

```
(:action drive
 :parameters (?t - truck ?from - location ?to - location ?pkg - package)
 :precondition (and (connected ?from ?to) (at ?from ?t) (in ?pkg ?t)
                    (verified ?pkg ?t) (not (flattire)))
 :effect (and (not (at ?from ?t)) (at ?to ?t) (probabilistic 0.4 (flattire))))
```

where action effects are (not (at ?from ?t)), (at ?to ?t), and (flattire), but the latest only happens with a probability of 0.4. This can be seen as two different action outcomes:

- o_1, where (not (at ?from ?t)) and (at ?to ?t) happens with probability 0.6.
- o_2, where (not (at ?from ?t)), (at ?to ?t), and (flattire) happens with probability 0.4.

There is a second source of uncertainty in action *verify*:

```
(:action verify
 :parameters (?pkg - package ?t - truck ?l - location ?to - location)
 :precondition (and (at ?l ?t) (in ?pkg ?t)
 :effect (and (probabilistic 0.8 (verified ?pkg ?t))))
```

where the action's effect is (verified ?pkg ?t), but it only happens with a probability of 0.8. Otherwise, the action does nothing.

Figure 2.10 shows a simple probabilistic Logistics problem where there is a package and a truck at location *a*, and the package needs to be delivered to location *c*. The package can be loaded on the truck. The truck can drive between locations when it does not have a flat tire and it is verified that the package is inside the truck. The verification of the package may fail with probability 0.2. In addition, a tire may go flat during the drive with a probability of 0.4. A flat tire can be changed, if the truck is at a locations *a* or *d* where there is a spare tire. Finally, before unloading the package from the truck, it must be scanned.

Probabilistic planning is an active research field. Several different techniques are currently being used to solve probabilistic planning problems. We classify these techniques in four main groups that are described below.

Solving Markov Decision Processes

Markov Decision Processes (MDPs) (Puterman, 1994) is a popular family of models for analyzing probabilistic planning problems. An MDP is defined as a tuple $M = \langle S, A, T, C, G \rangle$ where S is a finite set of all possible states; A is a finite set of all actions an agent can take; $T : S \times A \mapsto S'$ is a transition probability function

```
(define (domain logistics)
 (:requirements :strips :typing :probabilistic-effects)
 (:types truck package location)
 (:predicates (connected ?a1 ?a2 - location)
              (at ?l - location ?pkg - package)
              (at ?l - location ?t - truck)
              (in ?pkg - package ?t - truck)
              (scanned ?pkg - package ?t - truck)
              (verified ?pkg - package ?t - truck)
              (flattire) (spare ?l - location) (hasspare))

 (:action drive
  :parameters (?t - truck ?from - location ?to - location ?pkg - package)
  :precondition (and (connected ?from ?to) (at ?from ?t) (in ?pkg ?t) (verified ?pkg ?t) (not (flattire)))
  :effect (and (not (at ?from ?t)) (at ?to ?t) (probabilistic 0.4 (flattire))))

 (:action get-tire
  :parameters (?l - location ?t - truck)
  :precondition (and (at ?l ?t) (spare ?l))
  :effect (and (not (spare ?l)) (hasspare)))

 (:action change
  :precondition (hasspare)
  :effect (and (not (hasspare)) (not (flattire))))

 (:action load
  :parameters (?pkg - package ?l - location ?t - truck)
  :precondition (and (at ?l ?t) (at ?l ?pkg))
  :effect (and (in ?pkg ?t) (not (at ?l ?pkg))))

 (:action verify
  :parameters (?pkg - package ?t - truck ?l - location)
  :precondition (and (in ?pkg ?t) (at ?l ?t))
  :effect (and (probability 0.8 (verified ?pkg ?t))))

 (:action scan
  :parameters (?pkg - package ?t - truck)
  :precondition (and (in ?pkg ?t) (verified ?pkg ?t))
  :effect (and (scanned ?pkg ?t)))

 (:action unload
  :parameters (?pkg - package ?t - truck ?l - location)
  :precondition (and (at ?l ?t) (scanned ?pkg ?t))
  :effect (and (not (in ?pkg ?t)) (at ?l ?pkg)))
```

```
(define (problem logistics-p01)
 (:domain logistics)
 (:objects a b c d e - location trk - truck pkg - package
 (:init (connected a b) (connected a d) (connected b c)
        (connected d e) (connected e c) (at a trk) (at a pkg)
        (spare a) (spare d) (spare e) (not (flattire)))
 (:goal (at c pkg)))
```

Figure 2.9: A PPDDL domain description on the Logistics-Tire Domain.

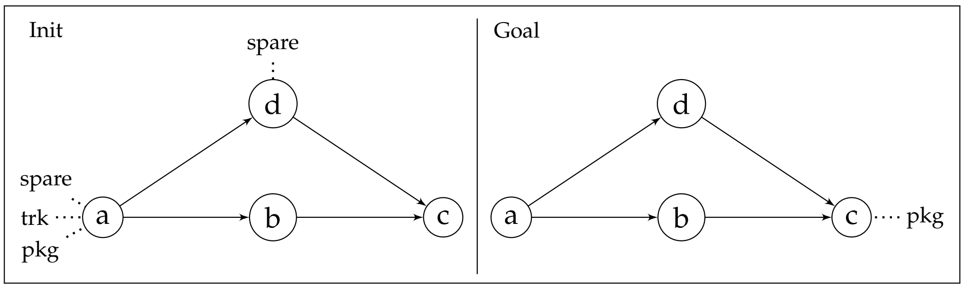


Figure 2.10: Initial and goal states for a simple probabilistic Logistics problem.

that returns a new state S' according to a probability distribution $P_T(S'|S, A)$; $C : S \times A \mapsto \mathbb{R}$ is a cost function that gives a finite numeric cost value $C(S, A)$, which represents the cost for taking action A in S; and G is the goal state. A solution to an MDP is a *policy* $\pi : S \mapsto A$, which is a mapping from states to actions. (In other words, a rule for action selection.) The goal is to find a policy that minimizes the expected cost. An optimal policy π^* can be found using a Dynaminc Programming (DP) algorithm such as Value Iteration (Bellman, 1957) or Policy Iteration (Howard, 1960).

Value Iteration is a DP method that computes an optimal policy and its value starting with an initial evaluation function V_0 and a parameter ε for detecting convergence to an optimal evaluation function. For each state $s \in S$, the algorithm first improves f by performing dynamic programming update as follows:

$$V_n(s) = \min_{a \in A(s)} \left\{ c_s(a) + \sum_{s' \in S} P_T(s'|s, a)\, V_{n-1}(s') \right\} \tag{2.13}$$

where $c_s(a)$ is the expected cost of taking action a in state s, $P_T(s'|s, a)$ is the probability that taking action a in state s results in state s', and $V_{n-1}(s')$ is the value of $V(s')$ at iteration $n - 1$.

This process is repeated until a convergence is reaches. In other words, when V_n is less than or equal to ε. Then, a policy can be extracted from f as follows:

$$\pi(s) = \operatorname*{argmin}_{a \in A(s)} \left\{ c_s(a) + \sum_{s' \in S} P_T(s'|s, a)\, V_{n-1}(s') \right\} \tag{2.14}$$

Policy Iteration is another DP method that computes an optimal policy starting with an initial policy π. It interleaves policy improvement with policy evaluation. It first computes the evaluation function $V^{\pi_{n-1}}$ for policy π by solving the following linear equation:

$$V_n^{\pi}(s) = c_s(\pi(s)) + \sum_{s' \in S} P_T(\pi(s'|s, a))\, V_{n-1}^{\pi}(s') \tag{2.15}$$

Then, for each state $i \in S$ it improves the policy as follows:

$$\pi'(s) = \operatorname*{argmin}_{a \in A(s)} \left\{ c_s(a) + \sum_{s' \in S} P_T(s'|s,a)\, V^{\pi}_{n-1}(j) \right\} \tag{2.16}$$

This process is repeated until convergence is reached. In order words, when π' is the same as π.

The disadvantage of Value Iteration and Policy Iteration is that they evaluate the entire state space. For this reason, Barto, Bradke, and Singh (1995) proposed the real-time dynamic algorithm (RTDP) that minimizes the search space of DP. The RTDP algorithm involves a sequence of *trials* to investigate choices of actions for each state. Each trial consists of (1) selecting the best action a that minimizes the cost of reaching the goal starting in s, (2) updating the value $V_n(s)$ using Equation 2.13, and (3) transitioning to a successor state s' under a. A significant drawback to RTDP is the lack of convergence detection. To overcome this problem, Bonet and Geffner (2003a) developed Labeled RTDP (LRTDP), which works in the same way as RTDP, but guarantees convergence by labeling visited states.

One feature that almost all approaches share is that all potential action outcomes in the plan are taken into account. That is, the plan includes a contingency branch for every possible outcome that may occur, given the actions in the plan. These approaches generate robust plans, but can be computationally expensive.

Using contingency planning

These approaches use planning graphs to compute estimates of probability that propositions can be achieved and actions can be performed. This information can be used to guide a probabilistic planner toward the most likely plan for achieving the goals. In particular, Blum and Langford (1999) extend the Graphplan representation to use it for probabilistic planning, and developed PGraphplan and TGraphplan. PGraphplan is a planner based on a forward-chaining search through a plan graph. It finds optimal (non-concurrent) contingency plans via dynamic programming using the information stored in the plan graph to prune the search. Since mutual exclusion relations computed in a plan graph are not helpful in a forward search, PGraphplan propagates information backwards through a plan graph to identify states from which the goal is probably unreachable. TGraphplan is a minor extension of the original Graphplan algorithm. It finds a non-optimal single path to the goal with highest probability. During the TGraphplan backwards search, the probability of the path is determined by recursively multiplying the probability of a step succeeding by the probability of the partial plan already explored. Paragraph (Little and Thiébaux, 2006) is another planner that also applies probabilistic plan graph, nogood learning, mutex reasoning, and goal regression techniques to probabilistic planning. It generates an optimal contingent plan by concatenating sub-trajectories generated by the plan graph. Bryce, Kambhampati, and Smith (2006) also use a plan graph to compute estimates of probability. However, they use an AO* algorithm in seeking a plan solution.

Translation into deterministic planning problems

These approaches translate the given planning problem into a classical planning problem that is then solved by a classical planner (Palacios and Geffner, 2009). Other approaches combine the translation-based technique with replanning (Albore et al., 2009), or replanning and sampling (Brafman and Shani, 2012), getting better results. It is important to note that these planners deal with uncertainty in the initial state and conditional effects. Also, for the most part these planners only deal with disjunctive uncertainty not probabilistic uncertainty.

Determinization-based techniques

Determinization consists of transforming the given probabilistic planning problem into a deterministic planning problem and using heuristic functions based on relaxed plans to guide a deterministic planner in the search for a deterministic plan. As a result, these planners generate a solution that is executed until the observed state differs from what is expected according to the next step in the solution. If this happens, replanning is performed to seek another plan solution. The insight behind this idea is that a deterministic planning problem can be solved more efficiently than the equivalent probabilistic MDP. The FF-Replan planner (Yoon et al., 2007) has been the pioneer in this paradigm, and it has proven very successful on many problems. It proposes two different methods of determinization:

- *single-outcome determinization*, which generates a deterministic action for each probabilistic action by selecting the outcome with the highest probability. To illustrate this, consider the probabilistic action (drive) in Figure 2.9. The most likely outcome is that the car successfully drive from one position to another. This determinization technique generates action (drive-1) for that outcome:

```
(:action drive-1
 :parameters (?t - truck ?from - location ?to - location ?pkg - package)
 :precondition (and (connected ?from ?to) (at ?from ?t) (in ?pkg ?t)
                    (verified ?pkg ?t) (not (flattire)))
 :effect (and (not (at ?from ?t)) (at ?to ?t)))
```

- *all-outcomes determinization*, which generates a deterministic action for each outcome of a probabilistic action. To illustrate this, consider again the probabilistic action (drive) in Figure 2.9. The probabilistic actions is determinized generating action (drive-1), where the car successfully drives from one location to another, and action (drive-2), where the car successfully drives from one location to another, but gets a flat tire:

```
(:action drive-1
 :parameters (?t - truck ?from - location ?to - location ?pkg - package)
 :precondition (and (connected ?from ?to) (at ?from ?t) (in ?pkg ?t)
                    (verified ?pkg ?t) (not (flattire)))
 :effect (and (not (at ?from ?t)) (at ?to ?t)))

(:action drive-2
 :parameters (?t - truck ?from - location ?to - location ?pkg - package)
 :precondition (and (connected ?from ?to) (at ?from ?t) (in ?pkg ?t)
                    (verified ?pkg ?t) (not (flattire)))
 :effect (and (not (at ?from ?t)) (at ?to ?t) (flattire)))
```

The first method may neglect important possible effects, while the second method considers every possible probabilistic outcome and may produce plans that rely on low probability outcomes.

Despite the success of FF-Replan, it does not make use of the probabilistic information in the domain description, which may result in frequent replanning. For this reason, most recent probabilistic planners follow the same line as FF-Replan, but deal with the probabilities of action outcomes (Kalyaman and Givan, 2008; Keyder and Geffner, 2008b; Wu et al., 2011). In particular, the work done by Jiménez, Coles, and Smith (2006) turns the probability information into costs. They make use of all-outcomes-determinization technique and for each new determinized action created, they transform the probability of the outcome into a cost equal to the negative logarithm of the probability. Then, they search for an optimal plan using a numeric deterministic planner that minimizes cost.

These approaches perform fast planning. However, they cannot anticipate possible deviations from the linear plan that is generated by the deterministic planner and, therefore, get stuck in dead-end states. FFH$^+$ (Yoon et al., 2008, 2010) is an improved version of FF-Replan that avoids that problem. It applies Hindsight Optimization, and on-line anticipatory strategy, for action selection. Determinization in hindsight samples actions according to their distribution (incorporates probability information). Then, it solves each determinized problem using an off-the-self deterministic planner at each step, and analyzes their solutions to select the action that leads to the best outcome. RFF (Teichteil-Königsbuch et al., 2010) is another determinized-based probabilistic planner that attempts to avoid dead-ends states in advance as FFH$^+$. RFF performs all-outcomes-determinization and then uses this relaxation to compute an off-line policy by generating consecutive execution paths leading to the goal from the initial states using the FF planner. The resulting policy has low probability of failing.

2.4 Summary

This chapter presented a review of several key concepts in classical planning, planning-based goal recognition, and probabilistic planning mentioned in subsequent chapters. In particular, it offered an in-depth review of the planning problem, the planning technique based on heuristic search, and heuristic estimators in the state-of-the-art due to the significant influence to the current work. Then, it introduced the problem of goal recognition, a new active area in automated planning, which is also addressed in this dissertation. It finally reviewed the probabilistic planning problem, and mentioned the most remarkable techniques used to solve probabilistic planning problems.

Chapter 3

Cost Estimates in a Plan Graph using Interaction

This chapter starts by describing the classical cost propagation process in a plan graph and its use for computing cost estimates. Then, it presents a novel technique to propagate cost through a plan graph that computes more accurate estimations of cost. Then, it proposes a family of heuristic functions based on the mentioned cost plan graph propagation technique. Finally, it shows an accuracy evaluation of the resulting family of heuristics and their application in classical planning.

3.1 Classical cost propagation in a plan graph

Simple propagation of cost estimates in a plan graph is a technique that has been used in a number of planning systems (Bonet et al., 1997; Nguyen et al., 2002; Do and Kambhampati, 2002). The computation of cost estimates starts from the initial conditions and works progressively forward through each successive layer of the plan graph. For level 0 it is assumed that the cost of the propositions at this level is zero. With this assumption, the propagation starts by computing the cost of the actions at level zero.

In general, the cost of performing an action a at level l with a set of preconditions $\mathcal{P}_a$ is equal to the cost of achieving its preconditions. This may be computed in two different ways:

1. Maximization: the cost of an action is equal to the cost of reaching its costliest precondition:
$$cost(a) = \max_{x_i \in \mathcal{P}_a} cost(x_i) \tag{3.1}$$

2. Summation: the cost of an action is equal to the cost of reaching all its preconditions:
$$cost(a) = \sum_{x_i \in \mathcal{P}_a} cost(x_i) \tag{3.2}$$

The first method assumes the possibility of dependence among the preconditions of an action. This is an admissible assumption since it underestimates the cost of an action. On the other hand, the second method assumes independence among all preconditions of an action. Although this heuristic is non-admissible, it is typically more informative and compelling in practice. For the purpose of this thesis, we make use of the Summation technique as in Equation 3.2 for estimating the cost of an action.

The cost of achieving a proposition x at level l, achieved by the actions $\mathcal{A}_x$ at the preceding level is the minimum cost among all $a \in \mathcal{A}_x$. It is defined as:

$$cost(x) = \min_{a \in \mathcal{A}_x} \{cost(a) + \mathcal{C}ost(a)\} \tag{3.3}$$

where $\mathcal{C}ost(a)$ is the cost of applying action a, and $cost(a)$ is given by Equations 3.1 and 3.2.

Figure 3.1 shows the first layers of the plan graph for the simple Logistics problem shown in Figure 2.1. The numbers above propositions and actions are the costs associated with each one computed during the cost propagation process. The highlighted costs are the ones computed below as an example. In particular, at level 0 the cost for action (load pkg a trk) is zero since its preconditions are true in the initial state, so the cost of its effect (in trk pkg) is one at level 1:

$$cost(\text{load pkg a trk}) = cost(\text{at a pkg}) + cost(\text{at a trk}) = 0 + 0 = 0$$

$$cost(\text{in trk pkg}) = cost(\text{load pkg a trk}) + \mathcal{C}ost(\text{load pkg a trk}) = 0 + 1 = 1$$

In the next actions' layer, the cost for (verify pkg a trk) is 1, the sum of the cost of its preconditions, so its effect (verified pkg trk) has a cost of 2:

$$\begin{aligned} cost(\text{verify pkg a trk}) &= cost(\text{at a trk}) + cost(\text{in trk pkg}) = 0 + 1 = 1 \\ cost(\text{verified pkg trk}) &= cost(\text{verify pkg a trk}) + \mathcal{C}ost(\text{verify pkg a trk}) \\ &= 1 + 1 = 2 \end{aligned}$$

Figure 3.2 shows the continuation of the plan graph in Figure 3.1 for the simple Logistics problem. At level 2 we know by intuition that the cost of actions

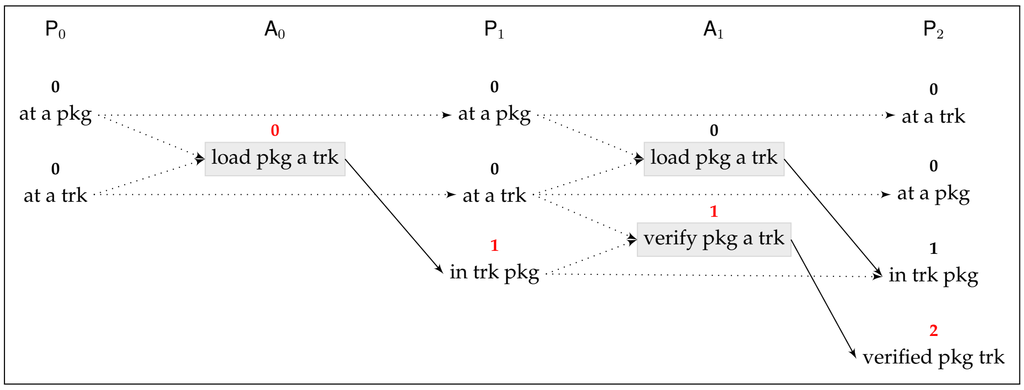


Figure 3.1: Example of a classical cost propagation in a plan graph.

(scan pkg trk) and (drive trk a b) is two. However, the sum of each action's preconditions gives a cost of three:

$$cost(\text{scan pkg trk}) = cost(\text{in trk pkg}) + cost(\text{verified pkg trk}) = 1 + 2 = 3$$

$$cost(\text{drive trk a b}) = \left\{ \begin{array}{l} cost(\text{at a trk}) + cost(\text{in trk pkg}) + \\ cost(\text{verified pkg trk}) \end{array} \right\} = 0 + 1 + 2 = 3$$

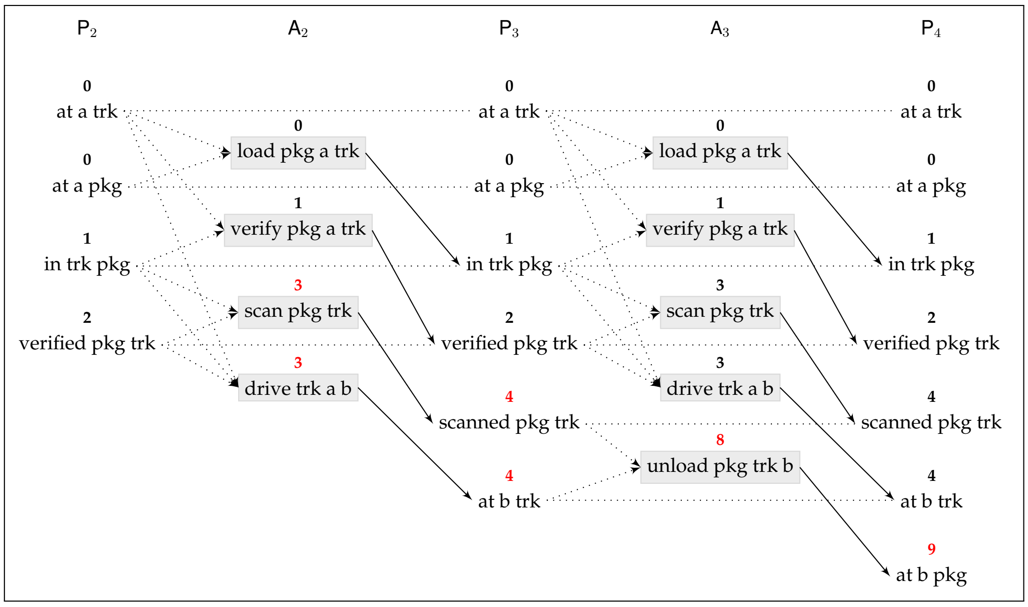


Figure 3.2: Example of a classical cost propagation in a plan graph (continued).

The problem here is that the proposition (verified pkg trk) is not independent of the proposition (in trk pkg) since the action (verify pkg a trk) has (in trk pkg) as a precondition. Therefore, there is synergy between the two propositions that is not considered in simple cost propagation using the Summation heuristic.

Continuing with the simple cost propagation for the current example, the cost of propositions (scanned pkg trk) and (at b pkg) at level 3 is:

$$cost(\text{scanned pkg trk}) = cost(\text{scan pkg trk}) + \mathcal{C}ost(\text{scan pkg trk}) = 3 + 1 = 4$$

$$cost(\text{at b pkg}) = cost(\text{drive trk a b}) + \mathcal{C}ost(\text{drive trk a b}) = 3 + 1 = 4$$

This propagation results in (unload pkg trk b) having the cost:

$$cost(\text{unload pkg trk b}) = \quad cost(\text{scanned pkg trk}) + cost(\text{at b pkg}) = 4 + 4 = 8$$

Taking the above calculations into consideration, a plan graph is built in the same way that an ordinary plan graph is created. The construction process finishes when two consecutive propositions layers are identical and there is *quiescence* in cost. Quiescence is reached when the cost for each proposition and action in the plan graph no longer changes. On completion, each possible goal proposition has an estimated cost of been achieved.

In Section 2.1, we show that the optimal cost for the current example is 5. Using the simple cost propagation approach, the goal proposition (at b pkg) has a cost of 9 when the propagation finishes. That is:

$$cost(\text{at b pkg}) = cost(\text{unload pkg trk b}) + \mathcal{C}ost(\text{unload pkg trk b}) = 8 + 1 = 9$$

This cost overestimates the optimal cost because of the assumption of independence among all the preconditions of an action. In the next section, we present a method that estimates the degree of dependence between pairs of propositions and pairs of actions in a plan graph, and thus computes more accurate estimates of cost.

3.2 Cost and Interaction propagation in a plan graph

The technique for cost propagation in a plan graph described in the previous section starts from the initial conditions and works progressively forward through each successive layer of the plan graph. The cost of performing an action is equal to the cost of achieving its preconditions. The cost of achieving a proposition is the minimum cost among all the actions that produce the proposition. This method assumes independence among the preconditions of an action. Therefore, the cost can be an underestimate if some of the preconditions interfere with each other, and can be an overestimate if some of the preconditions are achieved by the same action. For this reason, the technique presented in this chapter for cost propagation introduces the notion of *Interaction* (I), which captures the degree of dependence (positive or negative) between pairs of propositions and pairs of actions in the plan graph (Bryce and Smith, 2006).

3.2.1 Interaction

Interaction is a value that represents how more or less costly it is that two propositions or actions are established together instead of independently. This concept is a generalization of the mutual exclusion concept used in classical plan graphs. Formally, the optimal Interaction, I^*, considers n-ary Interaction relationships among propositions and among actions in the plan graph, and it is defined as:

$$I^*(p_0, ..., p_n) = cost^*(p_0 \wedge p_1 \wedge ... \wedge p_n) - \{cost^*(p_0) + ... + cost^*(p_n)\} \tag{3.4}$$

where the term $cost^*(p_0 \wedge ... \wedge p_n)$ is the minimum cost among all the possible plans that achieve all the members in the set. Computing I^* would be computationally prohibitive. As a result, we limit the calculation of these values to pairs of propositions and pairs of actions in each level of a plan graph. In other words, binary Interaction:

$$I^*(p, q) = cost^*(p \wedge q) - \{cost^*(p) + cost^*(q)\} \tag{3.5}$$

It has the following features:

$$I^*(p, q) \text{ is } \begin{cases} < 0 & \text{if } p \text{ and } q \text{ are } \textit{synergistic} \\ = 0 & \text{if } p \text{ and } q \text{ are } \textit{independent} \\ > 0 & \text{if } p \text{ and } q \textit{ interfere} \\ \infty & \text{if } p \text{ and } q \text{ are } \textit{mutually exclusive} \end{cases}$$

I provides information about the degree of interference or synergy between pairs of propositions and pairs of actions in a plan graph. When $0 < I(p, q) < \infty$ it means that there is some interference between the best plans for achieving p and q, so it is harder (more costly) to achieve them both than to achieve them independently. In the extreme case, $I = \infty$, the propositions or actions are mutually exclusive. Similarly, when $I(p, q) < 0$ the two elements are synergistic, which means that the cost of establishing both p and q is less than the sum of the costs for establishing the two independently. However, this cost cannot be less than the cost of establishing the most difficult of p and q. As a result $I(p, q)$ is bounded below by $-\min\{cost(p), cost(q)\}$, where $cost$ is the estimated cost propagated in the plan graph. This low bound will be discussed in more detail in Section 3.2.4.

The computation of cost and Interaction information begins at level zero of the plan graph and sequentially proceeds to higher levels. For level zero it is assumed that (1) the cost of each proposition at this level is 0, and (2) the Interaction between each pair of propositions at this level is 0, which means independence between propositions. Neither of these assumptions are essential, but they are adopted in this work for simplicity.

3.2.2 Computing action cost and Interaction

The cost and Interaction information of a propositions layer at a given level of the plan graph is used to compute the cost and the Interaction information for the subsequent actions' layer. Considering an action a at level l with a set of preconditions $\mathcal{P}_a$, the estimation of how costly it is to execute an action is the cost of achieving all its preconditions plus the Interaction between all pairs of propositions:

$$cost(a) = cost(\mathcal{P}_a) \approx \sum_{x_i \in \mathcal{P}_a} cost(x_i) \; + \sum_{\substack{(x_i, x_j) \in \mathcal{P}_a \\ j > i}} I(x_i, x_j) \tag{3.6}$$

For instance, consider levels 2 and 3 of the plan graph shown in Figure 3.3 for the simple Logistics problem. At level 2 propositions (in trk pkg) and (verified pkg trk) have a cost of 1 and 2, respectively, and an Interaction value of -1. This means that, the cost for actions (scan pkg trk) and (drive trk a b) is two (instead of three for simple cost propagation):

$$cost(\text{scan pkg trk}) = \left\{ \begin{array}{l} cost(\text{at a trk}) + cost(\text{in trk pkg}) + \\ I(\text{at a trk, verified pkg trk}) \end{array} \right\} = 1 + 2 - 1 = 2$$

$$cost(\text{drive trk a b}) = \left\{ \begin{array}{l} cost(\text{at a trk}) + \\ cost(\text{in trk pkg}) + \\ cost(\text{verified pkg trk}) + \\ I(\text{at a trk, in trk pkg}) + \\ I(\text{at a trk, verified pkg trk}) + \\ I(\text{in trk pkg, verified pkg trk}) \end{array} \right\} = 0 + 1 + 2 + 0 + 0 - 1 = 2$$

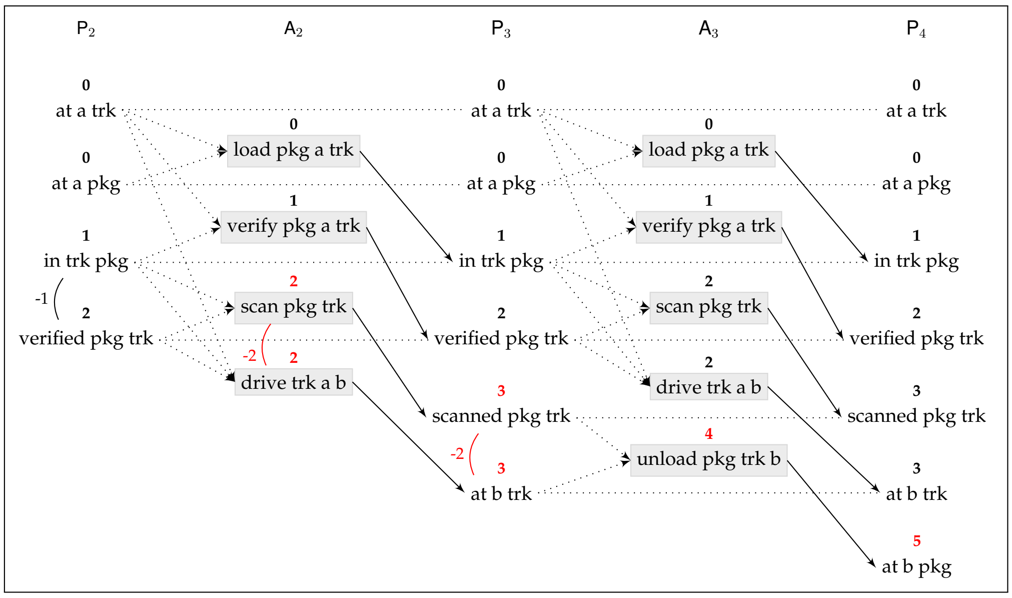


Figure 3.3: Example of cost and Interaction propagation in a plan graph.

The next step is to compute the Interaction between actions. The Interaction between two actions a and b at level l, with sets of preconditions $\mathcal{P}_a$ and $\mathcal{P}_b$ respectively, is defined as:

$$I(a,\, b) \;\; is \;\; \begin{cases} \infty & \text{if } a \text{ and } b \text{ are mutex by inconsistent effects or interference} \\ \\ cost(a \wedge b) - cost(a) - cost(b) & \text{otherwise} \end{cases} \tag{3.7}$$

If the actions are mutex by inconsistent effects or interference, then the Interaction is infinity. Otherwise, the cost of the conjunction of a and b, $cost(a \wedge b)$, is $cost(\mathcal{P}_a \cup \mathcal{P}_b)$, i.e., the cost of the union of their preconditions. This is approximated as in Equation 3.6 by:

$$cost(\mathcal{P}_a \cup \mathcal{P}_b) \approx \sum_{x_i \in \mathcal{P}_a \cup \mathcal{P}_b} cost(x_i) \; + \sum_{\substack{(x_i,x_j) \in \mathcal{P}_a \cup \mathcal{P}_b \\ j>i}} I(x_i, x_j) \tag{3.8}$$

where the cost of performing two actions a and b will be the sum of the cost of achieving all their preconditions plus the Interaction between all pairs of preconditions.

The Interaction above can be simplify. To illustrate that, consider a simple problem with operator A, which preconditions are x, y, and t, and operator B, which preconditions are x, y, and z. Assuming that A and B are not mutually exclusive, the Interaction between actions A and B will be:

$$I(A,B) = cost(\mathcal{P}_A \cup \mathcal{P}_B) - cost(A) - cost(B) \tag{3.9}$$

where:

$$\begin{aligned} cost(\mathcal{P}_A \cup \mathcal{P}_B) &= cost(t) + cost(x) + cost(y) + cost(z) + I(t,x) + I(t,y) + \\ &\quad I(t,z) + I(x,y) + I(x,z) + I(y,z) \\ \\ cost(A) &= cost(t) + cost(x) + cost(y) + I(t,x) + I(t,y) + I(x,y) \\ \\ cost(B) &= cost(x) + cost(y) + cost(z) + I(x,y) + I(x,z) + I(y,z) \end{aligned}$$

Therefore, the Interaction between A and B can be rewritten as:

$$\begin{aligned} I(A,B) &= \cancel{cost(t)} + \cancel{cost(x)} + \cancel{cost(y)} + \cancel{cost(z)} + \cancel{I(t,x)} + \cancel{I(t,y)} + \\ &\quad I(t,z) + \cancel{I(x,y)} + \cancel{I(x,z)} + \cancel{I(y,z)} - \cancel{cost(t)} - \cancel{cost(x)} - \\ &\quad \cancel{cost(y)} - \cancel{I(t,x)} - \cancel{I(t,y)} - \cancel{I(x,y)} - cost(x) - cost(y) - \\ &\quad \cancel{cost(z)} - I(x,y) - \cancel{I(x,z)} - \cancel{I(y,z)} \\ &= I(t,z) - cost(x) - cost(y) - I(x,y) \end{aligned}$$

In general:

$$I(a,b) \approx \sum_{\substack{x_i \in \mathcal{P}_a - \mathcal{P}_b \\ x_j \in \mathcal{P}_b - \mathcal{P}_a}} I(x_i,x_j) - \sum_{x_i \in \mathcal{P}_a \cap \mathcal{P}_b} cost(x_i) + \sum_{\substack{(x_i,x_j) \in \mathcal{P}_a \cap \mathcal{P}_b \\ j > i}} I(x_i,x_j)$$

where the first term gets the Interaction between unique preconditions for each action, the second term removes duplicated preconditions, and the third term gets the Interaction between common pairs of preconditions.

For the example in Figure 3.3, the Interaction between actions (scan pkg trk) and (drive trk a b) would be:

$$\begin{aligned} I(\text{scan pkg trk, drive trk a b}) &\approx \left\{ \begin{array}{l} cost(\text{in trk pkg}) + \\ cost(\text{verified pkg trk}) + \\ I(\text{in trk pkg, verified pkg trk}) \end{array} \right\} \\ &= -(1+2-1) = -2 \end{aligned}$$

The fact that I(scan pkg trk, drive trk a b) $= -2$ means that there is some degree of synergy between actions (scan pkg trk) and (drive trk a b). This synergy comes from the fact that the two actions have two common preconditions, (in trk pkg) and (verified pkg trk).

3.2.3 Computing proposition cost and Interaction

The next step consists of calculating the cost of the propositions at the next level. This cost is calculated in the same way as in Equation 3.3. In this calculation, all the possible actions at the previous level that achieve each proposition need to be taken into account. We make the usual optimistic assumption that the least expensive action can be used. Therefore, the cost of a proposition is the minimum cost among all the actions that produce the proposition. Formally, for a proposition x at level l achieved by actions $\mathcal{A}_x$ at the preceding level, the cost is calculated as:

$$cost(x) = \min_{a \in \mathcal{A}_x} \{cost(a) + \mathcal{C}ost(a)\} \tag{3.10}$$

In the current example, the cost of propositions (scanned pkg trk) and (at b pkg) at level 3 is:

$cost$(scanned pkg trk) $=$ $cost$(scan pkg trk) $+$ $\mathcal{C}ost$(scan pkg trk) $= 2 + 1 = 3$

$cost$(at b pkg) $=$ $cost$(drive trk a b) $+$ $\mathcal{C}ost$(drive trk a b) $= 2 + 1 = 3$

Finally, we compute the Interaction between propositions. In order to calculate the Interaction between two propositions x and y at a level l, we need to consider all the possible ways to achieve both propositions. In other words, all the actions that achieve the pair of propositions and the Interaction between them. Suppose that $\mathcal{A}_x$ and $\mathcal{A}_y$ are the sets of actions that achieve propositions x and y respectively at level l. The Interaction between x and y is then:

$$\begin{aligned}
I^*(x,y) &= cost^*(x \wedge y) - cost^*(x) - cost^*(y) \\
&= \min_{\substack{a \in \mathcal{A}_x \\ b \in \mathcal{A}_y}} \left\{ cost(a \wedge b) \right\} - cost^*(x) - cost^*(y) \\
&\approx \min \left\{ \begin{array}{l} \displaystyle\min_{a \in \mathcal{A}_x \cap \mathcal{A}_y} cost(a) + \mathcal{C}ost(a) - cost(x) - cost(y) \\ \\ \displaystyle\min_{\substack{a \in \mathcal{A}_x - \mathcal{A}_y \\ b \in \mathcal{A}_y - \mathcal{A}_x}} cost(a) + \mathcal{C}ost(a) + cost(b) + \mathcal{C}ost(b) + I(a,b) \end{array} \right\} \\
&\quad -cost(x) - cost(y) \qquad (3.11)
\end{aligned}$$

where actions are divided into those that accomplish both propositions x and y, and those that accomplish only one of them.

Returning to the current example, consider the calculation of the Interaction between (scanned pkg trk) and (at b pkg) at level 3 where the only way to achieve them both is by performing (scan pkg trk) and (drive trk a b). In this case, the Interaction is calculated as follows:

$$\begin{aligned}
I(\text{scanned pkg trk, at b pkg}) &\approx \left\{ \begin{array}{l} cost(\text{scan pkg trk}) + \mathcal{C}ost(\text{scan pkg trk}) + \\ cost(\text{drive trk a b}) + \mathcal{C}ost(\text{drive trk a b}) + \\ I(\text{scan pkg trk, drive trk a b}) \end{array} \right\} \\
&\quad -cost(\text{scanned pkg trk}) - cost(\text{at b pkg}) \\
&= \{2+1+2+1-2\} - 3 - 3 = -2
\end{aligned}$$

The fact that $I(\text{scanned pkg trk, at b pkg}) = -2$ means that there is synergy between having the package scanned and having it at b. This synergy comes from the fact that the actions (scan pkg trk) and (drive trk a b) have two common preconditions (in trk pkg) and (verified pkg trk).

Using Equations 3.6, 3.7, 3.10, and 3.11 we can build a plan graph and propagate cost in the same way as for simple cost propagation. The construction process finishes when two consecutive propositions layers are identical and there is quiescence in cost and Interaction for all propositions and actions in the plan graph. On completion, each possible goal proposition has an estimated cost of being achieved, and there is an Interaction estimation between each pair of goal propositions.

In the simple Logistics example, if we keep propagating cost and Interaction information, we get that the cost of (unload pkg trk b) at level 3 is four as expected:

$$cost(\text{unload pkg trk b}) = \left\{ \begin{array}{l} cost(\text{scanned pkg trk}) + \\ cost(\text{at b pkg}) + \\ I(\text{scanned pkg trk, at b pkg}) \end{array} \right\} = 3 + 3 - 2 = 4$$

Therefore, the goal proposition (at b pkg) has a cost of 5 when the propagation finishes. That is:

$$cost(\text{at b pkg}) = cost(\text{unload pkg trk b}) + \mathcal{C}ost(\text{unload pkg trk b}) = 4 + 1 = 5$$

which is the same as the optimal cost shown in Section 2.1.

3.2.4 Lower bound on cost and Interaction

Because the costs in Equations 3.6 and 3.10 are estimated based on binary Interaction, the resulting calculations can sometimes underestimate cost and Interaction. Therefore, we can improve estimates by considering lower bounds on *action cost*, *action Interaction*, and *propositions Interaction*. The reason for these lower bounds are shown in next subsections.

Lower bounds on action cost

The cost of an action a should be at least the maximum cost among all its preconditions, if they are synergistic among all of them. That is:

$$cost(a) \;\geq\; \max_{x \in \mathcal{P}_a} cost(x)$$

To illustrate this, consider a simple problem with the following two unit-cost operators:

$$\begin{array}{rcl} A & : & p \rightarrow x,\, y,\, z \\ B & : & x,\, y,\, z \rightarrow t \end{array} \qquad (3.12)$$

Figure 3.4 shows a partial planning graph for the operators described in Equation 3.12. The number next to each proposition and action refers to the estimated cost computed for each proposition and action during the cost propagation process. The number next to red edges refers to the Interaction value between the pair of propositions that each edge connects. The example in the figure starts at

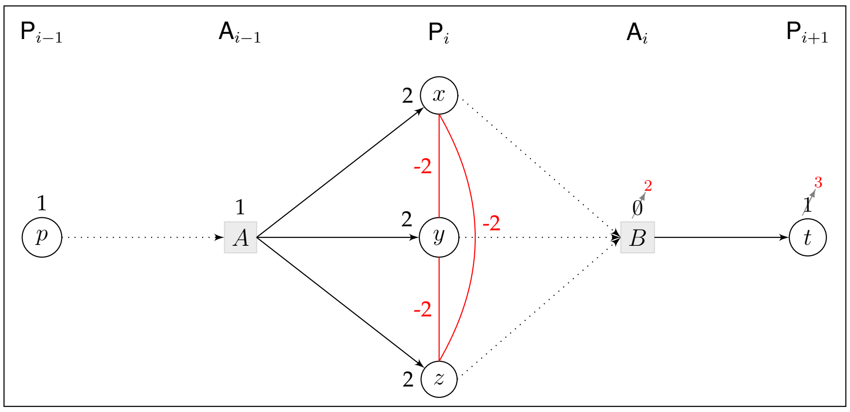


Figure 3.4: Example of lower bounds on action cost during the propagation of cost and Interaction information in a plan graph.

level $i-1$ with propositions layer P_{i-1}, which contains a single proposition p with $cost = 1$. Therefore, the estimated cost of action A is:

$$cost(A) = cost(p) = 1$$

Next step is to compute the estimated costs of propositions at level P_i, which have the following values:

$$\begin{aligned}
cost(x) &= cost(A) + \mathcal{C}ost(A) &= 1+1 &= 2\\
cost(y) &= cost(A) + \mathcal{C}ost(A) &= 1+1 &= 2\\
cost(z) &= cost(A) + \mathcal{C}ost(A) &= 1+1 &= 2
\end{aligned}$$

The computation of the estimated cost of a proposition is followed by the Interaction computation between pairs of them. The Interactions values are:

$$\begin{aligned}
I(x,y) &= cost(A) + \mathcal{C}ost(A) - cost(x) - cost(y) &= 1+1-2-2 &= -2\\
I(x,z) &= cost(A) + \mathcal{C}ost(A) - cost(x) - cost(z) &= 1+1-2-2 &= -2\\
I(z,y) &= cost(A) + \mathcal{C}ost(A) - cost(y) - cost(z) &= 1+1-2-2 &= -2
\end{aligned}$$

Next step is to compute the cost of operator B at layer A_i. B has x, y, and z as preconditions. However, the estimated cost of action B is the sum of the estimated cost of each of its preconditions combined with the Interaction between, which is zero:

$$\begin{aligned}
cost(B) &= cost(x) + cost(y) + cost(z) + I(x,y) + I(x,z) + I(y,z)\\
&= 2+2+2-2-2-2 = 0
\end{aligned}$$

In this case, the binary Interaction information is underestimating the cost of B, which should be at least the maximum cost among all its preconditions (if they are synergistic among all of them). The optimal cost of B should consider the Interaction among x, y, and z, which is:

$$\begin{aligned} I^*(x,y,z) &= cost^*(A) + \mathcal{C}ost(A) - cost^*(x) - cost^*(y) - cost^*(z) \\ &= 1 + 1 - 2 - 2 - 2 = -4 \end{aligned}$$

Therefore, the optimal cost of B is:

$$cost^*(B) \;=\; cost^*(x) + cost^*(y) + cost^*(z) + I^*(x,y,z) \;=\; 2 + 2 + 2 - 4 = 2$$

In this case, by enforcing the lower bounds from Equation 3.2.4 we get that the probability of B is at least 2.

Lower bounds on action and proposition Interaction

The approximate binary Interaction between two elements should always be bounded below by $-\min\{cost(x),\, cost(y)\}$. That is:

$$\begin{aligned} I^*(x,y) &= cost^*(x \wedge y) - \{cost^*(x) + cost^*(y)\} \\ &= \max\{cost^*(x), cost^*(y)\} - cost^*(x) - cost^*(y) \end{aligned}$$

If $\max\{cost(x),\, cost(y)\} \;=\; cost(x)$, then:

$$I(x,y) \;=\cancel{cost(x)} - \cancel{cost(x)} - cost(y) \;=\; -cost(y)$$

If $\max\{cost(x),\, cost(y)\} \;=\; cost(y)$, then:

$$I(x,y) \;=\cancel{cost(y)} - cost(x) - \cancel{cost(y)} \;=\; -cost(x)$$

Therefore:

$$I(x,y) \;\geq\; -\min\{cost(x),\, cost(y)\} \tag{3.13}$$

Considering these bounds during the Interaction computation result in more accurate cost estimates. To illustrate this, consider a simple problem with the following two unit-cost operators:

$$\begin{aligned} A &: \; x \rightarrow p \\ B &: \; y,\, z \rightarrow q \end{aligned} \tag{3.14}$$

Figure 3.5 shows a partial planning graph for the operators described in Equation 3.14. The number above the propositions and actions refers to the estimated cost computed for each proposition and action during the cost propagation process. The number next to the edges refers to the Interaction value between the pair of propositions that each edge connects. The example in the figure starts at level i with propositions layer P_i, which contains propositions x, y, and z, with costs 1.28, 1.6, and 1.28 respectively, and $I(x, y) = I(x, z) = I(y, z) = -1.28$. The estimated cost of actions A and B is:

$$\begin{aligned} cost(A) &= cost(x) = 1.28 \\ cost(B) &= cost(y) + cost(z) + I(y, z) = 1.6 + 1.28 - 1.28 = 1.6 \end{aligned}$$

The next step is to compute the estimated Interaction of actions at level A_i. Using Equation 3.10, we compute the Interaction between actions A and B, which is:

$$I(A, B) = I(x, y) + I(x, w) = -1.28 - 1.28 = -2.56$$

In this case, according to the bounds in Equation 3.13 we know that the Interaction between A and B should not be lower than:

$$-\min\{cost(A), cost(B)\} = -\min\{1.28, 1.6\} = -1.28$$

Therefore, considering these bounds during the Interaction computation results in more accurate cost estimates.

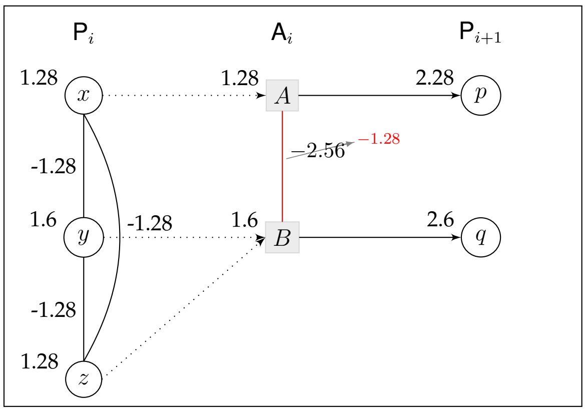


Figure 3.5: Example of lower bounds on action Interaction during the propagation of cost and Interaction information in a plan graph.

3.3 A heuristic estimator based on Interactions

The plan graph described in the previous section includes, for each proposition and action in the plan graph, an approximate cost of being achieved. These cost estimates may be used as heuristic estimators. The next subsections describe two different methods that make use of this information to compute heuristic functions. The first method computes the estimated cost using the information given in the plan graph, while the second one builds a cost-relaxed plan where the propagated cost information is taken into account.

3.3.1 The h^I heuristic

The h^I heuristic (E-Martín et al., 2015b) is based directly on the cost and Interaction information computed in the plan graph. It defines the estimated cost of achieving a (possibly conjunctive) goal $G = \{g_1, ..., g_n\}$ as:

$$h^I = Cost(G) \approx \sum_{g \in G} cost(g) + \sum_{\substack{(g_i, g_j) \in G \\ i < j}} I(g_i, g_j) \tag{3.15}$$

The Interaction information helps to compute more accurate estimates of cost when subgoals interfere or are synergistic with each other. However, the fact that the Interaction computation is binary makes the heuristic h^I non-admissible. Therefore, the estimated cost is an approximation of the optimal cost. Essentially, when all the preconditions of each action are independent of each other, the heuristic h^I reduces to the heuristic h^+. Otherwise, h^I can be greater or less than h^+, if there is ternary Interaction not captured.

3.3.2 The h^I_{rp} heuristic

The h^I_{rp} (E-Martín et al., 2015b) heuristic is based on computing a relaxed plan using the cost information in the plan graph. It uses the algorithm described in Figure 2.5. However, the more sophisticated strategy is to make use of cost and Interaction information in the plan graph when selecting actions for the relaxed plan, instead of the number of action's preconditions. In particular, to achieve a particular subgoal at a level l, the relaxed plan extraction process chooses the action that minimizes the cost of achieving each individual goal at level l. The sum of the costs of the actions in the relaxed plan π provides an estimate of cost for the goal. Therefore, the heuristic is defined as:

$$h^I_{rp} = Cost(\pi) \approx \sum_{a \in \pi} cost(a) \tag{3.16}$$

Like the h^I heuristic, the h^I_{rp} heuristic is non-admissible.

3.4 Difference between h^2 and h^I

Intuitively, h^I might seem similar to an additive version of the h^2 heuristic (Haslum and Geffner, 2000), where the $\max$ operator is replaced by the summation operator. However, there are some important subtleties in defining such a heuristic. The h^2 and h^I heuristics consider pairs of elements to approximate the cost of achieving a goal state s_g from any state s_n. The h^2 heuristic is admissible since it defines costs of sets by the maximum over the elements in the set, while the h^I heuristic is non-admissible since it defines costs of sets by the sum over the elements in the set and the binary Interaction between them. As previously described, the h^2 heuristic defines the cost of an action a with set of preconditions $\mathcal{P}_a$ as:

$$h^2(a) \quad = \quad \min \begin{cases} h^2(x) & \text{if } |\mathcal{P}_a| = 1 \text{ and } x \in \mathcal{P}_a \\ \\ h^2(x \wedge y) & \text{if } |\mathcal{P}_a| = 2 \text{ and } (x,y) \in \mathcal{P}_a \\ \\ \max\limits_{x,y \in \mathcal{P}_a} \{h^2(x \wedge y)\} & \text{otherwise} \end{cases}$$

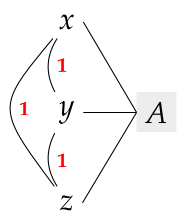
where the cost of the conjunction of two propositions x and y is:

$$h^2(x \wedge y) \quad = \quad \min \begin{cases} \min\limits_{a \in O(x,y)} \; \mathcal{C}ost(a) \; + \; h^2(\mathcal{P}_a) \\ \\ \min\limits_{a \in O(x|y)} \; \mathcal{C}ost(a) \; + \; h^2(\mathcal{P}_a \cup \{y\}) \\ \\ \min\limits_{a \in O(y|x)} \; \mathcal{C}ost(a) \; + \; h^2(\mathcal{P}_a \cup \{x\}) \end{cases}$$

The simplest additive alternative of the h^2 heuristic, which we call h^{2+}, is to simply replace the $\max$ operator by the summation operator. That is:

$$h^{2+}(a) = \sum_{(x,y) \in \mathcal{P}_a} h^{2+}(x \wedge y) \quad \text{when } |\mathcal{P}_a| > 2$$

However, this technique may significantly overestimate costs. To illustrate this, consider the operator A in Figure 3.6, which has three preconditions x, y, and z whose individual costs are 0, 1, and 1, respectively, and pairs costs are $cost(x, y) = 1$, $cost(x, z) = 1$, and $cost(y, z) = 1$. Given the individual costs we know that the cost of action a should be at least 1, and at most 2. However, the cost of a is 3, since it is the summation of the pair costs between all preconditions, which over counts the cost of the propositions.



$$\Longrightarrow\ h^{2+}(A) = h^{2+}(\mathcal{P}_A) = h^{2+}(x, y) + h^{2+}(x, z) + h^{2+}(y, z) = 3$$

Figure 3.6: An example of the over-counting costs using the simplest additive alternative to the h^2 heuristic.

On the other hand, the h^I heuristic defines the cost of an action a with a set of preconditions $\mathcal{P}_a$ as:

$$\begin{aligned}
\text{when } n = |\mathcal{P}_a| &= 1:\\
cost(a) &= cost(x)\\
\text{when } n = |\mathcal{P}_a| &= 2:\\
cost(a) &= cost(x) + cost(y) + I(x, y)\\
&= \cancel{cost(x)} + \cancel{cost(y)} + cost(x \wedge y) - \cancel{cost(x)} - \cancel{cost(y)}\\
&= cost(x \wedge y)\\
\text{when } n = |\mathcal{P}_a| &= 3:\\
cost(a) &= cost(x) + cost(y) + cost(z) + I(x, y) + I(x, z) + I(y, z)\\
&= \cancel{cost(x)} + \cancel{cost(y)} + \cancel{cost(z)} + cost(x \wedge y) - \cancel{cost(x)} - \cancel{cost(y)} +\\
&\quad cost(x \wedge z) - cost(x) - \cancel{cost(z)} + cost(y \wedge z) - cost(y) - cost(z)\\
&= cost(x \wedge y) + cost(x \wedge z) + cost(y \wedge z) - cost(x) - cost(y) - cost(z)\\
\text{when } n = |\mathcal{P}_a| &= 4:\\
cost(a) &= cost(x) + cost(y) + cost(z) + cost(w) + I(x, y) + I(x, z) + I(x, w)\\
&= \cancel{cost(x)} + \cancel{cost(y)} + \cancel{cost(z)} + \cancel{cost(w)} + cost(x \wedge y) - \cancel{cost(x)} -\\
&\quad \cancel{cost(y)} + cost(x \wedge z) - cost(x) - \cancel{cost(z)} + cost(x \wedge w) - cost(x) -\\
&\quad \cancel{cost(w)} + cost(y \wedge z) - cost(y) - cost(z) + cost(y \wedge w) - cost(y) -\\
&\quad cost(w) + cost(z \wedge w) - cost(z) - cost(w)\\
&= cost(x \wedge y) + cost(x \wedge z) + cost(x \wedge w) + cost(y \wedge z) + cost(y \wedge w) +\\
&\quad cost(z \wedge w) - 2cost(x) - 2cost(y) - 2cost(z) - 2cost(w)
\end{aligned}$$

In general:

$$cost(a) = \sum_{\substack{(x,y)\in\mathcal{P}_a \\ x\neq y}} cost(x \wedge y) \;-\; \sum_{x\in\mathcal{P}_a} (n-2)cost(x) \tag{3.17}$$

where the cost of the conjunction of two propositions x and y produced by actions a and b respectively is:

$$\begin{aligned}
cost(x \wedge y) &= \min_{\substack{a\in O(x|y) \\ b\in O(y|x)}} \{cost(a) + \mathcal{C}ost(a) + cost(b) + \mathcal{C}ost(b) + I(a,b)\} \\
&= \min_{\substack{a\in O(x|y) \\ b\in O(y|x)}} \left\{ \begin{array}{l} \cancel{cost(a)} + \mathcal{C}ost(a) + \cancel{cost(b)} + \mathcal{C}ost(b) + cost(a \wedge b) - \\ \cancel{cost(a)} - \cancel{cost(b)} \end{array} \right\} \\
&= \mathcal{C}ost(a) + \mathcal{C}ost(b) + cost(a \wedge b) \\
&= \mathcal{C}ost(a) + \mathcal{C}ost(b) + cost(\mathcal{P}_a \cup \mathcal{P}_b)
\end{aligned}$$

The Interaction information is the main key to prevent over-counting cost while using an additive heuristic. The second term in Equation 3.17 subtracts propositions' individual costs as many times as is needed to not duplicate them. This feature is given by the use of Interaction during cost propagation in a plan graph and underlines our novel heuristic. The h^{2+} heuristic that we just defined is therefore not a sensible candidate for an additive version of h^2 because it double-counts the cost of the preconditions. In the next subsections, we propose three alternatives that overcome this problem.

3.4.1 Ordered Greedy Covering

The Ordered Greedy Covering alternative, namely h^{2+}_g, defines the cost of an action a with the set of preconditions $\mathcal{P}_a$ as the summation of pairs $(x,y) \in \mathcal{P}_a$ where pairs (x,y) are chosen according to the order of preconditions $\mathcal{P}_a$, until all the elements in $\mathcal{P}_a$ are covered. That is:

$$h^{2+}_g(\mathcal{P}_a) = \begin{cases} \sum\limits_{i=1,3,\ldots,n-2} h^{2+}_g(x_i \wedge x_{i+1}) \;+\; h^{2+}_g(x_n) & n \text{ is odd} \\[2ex] \sum\limits_{i=1,3,\ldots,n-1} h^{2+}_g(x_i \wedge x_{i+1}) & n \text{ is even} \end{cases} \tag{3.18}$$

To illustrate this technique, consider the example from Figure 3.7 that shows an operator A whose set of preconditions is $\{x, y, z, t, q\}$. The h^{2+}_g estimation will be given by summation of the cost h_g^{2+} according to the order of the preconditions in the set. This technique first takes the pair of preconditions x and y, then the pair of preconditions z and t, and finally precondition q.

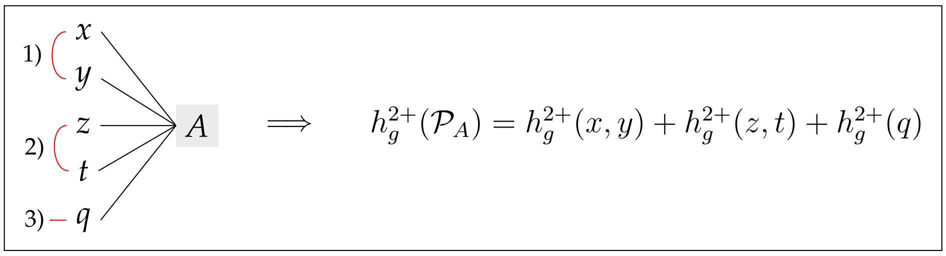

1)
2)
3)
x
y
z
t
q
A
$\Longrightarrow$
$h_g^{2+}(\mathcal{P}_A) = h_g^{2+}(x,y) + h_g^{2+}(z,t) + h_g^{2+}(q)$

Figure 3.7: An example of the ordered greedy covering alternative.

This is a simple and inexpensive way to prevent the over-counting problem.

3.4.2 Cost Greedy Covering

The Cost Greedy Covering alternative, namely h_c^{2+}, defines the cost of an action a with set of preconditions $\mathcal{P}_a$ as the summation of pairs $(x,y) \in \mathcal{P}_a$ where, at each stage, the pair (x,y) with highest cost is chosen, until all the elements in $\mathcal{P}_a$ are covered. That is:

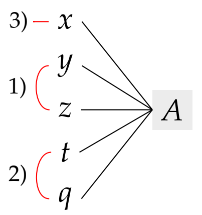
$$h_c^{2+}(\mathcal{P}_a) \quad = \quad p_n = \left\{ \{x_i, x_j\} = \underset{\substack{\mathcal{P}_{a_{n-1}} = \{\mathcal{P}_a - p_{n-1}\} \\ (x_i,x_j) \in \mathcal{P}_{a_{n-1}} \\ i<j}}{\operatorname{argmax}} h_c^{2+}(x_i, x_j) \right\} \tag{3.19}$$

To illustrate this technique, consider the example from Figure 3.8 that shows an operator A whose set of preconditions is $\{x, y, z, t, q\}$. First, it selects the pair of propositions with the highest h_c^{2+} value, (y, z) in the example. Then it excludes the other pairs that contain the previously selected preconditions. This step is repeated until all the preconditions of A are covered. The h_c^{2+} estimation will be given by summation of the cost h_c^{2+} of each selected pair.

This technique is slightly more expensive than the Order Greedy Covering technique because the cost of all pairs needs to be computed. However, the cost is still polynomial in the number of actions' preconditions.

3.4.3 Max Covering

The Max Covering alternative, namely h_m^{2+}, defines the cost of an action a with set of preconditions $\mathcal{P}_a$ as the summation of pairs $(x,y) \in \mathcal{P}_a$ that maximizes the cost of a. That is:



$$\Longrightarrow h_c^{2+}(\mathcal{P}_A) = \max \left\{ \begin{array}{r} \cancel{h_c^{2+}(x,y)}, \cancel{h_c^{2+}(x,z)}, h_c^{2+}(x,t), h_c^{2+}(x,q) \\ \mathbf{h_c^{2+}(y,z)}, \cancel{h_c^{2+}(y,t)}, \cancel{h_c^{2+}(y,q)} \\ \cancel{h_c^{2+}(z,t)}, \cancel{h_c^{2+}(z,q)} \\ h_c^{2+}(t,q) \end{array} \right\}$$

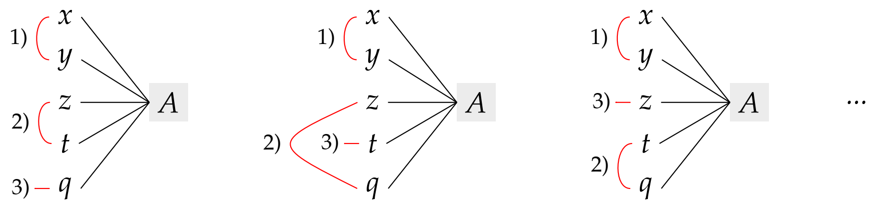
$$h_c^{2+}(\mathcal{P}_A) = \max \left\{ \begin{array}{r} \cancel{h_c^{2+}(x,t)}, \cancel{h_c^{2+}(x,q)} \\ \mathbf{h_c^{2+}(t,q)} \end{array} \right\}$$

$$h^{2+}(\mathcal{P}_A) = h_c^{2+}(y,z) + h_c^{2+}(t,q) + h_c^{2+}(x)$$

Figure 3.8: An example of the ordered greedy covering alternative.

$$h_m^{2+}(\mathcal{P}_a) = \max_{S \in P} \begin{cases} \sum\limits_{i=1,3,\ldots,n-2} h_m^{2+}(s_i \wedge s_{i+1}) \; + \; cost(s_n) & n \text{ is odd} \\ \\ \sum\limits_{i=1,3,\ldots,n-1} h_m^{2+}(s_i \wedge s_{i+1}) & n \text{ is even} \end{cases} \tag{3.20}$$

To illustrate this technique, consider the example from Figure 3.9 that shows an operator A whose set of preconditions is $\{x, y, z, t, q\}$. This technique will compute the cost of all possible permutations of the set of preconditions. The h_m^{2+} will be given by the permutation that maximizes the cost.



$$h_m^{2+}(\mathcal{P}_A) = \max \left\{ \begin{array}{l} (h_m^{2+}(x,y) + h_m^{2+}(z,t) + h_m^{2+}(q)), (h_m^{2+}(x,y) + h_m^{2+}(z,q) + h_m^{2+}(t)) \\ (h_m^{2+}(x,y) + h_m^{2+}(t,q) + h_m^{2+}(z)), (h_m^{2+}(y,z) + h_m^{2+}(x,t) + h_m^{2+}(q)) \\ (h_m^{2+}(y,z) + h_m^{2+}(x,q) + h_m^{2+}(t)), (h_m^{2+}(y,z) + h_m^{2+}(t,q) + h_m^{2+}(x)) \\ \ldots \end{array} \right\}$$

Figure 3.9: An example of the max covering alternative.

This technique is quite expensive for operators with many preconditions since the computation is exponential in the number of preconditions for an operator.

3.5 Cost estimation accuracy evaluation

Sections 3.3.1 and 3.3.2 describe two non-admissible heuristic based on the plan graph with Interactions described in Section 3.2. Sections 3.4.1, 3.4.2, and 3.4.3 describe three alternatives for an additive version of the h^2 heuristic. To evaluate the accuracy of these heuristics, we performed an experimental evaluation by comparing the *estimated cost* against the *optimal cost* for a selection of problems. The optimal cost of each problem was computed using the optimal planner HSP$^*_f$ (Haslum, 2008). In addition to the h^I and h^I_{rp} heuristics, the evaluation has been done for h^+ and h^+_{rp} heuristics, which are the versions of h^I and h^I_{rp} without Interaction. In these cases, the plan graph is built using the classical cost propagation. Likewise, the evaluation has been done for the set-additive heuristic h^s_a.

The following plots and tables show the results of this evaluation on seven well-known planning domains with fifteen problems each. For each heuristic technique in each domain, each column shows the following measures:

- r: the ratio of the *estimated cost* to the *optimal cost* per problem.
- M: the mean of the ratio among the *solved* problems.
- σ: the standard deviation of the ratio among the *solved* problems.
- p-value: Shapiro-Wilk normality test (Shapiro and Wilk, 1965) (to analyze whether data are normally distributed).

The next subsection shows the results for each tested domain. The symbol "-" means the HSP$^*_f$ does not solve the problem. Bold values symbolize the closest cost estimates and lowest variance cost estimate.

Blocksword domain

The Blocksword domain is based on the classical Blocksworld domain, but with six blocks labeled with letters and arranged randomly. The goal is to find the sequence of actions that appropriately arranges the blocks.

Table 3.1 and Figure 3.10 show the results in this domain. The h^I heuristic computes cost estimates that are considerably closer to the optimal, and more consistent (lower variance) than the estimated costs computed by the other heuristic techniques. The h^s_a, h^I_{rp}, h^+_{rp}, h^{2+}_g, h^{2+}_c, and h^{2+}_m heuristics compute the same cost estimate for these problems, which is always an underestimation of the optimal cost. The h^+ heuristic overestimates costs significantly. In addition, for all the heuristic techniques except h^+, we do not accept normality with a confidence level of $\alpha = 0.05$.

Table 3.1: Heuristic techniques accuracy evaluation in the Blocksword domain. Heuristics marked with * do not pass the normality test with $\alpha = 0.05$.

Heuristic	r															M	σ	p-value
	p01	p02	p03	p04	p05	p06	p07	p08	p09	p10	p11	p12	p13	p14	p15			
h_a^s	0.91	1	1	0.89	0.83	1	1	0.83	0.7	1	1	1	0.66	0.82	0.87	0.91	0.108	0.003735*
h^I	1	1.12	1	1.1	1.2	1	1	0.75	1	1	1	1	1	1.04	1.15	**1.025**	**0.099**	0.003401*
h^+	1.16	1.37	1.35	1.21	1.2	1	1	1.7	0.7	1	1.37	1.37	0.66	1.08	1.15	1.158	0.259	0.5556
h_{rp}^I	0.91	1	1	0.89	0.83	1	1	0.95	0.7	1	1	1	0.66	0.82	0.87	0.91	0.108	0.003735*
h_{rp}^+	0.91	1	1	0.89	0.83	1	1	0.95	0.7	1	1	1	0.66	0.82	0.87	0.91	0.108	0.003735*
h_g^{2+}	0.91	1	1	0.89	0.83	1	1	0.95	0.7	1	1	1	0.66	0.82	0.87	0.91	0.108	0.003735*
h_c^{2+}	0.91	1	1	0.89	0.83	1	1	0.95	0.7	1	1	1	0.66	0.82	0.87	0.91	0.108	0.003735*
h_m^{2+}	0.91	1	1	0.89	0.83	1	1	0.95	0.7	1	1	1	0.66	0.82	0.87	0.91	0.108	0.003735*

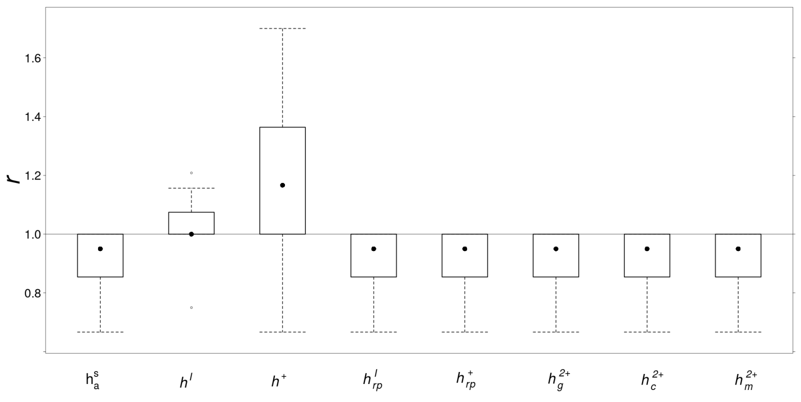


Figure 3.10: Ratio mean and standard deviation in the Blocksword domain.

Campus domain

The Campus domain is about the different activities that a student may perform at college. The goal is to find the sequence of actions that reaches the set of activities that the student wants to perform.

Table 3.2 and Figure 3.11 show the results for the Campus domain. The h^I and h_{rp}^I heuristics compute cost estimates that are considerably closer to the optimal, and more consistent than the estimated costs computed by the other heuristics estimators. The h_g^{2+}, h_c^{2+}, and h_c^{2+} heuristics compute better cost estimates than the h_a^s heuristic. The h^+ heuristic again badly overestimates. In addition, for the h^I and h_{rp}^I heuristics, we do not accept normality with a confidence level of $\alpha = 0.05$.

Elevator domain

The Elevator domain consists of a building with n fast elevators that stop in even floors and have a capacity of x passengers. Furthermore, there are m slow elevators that stop at all the floors and have a capacity of y passengers. The goal is to transport the passenger to the appropriate floor.

Table 3.2: Heuristic techniques accuracy evaluation in the Campus domain. Heuristics marked with * do not pass the normality test with $\alpha = 0.05$.

Heuristic	r															M	σ	p-value
	p01	p02	p03	p04	p05	p06	p07	p08	p09	p10	p11	p12	p13	p14	p15			
h_a^s	0.85	0.92	0.87	0.81	0.87	0.85	0.75	1	0.85	0.87	1	0.96	0.92	0.92	0.81	0.888	0.068	0.6139
h^I	1	0.89	1	1	1	1	1.12	0.92	1	1	0.92	0.89	0.96	0.96	1	**0.98**	0.055	0.01043*
h^+	2.5	2.67	2.93	2.81	2.68	2.64	2.68	2.81	2.5	2.93	2.81	2.71	2.53	2.53	2.81	2.707	0.141	0.2166
h_{rp}^I	1	0.92	1	1	0.93	1	0.87	1	1	1	1	1.1	0.92	0.92	1	**0.98**	**0.051**	7.836×10^{-3}*
h_{rp}^+	0.85	0.92	0.87	0.81	1.06	0.85	0.75	1	0.85	0.87	1	1.07	0.92	0.92	0.81	0.907	0.09	0.5029
h_g^{2+}	0.85	0.92	0.87	0.81	1.06	0.85	0.75	1	0.85	0.87	1	1.07	0.92	0.92	0.81	0.907	0.09	0.5029
h_c^{2+}	0.85	0.92	0.87	0.81	1.06	0.85	0.87	1	0.85	0.87	1	1.07	0.92	0.92	0.81	0.916	0.08	0.08807
h_m^{2+}	0.85	0.92	0.87	0.81	1.06	0.85	0.75	1	0.85	0.87	1	1.07	0.92	0.92	0.81	0.907	0.09	0.5029

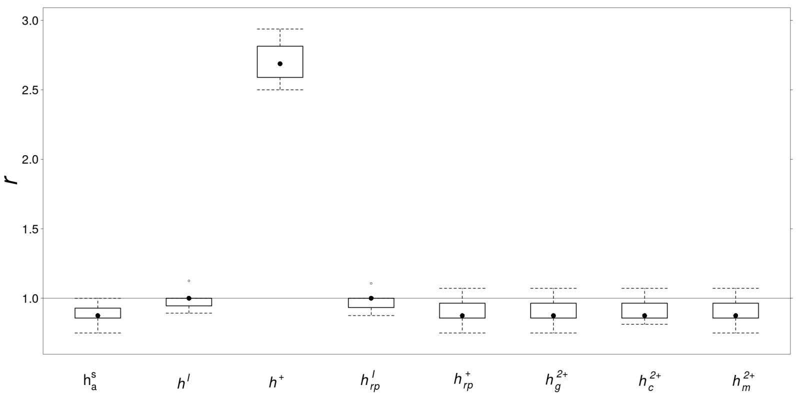


Figure 3.11: Ratio mean and standard deviation in the Campus domain.

Table 3.3 and Figure 3.12 show the results for the Elevator domain. The h^I heuristic computes cost estimates that are closer to the optimal compared to the other heuristic techniques. The h_a^s heuristic computes estimated costs that are more consistent than the estimated costs computed by the other heuristics estimators. The h_g^{2+}, h_c^{2+}, h_m^{2+}, and h_{rp}^I heuristics compute cost estimates that consistently overestimate cost, and the h^+ heuristic badly overestimates costs. In addition, for all the heuristic techniques except h^+ and h^I, we do not accept normality with a confidence level of $\alpha = 0.05$.

Floortile domain

The Floortile domain is about a set of robots that paint floor tiles with two colors. The robots can move up, down, left, and right, and can paint with one color at a time the tile that is up or down them. Once the tile is painted, the robot cannot stand on it. The goal is to paint the floor following the given color pattern.

Table 3.4 and Figure 3.13 show the results for the Floortile domain. The h_{rp}^I, h_{rp}^+, h_a^s, h_g^{2+}, h_c^{2+}, and h_m^{2+} heuristics compute cost estimates that are closer to the optimal, and more consistent than the estimated costs computed by h^I and h^+, which both seems to perform poorly in this particular domain. As before, the h^+

Table 3.3: Heuristic techniques accuracy evaluation in the Elevator domain. Heuristics marked with * do not pass the normality test with $\alpha = 0.05$.

Heuristic	r															M	σ	p-value
	p01	p02	p03	p04	p05	p06	p07	p08	p09	p10	p11	p12	p13	p14	p15			
h_a^s	0.8	0.8	1.11	0.85	0.9	0.83	-	1	1	0.78	1.06	0.84	0.94	0.95	1	0.922	**0.1**	$1.017\times10^{-4}*$
h^I	0.7	1.8	1	1.85	0.72	0.75	-	1	1.3	0.73	0.66	0.57	0.94	0.85	0.94	**0.99**	0.384	0.08806
h^+	2.2	2.8	2.55	3.14	2.45	2.16	-	2.13	2	2.26	5.2	3.52	4	4.71	4.35	3.107	1.031	0.3993
h_{rp}^I	1.7	2.2	1.77	1.71	1.72	1.5	-	1.46	1.6	1.42	1.73	1.68	1.68	1.57	1.76	1.681	0.18	$5.285\times10^{-5}*$
h_{rp}^+	1.7	2.4	1.77	1.14	1.72	1.5	-	1.46	1.6	1.42	1.73	1.68	1.68	1.57	1.7	1.651	0.263	$7.243\times10^{-4}*$
h_g^{2+}	1.7	2.4	1.77	1.14	1.72	1.5	-	1.46	1.6	1.42	1.73	1.68	1.631	1.57	1.7	1.647	0.263	$7.7\times10^{-4}*$
h_c^{2+}	1.7	2.4	1.77	1.28	1.72	1.5	-	1.46	1.6	1.42	1.73	1.73	1.631	1.57	1.76	1.665	0.248	$5.013\times10^{-4}*$
h_m^{2+}	1.7	2.4	1.77	1.14	1.72	1.5	-	1.46	1.6	1.42	1.73	1.68	1.68	1.57	1.7	1.651	0.263	$7.243\times10^{-4}*$

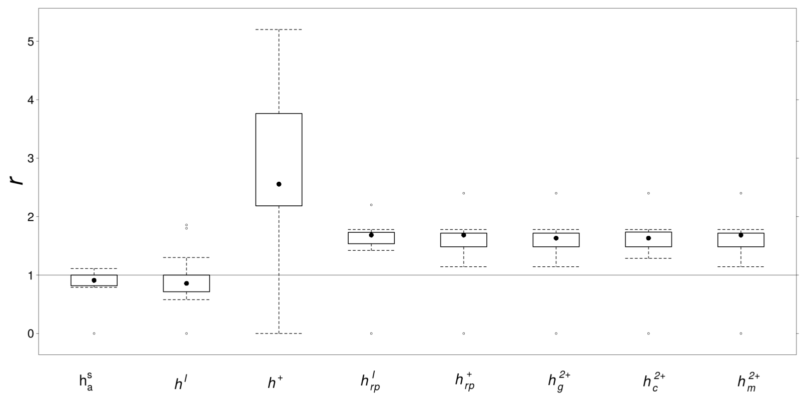


Figure 3.12: Ratio mean and standard deviation in the Elevator domain.

heuristic significantly overestimates costs, while the h^I heuristic underestimates costs on all the problem except for one problem. In addition, for the h^+ and h^I heuristics, we do not accept normality with a confidence level of $\alpha = 0.05$.

Grid domain

The Grid domain is about an agent that moves along a grid and has the task of transporting keys from some cells to others. The goal is to find the sequence of actions that transport the keys to the pertinent cells.

Table 3.5 and Figure 3.14 show the results for the Grid domain. The h^I heuristic computes cost estimates that are closer to the optimal, and more consistent, followed by the h_a^s heuristic, then the h_g^{2+}, h_c^{2+}, and h_m^{2+} heuristics, and finally the h_{rp}^I and h_{rp}^+ heuristics. All the heuristics generally underestimate cost except h^+, which overestimates on many problems. In addition, for all the heuristic techniques, we do not accept normality with a confidence level of $\alpha = 0.05$.

Intrusion domain

The Intrusion domain is about a hacker who might try different ways to attack on a set of servers. The goal is to find the sequence of actions that the hackers have

Table 3.4: Heuristic techniques accuracy evaluation in the Floortile domain. Heuristics marked with * do not pass the normality test with $\alpha = 0.05$.

Heuristic	r															M	σ	p-value
	p01	p02	p03	p04	p05	p06	p07	p08	p09	p10	p11	p12	p13	p14	p15			
h_a^s	1.2	1.09	0.92	1	0.88	0.86	0.75	1.01	0.8	1.03	0.86	0.83	1.15	0.91	0.94	0.951	**0.125**	0.7949
h^I	1	1	1.12	0.86	1	1	0.83	0.33	0.66	0.16	0.29	0.09	0.12	0.48	0.25	0.614	0.366	0.03955*
h^+	1.2	1.27	1.04	1.2	1.02	1.02	1	1.28	1.04	1.66	1.18	1.41	1.85	1.19	1.27	1.246	0.233	0.01273*
h_{rp}^I	1	1.36	1.16	1.13	1.05	0.88	0.83	1.03	0.86	0.83	0.76	0.75	0.81	0.92	0.88	**0.954**	0.164	0.1637
h_{rp}^+	1	1.36	1.16	1.13	1.05	0.88	0.83	1.03	0.86	0.83	0.76	0.75	0.81	0.92	0.88	**0.954**	0.164	0.1637
h_g^{2+}	1	1.36	1.16	1.13	1.05	0.88	0.83	1.03	0.86	0.83	0.76	0.75	0.81	0.92	0.88	**0.954**	0.164	0.1637
h_c^{2+}	1	1.36	1.16	1.13	1.05	0.88	0.83	1.03	0.86	0.83	0.76	0.75	0.81	0.92	0.88	**0.954**	0.164	0.1637
h_m^{2+}	1	1.36	1.16	1.13	1.05	0.88	0.83	1.03	0.86	0.83	0.76	0.75	0.81	0.92	0.88	**0.954**	0.164	0.1637

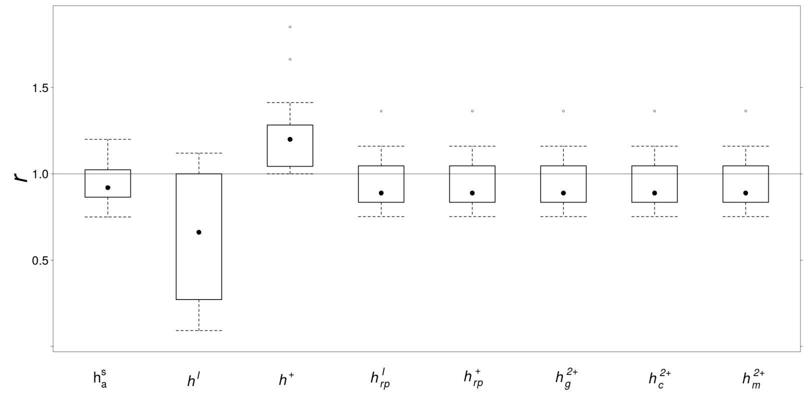


Figure 3.13: Ratio mean and standard deviation in the Floortile domain.

Table 3.5: Heuristic techniques accuracy evaluation in the Grid domain. Heuristics marked with * do not pass the normality test with $\alpha = 0.05$.

Heuristic	r															M	σ	p-value
	p01	p02	p03	p04	p05	p06	p07	p08	p09	p10	p11	p12	p13	p14	p15			
h_a^s	1	1	0.91	1	1	0.8	0.86	1	1	1	0.93	0.93	1	1	1	0.963	0.058	$1.41\times10^{-4}*$
h^I	1	1	1	1	1	1	0.93	1	1	1	1	1	1	1	1	**0.995**	**0.016**	$9.834\times10^{-8}*$
h^+	1	1	1.34	1.38	1.6	1.29	2.76	1.48	1	1	1	1	1	1	1	1.258	0.451	$3.455\times10^{-5}*$
h_{rp}^I	1	1	0.91	1	1	0.8	0.86	1	1	1	0.93	0.93	1	1	1	0.963	0.058	$1.41\times10^{-4}*$
h_{rp}^+	1	1	0.91	1	1	0.8	0.86	1	1	1	0.93	0.93	1	1	1	0.963	0.058	$1.41\times10^{-4}*$
h_g^{2+}	1	1	0.91	1	1	0.8	0.86	1	1	1	0.93	0.93	1.11	1	1	0.971	0.069	0.01059*
h_c^{2+}	1	1	0.91	1	1	0.8	0.86	1	1	1	0.93	0.93	1.11	1	1	0.971	0.069	0.01059*
h_m^{2+}	1	1	0.91	1	1	0.8	0.86	1	1	1	0.93	0.93	1.11	1	1	0.971	0.069	0.01059*

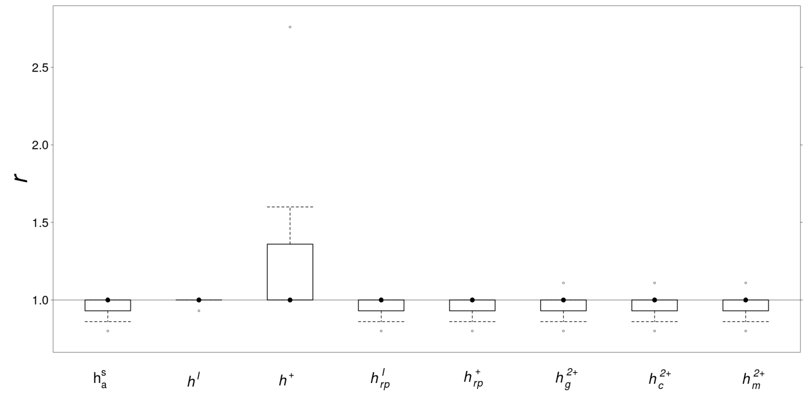


Figure 3.14: Ratio mean and standard deviation in the Grid domain.

to perform to attack a set of servers.

Table 3.6 and Figure 3.15 show the results for the Intrusion domain. All the heuristic techniques, except h^I and h^+, compute cost estimates equal to the optimal cost. The h^I produces the optimal value except for one problem where it is very close. Again, h^+ significantly overestimates costs. In addition, for the h^+ and h^I heuristic techniques, we do not accept normality with a confidence level of $\alpha = 0.05$. For the rest of the heuristics, the Shapiro-Wilk test is not applicable.

Table 3.6: Heuristic techniques accuracy evaluation in the Intrusion domain. Heuristics marked with * do not pass the normality test with $\alpha = 0.05$.

Heuristic	r															M	σ	p-value
	p01	p02	p03	p04	p05	p06	p07	p08	p09	p10	p11	p12	p13	p14	p15			
h_a^s	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**	**0**	-
h^I	1	1	1	1	1	1	1	1	0.9	1	1	1	1	1	1	0.993	0.024	$9.834\times10^{-8}*$
h^+	1	1.33	1.25	1.33	1.25	1.22	1.33	1.25	1.5	1.28	1.25	1.25	1.25	1.25	1.28	1.269	0.097	$3.641\times10^{-3}*$
h_{rp}^I	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**	**0**	-
h_{rp}^+	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**	**0**	-
h_g^{2+}	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**	**0**	-
h_c^{2+}	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**	**0**	-
h_m^{2+}	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**	**0**	-

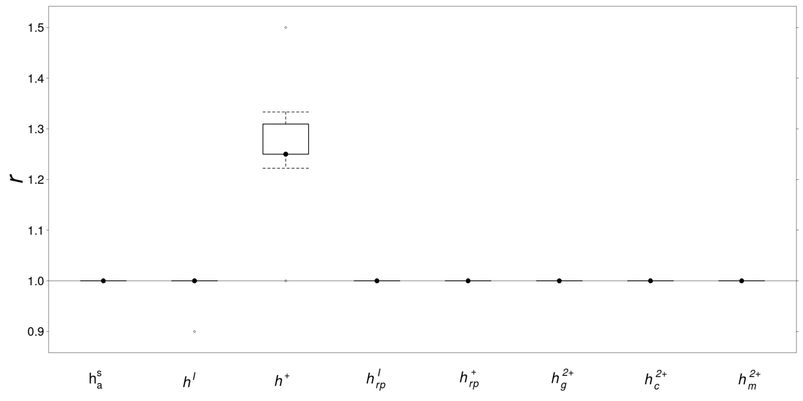


Figure 3.15: Ratio mean and standard deviation in the Intrusion domain.

Kitchen domain

The Kitchen domain is about preparing different meals: breakfast, lunch, or dinner. The goal is to find out the sequence of actions that prepares the desired meal.

Table 3.7 and Figure 3.16 show the results for the Kitchen domain. All the heuristic techniques, except h^I and h_{rp}^I, compute cost estimates equal to the optimal cost. The h^I and h_{rp}^I heuristics produce cost estimates close to the optimal value. However, h_{rp}^I overestimates costs while h^I underestimates. In addition, for the h^+ and h^I heuristic techniques, we do not accept normality with a confidence level of $\alpha = 0.05$. For the rest of the heuristics, the Shapiro-Wilk test is not applicable.

Table 3.7: Heuristic techniques accuracy evaluation in the Kitchen domain. Heuristics marked with * do not pass the normality test with $\alpha = 0.05$.

Heuristic	r															M	σ	p-value
	p01	p02	p03	p04	p05	p06	p07	p08	p09	p10	p11	p12	p13	p14	p15			
h_a^s	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**	**0**	-
h^I	0.97	0.97	0.97	0.97	1	1	1	1	1	1	1	1	1	1	1	0.994	0.009	$1.139 \times 10^{-5} *$
h^+	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**	**0**	$1.139 \times 10^{-5} *$
h_{rp}^I	1.06	1.06	1.06	1.06	1	1	1	1	1	1	1	1	1	1	1	1.017	0.028	-
h_{rp}^+	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**	**0**	-
h_g^{2+}	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**	**0**	-
h_c^{2+}	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**	**0**	-
h_m^{2+}	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**	**0**	-

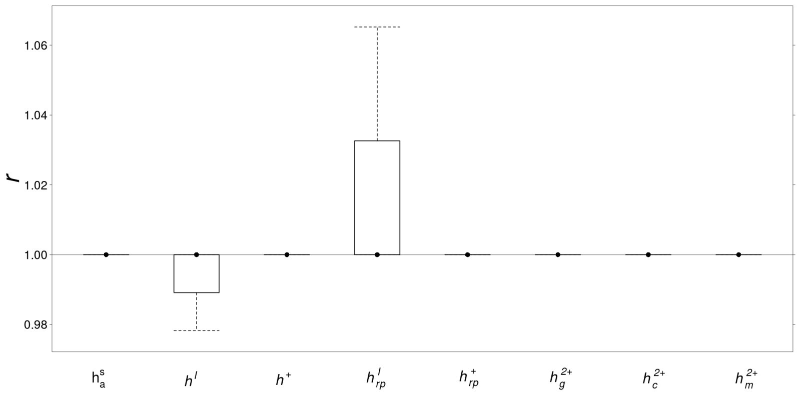


Figure 3.16: Ratio mean and standard deviation in the Kitchen domain.

Logistics domain

The Logistics domain is about delivering packages using planes and trucks between cities. The goal is to find the sequence of actions that transports each package to its final destination.

Table 3.8 and Figure 3.17 show the results for the Logistics domain. The h^I heuristic computes cost estimates that are closer to the optimal, while h_{rp}^I computes cost estimates that are a bit more consistent, but overestimates costs. The h^+, h_g^{2+}, h_c^{2+}, h_m^{2+}, and h_a^s heuristics consistently underestimate costs that are less closer and consistent. In addition, for all the heuristic techniques except h^I, we do not accept normality with a confidence level of $\alpha = 0.05$.

Accuracy evaluation summary

The cost estimation accuracy evaluation performed highlights some important ideas:

1. There is no significant difference among the h_g^{2+}, h_c^{2+}, and h_m^{2+} heuristics.

2. The h^{2+} family of heuristics does about the same as the h_{rp}^+ heuristic in all

Table 3.8: Heuristic techniques accuracy evaluation in the Logistics domain. Heuristics marked with * do not pass the normality test with $\alpha = 0.05$.

Heuristic	r															M	σ	p-value
	p01	p02	p03	p04	p05	p06	p07	p08	p09	p10	p11	p12	p13	p14	p15			
h^s_a	0.71	0.75	1	1	0.71	0.818	1	1	1	1	0.75	1	1	0.7	0.846	0.886	0.126	$8.055\times10^{-4}*$
h^I	0.92	1.08	1	1.08	0.92	0.72	1	1	1	1	1.08	0.88	1	1.2	0.76	**0.979**	0.116	0.2095
h^+	0.71	1.16	1	1	0.71	0.81	1	1	1	1	1	1	1	1	0.84	0.95	0.118	$1.311\times10^{-3*}$
h^I_{rp}	1	1.16	1	1.16	1	1	1	1	1	1	1.16	1	1	1.2	0.84	1.036	**0.092**	$1.082\times10^{-3*}$
h^+_{rp}	0.71	0.75	1	1	0.71	0.81	1	1	1	1	0.75	1	1	0.7	0.84	0.886	0.126	$8.055\times10^{-4}*$
h^{2+}_g	0.71	0.75	1	1	0.71	0.81	1	1	1	1	0.75	1	1	0.7	0.84	0.886	0.126	$8.055\times10^{-4}*$
h^{2+}_c	0.71	0.75	1	1	0.71	0.81	1	1	1	1	0.75	1	1	0.7	0.84	0.886	0.126	$8.055\times10^{-4}*$
h^{2+}_m	0.71	0.75	1	1	0.71	0.81	1	1	1	1	0.75	1	1	0.7	0.84	0.886	0.126	$8.055\times10^{-4}*$

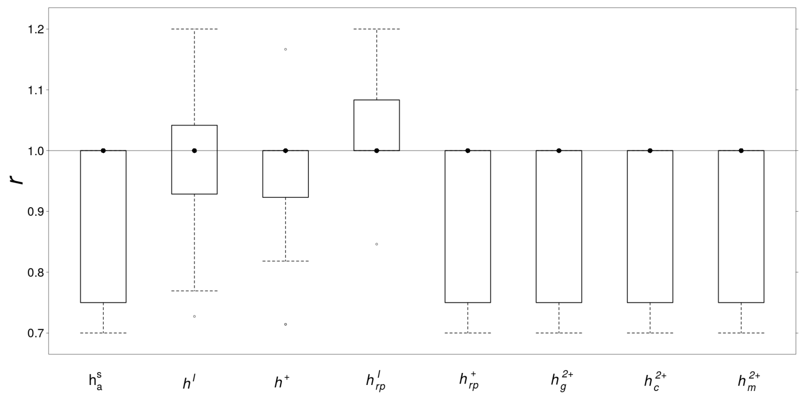


Figure 3.17: Ratio mean and standard deviation in the Logistics domain.

the domains – the h^+_{rp} heuristic eliminates the overestimation problem of the h^+ heuristic.

3. The h^s_a, h^+_{rp}, and h^{2+} heuristics perform similarly in all the domains except in the Elevator domain where h^s_a underestimates costs while h^+_{rp} and h^{2+} overestimate them.

4. The h^+ heuristic badly overestimates costs in most of the tested domains.

5. The h^I_{rp} and h^s_a heuristics are inconsistent – occasionally perform well, but often do not.

6. Overall, the h^I heuristic is the most consistent and computes cost estimates that are closer to the optimal.

We assume that the generated data for each heuristic technique are not normally distributed. This means that we cannot apply parametric statistics. Therefore, we have performed a non-parametric Kruskal-Wallis (Kruskal and Allen, 1952) test to find statistical significance difference of each heuristic technique among the different tested domains. In other words, if the domain induces a significant difference in the ratio. The Kruskal-Wallis test is performed considering each heuristic technique as a dependent variable, and the domain as a factor with 8 levels (domains) with 15 samples (problems). For each heuristic technique, each column shows χ^2 as the test statistic, *df* as the degrees of freedom of the test (the number of factors minus 1), and p-value as the Kruskal-Wallis test result. Table 3.9 shows the results of this test. For each heuristic technique among the different domains the test assumes that there is significant difference assuming $\alpha = 0.05$.

Table 3.9: Kruskal-Wallis test on heuristics functions. Those p-value with significant difference are marked with * ($\alpha = 0.05$).

Heuristic	χ^2	df	p-value
h_a^s	27.0195	7	$3.306\times10^{-4}*$
h^I	26.1848	7	$4.668\times10^{-4}*$
h^+	79.7514	7	$1.54\times10^{-14}*$
h_{rp}^I	49.4077	7	$1.88\times10^{-8}*$
h_{rp}^+	46.1926	7	$8.019\times10^{-8}*$
h_g^{2+}	45.8753	7	$9.244\times10^{-8}*$
h_c^{2+}	45.775	7	$9.669\times10^{-8}*$
h_m^{2+}	45.8755	7	$9.243\times10^{-8}*$

3.6 Evaluation of the h^I family of heuristics in planning

As a result of the increased accuracy and stability of the previously described h^I family of heuristics, it is natural to try to use them for planning purposes. The h^I heuristics were compared in the context of progression planning using the MetricFF planner (Hoffmann, 2003) with an A_ε^* search strategy (Pearl and Kim, 1982). The successors evaluation of the current state is based on the cost estimate computed by the h^I or h_{rp}^I heuristics, where the best successor is the one with the lowest cost. We have implemented four variations of the heuristic function in the MetricFF planner:

- MetricFF^I: h^I as heuristic function (Equation 3.15), and cost propagation through the plan graph considering Interaction information.

- MetricFF$^+$: h^I as heuristic function (Equation 3.15), and cost propagation through the plan graph not considering Interaction information.

- MetricFF$^I_{rp}$: h^I_{rp} as heuristic function (Equation 3.16), and cost propagation through the plan graph considering Interaction information.
- MetricFF$^+_{rp}$: h^I_{rp} as heuristic function (Equation 3.16), and cost propagation through the plan graph not considering Interaction information.

We have developed another approach using the MetricFF planner again, where the h^I heuristic is used in the first k levels of the search process, and then the h^+ heuristic, namely MetricFF^I_k. The idea behind this strategy is to compute more accurate, but computational expensive, cost estimates, early in the search process to guide it towards lower cost plans when the heuristic is less informative. Then, a faster heuristic is used for the reminder of the search. The MetricFF^I_k planner has been tested with $k = 1$ up to 3.

We compare these eight variants against several satisficing planners that deal with action costs such as: HSP$^*_f$, LPG (Gerevini et al., 2003), SGPlan$_6$ (Chen et al., 2006), and MetricFF (Hoffmann, 2003). LPG is run using LPG-speed (LPG$_s$) that computes the first solution, and LPG-quality (LPG$_q$) that computes the best solution. MetricFF is run under three different approaches that perform cost minimization using a weighted A* algorithm (MetricFF$_3$), cost minimization using a A$^*_\varepsilon$ algorithm (MetricFF$_4$), and cost minimization using an enforced hill-climbing algorithm and then A^*_ε (MetricFF$_5$).

The following plots and tables show the results of this evaluation on the same planning domains as before: Blocksword, Campus, Floortile, Intrusion, Kitchen, and Logistics domains. We do not show results for the Elevator and Grid domains because most of the planners had difficulties solving them so that we could not perform the study properly. Again, each domain has fifteen problems each. For each planning technique in each domain, each column shows the following measures:

- *r*: the ratio of the *estimated cost* to the *optimal cost* per problem.
- *M*: the mean of the ratio among the *solved* problems.
- σ: the standard deviation of the ratio among the *solved* problems.
- p-value: Shapiro-Wilk normality test (to analyze whether data are normally distributed).

The next subsection shows the results for each tested domain. Once again, the symbol "-" means the HSP$^*_f$ does not solve the problem. Bold values symbolize the closest cost estimates and lowest variance cost estimate.

Blocksword domain

Table 3.10 and Figure 3.18 show the results in this domain. MetricFF$^+_{rp}$ reaches solutions equal to the optimal. MetricFF^I, MetricFF$^I_{rp}$, MetricFF$^I_1$ and MetricFF$^I_2$ produce the optimal solution except in one problem, where the MetricFF$^I_1$ and MetricFF$^I_2$ planners produce the closest solution. MetricFF$^+$ and MetricFF$^I_3$ produce the optimal solution except in two problems. The rest of the planners consistently produce low accurate solutions in most of the problems. In addition, for all the planners except MetricFF$_5$, we do not accept normality with a confidence level of $\alpha = 0.05$. The Shapiro-Wilk test is not applicable in the MetricFF$^+_{rp}$ planner.

Table 3.10: Accuracy evaluation in the Blocksword domain. Heuristics marked with * do not pass the normality test with $\alpha = 0.05$.

Planner	Problem															M	σ	p-value
	p01	p02	p03	p04	p05	p06	p07	p08	p09	p10	p11	p12	p13	p14	p15			
LPG$_s$	1.33	1.18	1	1.21	1.08	1	1	1.1	1.15	1	1	1	1.16	1.34	1.53	1.14	0.155	0.01287*
LPG$_q$	1	1	1	1.21	1.25	1	1	1	1	1	1	1	1	1.13	1	1.039	0.081	$6.229\times10^{-6}*$
SGPlan$_6$	1	1	1	1	1	1	1	1	1	1	1	1	2.33	1	2.68	1.201	0.517	$9.624\times10^{-7}*$
MetricFF$_3$	1	1	1	1	1.25	1	1	1	1	1	1	1	1	1.26	1.18	1.046	0.094	$5.393\times10^{-6}*$
MetricFF$_4$	1.08	1	1	1.21	1.37	1	1	1	1.15	1.8	1	1	1.16	1.39	3.5	1.311	0.623	$6.978\times10^{-6}*$
MetricFF$_5$	1.41	1.75	1.35	1.21	1.45	1.8	1.75	1.1	1.15	1.8	1.37	1.37	1.16	1.47	1.34	1.435	0.231	0.07975
MetricFF^I	1	1	1	1	1	1	1	1.25	1	1	1	1	1	1	1	1.016	0.062	$9.834\times10^{-8}*$
MetricFF$^+$	1	1	1	1	1	1	1	1.15	1	1.6	1	1	1	1	1	1.05	0.151	$3.886\times10^{-7}*$
MetricFF$^I_{rp}$	1	1	1	1	1	1	1	1	1	1	1	1	1	1.26	1	1.017	0.065	$9.834\times10^{-8}*$
MetricFF$^+_{rp}$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**	**0**	-
MetricFF$^I_1$	1	1	1	1	1	1	1	1.15	1	1	1	1	1	1	1	1.01	0.037	$9.834\times10^{-8}*$
MetricFF$^I_2$	1	1	1	1	1	1	1	1.15	1	1	1	1	1	1	1	1.01	0.037	$9.834\times10^{-8}*$
MetricFF$^I_3$	1	1	1	1	1	1.26	1	1.15	1	1	1	1	1	1	1	1.027	0.073	$9.568\times10^{-7}*$

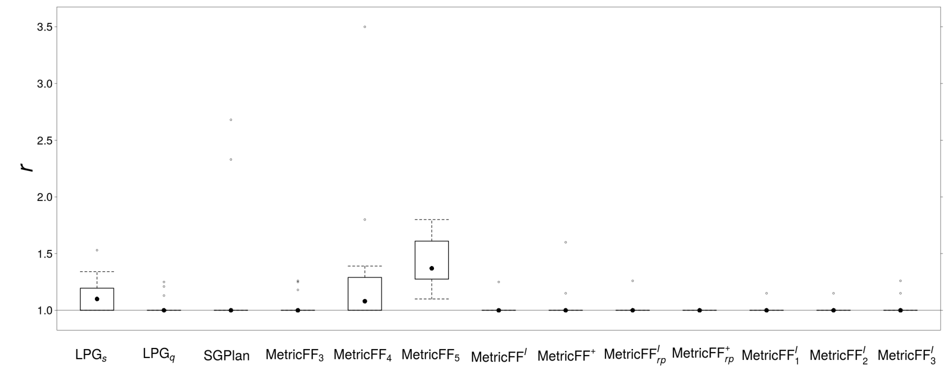


Figure 3.18: Ratio mean and standard deviation in the Blocksword domain.

Campus domain

Table 3.11 and Figure 3.19 show the results in this domain. MetricFF$_3$, MetricFF$_5$, MetricFF$^I_{rp}$, and MetricFF$^+_{rp}$ reach solutions equal to the optimal. SGPlan$_6$ and MetricFF^I produce the optimal solution except in one problem. The MetricFF$^+$,

MetricFF$_2^I$, and MetricFF$_3^I$ planners produce the optimal solution except in two problems. The rest of the planning techniques, that is, LPG$_s$,LPG$_q$, MetricFF$_4$, and MetricFF$_1^I$, produce less closer and consistent solutions in most of the problems. In addition, for all the planners except LPG$_s$, we do not accept normality with a confidence level of $\alpha = 0.05$. The Shapiro-Wilk test is not applicable in the MetricFF$_3$, MetricFF$_{rp}^I$, and MetricFF$_{rp}^+$ planners.

Table 3.11: Accuracy evaluation in the Campus domain. Heuristics marked with * do not pass the normality test with $\alpha = 0.05$.

Planner	Problem															M	σ	p-value
	p01	p02	p03	p04	p05	p06	p07	p08	p09	p10	p11	p12	p13	p14	p15			
LPG$_s$	1.35	1.1	1.43	1.12	1	1.35	1.18	1.07	1.28	1.25	1.07	1.03	1.21	1.14	1.25	1.193	0.124	0.7519
LPG$_q$	1	1.07	1.12	1	1.12	1.07	1.12	1.07	1.28	1	1.18	1.03	1	1.03	1.31	1.096	0.096	0.017*
SGPlan$_6$	1	1	1	1	1	1.14	1	1	1	1	1	1	1	1	1	1.009	0.035	$9.834\times10^{-8}*$
MetricFF$_3$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**	**0**	-
MetricFF$_4$	1.85	1.25	1.37	1.37	1.37	1.71	1.56	1.33	1.85	1.43	1.33	1.07	1.14	1.14	1.37	1.413	0.233	0.1104
MetricFF$_5$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**	**0**	-
MetricFF^I	1	1	1	1	1	1	1	1	1.07	1	1	1	1	1	1	1.004	0.017	$9.834\times10^{-8}*$
MetricFF$^+$	1	1	1	1	1	1	1	1.03	1	1	1.03	1	1	1	1	1.004	0.012	$7.525\times10^{-7}*$
MetricFF$_{rp}^I$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**	**0**	-
MetricFF$_{rp}^+$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**	**0**	-
MetricFF$_1^I$	1.07	1	1.18	1.12	1	1	1.12	1.03	1	1.18	1.03	1	1	1	1.18	1.063	0.074	$1.323\times10^{-3}*$
MetricFF$_2^I$	1	1	1	1	1	1	1	1.03	1	1	1.07	1	1	1	1	1.007	0.02	$7.395\times10^{-7}*$
MetricFF$_3^I$	1	1	1	1	1	1	1	1.03	1	1	1.03	1	1	1	1	1.004	0.012	$7.525\times10^{-7}*$

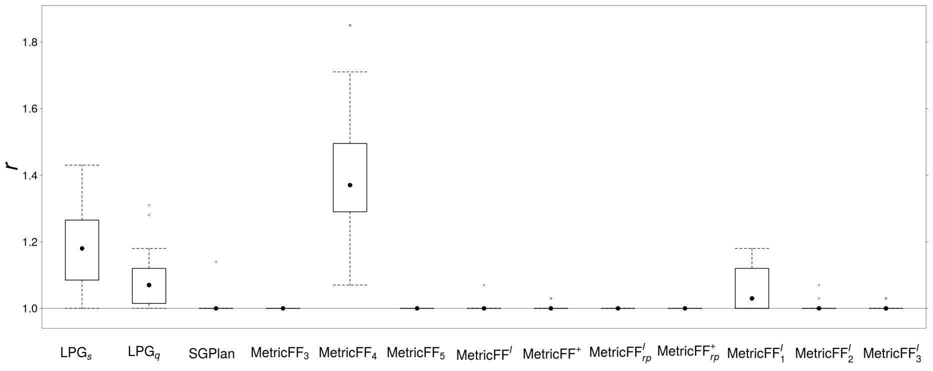


Figure 3.19: Ratio mean and standard deviation in the Campus domain.

Floortile domain

Table 3.12 and Figure 3.20 show the results in this domain. LPG$_s$ and SGPlan$_6$ solve all the problems, although it produces low accurate solutions. LPG$_q$ solve all the problems except one, with low accurate solutions. MetricFF$_3$ solves half of the problems and produces solutions that are close to the optimal in most of the cases. The rest of the planning techniques solve one-third of problems where MetricFF^I, MetricFF$^+$, MetricFF$_1^I$, and MetricFF$_3^I$ reach the optimal solution, and MetricFF$_{rp}^I$, MetricFF$_{rp}^+$, and MetricFF$_2^I$ produce less closer solutions. In addition, for all the planners except LPG$_s$ and SGPlan$_6$, we do not accept normality with a confidence level of $\alpha = 0.05$.

Table 3.12: Accuracy evaluation in the Floortile domain. Heuristics marked with * do not pass the normality test with $\alpha = 0.05$.

Planner	Problem															M	σ	p-value
	p01	p02	p03	p04	p05	p06	p07	p08	p09	p10	p11	p12	p13	p14	p15			
LPG$_s$	1	1.36	1.32	1.53	1.76	4.94	2	4.55	5.46	7.65	7.79	5.2	4.74	4.32	4.16	3.855	2.177	0.09062
LPG$_q$	1	1	2.84	1	1.58	1.5	1.16	1.4	1.55	1.44	1.18	1.64	-	1.42	1.13	1.421	0.45	0.01873*
SGPlan$_6$	1	1.36	1.8	2.06	1.52	1.94	1.54	1.5	1.72	1.65	1.37	1.49	1.43	1.78	1.47	1.58	0.252	0.7718
MetricFF$_3$	1	1	1	1.4	1	1	1.29	-	1.09	-	-	-	-	-	-	1.097	0.148	$9.41\times10^{-4}*$
MetricFF$_4$	1	1	3.16	1.93	3.08	-	-	-	-	-	-	-	-	-	-	2.036	0.951	$7.706\times10^{-5}*$
MetricFF$_5$	1	1.72	2.4	1.53	1.64	-	-	-	-	-	-	-	-	-	-	1.661	0.448	$1.669\times10^{-4}*$
MetricFF^I	1	1	1	1	1	1	-	-	-	-	-	-	-	-	-	**1**	**0**	$4.904\times10^{-5}*$
MetricFF$^+$	1	1	1	1	1	1	-	-	-	-	-	-	-	-	-	**1**	**0**	$4.904\times10^{-5}*$
MetricFF$^I_{rp}$	3.4	1.36	1.16	1	1	-	-	-	-	-	-	-	-	-	-	1.584	0.917	$4.803\times10^{-5}*$
MetricFF$^+_{rp}$	1	1.36	1.16	1	1	-	-	-	-	-	-	-	-	-	-	1.104	0.143	$9.375\times10^{-5}*$
MetricFF$^I_1$	1	1	1	1	1	1	-	-	-	-	-	-	-	-	-	**1**	**0**	$4.904\times10^{-5}*$
MetricFF$^I_2$	1	1	1.04	1	1	1	-	-	-	-	-	-	-	-	-	1.006	0.014	$5.932\times10^{-5}*$
MetricFF$^I_3$	1	1	1	1	1	1	-	-	-	-	-	-	-	-	-	**1**	**0**	$4.904\times10^{-5}*$

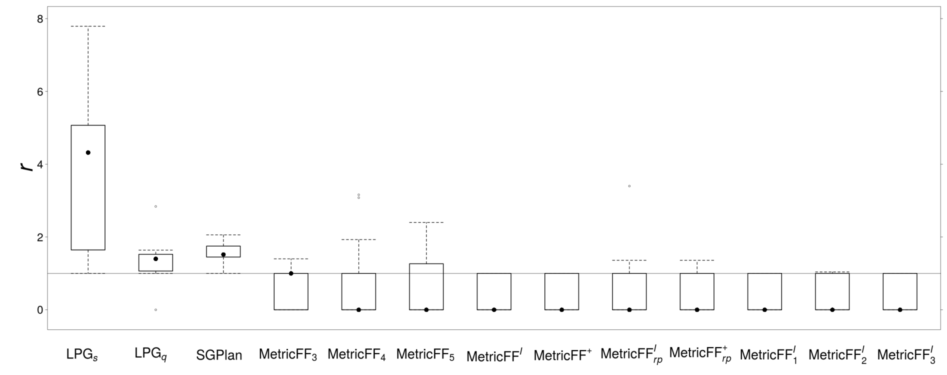


Figure 3.20: Ratio mean and standard deviation in the Floortile domain.

Intrusion domain

For the Intrusion domain, all the planning techniques produce solutions equal to the optimal cost. As a consequence of this, the Shapiro-Wilk test is not applicable.

Kitchen domain

Table 3.13 and Figure 3.21 show the results in this domain. MetricFF$_3$, MetricFF$_4$, MetricFF$^I_1$, MetricFF$^I_2$, and MetricFF$^I_3$ produce solutions equal to the optimal. The MetricFF$_5$ and SGPlan$_6$ planners produce less accurate solutions. LPG$_q$ and LPG$_s$ find the optimal solution except in one and two problems respectively. MetricFF^I finds the optimal solution except in four problems because the planner does not solve them. The rest of the planners produce the optimal solution except in one problem that planners are not able to solve it. In addition, for all the planners, we do not accept normality with a confidence level of $\alpha = 0.05$. The Shapiro-Wilk test is not applicable in the MetricFF$^I_1$, MetriFF$^I_2$, and MetricFF$^I_3$ planners.

Table 3.13: Accuracy evaluation in the Kitchen domain. Heuristics marked with * do not pass the normality test with $\alpha = 0.05$.

Planner	Problem															M	σ	p-value
	p01	p02	p03	p04	p05	p06	p07	p08	p09	p10	p11	p12	p13	p14	p15			
LPG$_s$	1	1	1	1	1	1	1	1	1.3	1.3	1	1	1	1	1.3	1.06	0.12	3.481×10^{-6}*
LPG$_q$	1.06	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1.004	0.016	9.834×10^{-8}*
SGPlan$_6$	1.06	1.06	1.06	1.06	1.13	1.13	1.13	1.13	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.192	0.103	8.669×10^{-4}*
MetricFF$_3$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**	**0**	4.252×10^{-4}*
MetricFF$_4$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**	**0**	1.139×10^{-5}*
MetricFF$_5$	1.04	1.04	1.04	1.04	1	1	1	1	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.104	0.09	9.834×10^{-8}*
MetricFF^I	-	-	-	-	1	1	1	1	1	1	1	1	1	1	1	**1**	**0**	9.834×10^{-8}*
MetricFF$^+$	1	1	1	1	1	1	1	1	-	1	1	1	1	1	1	**1**	**0**	9.834×10^{-8}*
MetricFF$^I_{rp}$	1	1	1	1	1	1	1	1	-	1	1	1	1	1	1	**1**	**0**	9.834×10^{-8}*
MetricFF$^+_{rp}$	1	1	1	1	1	1	1	1	-	1	1	1	1	1	1	**1**	**0**	9.834×10^{-8}*
MetricFF$^I_1$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**	**0**	-
MetricFF$^I_2$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**	**0**	-
MetricFF$^I_3$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**	**0**	-

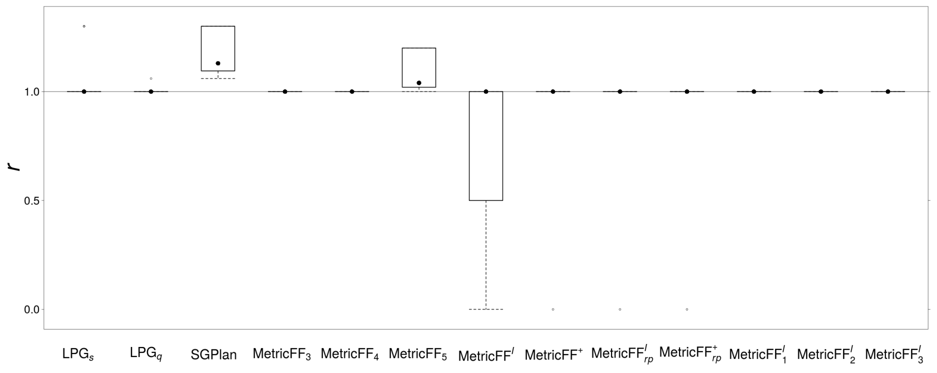


Figure 3.21: Ratio mean and standard deviation in the Kitchen domain.

Logistics domain

Table 3.14 and Figure 3.22 show the results in this domain. MetricFF$^+_{rp}$ produces solutions equal to the optimal. The rest of the planning techniques produce solutions closer to the optimal. In particular, MetricFF+, MetricFF$^I_{rp}$, and MetricFF$^I_1$ produce the same exact solutions among them. The results produce by MetricFF$^I_1$ and MetricFF$^I_2$ are a bit more closer to the optimal and consistent, followed by the MetricFF^I and MetricFF^3 planners. LPG$_s$, LPG$_q$, SGPlan$_6$, MetricFF$_4$, and MetricFF$_5$ produce low accurate solutions. In addition, for all the planners except MetricFF$^+_{rp}$ where the Shapiro-Wilk test is not applicable, we do not accept normality with a confidence level of $\alpha = 0.05$.

Table 3.14: Accuracy evaluation in the Logistics domain. Heuristics marked with * do not pass the normality test with $\alpha = 0.05$.

Planner	Problem															M	σ	p-value
	p01	p02	p03	p04	p05	p06	p07	p08	p09	p10	p11	p12	p13	p14	p15			
LPG_s	1.14	1.25	1.41	1.58	1.28	1.36	1.08	1.2	1	1.16	2.25	1.77	1	1.4	1.07	1.333	0.321	0.01351*
LPG_q	1.21	1.08	1.08	1	1	1.27	1.41	1	1.08	1.16	1.5	1.22	1	1	1.3	1.156	0.156	0.04034*
SGPlan_6	1	1	1.16	1.16	1	1.18	1.16	1.2	1	1	1	1.22	1.16	1	1	1.084	0.091	$5.484\times10^{-4}*$
MetricFF_3	1.07	1	1	1.16	1.07	1	1	1	1	1	1	1	1	1	1	1.02	0.045	$4.613\times10^{-6}*$
MetricFF_4	1.35	1.08	2.91	1.83	1.35	1.63	1.08	1	1.75	1.08	1.08	1	1.08	1.3	2.23	1.453	0.525	$3.122\times10^{-3}*$
MetricFF_5	1.14	1.08	1.08	1.08	1.07	1.18	1	1	1.08	1.08	1.08	1.44	1.08	1.6	1.23	1.15	0.158	$3.698\times10^{-4}*$
MetricFF^I	1	1	1	1	1	1.09	1	1	1	1	1	1	1	1.2	1	1.019	0.053	$7.793\times10^{-7}*$
MetricFF^+	1	1	1	1.08	1	1.18	1	1	1	1	1.08	1.22	1	1	1	1.038	0.07	$2.246\times10^{-5}*$
MetricFF^I_{rp}	1	1	1	1.08	1	1.18	1	1	1	1	1.08	1.22	1	1	1	1.038	0.07	$2.246\times10^{-5}*$
MetricFF^+_{rp}	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**	**0**	-
MetricFF^I_1	1	1	1	1.08	1	1.18	1	1	1	1	1	1.22	1	1	1	1.032	0.07	$4.866\times10^{-5}*$
MetricFF^I_2	1	1	1	1	1	1.18	1	1	1	1	1	1	1	1	1	1.012	0.045	$9.834\times10^{-5}*$
MetricFF^I_3	1	1	1	1	1	1.18	1	1	1	1	1	1	1	1	1	1.012	0.045	$9.834\times10^{-5}*$

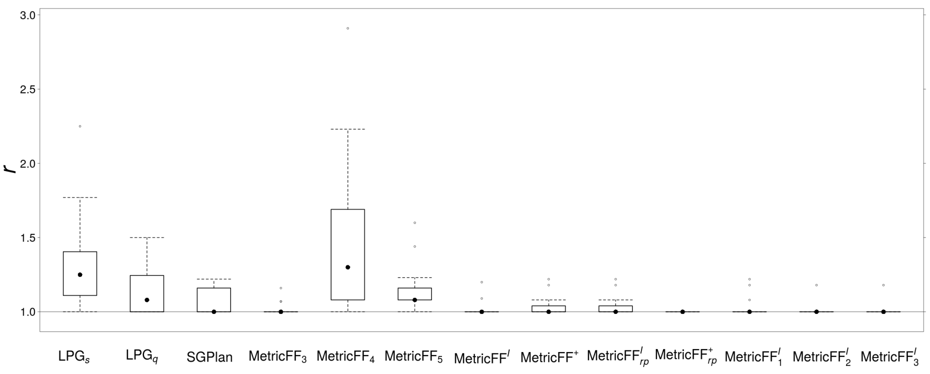


Figure 3.22: Ratio mean and standard deviation in the Logistics domain.

Accuracy evaluation summary in planning

The cost estimation accuracy evaluation performed highlights some important ideas:

1. MetricFF^+, MetricFF^I, and MetricFF^I_2 have a similar behavior in most of the domains.
2. MetricFF^+_{rp} and MetricFF^I_{rp} have similar behaviors in most the domains except in the Floortile domain where MetricFF^I_{rp} does not perform well for one of the problems.
3. The LPG_q planner is a bit more consistent and computes better estimates than the LPG_s planner.
4. The MetricFF_3, MetricFF_4, and MetricFF_5 planners are inconsistent – occasionally perform well, but often do not.
5. The LPG_s, LPG_q, and SGPlan_6 planners scale better than the rest of the planners.

For computation time, Table 3.15 shows the average time in seconds taken to solve the problem per domain and planner. For most domains, HSP$^*_f$ takes less time to solve a problem than any MetricFF^I and MetricFF$^+$ variations. In addition, it is noticeable that the use of the h^I heuristic in only the first two levels of the search process benefits the performance in time, and in accuracy for some cases.

Table 3.15: Time evaluation progression planning in seconds.

Domain	Blocks	Campus	Floortile	Intrusion	Kitchen	Logistics
HSP$^*_f$	8.302	0.054	63010	0.022	2.833	0.169
MetricFF^I	76.86	23.46	1658.6	143.53	500.66	79.46
MetricFF$^+$	9.6	6.4	1132	23.73	27.53	25.93
MetricFF$^I_{rp}$	184.46	17	1862.06	24.93	43.13	82.46
MetricFF$^+_{rp}$	18.8	10.26	1483.26	20.46	26.46	26.6
MetricFF$^I_1$	14.06	9.66	1114.8	32.53	39	29.93
MetricFF$^I_2$	17.46	11.26	1115.4	43.4	48.13	34.66
MetricFF$^I_3$	21.46	12.66	1115.66	56.66	57.4	38.86

We assume that the generated data for each planning technique are not normally distributed. This means that we cannot apply parametric statistics. Therefore, we have performed a non-parametric Kruskal-Wallis test to find statistical significance of each planning technique among the different tested domains. In other words, if the domain induces a significant difference in the ratio. The Kruskal-Wallis test is performed considering each planning technique as a dependent variable, and the domain as a factor with 6 levels (domains) with 15 samples (problems). For each planning technique, each column shows χ^2 as the test statistic, *df* as the degrees of freedom of the test (the number of factors minus 1), and p-value as the Kruskal-Wallis test result. Table 3.16 shows the results of this test. For each planning technique among the different domains the test assumes that there is significant difference assuming $\alpha = 0.05$.

3.7 Conclusions

This chapter presented two heuristic techniques, h^I and h^I_{rp}, based on a plan graph and Interaction information to compute more accurate cost estimates. This heuristic provides cost estimates that are generally closer to the optimal value and have lower variance than cost estimates computed by other cost-based heuristic estimators such as h^+ and h^s_a. The key to this improvement is the use of Interaction information.

Table 3.16: Kruskal-Wallis test on planning techniques. Those p with significant are marked with * ($\alpha = 0.05$).

Planner	χ^2	df	p-value
LPG$_s$	52.1139	5	$5.112\times10^{-10}*$
LPG$_q$	34.8267	5	$1.629\times10^{-6}*$
SGPlan$_6$	57.6051	5	$3.794\times10^{-11}*$
MetricFF$_3$	11.4905	5	0.04248*
MetricFF$_4$	44.1349	5	$2.175\times10^{-8}*$
MetricFF$_5$	47.2824	5	$4.976\times10^{-9}*$
MetricFF^I	33.9184	5	$2.472\times10^{-6}*$
MetricFF$^+$	35.1789	5	$1.286\times10^{-6}*$
MetricFF$^I_{rp}$	20.373	5	$1.063\times10^{-3}*$
MetricFF$^+_{rp}$	27.5519	5	$4.452\times10^{-5}*$
MetricFF$^I_1$	42.5597	5	$4.538\times10^{-8}*$
MetricFF$^I_2$	30.0312	5	$1.454\times10^{-5}*$
MetricFF$^I_3$	38.2974	5	$3.288\times10^{-7}*$

As a consequence of the stability and accuracy of the h^I heuristic, we applied it to classical planning. We demonstrated the accuracy/cost trade-off and its variation over different planning problems. Unfortunately, the computational overhead of h^I is high and usually does not pay off the actual search process.

Chapter 4

The h^I Family of Heuristics in Goal Recognition

This chapter presents a goal recognition technique based on the h^I heuristic. It first describes the techniques involved in the development of our approach, and then shows an empirical study. Next, it presents a goal recognition problem that involves free-flying robots monitoring ISS crew activities where we apply our goal recognition technique. Finally, it proposes a theoretical framework for goal recognition problems with uncertain observations.

4.1 Fast Goal Recognition (FGR) based on h^I

Ramirez and Geffner (2010) provide a very compelling domain-independent theory of the likelihood of a goal, based on how much the observed actions contribute to achieving that goal. This theory is rooted in the space of all possible plans that might be used to achieve each goal, and the assumption that less costly plans are more likely. It does not rely on hand-coded plan libraries or evidence-style networks. The drawback to this approach is its computational overhead. To overcome this issue, we develop FGR (E-Martín et al., 2015a) where we make use of Ramirez's framework presented in Section 2.2.1, but make use of:

1. Cost and Interaction estimates using the plan graph described in Section 3.2.
2. The pruning technique of IPR described in Section 2.2.2.

As previously mentioned, for each goal set $G \in \mathcal{G}$ Ramirez's framework computes the difference between $Cost(G|O)$ and $Cost(G|\overline{O})$. We can use approximations to compute these costs. In particular, we can compute $Cost(G|O)$ using IPR and the h^I heuristic. We do that by pruning the plan graph to remove everything inconsistent with the observations. The cost and Interaction information is then propagated through the pruned graph. These new cost values are used to estimate $Cost(G|O)$ using Equation 3.15. In order to compute $Cost(G|\overline{O})$, we use a plan graph with cost and Interaction information and the h^I heuristic. We do that by approximating $Cost(G|\overline{O})$ as just $Cost(G)$. We can do this because, in most cases, $Cost(G) = Cost(G|\overline{O})$ since there are multiple possible paths to the goal. In order for the two costs to be different, the observation sequence would need to consist exclusively of optimal landmarks (Hoffmann et al., 2004).

To illustrate, consider the example shown in Figure 4.1, where an agent can move *up*, *left*, and *right* at cost 1. It has two possible goals, G_1 and G_2, and the observed sequence is $O = (o_1)$. For goal G_1, $Cost(G_1|O) = 3$, and $Cost(G_1|\overline{O}) = Cost(G_1) = 3$. (Both costs are the same since o_1 is on an optimal path to G_1 and there is another optimal path that reaches G_1, but does not include o_1.) Hence, $\Delta(G_1, O) = 0$, and $Pr(G_1|O) = \alpha(0.5)$. For goal G_2, $Cost(G_2|O) = 2$ and $Cost(G_2|\overline{O}) = 4$ since avoiding o_1 requires a suboptimal plan for G_2. This results in $\Delta(G_2, O) = -2$, and $Pr(G_2|O) = \alpha(0.88)$. This means that G_2 is more likely to occur than G_1. This result is somewhat counterintuitive. The fact that the sequence of observed actions consists exclusively of optimal action landmarks for G_2 causes $Cost(G_2|\overline{O})$ to differ from $Cost(G_2)$ yielding a higher probability for G_2. However, in most cases, $Cost(G|\overline{O}) = Cost(G)$ because there are multiple possible paths to a goal. In order for the two costs to be different, the observation sequence would need to consist exclusively of optimal landmarks (like the case for G_2 above). For example, if we were to shift G_1 and G_2 left one column there would now be multiple optimal paths to G_2 so $Cost(G_2|\overline{O})$ would equal to $Cost(G_2)$. This would result in both $\Delta(G_1, O) = 0$ and $\Delta(G_2, O) = 0$ yielding the same probability for both goals.

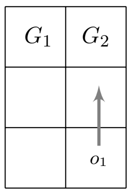


Figure 4.1: A 3x3 plan network for goals G_1 and G_2.

As with Ramirez, we define a goal recognition problem as $T = \langle P, \mathcal{G}, O, Pr \rangle$, which includes the problem P (planning domain and initial conditions), the set of possible goals G, a set of observations O, and a prior distribution over the possible goals G. It is also assumed that the sequence of observed actions may be incomplete, but is accurate (not noisy). As mentioned previously, this work also makes use of the IPR technique that assumes knowledge in the time step in which each action in the observation sequence happens. This assumption is generally valid for domains that involve monitoring – for instance, a robotic observer watching the activities of a person; it may not see or identify every activity, but it knows the times where it does observe something. However, it may be restrictive in other domains. In order to cover both assumptions, this work covers two different cases. In the first one, the time step for each observed action in O is assumed to be known. In the second one, like in Ramirez'z approach, the time step for each observed action in O is not required. As a result, IPR is modified to relax this assumption.

4.1.1 Computing goal probabilities

The combination of the plan graph cost estimation technique described in Section 3.2, and the observation pruning technique described in Section 2.2.2 and Section 4.1.2 allow fast estimation of cost differences $\Delta(G, O)$, and as a result probability estimates for the possible goals $G \in \mathcal{G}$.

Figure 4.2 shows the high-level algorithm for FGR used to solve a goal recognition problem, which may be summarized in the following steps:

1. Build a plan graph for the problem P (domain plus initial conditions) and propagate cost and Interaction information through this plan graph according to the technique described in Section 3.2.

2. For each (possibly conjunctive) goal $G \in \mathcal{G}$ estimate the $Cost(G)$ from the plan graph using Equation 3.15.

3. Prune the plan graph, based on the observed actions O, using the technique described in Sections 2.2.2 and 4.1.2.

4. Compute new cost and Interaction estimates for this pruned plan graph, considering only those propositions and actions labeled 0 or 1.

5. For each (possibly conjunctive) goal $G \in \mathcal{G}$:

    a. Estimate the $Cost(G|O)$ from the cost and Interaction estimates in the pruned plan graph, again using Equation 3.15. The pruned plan graph may discard propositions and/or actions in the plan graph necessary to reach the goal. This constraint provides a way to discriminate possible goals. However, it may imply that (1) the real goal is discarded, or (2) the calculated costs are less accurate. Therefore, computation of $Cost(G|O)$ has been developed under two strategies:

i. $Cost(G|O)$ is computed using the pruned plan graph.

ii. $Cost(G|O)$ is computed after the pruned plan graph is expanded to quiescence again. This will reintroduce any pruned goals that are still possible given the observations.

b. Compute $\Delta(G, O)$ using Equation 2.11, and using Equation 2.12 compute the probability $Pr(G|O)$ for the goal given the observations.

```
FAST GOAL RECOGNITION ($P, O, \mathcal{G}$)

$O$              ≡  an observed actions sequence
$\mathcal{G}$    ≡  a set of possible goals or hypothesis
$g$              ≡  a goal $g \in \mathcal{G}$
$pg$             ≡  a plan graph with cost and Interaction
$pruned–pg$      ≡  a pruned plan graph
PR               ≡  a probability distribution over $\mathcal{G}$

1. $pg \leftarrow$ BUILDPLANGRAPH($P$, $\mathcal{G}$)
2. for each $g \in \mathcal{G}$
     COMPUTECOST($G$, $pg$)
3. $pruned–pg \leftarrow$ PRUNEPLANGRAPH($pg$, $O$)
4. UPDATECOSTPLANGRAPH($pruned–pg$)
5. for each $g \in \mathcal{G}$
     a. COMPUTECOST($G$, $pruned–pg$)
     b. PR $\leftarrow$ PR $\cup$ COMPUTEPROBABILITY($G$)
6. return PR
```

Figure 4.2: The FGR pseudo-algorithm.

To illustrate this computation, consider the example shown in Figure 2.7. There are two possible goals, G_1 and G_2, and $O = (o_1)$ as the observed sequence. For goal G_1, $Cost(G_1|O) = 4$ and $Cost(G_1) = 4$. (Both costs are the same since o_1 is on an optimal path to G_1 and there is another optimal path that reaches G_1, but does not include o_1.) Hence, $\Delta(G_1, O) = 0$, and $Pr(G_1|O) = \alpha(0.5)$. Similarly, $Cost(G_2|O) = 3$ and $Cost(G_2) = 3$. This results in $\Delta(G_2, O) = 0$, and $Pr(G_2|O) = \alpha(0.5)$. This means that G_1 and G_2 are equally likely to occur given $O = (o_1)$. Regardless of the use of $Cost(G)$ instead of $Cost(G|\overline{O})$, the result is the same as in Ramirez's approach.

As a more complicated example, consider again the simple logistics problem from Section 2.2.2. Suppose that the possible goals are $g_1 = \{$(scanned pkg trk)$\}$, $g_2 = \{$(scanned pkg trk), (at b pkg)$\}$, and $g_3 = \{$(at a pkg)$\}$. Propagating cost and

Interaction information through the plan graph results in:

$$Cost(g_1) = Cost(\text{scanned pkg trk}) = 3$$

$$Cost(g_1|O) = Cost(\text{scanned pkg trk}) = 4$$

$$Cost(g_2) \approx \left\{ \begin{array}{l} Cost(\text{scanned pkg trk}) + Cost(\text{at b pkg}) + \\ I(\text{scanned pkg trk, at b pkg}) \end{array} \right\} = 2 + 5 - 2 = 5$$

$$Cost(g_2|O) \approx \left\{ \begin{array}{l} Cost(\text{scanned pkg trk}) + Cost(\text{at b pkg}) + \\ I(\text{scanned pkg trk, at b pkg}) \end{array} \right\} = 2 + 5 - 2 = 5$$

Thus, the the cost difference is:

$$\begin{array}{lllllll} \Delta(g_1, O) & = & Cost(g_1|O) - Cost(g_2) & = & 4 - 3 & = & 1 \\ \Delta(g_2, O) & = & Cost(g_1|O) - Cost(g_3) & = & 5 - 5 & = & 0 \end{array}$$

As a result:

$$\begin{array}{lll} Pr(g_2|O) & = & \alpha \dfrac{\exp\{-1\}}{1 + \exp\{-1\}} = \alpha(0.27) \\ Pr(g_3|O) & = & \alpha \dfrac{\exp\{0\}}{1 + \exp\{0\}} = \alpha(0.5) \end{array}$$

With regard to hypothesis $g_3 = \{(\text{at a pkg})\}$, the plan graph dismisses this hypothesis as a solution because once the plan graph is pruned, proposition (at a pkg) is labeled as -1. Therefore:

$$Cost(g_3|O) \approx Cost(\text{at a pkg}) = \infty$$

As a result:

$$Pr(g_3|O) = \alpha \frac{\exp\{-\infty\}}{1 + \exp\{-\infty\}} = \frac{0}{1} = \alpha(0)$$

Assuming uniform priors, $Pr(G)$, after normalizing the probabilities, this technique gives as a result $Pr(g_1|O) = 0.35$, $Pr(g_2|O) = 0.65$, and $Pr(g_3|O) = 0$. Hence, the goal g_2 is the most likely goal in this example given the observations of actions (verify pkg trk a) and (drive trk a b).

4.1.2 Relaxing the time step assumption in IPR

Jigui and Minghao's pruning technique can be modified in order to relax the assumption of knowing the time step of each action in the observed sequence. Like Ramirez and Geffner (2010), the sequence of observed actions is sequential. Initially, an *earlier start time* (*est*) i is assigned to each action o in the observed sequence. The *est* is given by the order of each action in the observed sequence. To illustrate this, given the sequence of observed actions $(o_0, o_1, ..., o_i)$, the *est* for each action would be: *est*(o_0)=0, *est*(o_1)=1, *est*(o_2)=2, etc. Once the pruning process starts, an observed action o is possible to be observed at the assigned level i, if all its preconditions are true (value 1) and/or unknown (value 0) and they are not mutually exclusive at level $i - 1$. Otherwise, the action cannot be executed at that level, which results in an update of the *est* of each remaining action in the observed sequence. For instance, consider the initial observed sequence where *est*(o_0)=0, *est*(o_1)=1, *est*(o_2)=2, and o_0 can be executed at level 0. If o_1 cannot be executed at level 1, then *est*(o_1)=2 and *est*(o_2)=3. If necessary, the plan graph will be expanded until an *est* is assigned to each observed action in the sequence.

To illustrate this method, consider the simple Logistics example in Figure 2.1 whose plan graph is shown in Figure 2.8. Suppose that the sequence of observed actions is (verify pkg trk a) and (drive trk a b), with initial *est* 0 and 1 respectively. Action (verify pkg trk a) is initially assumed to be at level 0, but this cannot be the case because its preconditions are not true at level 0. Therefore, the *est* for (verify pkg trk a) is updated to 1, and the *est* for (drive trk a b) is updated to 2. The consequence of this updating is that each observed action is assumed to occur at the earliest possible time consistent with both the observation sequence and the constraints found in constructing the plan graph, using the Interaction information.

4.1.3 Experimental evaluation

This section describes an experimental evaluation on the planning domains used by Ramírez and Geffner: Blocksword, Campus, Grid, Intrusion, Kitchen, and Logistics. Each domain has 15 problems. The hypotheses set and actual goal for each problem were randomly chosen with the priors on the goal sets assumed to be uniform. For each problem in each of the domains, the LAMA planner (Richter and Westphal, 2010) was run to solve the problem for the actual goal. The actual plans generated were used to chose the sequences of actions. We selected the LAMA planner to consider optimal and non-optimal actual plans. The experiments were conducted on an Intel Xeon CPU E5-1650 processor running at

3.20GHz with 32 GB of RAM.

Ramirez evaluates his technique using HSP$^*_f$ (Haslum, 2008), an optimal planner, and LAMA, a satisficing planner that is used in two modes: as a greedy planner that stops when it finds the first plan (LAMA$_G$), and as a planner that returns the best plan found in a given time limit (LAMA). For the purpose of our tests, Ramirez's technique is also evaluated using the heuristic goal set estimator h^s_a, which requires no search since the cost is given by a heuristic. The goal recognition technique presented in the previous section is compared against Ramirez's technique using those three planners and h^s_a, on the aforementioned domains, using a range of time limits from 5 seconds up to 1800 seconds (see Appendix A for more details). Our technique FGR, is run under eight variations, with and without observation times, with and without extension of the plan graph after pruning, with and without Interaction, and with and without relaxed plans. Not all the combinations make sense. Here are the ones we evaluate (some additional combinations in Appendix A):

- FGR^{I}T: the propagation of cost information through the plan graph considers Interaction information, and the time step for each observed action is known.
- FGR^{I}ET: same as above, but the pruned plan graph is expanded until quiescence.
- FGR^I: the propagation of cost information through the plan graph considers Interaction information, but observation times are unknown.
- FGR^{I}E: same as above, but the pruned plan graph is expanded until quiescence.
- FGR$^+$: the propagation of cost information through the plan graph does not consider Interaction information, and observation times are unknown.
- FGR^{FF}: builds a simple relaxed plan without utilizing cost information, and observation times are unknown.
- FGR$^I_{rp}$: builds a cost-relaxed plan using a plan graph that propagates cost using Interaction information, and observation times are unknown.
- FGR$^+_{rp}$: builds a cost-relaxed plan using a plan graph that propagates cost without using Interaction information, and observation times are unknown.

This section is divided into two subsections, each one corresponding to the test performed. In the first one, the set of observed actions for each recognition problem was randomly taken from the subset of the plan solution, ranging from 100% of the actions, down to 10% of the actions. In the second one, the set of observed actions for each recognition problem was considered to be the prefix of the plan solution, ranging from 100% of the actions, down to 10% of the actions. The reason for this test is to model incremental goal recognition as actions are observed. Priors on the goal sets are assumed to be uniform for both tests.

Results for both tests are summarized in Tables 4.1 and 4.2 where for each planner, each column shows average performance over the 15 problems in each domain. The first row in the table represents the optimal solution where HSP$^*_f$ computes $\Delta(G, O)$ using Equation 2.11. The second row in the table represents the optimal solution where HSP$^*_f$ computes $\Delta(G, O)$ using $Cost(G)$ instead of $Cost(G|\overline{O})$. We denote the latest as gHSP$^*_f$. Both approaches were allowed to run for an unlimited amount of time. The next rows in the table represent suboptimal solutions: first, solutions generated by LAMA and LAMA$_G$; second, solutions generated by FGR with Interaction and time step information; third, solutions generated by FGR with Interaction and no time step information; and finally, solutions generated by relaxed plan-based approaches. For each technique, rows represent different measures of quality and performance:

- T shows the average time taken for solving all the problems.
- Q shows the fraction of times the actual goal was among the goals found to be the most likely.
- S shows the *spread*, i.e., the average number of goals $G \in \mathcal{G}$ that were found to be the most likely.
- Q_{20} and Q_{50} show the fraction of times the actual goal is in the top 20% and top 50% of the ranked goals. Although Q might be less than 1 for some problem, Q_{20} or Q_{50} might be 1, indicating that the actual goal was "close" to the top.
- d is the mean distance between the probability scores produced for all the goal candidates, and the probability scores produced by gHSP$^*_f$. More precisely, if the set of possible goals is $\{G_1, ..., G_n\}$, a method produces probabilities estimates $\{e_1, ..., e_n\}$ for those goals, and gHSP$^*_f$ produces probabilities $\{p_1, ..., p_n\}$ for those goals, d is then defined as:

$$d = \frac{1}{n}\left\{\sum_{i=1}^{n} |e_i - p_i|\right\} \tag{4.1}$$

The ideal result has $Q = 1$, which means that the actual goal is found with the highest probability, and $S = 1$, which means that the tested approach discriminates among the possible goals very well since there is only one goal set with the highest probability.

Random observations

Table 4.1 summarizes the results when observations are randomly chosen. In all the domains except Grid, Kitchen, and Logistics, HSP$^*_f$ finds the actual goal with highest probability ($Q = 1$). In all the domains except Grid and Logistics, gHSP$^*_f$ also finds the actual goal with highest probability ($Q = 1$). In Logistics, the value of Q degrades with lower percentages of observed actions. The *spread*, S, also increases as the percentage of observed actions decreases, because there is not enough information to distinguish among different possible goals. Results for HSP$^*_f$ and gHSP$^*_f$ are the same except for the Kitchen and Grid domains where gHSP$^*_f$ has a slightly better quality results since the fraction of times that gHSP$^*_f$ finds the actual goal is higher. In other words, the Q value is higher. As a consequence, S is slightly higher as well. Another key point is that gHSP$^*_f$ is significantly faster than HSP$^*_f$ because it is generally easier to plan for G than for $G|\overline{O}$.

LAMA and LAMA$_G$ solve all the problems for the Blocksword domain within 1800 seconds, which is actually slower than gHSP$^*_f$. However, the quality Q is high, which means that the actual goal is among the most likely goals for most problems. In the Campus domain, LAMA and LAMA$_G$ solve all the problems within 3 and 0.6 seconds respectively. They generate high quality solution since they find the actual goal with highest probability ($Q = 1$) except for low percentage of observed actions where it degrades a bit. Both approaches discriminate very well among the possible goals ($S = 1$). In the Grid domain, LAMA and LAMA$_G$ solve all the problems within 300 and 80 seconds respectively. (In both cases, the computation time decreases as the percentage of observed actions drops.) LAMA finds the same solution as gHSP$^*_f$, that is, the optimal solution, for all the percentages of observed actions excepts for the 70%. LAMA$_G$ finds a high quality Q solutions. In the Intrusion domain, LAMA solves all the problems within 1300 seconds. The computation time decreases to 75 seconds as the percentage of observed actions drops. It finds the optimal solution (same as gHSP$^*_f$). LAMA$_G$ solves all the problems in the Intrusion domain within 3 seconds. It finds the optimal solution for low percentages of observed actions. However the spread increases along with the percentage of observed actions, which means that the approach does not discriminate among the possible goals. In the Kitchen domain, LAMA and LAMA$_G$ find the optimal solution (same as gHSP$^*_f$) for all the problems except for the 100% of observed actions set. However, LAMA$_G$ is faster than LAMA. In the Logistics domains, LAMA solves all the problems, and produces high quality Q solutions within 42 seconds as long as the percentage of observed actions is high. The computation time decreases to around 11 seconds as the percentage of observed actions drops. For LAMA$_G$ the problems are solved within 5 seconds. When the percentage of observed actions is high, the fraction of times where the approach finds the actual solution among all possible goals is good. The quality degrades as the percentage of observed actions drops.

The FGR^{I}T and FGR^I heuristics solve all the problems within 2 seconds. FGR^{I}E solves all the problems within 10 seconds, and FGR^{I}ET solves them within 12 seconds except for problems in the Grid domain where FGR^{I}T and FGR^I solve them within 115 seconds. FGR^{I}E solves all the problems within 130 seconds, and FGR^{I}ET solves them within 125 seconds. In general, the four approaches quickly provide high quality Q solutions when the percentage of observed actions is high, although this degrades a bit as the percentage drops. Comparing FGR^I with and without observation times, FGR^{I}T and FGR^{I}TE sometimes produce slightly higher quality Q solutions than FGR^I and FGR^{I}E, where the time step is unknown. However, this difference is small and only occurs in a few instances for the Blocksword and Logistics domains. Considering the expansion of the pruned plan graph, it was expected that FGR^{I}ET and FGR^{I}E dominated FGR^{I}T and FGR^I, respectively, in terms of goal recognition accuracy. Surprisingly, this is not the case in the studied domains. In some cases, FGR^{I}ET and FGR^{I}E show a small improvement, but in others the quality Q of the solutions drops. The hypothesis is that the pruned goals are sufficiently unlikely that reintroducing them does not significantly impact the resulting probability distribution. Considering the use of Interaction estimates, FGR^I gets higher quality Q solutions than FGR$^+$, except for some cases of Blocksword and Logistics domains for 30% and 10% of observed actions sets.

Heuristics based on relaxed plans produce quick solutions. In particular, the h^s_a heuristic solves all the problems within a second. In all the domains except Logistics, the quality Q of the solution is 1, which means that the actual goal is among the most likely goals for all the problems. However, the *spread* is very high, which means that the approach does not discriminate among the possible goals very well. In Logistics, the quality Q of the solution is very low, although it improves significantly in the top 20% set of the ranked goals. The FGR^{FF} heuristic produces solutions within a second, while FGR$^I_{rp}$ and FGR$^+_{rp}$ heuristics find solutions within 2 seconds. However, FGR^{FF} solutions have lower quality Q

compared to FGR^I_{rp} and FGR^+_{rp}, with the exception of the Kitchen domain where it produces better quality Q solutions than FGR^I_{rp}. Although we expected FGR^I_{rp} to perform better that FGR^+_{rp} because of the Interaction information, they give almost the same results. FGR^I_{rp} performs slightly better in the Blocksword domain when the percentage of observed actions is 50% or lower, and in some cases of the Logistics domain. The rest of the cases and for the Intrusion domain, their performance is the same, except in the Kitchen domain where the quality Q of the solutions produced by FGR^+_{rp} increases considerably. As mentioned previously in Section 3.5, the hypothesis is that while constructing a cost-relaxed plan, the algorithm only considers the actions that minimize the cost, but not the Interaction between/among them. Those selected actions might be the same as the ones where the Interaction during cost propagation is not considered. In general, cost-relaxed plan estimates produce fast quality solutions, although they are slightly poorer in quality than FGR^I.

Sequential observations

Table 4.2 summarizes the results when observations are the prefix of the actual plan. This produces essentially the same results as the previous test for all the different techniques. HSP^*_f finds the actual goal with highest probability ($Q = 1$) for all the cases except in Blocksword and Intrusion when the percentage of observed actions is 70%, and Logistics when the percentage of observed actions is 70% or lower. In all the domains except Logistics, $g\text{HSP}^*_f$ also finds the actual goal with highest probability. In Logistics, the value of Q degrades with lower percentages of observed actions. The *spread* increases as the percentage of observed actions decreases because there is not enough information to distinguish among different possible goals. Results for $g\text{HSP}^*_f$ are slightly better than HSP^*_f in Blocksword and Logistics domains. Like in the previous test, $g\text{HSP}^*_f$ is significantly faster than HSP^*_f, because it is generally easier to plan for G than for $G|\overline{O}$.

LAMA and LAMA_G solve all the problems for the Blocksword domain within 1800 seconds, which is actually slower than HSP^*_f. However, the quality Q is high, which means that the actual goal is among the most likely goals for most problems. In the Campus and Grid domains, LAMA finds the optimal solution (same as $g\text{HSP}^*_f$) within 3 seconds, while LAMA_G gets high quality Q solutions within a second. In the Grid domain, LAMA and LAMA_G find the optimal solution within 100 seconds. In the Intrusion domain, LAMA solves all the problems and produces the optimal solution within 1300 seconds. (The computation time decreases to around 55 seconds as the percentage of observed actions drops.)

Table 4.1: Goal recognition with random observations.

Domain		Blocks					Campus					Grid				
Approach	%O	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10
HSP^*_f	T	1080.6	729.06	529.04	405.1	405.14	1.78	1.11	0.55	0.28	0.28	224.67	144.5	79.83	41.47	37.42
	Q	1	1	0.86	0.86	0.86	1	1	1	1	1	1	0.2	0.66	0.86	0.73
	S	1.06	1.13	3.73	9.33	9.33	1	1	1	1	1	1	1.06	1.93	3.6	3.93
$g\text{HSP}^*_f$	T	531.26	446.26	379.81	357.94	357.94	1.22	0.8	0.5	0.32	0.32	114.12	79.76	52.92	39.20	38.8
	Q	1	1	1	1	1	1	1	1	1	1	1	0.2	0.8	0.93	0.8
	S	1.06	1.13	4.06	11.46	11.46	1	1	1	1	1	1	1.2	2.26	3.86	4.2
LAMA	T	1603.24	1522.96	1260.6	1077.62	1082.15	2.74	1.43	0.84	0.52	0.52	294.75	208.70	62.12	20.38	16.52
	Q	0.33	0.8	0.8	0.93	1	1	1	1	1	1	1	0.26	0.8	0.93	0.8
	S	1	1.13	3.86	10.4	10.93	1	1	1	1	1	1	1.2	2.26	3.86	4.2
	Q_{20}	1	1	1	1	1	1	1	1	1	1	1	0.66	0.93	1	0.86
	Q_{50}	1	1	1	1	1	1	1	1	1	1	1	1	0.93	1	0.86
	d	0.042	0.017	0.009	0.001	0.002	0.066	0	0	0	0	0	0	0	0	0
LAMA_G	T	849.08	840.76	814.95	803.04	809.10	0.57	0.56	0.4	0.38	0.38	81.36	40.27	19.26	10.16	9.76
	Q	1	0.8	0.73	0.66	0.46	1	1	1	0.8	0.8	1	0.33	0.73	0.86	0.73
	S	2.26	1.2	3	6.20	4.2	1	1	1	1	1	1	1.8	2	3.4	3.73
	Q_{20}	1	1	1	1	1	1	1	1	0.8	0.8	1	0.66	0.93	1	0.86
	Q_{50}	1	1	1	1	1	1	1	1	1	1	1	1	0.93	1	0.86
	d	0.005	0.02	0.024	0.014	0.015	0.066	4.9×10^5	0.02	0.19	0.19	6.9×10^5	0.029	0.012	0.015	0.015
FGR^IT	T	0.94	1.07	1.28	1.42	1.43	0.27	0.29	0.33	0.38	0.38	106.84	108.53	112.21	114.01	114.71
	Q	1	0.93	0.53	0.13	0.13	1	1	1	0.93	0.93	1	0.2	0.86	0.93	0.86
	S	2.33	2.33	1.53	1.8	1.8	1	1	1	1.06	1.06	1	1.93	3.2	4.2	4.2
	Q_{20}	1	0.93	0.66	0.53	0.53	1	1	1	0.93	0.93	1	0.2	0.86	0.93	0.86
	Q_{50}	1	1	0.8	0.86	0.86	1	1	1	0.93	0.93	1	0.2	0.86	0.93	0.86
	d	0.004	0.016	0.035	0.022	0.022	0.066	2.4×10^5	0.002	0.098	0.098	0.004	0.077	0.07	0.056	0.048
FGR^IET	T	12.59	8.94	4.6	3.68	3.67	0.39	0.39	0.4	0.42	0.42	127.22	125.25	124.59	123.92	124.81
	Q	0.93	0.73	0.46	0.13	0.13	1	1	1	0.93	0.93	1	0.2	0.86	0.93	0.86
	S	1.06	1.13	1.86	1.73	1.73	1	1	1	1.06	1.06	1	2	3.4	4.2	4.2
	Q_{20}	1	0.93	0.66	0.4	0.4	1	1	1	0.93	0.93	1	0.2	0.86	0.93	0.86
	Q_{50}	1	1	0.86	0.8	0.8	1	1	1	0.93	0.93	1	0.26	0.86	0.93	0.86
	d	7.4×10^3	0.04	0.03	0.015	0.015	0.066	5.6×10^5	6.3×10^3	0.098	0.098	0.004	0.078	0.085	0.056	0.048
FGR^I	T	1	1.03	1.25	1.44	1.45	0.27	0.29	0.35	0.39	0.39	107.95	109.83	110.98	115.92	115.77
	Q	1	0.93	0.46	0.13	0.13	1	1	1	0.93	0.93	1	0.26	0.86	0.93	0.86
	S	1.06	6.13	2.33	1.73	1.73	1	1.6	1.53	1.13	1.13	1	2.06	3.80	3.93	3.93
	Q_{20}	1	0.93	0.6	0.46	0.46	1	1	1	0.93	0.93	1	0.26	0.86	0.93	0.86
	Q_{50}	1	1	0.8	0.8	0.8	1	1	1	0.93	0.93	1	0.26	0.86	0.93	0.86
	d	1.1×10^3	0.023	0.035	0.017	0.017	0.066	0.266	0.271	0.09	0.09	0.004	0.071	0.121	0.05	0.041
FGR^IE	T	12.55	8.31	6.09	3.67	3.67	0.39	0.44	0.43	0.43	0.43	127.80	130.82	121.87	126.25	125.93
	Q	0.93	0.73	0.46	0.13	0.13	1	1	1	0.93	0.93	1	0.26	0.86	0.93	0.86
	S	1.06	1.2	2.53	1.66	1.66	1	1	1.06	1.13	1.13	1	2.2	3.93	3.93	3.93
	Q_{20}	1	0.93	0.66	0.33	0.33	1	1	1	0.93	0.93	1	0.26	0.86	0.93	0.86
	Q_{50}	1	1	0.86	0.8	0.8	1	1	1	0.93	0.93	1	0.26	0.86	0.93	0.86
	d	7.4×10^3	0.041	0.031	0.014	0.014	0.066	1×10^1	0.042	0.09	0.09	0.004	0.071	0.121	0.05	0.041
FGR^+	T	0.76	0.6	0.64	0.78	0.78	0.23	0.22	0.25	0.26	0.26	33.02	33.47	33.25	35.71	35.91
	Q	1	1	0.73	0.46	0.46	1	1	1	0.93	0.93	1	0.2	0.8	0.93	0.73
	S	1	11.33	6.6	2.06	2.06	1	1.66	1.46	1.26	1.26	1	1.53	3.33	3.66	3.86
	Q_{20}	1	1	0.86	0.6	0.6	1	1	1	0.93	0.93	1	0.2	0.8	0.93	0.73
	Q_{50}	1	1	0.86	0.73	0.73	1	1	1	0.93	0.93	1	0.2	0.8	0.93	0.73
	d	0.004	0.047	0.045	0.046	0.046	0.066	0.3	0.235	0.2	0.2	1.2×10^3	0.04	0.113	0.055	0.05
h^a_s	T	0.44	0.39	0.37	0.36	0.36	0.05	0.04	0.03	0.03	0.03	0.42	0.33	0.26	0.24	0.24
	Q	1	1	1	1	1	1	1	0.86	0.93	0.93	1	1	1	1	1
	S	20.26	20.26	20.26	20.26	20.26	2	2	1.8	1.86	1.86	6.66	6.66	6.6	6.4	6.4
	Q_{20}	1	1	1	1	1	1	1	0.86	0.93	0.93	1	1	1	1	1
	Q_{50}	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	d	0.086	0.08	0.055	0.031	0.031	0.466	0.466	0.495	0.347	0.347	0.252	0.238	0.208	0.132	0.126
FGR^I_{rp}	T	1	1.02	1.26	1.45	1.44	0.27	0.29	0.35	0.39	0.39	107.30	109.75	110.72	116.48	113.75
	Q	0.86	0.93	0.53	0.4	0.4	1	0.93	1	0.73	0.73	1	0.4	0.8	0.73	0.53
	S	2.33	4.93	3.2	4.66	4.66	1	1.53	1.53	1.06	1.06	1	3	2.26	2.6	2.66
	Q_{20}	0.86	0.93	0.73	0.6	0.6	1	0.93	1	0.73	0.73	1	0.4	0.8	0.73	0.53
	Q_{50}	0.93	0.93	0.8	0.86	0.86	1	0.93	1	0.73	0.73	1	0.4	0.8	0.73	0.53
	d	0.017	0.023	0.04	0.033	0.033	0.066	0.3	0.268	0.233	0.233	1.3×10^4	0.041	0.09	0.068	0.062
FGR^+_{rp}	T	1.33	0.7	0.64	0.78	0.78	0.31	0.26	0.25	0.26	0.26	33.11	33.44	33.10	35.66	35.64
	Q	1	1	0.73	0.33	0.33	1	1	1	0.86	0.86	1	0.4	0.93	0.8	0.6
	S	1	11.33	7	4.13	4.13	1	1.8	1.53	1.46	1.46	1	3.06	3.66	2.73	2.8
	Q_{20}	1	1	0.8	0.46	0.46	1	1	1	0.86	0.86	1	0.4	0.93	0.8	0.6
	Q_{50}	1	1	0.86	0.6	0.6	1	1	1	0.86	0.86	1	0.4	0.93	0.8	0.6
	d	0.004	0.047	0.04	0.044	0.044	0.066	0.366	0.304	0.267	0.267	1.3×10^4	0.045	0.094	0.072	0.066
FGR^{FF}	T	0.65	0.54	0.44	0.41	0.44	0.21	0.2	0.19	0.19	0.18	18.21	18.27	17.04	17.19	17.07
	Q	1	1	0.73	0.06	0.06	1	1	0.66	0.53	0.53	1	0.26	0.8	0.73	0.53
	S	1	13.8	9	2.26	2.26	1	1.8	1.4	1.13	1.13	1	1.73	2.73	2.73	2.8
	Q_{20}	1	1	0.73	0.33	0.33	1	1	0.66	0.53	0.53	1	0.26	0.8	0.73	0.53
	Q_{50}	1	1	0.73	0.46	0.46	1	1	0.66	0.53	0.53	1	0.26	0.8	0.73	0.53
	d	0.004	0.06	0.045	0.053	0.053	0.066	0.366	0.404	0.332	0.332	7×10^5	0.031	0.086	0.088	0.083

(continued)

Domain		Intrusion					Kitchen					Logistics				
Approach	$\%O$	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10
HSP^*_f	T	588.34	414.3	214.2	4.25	4.22	694.67	243.5	60.65	33.3	33.31	44.93	42.02	18.94	9.18	9.18
	Q	1	1	1	1	1	1	1	0.86	0.86	0.86	1	0.93	0.66	0.8	0.8
	S	1	1	1.06	4.46	4.6	1	1	1.2	1.26	1.26	1.06	1.06	2.26	3.6	3.6
$g\text{HSP}^*_f$	T	447.41	281.11	151.37	3.58	3.55	480.39	171.08	49.62	37.93	37.92	36.26	32.46	14.8	7.04	7.04
	Q	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1	1	1.06	4.46	4.6	1	1	1.33	1.4	1.4	1	1.13	2.26	3.6	3.6
FGR^IT	T	0.89	0.77	0.56	0.31	0.31	0.26	0.24	0.2	0.16	0.16	0.88	0.99	1.14	1.25	1.25
	Q	1	1	0.93	0.93	0.93	1	1	1	1	1	1	0.86	0.73	0.6	0.6
	S	1	1	1	4.4	4.53	1	1	1.2	1.26	1.26	1	1.13	1.73	2.8	2.8
	Q_{20}	1	1	1	0.93	0.93	1	1	1	1	1	1	0.93	0.8	0.73	0.73
	Q_{50}	1	1	1	1	1	1	1	1	1	1	1	0.93	0.86	0.86	0.86
	d	1.6×10^5	0.003	0.048	0.014	0.014	1.14×10^4	0.011	0.03	0.058	0.058	5.8×10^4	0.023	0.061	0.045	0.045
FGR^IET	T	1.29	1.18	0.99	0.74	0.74	0.28	0.26	0.23	0.19	0.19	8.51	4.36	3	2.83	2.85
	Q	1	1	0.93	0.93	0.93	1	1	1	1	1	0.86	0.66	0.66	0.6	0.6
	S	1	1	1	4.4	4.53	1	1	1.2	1.26	1.26	1.13	1.4	1.8	2.8	2.8
	Q_{20}	1	1	1	0.93	0.93	1	1	1	1	1	0.86	0.8	0.8	0.73	0.73
	Q_{50}	1	1	1	1	1	1	1	1	1	1	0.93	1	0.86	0.86	0.86
	d	1.6×10^5	0.003	0.048	0.014	0.014	1.14×10^4	0.011	0.03	0.058	0.058	0.017	0.054	0.068	0.045	0.045
FGR^I	T	0.89	0.49	0.21	0.21	0.21	0.26	0.19	0.14	0.13	0.13	0.88	1.01	1.19	1.26	1.26
	Q	1	1	0.93	0.93	0.93	1	1	1	1	1	1	0.86	0.53	0.6	0.6
	S	1	1	1	4.4	4.53	1	1	1.2	1.26	1.26	1	1.26	1.6	2.46	2.46
	Q_{20}	1	1	1	0.93	0.93	1	1	1	1	1	1	0.93	0.66	0.73	0.73
	Q_{50}	1	1	1	1	1	1	1	1	1	1	1	0.93	0.8	0.86	0.86
	d	1.6×10^5	0.003	0.048	0.014	0.014	1.14×10^4	0.011	0.03	0.058	0.058	5.8×10^4	0.025	0.073	0.043	0.043
FGR^IE	T	1.28	0.90	0.63	0.63	0.63	0.28	0.22	0.16	0.16	0.16	8.51	3.57	3.08	2.82	2.82
	Q	1	1	0.93	0.93	0.93	1	1	1	1	1	0.86	0.66	0.6	0.6	0.6
	S	1	1	1	4.4	4.53	1	1	1.2	1.26	1.26	1.13	1.26	1.73	2.46	2.46
	Q_{20}	1	1	1	0.93	0.93	1	1	1	1	1	0.86	0.8	0.66	0.73	0.73
	Q_{50}	1	1	1	1	1	1	1	1	1	1	0.93	1	0.73	0.86	0.86
	d	1.6×10^5	0.003	0.048	0.014	0.014	1.14×10^4	0.011	0.03	0.058	0.058	0.017	0.043	0.072	0.041	0.041
FGR^+	T	0.88	0.49	0.19	0.19	0.19	0.25	0.19	0.13	0.12	0.12	0.8	0.4	0.44	0.5	0.5
	Q	1	1	1	1	0.93	1	1	1	1	1	1	0.53	0.46	0.6	0.6
	S	1	1.13	1.13	4.06	3.93	1	1	1.33	1.4	1.4	1	3	2.13	2.93	2.93
	Q_{20}	1	1	1	1	0.93	1	1	1	1	1	1	0.6	0.6	0.66	0.66
	Q_{50}	1	1	1	1	1	1	1	1	1	1	1	0.66	0.73	0.8	0.8
	d	0.011	0.045	0.074	0.02	0.018	1×10^6	0.011	2×10^6	0.007	0.007	7.5×10^4	0.07	0.077	0.052	0.052
h^a_s	T	0.46	0.32	0.27	0.25	0.25	0.06	0.05	0.04	0.04	0.04	0.34	0.32	0.3	0.29	0.3
	Q	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.93	0.93
	S	16.66	16.06	16.66	16.66	16.66	3	3	3	3	3	15	15	14.8	14.66	14.66
	Q_{20}	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.93	0.93
	Q_{50}	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.93	0.93
	d	0.117	0.117	0.114	0.071	0.07	0.414	0.407	0.274	0.216	0.216	0.115	0.11	0.093	0.068	0.068
FGR^I_{rp}	T	0.88	0.49	0.21	0.21	0.2	0.26	0.19	0.14	0.13	0.13	0.88	1	1.19	1.26	1.26
	Q	1	1	0.93	1	1	1	0.26	0.26	0.26	0.26	1	0.66	0.33	0.53	0.53
	S	1	7.46	2.4	4.6	4.6	1	1.53	1	1	1	1	1.66	2.53	2.86	2.86
	Q_{20}	1	1	0.93	1	1	1	0.26	0.26	0.26	0.26	1	0.8	0.53	0.86	0.86
	Q_{50}	1	1	1	1	1	1	1	1	1	1	1	0.86	0.66	0.86	0.86
	d	0	0.053	0.042	0.024	0.02	4.2×10^4	0.45	0.19	0.166	0.166	$4.8\times^4$	0.05	0.079	0.026	0.026
FGR^+_{rp}	T	0.89	0.51	0.19	0.19	0.19	0.29	0.21	0.13	0.12	0.13	0.97	0.43	0.45	0.5	0.5
	Q	1	1	0.93	1	1	1	1	1	1	1	1	0.53	0.4	0.66	0.66
	S	1	7.46	2.4	4.6	4.6	1	1.93	1.46	1.4	1.4	1	5.8	2.66	4.00	4.00
	Q_{20}	1	1	0.93	1	1	1	1	1	1	1	1	0.6	0.4	0.66	0.66
	Q_{50}	1	1	1	1	1	1	1	1	1	1	1	0.66	0.53	0.8	0.8
	d	0	0.053	0.042	0.024	0.02	3×10^4	0.262	0.068	0.014	0.014	0.002	0.083	0.08	0.052	0.052
FGR^{FF}	T	0.44	0.26	0.10	0.09	0.08	0.13	0.10	0.08	0.06	0.08	0.67	0.35	0.19	0.27	0.19
	Q	1	0.66	0.26	0.2	0.2	0.53	0.53	0.4	0.33	0.33	0.8	0.4	0.33	0.33	0.33
	S	16.66	9.93	1.33	2.66	2.66	2.06	1.53	1.26	1	1	1.2	3.86	1.6	2.86	2.86
	Q_{20}	1	0.66	0.6	0.26	0.26	0.53	0.53	0.4	0.33	0.33	0.8	0.46	0.4	0.33	0.33
	Q_{50}	1	0.73	0.73	0.66	0.66	1	1	0.8	0.86	0.86	0.93	0.53	0.6	0.8	0.8
	d	0.117	0.112	0.093	0.07	0.068	0.435	0.381	0.248	0.2	0.2	0.05	0.103	0.092	0.067	0.067

LAMA$_G$ solves all the problems in the Intrusion domain within 3 seconds. The quality Q of the solutions is very high. However, the *spread* is high, which means that it does not discriminate very well among possible goals. In the Kitchen domain, the quality Q of solutions given by LAMA are slightly better than the ones produced by LAMA$_G$. However, LAMA$_G$ is faster than LAMA. In the Logistics domain, LAMA solves all the problems and produces the optimal solution within 770 seconds. (The computation time decreases to around 12 seconds as the percentage of observed actions drops.) LAMA$_G$ solves all the problems within 5 seconds. When the percentage of observed actions is high, the quality Q of the solution is good and degrades as the percentage of observed actions drops.

Because the observations are the prefix of the actual plan, the solutions produced by FGR^{I}T and FGR^I, and FGR^{I}TE and FGR^{I}E are the same. The reason for this is that the assigned *est* to each observed actions, is the same as the actual time step of the action. Consequently, Table 4.2 only shows the results for FGR^I and FGR^{I}E. The FGR^I approach solves all the problems within 1.5 seconds, while FGR^{I}ET solves them within 12 seconds. In general, both approaches quickly provide high quality Q solutions when the percentage of observed actions is high, and degrade a bit as the percentage of observed actions drops. Considering the use of Interaction estimates, FGR^I gets higher quality solutions than FGR$^+$ except for some cases of Blocksword domain for the 30% and 10% of observed actions sets, and some cases of the Kitchen domain for the 10% of observed actions set.

As noted before, heuristics based on relaxed plans produce quick solutions. Specifically, the h^s_a heuristic repeatedly solves all the problems within a second, producing solutions where the actual goal is among the most likely goals for all the problems, but with a high *spread*. This happens for all the domains except in Logistics, where the quality Q of the solution is very low, but it improves significantly in the top 20% set of the ranked goals. The FGR^{FF} heuristic produces solutions within a second, while FGR$^I_{rp}$ and FGR$^+_{rp}$ heuristics find solutions within 1.5 seconds. However, FGR^{FF} solutions have lower quality Q compared to FGR$^I_{rp}$ and FGR$^+_{rp}$ with the exception of the Kitchen domain where, again, it produces better quality Q solutions than FGR$^I_{rp}$. The FGR$^I_{rp}$ heuristic performs slightly better than FGR$^+_{rp}$ in the Grid domain when the percentage of observed actions is 70% or lower, and in some cases of the Intrusion and Logistics domains. In the rest of the cases, their performance is the same, except in the Kitchen domain where the quality of the solutions produced by FGR$^+_{rp}$ increases considerably.

Table 4.2: Goal recognition with sequential observations.

Domain		Blocks					Campus					Grid				
Approach	$\%O$	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10
HSP^*_f	T	908.52	677.66	612.36	432.56	321.31	1.77	1.03	0.66	0.45	0.27	259.53	164.18	121.34	87.53	33.94
	Q	1	1	1	0.66	1	1	1	1	1	1	1	1	1	1	1
	S	1.06	2.2	3.13	8.86	14.73	1	1	1	1	1.13	6.66	6.66	6.66	6.66	6.66
$g\text{HSP}^*_f$	T	531.02	427.78	402.37	369.21	358.32	0.94	0.58	0.41	0.31	0.23	119.23	82.18	64.6	50.97	40.64
	Q	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	S	1.06	2.2	3.13	11.86	14.73	1	1	1	1	1.13	6.66	6.66	6.66	6.66	6.66
LAMA	T	1596.43	1471.80	1263.15	1085.34	1037.39	2.8	1.39	0.8	0.62	0.51	110.11	69.85	56.61	32.84	20.21
	Q	0.73	0.73	0.86	0.93	0.93	1	1	1	1	1	1	1	1	1	1
	S	1.06	2.13	2.86	11	13.73	1	1	1	1	1.13	6.66	6.66	6.66	6.66	6.66
	Q_{20}	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Q_{50}	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	d	0.026	0.021	0.005	2.1×10^3	1.1×10^3	0	0	0	0	0	0	0	0	0	0
LAMA_G	T	860.44	846.22	825.27	817.13	785.30	0.62	0.54	0.44	0.41	0.37	84.9	47.99	36.36	19.21	11.76
	Q	0.66	0.86	0.6	0.4	0.46	1	1	1	0.8	0.73	1	1	1	1	1
	S	1	1.53	1.6	3.73	5.86	1	1	1	1.06	1.06	6.66	6.66	6.66	6.66	6.66
	Q_{20}	1	1	1	1	1	1	1	1	0.8	0.73	1	1	1	1	1
	Q_{50}	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	d	0.026481	0.017978	0.028515	0.016922	0.016054	3×10^6	4.8×10^4	5.9×10^3	0.138	0.23	0	0	0	0	0
FGR^I	T	1	1.12	1.22	1.41	1.44	0.27	0.32	0.34	0.38	0.41	112.84	116.80	118.43	119.85	120.59
	Q	1	0.86	0.8	0.2	0.06	1	1	1	1	1	1	0.8	0.86	0.8	1
	S	1.06	3.73	2.46	2.06	1.46	1	1	1	1	1.13	1	1.4	1.53	2.4	4.46
	Q_{20}	1	0.93	0.93	0.46	0.33	1	1	1	1	1	1	0.93	0.86	0.8	1
	Q_{50}	1	0.93	0.93	0.8	0.66	1	1	1	1	1	1	1	1	0.93	1
	d	1.1×10^3	0.024	0.028	0.021	0.015	2×10^6	1.8×10^4	0.001	0.022	0.14	0.22	0.19	0.16	0.096	0.03
FGR^IE	T	12.6	5.98	4.4	3.63	3.58	0.39	0.39	0.4	0.42	0.44	133.03	128.05	127.42	129.38	129.83
	Q	0.93	0.86	0.8	0.2	0.06	1	1	1	1	1	1	0.8	0.86	0.8	1
	S	1.06	1.4	1.33	2	1.4	1	1	1	1	1.13	1	1.4	1.53	2.4	4.46
	Q_{20}	1	0.93	0.93	0.4	0.13	1	1	1	1	1	1	0.93	0.86	0.8	1
	Q_{50}	1	0.93	1	0.86	0.66	1	1	1	1	1	1	1	1	0.93	1
	d	7.4×10^3	0.023	0.021	0.014	0.011	1×10^6	6.8×10^5	6.1×10^4	0.022	0.14	0.217	0.186	0.156	0.09	0.03
FGR^+	T	0.76	0.59	0.61	0.75	0.78	0.23	0.24	0.27	0.25	0.27	33.33	33.54	34.34	35.48	36
	Q	1	0.93	0.66	0.46	0.4	1	1	1	1	1	1	0.73	0.73	0.66	0.66
	S	1	11.53	5	1.66	1.6	1	1.2	1	1.2	1.2	1	1.53	1.66	2	2.2
	Q_{20}	1	0.93	0.86	0.8	0.66	1	1	1	1	1	1	0.73	0.8	0.66	0.66
	Q_{50}	1	1	0.86	0.86	0.86	1	1	1	1	1	1	0.93	0.86	0.86	0.73
	d	0.004	0.045	0.042	0.05	0.055	2×10^6	0.1	9.3×10^4	0.126	0.093	0.248	0.2	0.181	0.108	0.064
h^a_s	T	0.43	0.39	0.39	0.36	0.36	0.05	0.04	0.03	0.03	0.03	0.42	0.32	0.28	0.25	0.24
	Q	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	S	20.26	20.26	20.26	20.26	20.26	2	2	2	2	2	6.66	6.66	6.66	6.66	6.66
	Q_{20}	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Q_{50}	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	d	0.086	0.08	0.055	0.031	0.031	0.466	0.466	0.495	0.347	0.347	0.252	0.238	0.208	0.132	0.126
FGR^I_{rp}	T	1	1.12	1.22	1.41	1.45	0.27	0.32	0.34	0.38	0.41	112.45	117.33	118.13	119.76	120.72
	Q	0.86	0.73	0.66	0.26	0.2	1	1	1	1	0.93	1	0.93	0.93	0.93	0.8
	S	2.33	3.86	3.26	4.33	5.66	1	1	1	1	1.13	1	1.53	1.86	2.53	3.4
	Q_{20}	0.86	0.73	0.66	0.46	0.46	1	1	1	1	0.93	1	1	0.93	0.93	0.8
	Q_{50}	0.93	1	1	0.93	1	1	1	1	1	0.93	1	1	1	1	0.8
	d	0.017	0.03	0.044	0.04	0.021	2×10^6	1.8×10^4	0.001	0.024	0.134	0.25	0.21	0.193	0.143	0.053
FGR^+_{rp}	T	0.76	0.59	0.61	0.75	0.78	0.23	0.24	0.27	0.25	0.27	33.33	33.54	34.34	35.48	36.00
	Q	1	0.93	0.66	0.46	0.4	1	1	1	1	1	1	0.73	0.73	0.66	0.66
	S	1	11.53	5	1.66	1.6	1	1.2	1	1.2	1.2	1	1.53	1.66	2	2.2
	Q_{20}	1	0.93	0.86	0.8	0.66	1	1	1	1	1	1	0.73	0.8	0.66	0.66
	Q_{50}	1	1	0.86	0.86	0.86	1	1	1	1	1	1	0.93	0.86	0.86	0.73
	d	0.004	0.045	0.042	0.05	0.055	2×10^6	0.1	9.3×10^4	0.126	0.093	0.248	0.2	0.181	0.108	0.064
FGR^{FF}	T	0.65	0.42	0.41	0.35	0.39	0.21	0.18	0.25	0.18	0.2	18.33	17.31	17.03	16.84	16.58
	Q	1	1	0.8	0.2	0.06	1	1	0.73	0.4	0.4	1	1	0.93	0.93	0.8
	S	1	14	8.93	4.13	1.86	1	1.4	1.2	1	1	1	2.2	1.73	2.46	3.53
	Q_{20}	1	1	0.8	0.6	0.46	1	1	0.73	0.4	0.4	1	1	0.93	0.93	0.8
	Q_{50}	1	1	0.8	0.66	0.53	1	1	0.73	0.4	0.4	1	1	0.93	0.93	0.86
	d	0.004	0.053	0.046	0.048	0.048	2×10^6	0.2	0.238	0.375	0.218	0.25	0.197	0.198	0.14	0.054

(continued)

Domain		Intrusion					Kitchen					Logistics				
Approach	$\%O$	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10
HSP$^*_f$	T	591.93	147.18	40.68	7.21	1.38	694.32	178.74	105.86	57.87	37.56	63.54	51.24	20.34	9.46	9.47
	Q	1	1	1	1	1	1	1	1	1	1	1	0.93	0.46	0.33	0.33
	S	1	1	1.06	1.2	5	1	1	1	1.4	1.4	1	1.13	1.46	1.46	1.46
gHSP$^*_f$	T	450.05	113.91	29.56	4.56	1.24	256.74	64.89	52.46	38.93	32.49	36.3	32.48	14.82	7.05	7.04
	Q	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1	1	1.06	1.2	5	1	1	1	1.4	1.66	1	1.13	2.26	3.6	3.6
LAMA	T	1302.52	551.54	213.17	104.16	55.12	357.29	164.37	133.43	107.19	97.92	772.77	400.97	50.44	11.43	11.45
	Q	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1	1	1.06	1.2	5	1	1	1	1.4	1.66	1	1.13	2.26	3.6	3.6
	Q_{20}	1	1	1	1	1	1	1	1	1	1	1	1	0.93	1	1
	Q_{50}	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	d	0	0	3.6×10^5	0	0	1.8×10^4	0	0	0	0	0	0	0	0	0
LAMA$_G$	T	3.35	2.63	2.40	2.19	2.02	0.42	0.36	0.34	0.32	2.69	5.53	5.05	4.57	4.39	4.49
	Q	1	1	0.93	1	1	1	1	1	1	1	1	0.8	0.4	0.46	0.46
	S	16.66	10	1.86	2.33	5	1.53	1.13	1	1.4	1.66	1	1.2	1.8	2.93	2.93
	Q_{20}	1	1	1	1	1	1	1	1	1	1	1	1	0.66	0.6	0.66
	Q_{50}	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.86	0.86
	d	8×10^6	0.013	0.026	0.041	0	0.001	7.4×10^4	5×10^3	0.037	0	1.3×10^3	0.045	0.07	0.063	0.063
FGR^I	T	0.88	0.5	0.3	0.2	0.2	0.26	0.19	0.16	0.13	0.13	0.88	1.01	1.18	1.26	1.26
	Q	1	0.93	0.93	1	1	1	1	1	1	0.73	1	0.86	0.53	0.6	0.6
	S	1	1	1	1.2	5	1	1	1	1.4	1.4	1	1.26	1.6	2.46	2.46
	Q_{20}	1	1	0.93	1	1	1	1	1	1	0.73	1	0.93	0.66	0.73	0.73
	Q_{50}	1	1	1	1	1	1	1	1	1	1	1	0.93	0.8	0.86	0.86
	d	1.6×10^5	0.004	0.003	1.6×10^4	0	6.6×10^5	6×10^3	0.026	0.034	0.06	5.8×10^4	0.025	0.073	0.043	0.043
FGR^{I}E	T	1.28	0.91	0.72	0.63	0.63	0.28	0.21	0.18	0.16	0.16	8.51	3.58	3.08	2.82	2.82
	Q	1	0.93	0.93	1	1	1	1	1	1	0.73	0.86	0.66	0.6	0.6	0.6
	S	1	1	1	1.2	5	1	1	1	1.4	1.4	1.13	1.26	1.73	2.46	2.46
	Q_{20}	1	1	0.93	1	1	1	1	1	1	0.73	0.86	0.8	0.66	0.73	0.73
	Q_{50}	1	1	1	1	1	1	1	1	1	1	0.93	1	0.73	0.86	0.86
	d	1.6×10^5	0.004	0.003	1.6×10^4	0	6.6×10^5	6×10^3	0.026	0.034	0.06	0.017	0.043	0.072	0.041	0.041
FGR$^+$	T	0.88	0.49	0.3	0.19	0.19	0.25	0.19	0.15	0.12	0.12	0.81	0.4	0.44	0.5	0.5
	Q	1	1	0.93	0.93	0.93	1	1	1	1	1	1	0.53	0.46	0.6	0.6
	S	1	1	1.2	1.06	3.66	1	1	1	1.4	1.66	1	3	2.13	2.93	2.93
	Q_{20}	1	1	1	1	1	1	1	1	1	1	1	0.6	0.6	0.66	0.66
	Q_{50}	1	1	1	1	1	1	1	1	1	1	1	0.66	0.73	0.8	0.8
	d	0.011	0.034	0.043	0.055	0.018	1.7×10^4	1×10^6	1×10^6	1×10^6	0	7.5×10^4	0.07	0.077	0.052	0.052
h^a_s	T	0.47	0.32	0.3	0.26	0.26	0.06	0.05	0.05	0.04	0.04	0.35	0.32	0.3	0.31	0.3
	Q	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.93	0.93
	S	16.66	16.06	16.66	16.66	16.66	3	3	3	3	3	15	15	14.8	14.66	14.66
	Q_{20}	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.93	0.93
	Q_{50}	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.93	0.93
	d	0.117	0.117	0.114	0.071	0.07	0.414	0.407	0.274	0.216	0.216	0.115	0.11	0.093	0.068	0.068
FGR$^I_{rp}$	T	0.88	0.49	0.3	0.2	0.2	0.26	0.19	0.16	0.13	0.13	0.88	1.01	1.19	1.26	1.27
	Q	1	1	1	1	1	1	0.26	0.26	0.26	0.26	1	0.66	0.33	0.53	0.53
	S	1	5.46	9.06	4.8	5.13	1	1.53	1.53	1	1	1	1.66	2.53	2.86	2.86
	Q_{20}	1	1	1	1	1	1	0.26	0.26	0.26	0.26	1	0.8	0.53	0.86	0.86
	Q_{50}	1	1	1	1	1	1	1	1	1	1	1	0.86	0.66	0.86	0.86
	d	0	0.042	0.085	0.072	0.004	6×10^4	0.436	0.361	0.164	0.183	$4.8\times^4$	0.05	0.079	0.026	0.026
FGR$^+_{rp}$	T	0.88	0.49	0.3	0.19	0.19	0.25	0.19	0.15	0.12	0.12	0.81	0.4	0.44	0.5	0.5
	Q	1	1	0.93	0.93	0.93	1	1	1	1	1	1	0.53	0.46	0.6	0.6
	S	1	1	1.2	1.06	3.66	1	1	1	1.4	1.66	1	3	2.13	2.93	2.93
	Q_{20}	1	1	1	1	1	1	1	1	1	1	1	0.6	0.6	0.66	0.66
	Q_{50}	1	1	1	1	1	1	1	1	1	1	1	0.66	0.73	0.8	0.8
	d	0.011	0.034	0.043	0.055	0.018	1.7×10^4	1×10^6	1×10^6	1×10^6	0	7.5×10^4	0.07	0.077	0.052	0.052
FGR^{FF}	T	0.44	0.29	0.18	0.09	0.09	0.13	0.11	0.08	0.07	0.07	0.67	0.31	0.19	0.22	0.26
	Q	1	0.73	0.13	0.13	0.13	0.53	0.53	0.53	0.53	0.53	0.8	0.4	0.33	0.33	0.33
	S	16.66	12.4	2.26	2.26	2.26	2.06	1.53	1.53	1	1	1.2	3.86	1.6	2.86	2.86
	Q_{20}	1	0.73	0.13	0.13	0.13	0.53	0.53	0.53	0.53	0.53	0.8	0.46	0.4	0.33	0.33
	Q_{50}	1	0.73	0.6	0.6	0.6	1	1	1	1	1	0.93	0.53	0.6	0.8	0.8
	d	0.117	0.115	0.115	0.105	0.051	0.435	0.373	0.331	0.103	0.155	0.05	0.103	0.092	0.067	0.067

4.2 Practical goal recognition for ISS crew activities

Recognizing an agent's goals from some or all of the agent's observed actions is an important technological capability for applications that involve cooperation between humans and machines, such as intelligent tutoring systems (Brown et al., 1977), monitoring user's needs (Pollack et al., 2003; Kautz et al., 2002), smart environments (Wu et al., 2007), and intelligent personal assistants (Weber and Pollack, 2008). For human space exploration, there has been increasing interest in the development of intelligent robots that can assist with astronauts activities. An example of this is the use of free-flying robots to assist astronauts aboard the International Space Station (ISS). The Smart Synchronized Position Hold, Engage, Reorient Experimental Satellites (SPHERES), free-flying robots currently aboard the ISS, are capable of functioning autonomously, conducting zero gravity robotics experiments, carrying mobile sensors, and inspecting items using a built in camera. They can also potentially offer supervision, advice, and support for the astronauts. However, goal recognition is essential for this capability, because an astronaut often doesn't fully communicate his or her intentions and objectives to a robotic assistant or to others in the vicinity.

In this section, we describe the ISS Crew Activities Domain (ISS-CAD) that focuses on maintenance tasks for the Environmental Control and Life Support System (ECLSS), and present an empirical evaluation of different goal recognition techniques applied to this domain.

4.2.1 ISS Crew Activities domain

The Environmental Control and Life Support System (ECLSS) is a critical subsystem aboard the ISS that provides oxygen and potable water, removes carbon dioxide, distributes cabin air between modules, maintains cabin temperature, humidity, and pressure levels, monitors and controls nitrogen, oxygen, carbon dioxide, methane, hydrogen, and water vapor levels, and filters particulates and microorganisms from the air. Maintaining the ECLSS equipment involves regular inspection, repair, and replacement of the components, which requires retrieving and using various tools, measurement instruments, and replacement parts from stowage in different modules aboard the space station.

The ISS is divided into modules where the ECLSS subsystems are located. An ECLSS subsystem, such as the Air Revitalization System (ARS), or Water Recovery System (WRS), may be present in more than one module. The ISS-CAD domain is concerned with the maintenance tasks that the astronaut must conduct for the ECLSS subsystems, and the astronaut's health. We developed our model of activities based on descriptions provided by Bagdigian (2008). The operators described in the domain involve astronauts moving between modules, taking, replacing, repairing, or inspecting components, measuring air and humidity levels using different instruments, cleaning modules, exercising, eating, monitoring brain activity, body radiation, blood pressure, etc. Propositions in the domain model the connection among modules, the location of astronauts, subsystems, tools, and components, the availability of instruments and components, the state of the electrical equipment, and the results of repair, replace, measure, etc, operators.

The ISS-CAD domain (E-Martín et al., 2015c) has a total of 47 operators and 96 goals. The length of a plan solution depends on the number of tasks to be performed, and the number of astronauts involved. Figure 4.3 shows a simplified description of an ISS-CAD problem (see Appendix B for the complete domain description). It consists of three actions (move), (get-replacement), and (replace-component), and a flight engineer (fe1). There are three ISS modules: Harmony, Destiny, and Unity (Harmony and Destiny are connected, and Destiny and Unity are connected). The goal is to replace a sensor in the ARS subsystem located in the Destiny module. A valid plan for the problem in Figure 4.3 is the following sequence of actions:

$$\pi = \left(\begin{array}{l} \text{(move fe1 Harmony Destiny),} \\ \text{(move fe1 Destiny Unity),} \\ \text{(get-replacement sensor fe1 Unity),} \\ \text{(move fe1 Unity Destiny)} \\ \text{(replace-component sensor ars Destiny fe1)} \end{array} \right)$$

4.2.2 Experimental evaluation on the ISS Crew Activities domain

This section shows results on the ISS Crew Activities Domain, which consists of 30 problems that were automatically generated. The location of astronauts, subsystems, tools, and components, the state (on, off, enabled, disabled) of the electrical equipment, and the availability of tools and components at the initial state for each problem were randomly generated. Connection between locations are set according to the ISS assembly. The hypotheses file consists of all possible goals that astronauts can perform. The actual goal for each problem was chosen at random with the priors on the goal sets assumed to be uniform. As in the previous section, for each problem we ran the LAMA planner to solve the problem for the actual goal. We have conducted a test where the observed sequence

```
(define (domain ISS-CAD)
 (:requirements :strips :typing :action-costs)
 (:types crew module system component tool)
 (:predicates (connected ?m1 ?m2 - module) (at ?c - crew ?m - module)
              (in ?c - component ?s - system ?m - module)
              (replacement-in ?t - component ?m - module)
              (taken-replacement ?t - component ?c - crew)
              (replaced ?cp - component ?s - system ?m - module ?c - crew))

 (:functions (total-cost))

 (:action move
  :parameters (?c - crew ?m1 ?m2 - module)
  :precondition (and (at ?c ?m1) (connected ?m1 ?m2))
  :effect (and (not (at ?c ?m1)) (at ?c ?m2) (increase (total-cost) 1)))

 (:action get-replacement
  :parameters (?t - component ?c - crew ?m - module)
  :precondition (and (at ?c ?m) (replacement-in ?t ?m))
  :effect (and (taken-replacement ?t ?c) (increase (total-cost) 20)))

 (:action replace-component
  :parameters (?o - component ?s - system ?m - module ?c - crew)
  :precondition (and (at ?c ?m) (in ?o ?s ?m) (taken-replacement ?o ?c))
  :effect (and (replaced ?o ?s ?m ?c) (not (taken-replacement ?o ?c))
               (increase (total-cost) 20)))
```

```
(define (problem ISS-CAD-1)
 (:domain ISS-CAD)
 (:objects fe1 - crew Harmony Destiny Unity - module
           ars - system sensor - component
 (:init (connected Harmony Destiny) (connected Destiny Harmony)
        (connected Destiny Unity) (connected Unity Destiny)
        (replacement-in sensor Unity) (in sensor ars Destiny)
        (at fe1 Harmony) (= (total-cost) 0))
 (:goal (replaced sensor ars Destiny fe1))
 (:metric minimize (total-cost)))
```

Figure 4.3: A fragment of a PDDL domain and problem description on the ISS Crew Activities Domain.

was the prefix of the actual plan, since we assume that the observed action sequence is provided by a free-flying robot monitoring the astronaut's activities. In this case, the observed sequence ranges from 100% of the actions, down to 50% of the actions. We did not include a test where the observed sequence is lower than 50% because the quality of the solutions were poor for all the approaches. The reason for this is that the first steps of the plan solution only correspond to actions where the astronaut moves among the different modules collecting tools and equipment, which is usually not enough information to discriminate among the possible goals.

For the purpose of this test, Ramírez technique is evaluated using the heuristic estimator h_a^s, and the HSP$_f^*$ and LAMA$_G$ planners.

We have performed two different sets of problems. In the first one, each goal in the hypotheses set involves a single astronaut in the domain definition. While in the second each goal in the hypotheses set involves more than one astronaut whose tasks interfere with each other.

Single astronaut tasks

The hypotheses set includes 96 goals that a single astronaut could potentially be doing. Table 4.3 shows the results when the sequence of observed actions is the prefix of actual plan (*sequential observations*). gHSP$_f^*$ finds the actual goal with the highest probability ($Q = 1$), and the spread increases as the percentages of observed actions drops. However, the computation time ranges from 687 to 1058 seconds, which makes the use of this planner impractical in this domain. LAMA$_G$ solves all the problems within 300 seconds and produces high quality Q solutions, which means that the actual goal is among the most likely goals for most of the problems. However, the spread increases considerably for lower percentages, which means that it does not discriminate well for lower numbers of observed actions. The h_a^s heuristic solves all the problems within 55 seconds, and finds the actual goal with the highest probability for all problems. However, it does not discriminate among the possible goals very well since the spread is very high for all the percentages. FGR^I and FGR$^+$ solve all the problems within 113 and 46 seconds respectively, and find the actual goal with the highest probability. In addition, the spread of the solution is very close to the optimal solution computed by gHSP$_f^*$, being slightly better for FGR^I. This means that these heuristic approaches discriminate among the possible goals quite well. FGR$^+$ is faster than FGR^I, and the quality Q of the solutions is only a bit lower. Interaction information helps to compute more accurate estimates of cost when subgoals interfere or are synergistic with each other. It appears that the reason FGR$^+$ works so well for these problems is that the degree of interference and/or synergy among subgoals is low.

We have performed a second test where the observed sequence was randomly taken to be a subset of the plan solution, ranging from 100% of the actions, down to 30% of the actions, even though it does not exactly fit the ISS-CAD problem where the free-flying robot is progressively monitoring the astronaut. Table 4.3

shows the results of this test, which produces essentially the same results as the previous test for all the different techniques. $g\text{HSP}^*_f$ finds the actual goal with the highest probability ($Q = 1$), but the computation time is high. LAMA_G solves all the problems within 300 seconds. It provides high quality Q solutions that degrade as the percentage of observed actions decreases. The h^s_a heuristic solves all the problems within 54–58 seconds, and finds high quality Q solutions. However, it does not discriminate among the possible goals very well (high S values). FGR^I solves all the problems within 88–105 seconds. The quality Q of the solutions decreases as the percentage of observed actions drops, and, in general, is lower than the solutions provided by LAMA_G (although the spread is lower for FGR^I). FGR^+ solves all the problems within 30-40 seconds. It finds a high quality Q solutions, but with relatively high spread for low percentages of observed actions. In general, when the sequence of observed actions is randomly chosen, solutions are lower quality Q than when the observed sequence is sequential.

Table 4.3: ISS-CAD Evaluation Single Astronaut Tasks.

		Sequential Observations						**Random Observations**			
Approach	$\%O$	100	90	80	70	60	50	100	70	50	30
$g\text{HSP}^*_f$	T	1058.79	919.33	889.02	733.28	692.01	687.14	1039.62	1066.83	937.19	789.86
	Q	1	0.96	0.96	0.96	0.93	0.93	1	1	0.96	0.93
	S	1.03	7.83	21.96	38.7	43.43	44.56	1.03	5.76	17.8	20.56
LAMA_G	T	300.33	301.89	298.26	296.41	294.83	293.58	192.26	188.87	186.56	185.91
	Q	1	0.8	0.7	0.8	0.86	0.86	1	0.83	0.76	0.76
	S	1.1	4.7	15.63	42.36	68.2	71.86	1.16	6.16	20.1	39.7
	Q_{20}	1	0.86	0.8	0.83	0.9	0.86	1	0.93	0.96	0.96
	Q_{50}	1	1	0.96	0.96	0.9	0.86	1	1	0.96	0.96
	d	0.01	0.012	0.01	0.008	0.004	0.003	0.01	0.011	0.011	0.01
h^s_a	T	56.93	56.65	56.31	55.71	55.21	55.15	58.15	56.31	54.6	54.23
	Q	0.93	0.93	0.93	0.93	0.93	0.93	0.86	0.86	0.86	0.86
	S	84.9	84.9	84.9	84.9	84.9	84.9	84.83	84.83	84.83	84.83
	Q_{20}	0.93	0.93	0.93	0.93	0.93	0.93	0.86	0.86	0.86	0.86
	Q_{50}	0.93	0.93	0.93	0.93	0.93	0.93	0.86	0.86	0.86	0.86
	d	0.02	0.015	0.009	0.005	0.004	0.004	0.02	0.017	0.012	0.012
FGR^I	T	100.81	103.36	107.58	111.88	112.72	113.01	87.58	93.85	101.51	106.06
	Q	1	0.66	0.66	0.56	0.56	0.56	1	0.76	0.5	0.43
	S	1.03	7.73	14.86	21.46	20.16	21.6	1.1	4.73	13.16	24.76
	Q_{20}	1	0.7	0.7	0.56	0.56	0.56	1	0.76	0.5	0.43
	Q_{50}	1	0.96	0.9	0.8	0.8	0.76	1	0.9	0.76	0.7
	d	0.0008	0.006	0.006	0.006	0.005	0.005	0.01	0.013	0.015	0.013
FGR^+	T	37.4	38.07	40.74	44.56	45.98	46.38	29.02	31.81	36.77	41.11
	Q	1	0.96	0.96	0.96	0.96	0.93	1	0.96	1	0.83
	S	2.03	15.56	27.8	45.66	47.33	47.33	2.63	13.13	37.26	39.53
	Q_{20}	1	0.96	0.96	0.96	0.96	0.96	1	0.96	1	0.86
	Q_{50}	1	1	1	0.96	0.96	0.96	1	0.96	1	0.86
	d	0.004	0.005	0.005	0.002	0.002	0.002	0.014	0.014	0.012	0.011

Multiple astronauts tasks

The hypotheses set includes 30 goals that two astronauts could potentially be doing. The tasks assigned to each astronaut are independent, but interfere with each other. Table 4.4 shows the results when the sequence of observed actions is the prefix of the actual plan (*sequential observations*). gHSP$^*_f$ finds the actual goal with the highest probability ($Q = 1$), and the spread increases as the percentages of observed actions drops. However, the computation time is again high enough that it makes the use of this planner impractical in this domain. LAMA$_G$ solves all the problems within 1400 seconds, which is quite high and makes the approach impractical when multiple astronauts are involved. Nevertheless, it produces high quality Q solutions ($Q = 1$) when the percentage of observed actions is high, and degrades as the percentage of observed actions decreases. The spread is low for all the cases, except for the set of 90% of the observations, where the spread is unexpectedly high. The h^s_a heuristic solves all the problems within 55 seconds, but does not always find the actual goal with the highest probability. As before, it does not discriminated among the possible goals since the spread is very high for all the percentages. FGR^I solves all the problems within 300-415 seconds. It finds the actual goal with high probability and Q decreases as the percentage of observed actions drops. The spread is fairly small, which means that the technique is able to discriminate among all the possible goals, although the spread increases as the percentages of observed actions drops. FGR$^+$ solves all the problems within 100-140 seconds, and finds lower quality Q solutions than FGR^I as expected for high percentage of observed actions. For lower percentage of observed actions the quality Q is better, but the spread is high for all percentage of actions. Therefore, it does not discriminate among all the possible goals as well as FGR$^+$. However, it performs better than LAMA$_G$ in terms of speed, and better than h^s_a in terms of quality Q and spread S.

As before, we performed a second test where the observed sequence was randomly taken to be a subset of this plan solution, ranging from 100% of the actions, down to 30% of the actions. Table 4.4 shows the results of this test, which produces essentially the same results as the previous test for all the different techniques. gHSP$^*_f$ finds the actual goal with the highest probability ($Q = 1$), but the computation time is high. LAMA$_G$ solves all the problems within 1470 seconds. It provides high Q solutions for high percentage of observed actions. The increase in computational time for LAMA$_G$ is noticeable when more that one astronaut is involved in comparison to the previous test, where there is a single astronaut. The h^s_a heuristic solves all the problems within 44–55 seconds, and finds high Q solutions. However, the spread is large so it does not discriminate among the possible goals very well. FGR^I solves all the problems within 300–400 seconds. The quality Q of the solutions decreases as the percentage of observed actions drops, and, in general, is slightly lower than the solutions provided by LAMA$_G$. However, FGR^I is almost three orders of magnitude faster than LAMA$_G$. FGR$^+$ solves all the problems within 130-170 seconds. It finds a lower quality Q solutions than FGR^I as we expected. This confirms that the use of Interaction information is beneficial when subgoals interfere with each other.

Table 4.4: ISS-CAD Evaluation Multiple Astronaut Tasks.

		Sequential Observations						**Random Observations**			
Approach	$\%O$	100	90	80	70	60	50	100	70	50	30
gHSP$^*_f$	T	11171.51	9992.57	10705.45	7504.84	5121.11	3244.15	11173.52	10473.08	7832.33	4013.13
	Q	1	1	1	0.7	0.9	0.7	1	1	1	0.7
	S	1	1	1	2.6	17.9	24.6	1	2	3.7	9.2
LAMA$_G$	T	1480.08	1480.41	1475.46	1475.15	1471.17	1470.05	1482.15	1478.54	1475.43	1471.8
	Q	1	1	0.8	0.8	0.5	0.3	1	1	1	0.3
	S	1	20.2	5.8	1.7	4.3	7.3	1	10.7	16.2	5.2
	Q_{20}	1	1	0.9	1	0.8	0.5	1	1	1	0.7
	Q_{50}	1	1	1	1	0.9	0.9	1	1	1	0.8
	d	0	0.016	0.014	0.011	0.016	0.017	0	0.011	0.014	0.017
h^s_a	T	54.5	51.84	49	46.17	45.29	44.74	54.62	48.77	44.9	43.59
	Q	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.6	0.5
	S	38.4	38.6	38.6	38.8	38.8	38.6	38.4	38.3	37.1	34.3
	Q_{20}	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.6	0.5
	Q_{50}	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.6	0.6
	d	0.036	0.036	0.034	0.028	0.019	6.2×10^{-3}	0.036	0.034	0.033	0.022
FGR^I	T	316.52	305.51	337.33	383.03	412.55	415.92	291.53	312.61	375.35	409.17
	Q	1	0.8	0.8	0.6	0.6	0.5	1	0.9	0.8	0.7
	S	1.2	4	1.2	3.2	3.7	7.6	1	6.6	2.2	10.6
	Q_{20}	1	0.8	0.9	0.6	0.6	0.6	1	0.9	0.8	0.7
	Q_{50}	1	0.8	0.9	0.7	0.8	0.7	1	0.9	0.8	0.8
	d	0	9.5×10^{-3}	0.013	0.012	0.021	0.021	0	0.012	0.015	0.013
FGR$^+$	T	99.19	96.54	96.14	109.31	142.07	136.02	132.96	116.98	140.4	169.17
	Q	0.9	0.7	0.7	0.8	0.8	0.9	0.9	0.6	0.9	0.8
	S	5.8	10.7	6.7	8.9	22.3	34.1	5.8	6.4	2.2	15.2
	Q_{20}	0.9	0.7	0.7	0.8	0.8	1	0.9	0.6	0.9	0.8
	Q_{50}	0.9	0.7	0.7	0.8	0.9	1	0.9	0.6	0.9	0.8
	d	4×10^{-3}	0.014	0.018	0.011	0.012	9×10^{-3}	4×10^{-3}	0.02	0.013	0.015

4.3 Uncertain Observations in Goal Recognition

Previously, we solved goal recognition problems where observations were assumed to be perfect. In practice, this is not very realistic. An important extension is to investigate goal recognition problems that involve uncertain observations. As an illustration of this problem, consider the ISS-CAD problem described in Figure 4.3, where a free-flying robot is observing an astronaut doing a task. There might be different sources of uncertainty in the robot observations because of different reasons such as: (1) there is an obstacle between the robot and the astronaut, (2) some actions may be hard to identify, (3) the robot is engaged in other tasks that do not allow continuous observation of the astronaut, etc.

Ramirez (2010) defines a classical goal recognition problem as a tuple $T = \langle P, \mathcal{G}, O, Pr \rangle$ where P is a planning domain and initial conditions, $\mathcal{G}$ is a set of possible goal sets or hypotheses, O is the observed action sequence $O = o_1, ..., o_n$, and Pr is the prior probability distribution over the goal sets $G \in \mathcal{G}$. The observation sequence O may be incomplete, but is sequentially ordered. The solution to a goal recognition problem is a probability distribution over each goal set $G \in \mathcal{G}$ giving the relative likelihood of each goal set. Ramirez later work (2011) introduces an approach to solve goal recognition problems with uncertain action outcomes, but it does not consider uncertain observations. Here, we introduce an extension that includes uncertain observations, but we are still considering deterministic actions. The syntax we propose for expressing uncertainty in the observability of actions is the inclusion of an *observability* expression in the PDDL action definition:

$$(\texttt{:observability}\ (p_1\ p_2))$$

where p_1 is the default probability that if the action happens, the robot observes it (*true positive*), and p_2 is the default probability that if the action does not happen, the robot observes it (*false positive*). For instance, consider the action *move* from the ISS-CAD domain, and a probability 0.9 considered as true positive (the false negative probability would be $1 - 0.9 = 0.1$); and a probability 0.2 considered as false positive (the true negative probability would be $1 - 0.2 = 0.8$). That is:

```
(:action get-replacement
 :parameters (?t - component ?c - crew ?m - module)
 :precondition (and (at ?c ?m) (replacement-in ?t ?m))
 :effect (and (taken-replacement ?t ?c) (increase (total-cost) 20))
 :observability (0.9 0.2))
```

However, action observability can be a function of parameters. In other words, the observability of an action depends on the parameters. For instance, consider an astronaut opening a drawer to get a replacement component such as a filter or a screw. In this case, it would be more difficult for the robot to identify that the astronaut is getting a screw than a filter since the screw is smaller and thinner than the filter. For this reason, we allow specific grounded statements (overriding the defaults) of the following form in the problem definition:

```
    (observable (get-replacement screw fe1 Unity) (0.7 0.3))
```

where the action of an astronaut `fe1` getting a screw in the module `Unity` is observed by the robot with a probability of 0.7, if it happens, and with a probability of 0.3, if it does not happen.

4.3.1 Optimal solution

In theory, we could again use Ramirez's framework to solve this goal recognition problem with uncertain observations. Ramirez characterizes the likelihood $Pr(O|G)$ in terms of cost differences for achieving G under two conditions: complying with the observations O, and not complying with the observations O. The optimal solution to this problem uses optimal planners to compute an accurate likelihood of the possible goals. As previously mentioned, for each goal set $G \in \mathcal{G}$ Ramirez's framework computes the difference between $Cost(G|O)$ and $Cost(G|\overline{O})$. The latter is affected by the uncertain observation, but, as before, it can be approximated as just $Cost(G)$, which is not affected. Therefore, it can be computed by solving the planning problem using an optimal planner, making its computation the same as before. However, $Cost(G|O)$ is affected by the uncertain observations, so it is necessary to introduce some changes in order to compute it. In the new problem formulation, the set O of uncertain observed actions is a set of observed action sequences. To illustrate this, consider the ISS-CAD problem where the observed sequence is:

$$O = \begin{pmatrix} \text{(move fe1 Harmony Destiny),} \\ \text{(move fe1 Destiny Unity),} \\ \text{(get-replacement sensor fe1 Unity)} \end{pmatrix}$$

where:

(observable (move fe1 Harmony Destiny) (1.0 0.0))

(observable (move fe1 Destiny Unity) (0.8 0.1))

(observable (get-replacement sensor fe1 Unity) (0.7 0.3))

The uncertain observed sequence is then a set S of all possible combinations of observed sequences s_i. In our example, the set S consists of the following subsets s_i:

$$\begin{aligned}
s_1 &= \begin{pmatrix} \text{move fe1 Harmony Destiny,} \\ \text{move fe1 Destiny Unity,} \\ \text{get-replacement sensor fe1 Unity} \end{pmatrix} \text{ with } pr(s_1) = 1(0.8)(0.7) = 0.56 \\
s_2 &= \begin{pmatrix} \text{move fe1 Harmony Destiny,} \\ \text{move fe1 Destiny Unity} \end{pmatrix} \text{ with } pr(s_2) = 1(0.8)(0.3) = 0.24 \\
s_3 &= \begin{pmatrix} \text{move fe1 Harmony Destiny,} \\ \text{get-replacement sensor fe1 Unity} \end{pmatrix} \text{ with } pr(s_3) = 1(0.2)(0.7) = 0.14 \\
s_4 &= \{\text{move fe1 Harmony Destiny}\} \text{ with } pr(s_4) = 1(0.2)(0.3) = 0.06 \\
s_5 &= \begin{pmatrix} \text{move fe1 Destiny Unity,} \\ \text{get-replacement sensor fe1 Unity} \end{pmatrix} \text{ with } pr(s_5) = 0(0.8)(0.7) = 0 \\
s_6 &= (\text{move fe1 Destiny Unity}) \text{ with } pr(s_6) = 0(0.8)(0.3) = 0 \\
s_7 &= (\text{get-replacement sensor fe1 Unity}) \text{ with } pr(s_7) = 0(0.2)(0.7) = 0 \\
s_8 &= (\varnothing) \text{ with } pr(s_8) = 0(0.2)(0.3) = 0
\end{aligned}$$

that is, $S = (s_1, s_2, s_3, s_4)$. Subsets s_5, s_6, s_7, and s_8 can be ignored because they have probability equal to zero.

The set S is needed to compute $Cost(G|O)$ because it contains all the possible sequences potentially performed by the agent and observed by the robot. This means that the problem needs to be solved for each observed action sequence $s \in S$. Therefore, $Cost(G|O)$ would be:

$$Cost(G|O) = Cost(G|S) = \sum_{s_i \in S} Cost(G|s_i)\, pr(s_i) \tag{4.2}$$

Once we have estimated $Cost(G|\overline{O}) \approx Cost(G)$ and $Cost(G|O)$, we can proceed to compute $Pr(G|O)$.

The proposed algorithm may be summarized in the following steps:

1. For each (possibly conjunctive) goal $G \in \mathcal{G}$ approximate $Cost(G|\overline{O})$ as $Cost(G)$ by solving the planning problem using the standard planner HSP$^*_f$.

2. Build the set S that consists of all the possible combinations of observed action sequences.

3. For each (possibly conjunctive) goal $G \in \mathcal{G}$, compute $Cost(G|S)$ as in Equation 4.2.

4. For each (possibly conjunctive) goal $G \in \mathcal{G}$, compute $\Delta(G, O)$ as in Equation 2.11.

5. For each (possibly conjunctive) goal $G \in \mathcal{G}$, compute $Pr(G|O)$ as in Equation 2.12.

This solution provides a probability distribution over the set of possible goals. However, it is computationally prohibitive because of the explosion in the number of possible observation sequences. For this reason, in the next section we present a faster approximate solution to this problem.

4.3.2 Approximate solution

As before, we assume a goal recognition problem $T = \langle P, \mathcal{G}, O, Pr \rangle$, which includes the problem P, i.e., a planning domain and initial conditions with observability information, the set of possible goal sets $G \in \mathcal{G}$, a set of observations O, and a prior distribution Pr over the possible goal sets $G \in \mathcal{G}$. It is also assumed that the sequence of observed actions may be incomplete, but we know the time step in which each action in the observation sequence happens.

The proposed approximate solution uses Bayesian networks (Pearl, 1988) to infer the probability of different actions and propositions in a plan graph given the observations. The Bayesian network (BN) essentially serves as the probabilistic version of the IPR technique used earlier. We then combine this with the plan graph cost estimation technique described in Section 3.2 to allow fast estimation of cost differences $\Delta(G, O)$, and as a result probability estimates for the possible goals $G \in \mathcal{G}$.

As in FGR, $Cost(G|\overline{O})$ is approximated as just $Cost(G)$. As previously mentioned, the latter cost is not affected by the uncertain observations, so we use the plan graph with cost and Interaction information and the h^I heuristic to compute it. On the other hand, to compute $Cost(G|O)$ we first use the BN to estimate the probability of the actions and propositions in the plan graph. Then, we use this information as we did with IPR to estimate $Cost(G|O)$ in the plan graph. We now define the BN and show how we use it. Then we present the algorithm used to compute $Cost(G|O)$.

Pearl (1988) defines a BN as a directed acyclic graph in which nodes represent variables and arcs represent the existence of direct casual influence between the linked variables. The strengths of these influences are expressed by forward conditional probabilities, also called Conditional Probability Tables (CPT), which describe the probability of being within a variable state given a combination of values of its predecessors' states. This model provides complete information about variables and their relationships and, therefore, it can be used to perform probabilistic queries about them. A BN is also widely used to infer posterior probabilities of the variables when other variables, called *evidence*, are observed. We take advantage of this capability to solve our goal recognition problem with uncertain observations.

A plan graph can be translated into a BN by including variables for each action and proposition, and connecting them together in the way preconditions and effects are connected to actions in a plan graph. An *action* variable represents a plan graph action node. A *proposition* variable represents a plan graph proposition node. Those proposition variables that represent the initial state in the plan graph are given as true in the BN.

Figure 4.4 shows an example of the form of the BN. Blue nodes represents evidence nodes that are known as true beforehand, for instance, variable (atHarmony-0) that represents the proposition in the initial state of the plan graph. Green nodes represent those variables for which state is unknown.

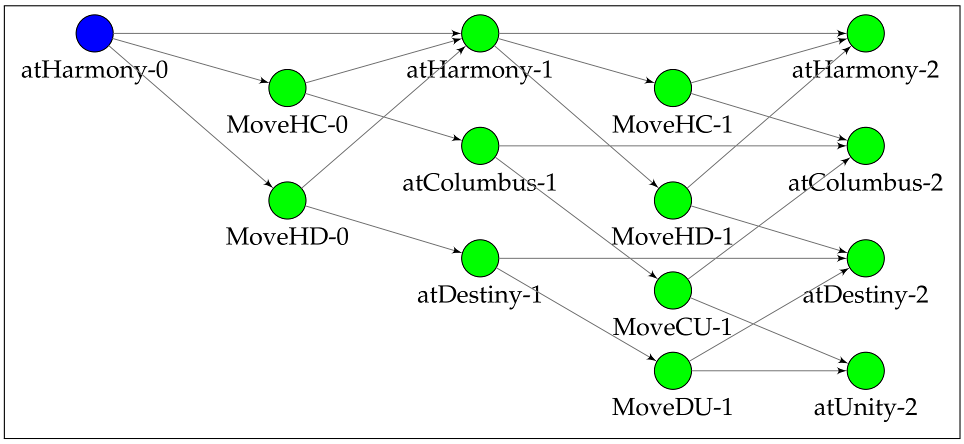


Figure 4.4: An incomplete Bayesian network for the ISS-CAD problem.

A plan graph, however, has more complicated details such as mutex relationships, noop operators, and, in this particular case, observed actions. In order to model a mutex relationship between two actions A and B, we need to create a *mutex* variable and connect both actions to it. To illustrate, consider the example in Figure 4.5(a), where there are two action variables A and B and a mutex variable AxB representing the mutex relationship between A and B. Mutex variables are given as true in the BN for all mutex relationships in the plan graph. In order to model noop operators, we define a *noop* variable. To illustrate, consider the example in Figure 4.5(b), where there is a proposition variable x and a noop variable noop-x. The noop variable has a single incoming arc from x at the previous level i, and a single outgoing arc to x at the next level $i + 1$. Finally, we define an *observation* variable to represent an observed action. To illustrate, consider the example in Figure 4.5(c), where there is an action variable A and an observation variable obs-A. Each action variable is connected to its observed variable. If the action is observed, its observation variable is assigned as true in the BN.

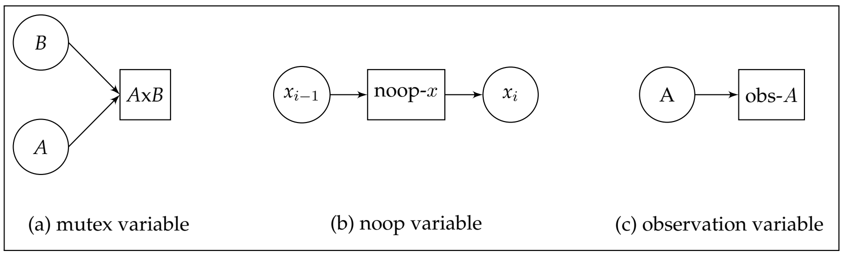


Figure 4.5: Graphical description of *mutex*, *noop*, and *observation* variables in a BN.

Figure 4.6 shows an example of the completed form of the BN. Again, blue nodes represent evidence nodes that are known as true beforehand – initial state propositions, mutex relations, and observed actions. Green nodes represent those variables for which state is unknown.

Each variable in the BN has an associated Conditional Probability Table (CPT) that gives the distribution of the variable for each combination of predecessor values. Those proposition variables without predecessors represent propositions in the initial state, and their CPTs only contain the prior probability of the variable being true or false. For simplicity, we assume that the initial state is observed to be true, although it is possible to make it uncertain. Table 4.5 illustrate the CPT of the proposition variable (atHarmony-0) in Figure 4.6, which is in the initial state. It is true with probability 1 and, therefore, is false with probability 0.

Table 4.5: Conditional probability table of a proposition variable (atHarmony-0).

Pr(atHarmony-0=True)	Pr(atHarmony-0=False)
1	0

An action variable is taken to be:

- False: if any of its preconditions is false.

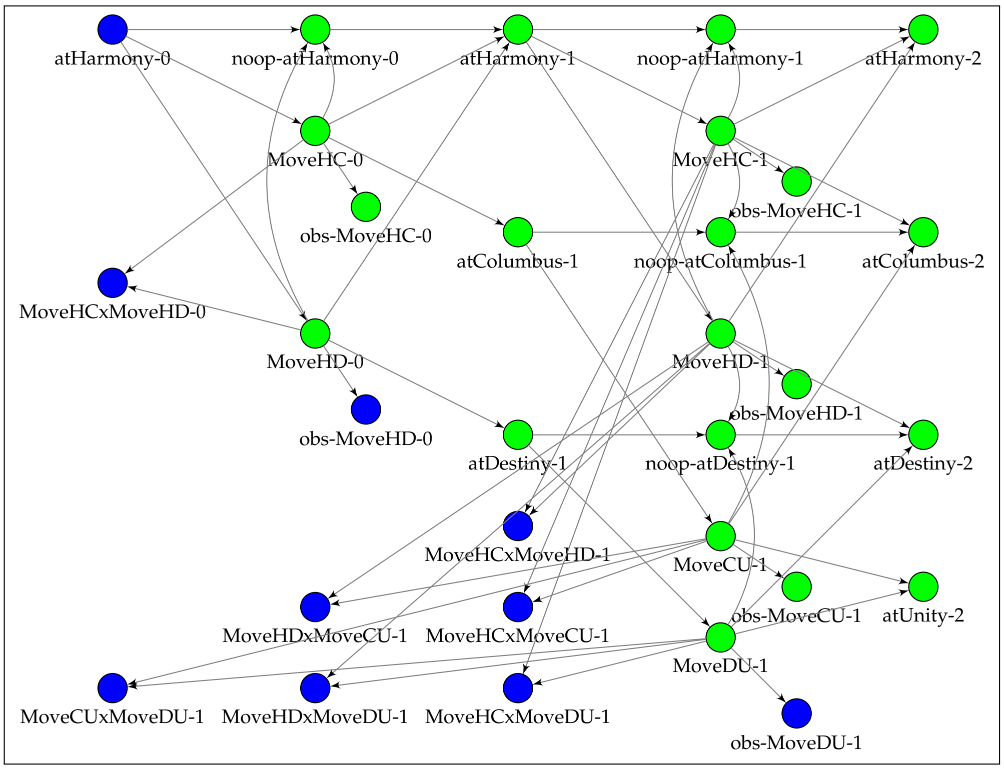


Figure 4.6: An incomplete Bayesian network for the ISS-CAD problem.

- Unknown: if all its preconditions are true.

Figure 4.7 shows this rule for defining the CPT of an action variable A with set of preconditions $\{x_1, x_2, \ldots, x_n\}$. In general, if one of more of the action's preconditions are false, the action is false. If an action's preconditions are true, the action is uncertain. This is because we cannot guarantee that an action happens just because its preconditions are true.

$$\begin{tikzcd}
x_1 \arrow[drr, no head] & & \\
x_2 \arrow[rr, no head] & & A \\
\vdots & & \\
x_n \arrow[uurr, no head] & &
\end{tikzcd}
\implies pr(A) \text{ is } \begin{cases} 0 & \text{if any } x_i = 0 \\ 0.5 & \text{if all } x_i = 1 \end{cases}$$

Figure 4.7: CPT rules for an action variable.

Table 4.6 illustrates the CPT of the action variable (moveHC-0) in Figure 4.6 whose precondition is (atHarmony-0). When (atHarmony-0) is true, the probability that (moveHC-0) is true or false is equal to 0.5. Otherwise, the probability that (moveHC-0) is true is equal to 0, and the probability that (moveHC-0) is false is equal to 1.

Table 4.6: Conditional probability table of an action variable (moveHC-0) with precondition (atHarmony-0).

atHarmony-0	Pr(moveHC-0=True)	Pr(moveHC-0=False)
True	0.5	0.5
False	0	1

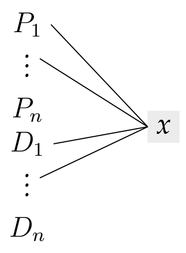
A proposition variable is taken to be:

- True: if there is an action that is true that adds it.
- False: if there is an action that is true that deletes it.

However, a proposition variable with one or more predecessors has a more complex CPT. In order to generate this CPT, we need to consider (1) the actions that can produce (*producers*) or delete (*deleters*) the proposition, (2) the mutex relationships, and (3) whether or not the proposition can persist. In general, a CPT for a proposition variable x with a set of producers $\mathcal{A}$ is defined as:

- True: if any of the producers is true and all of the deleters are false.
- False: if any of the deleters is true and all of the producers are false.
- Unknown: if both a producer and deleter are true.

Figure 4.8 shows this rule for defining the CPT of a proposition variable x with set of producers $\{P_1, \ldots, P_n\}$ and set of producers $\{D_1, \ldots, D_n\}$. In general when any of the producers is true and all of the deleters are false, x is true with probability 1. Conversely, when any of the deleters is true and all of the producers are false, x is true with probability 0. Finally, when both a deleter is true and a producer is true, x has an equal probability of 0.5 of being true or false.



$$\Longrightarrow \quad pr(x) \text{ is } \begin{cases} 1 & \text{if any } P_i = 1 \text{ and no } D_i = 1 \\ 0 & \text{if any } D_i = 1 \text{ and no } P_i = 1 \\ 0.5 & \text{if some } P_i = 1 \text{ and some } D_i = 1 \end{cases}$$

Figure 4.8: CPT rules for a proposition variable.

Table 4.7 illustrates the CPT of the proposition variable (atDestiny-2) in Figure 4.6 whose predecessors are (MoveHD-1), (MoveDU-1), and (noop-atDestiny-2). When (MoveHD-1) and (MoveDU-1) are true, (atDestiny-2) is unknown and, therefore, it has an equal probability of 0.5 of being true or false. This is because the two actions are mutually exclusive. When (MoveHD-1) is true and (MoveDU-1) is false, (atDestiny-2) is true with probability 1 and, therefore, false with probability 0. This is because (atDestiny-2) belongs to (MoveHD-1) add effects. Conversely, when (MoveHD-1) is false and (MoveDU-1) is true, (atDestiny-2) is true with probability 0 and, therefore, false with probability 1. This is because (atDestiny-2) belongs to (MoveDU-1) delete effects. Finally, when (MoveHD-1) and (MoveDU-1) are false and (noop-atDestiny-1) is true, (atDestiny-2) is true with probability 1. Otherwise, it is false with probability 1.

Table 4.7: Conditional probability table of a proposition variable (atDestiny-2).

MoveHD-1	MoveDU-1	noop-atDestiny-1	Pr(atDestiny-2=True)	Pr(atDestiny-2=False)
True	True	True	0.5	0.5
True	True	False	0.5	0.5
True	False	True	1	0
True	False	False	1	0
False	True	True	0	1
False	True	False	0	1
False	False	True	1	0
False	False	False	0	1

A mutex variable between two actions A and B is defined as:

- False: if both A and B are true.
- Unknown: if at least one of A and B is false.

Figure 4.9 shows this rule for defining the CPT of a mutex variable AxB for actions A and B. In general, when both actions are true, the mutex variable is false. Otherwise, the mutex variable may be true or false (unknown) with probability 0.5.

$$\begin{tikzcd} A \arrow[dr] & \\ & AxB \\ B \arrow[ur] & \end{tikzcd} \implies pr(A\text{x}B) \text{ is } \begin{cases} 0 & \text{if } A = 1 \text{ and } B = 1 \\ 0.5 & \text{if } A = 0 \text{ or } B = 0 \end{cases}$$

Figure 4.9: CPT rules for a mutex variable.

By setting all mutex variables to true in the BN, we prevent mutex actions from occurring in a symmetric way.

Table 4.8 illustrates the CPT of the mutex variable (MoveHCxMoveHD-0) in Figure 4.6, which represents the mutex relation between actions (MoveHC-0) and (MoveHD-0). When both actions (MoveHC-0) and (MoveHD-0) are true, the mutex variable is false with probability 1. Otherwise, the mutex variable is equally likely with probability 0.5.

Table 4.8: Conditional probability table of a mutex variable (MoveHCxMoveHD-0) for actions (MoveHC-0) and (MoveHD-0).

MoveHC-0	MoveHD-0	Pr(MoveHCxMoveHD-0=True)	Pr(MoveHCxMoveHD-0=False)
True	True	0	1
True	False	0.5	0.5
False	True	0.5	0.5
False	False	0.5	0.5

It might seen that we could just model noop variables as ordinary actions and add mutex relationships between the noop and all other actions that influence the proposition variable. However, this is not enough. We also need to force the noop to be true when all other actions that affect the variable are false. A noop variable of a proposition is, therefore, defined as:

- True: if the proposition is true in the previous level and all other actions that affect the proposition are false.

- False: if the proposition is not true in the previous level or there is another action that affects the proposition.

Figure 4.10 shows this rule for defining the CPT of a noop variable noop-x for proposition x with set of producers P_i and set of deleters D_i. A proposition variable will remain unchanged if there is no action that modifies it.

$$\begin{tikzcd} & P_i \arrow[d] & D_i \arrow[dl] \\ x \arrow[r] & \text{noop-}x & \end{tikzcd} \implies pr(\text{noop-}x) \text{ is } \begin{cases} 1 \text{ if } x_{i-1} = 1 \text{ and } P_i = 0 \text{ and } D_i = 0 \\ \\ 0 \text{ if } x_{i-1} = 0 \text{ or some } P_i = 1 \text{ and some } D_i = 1 \end{cases}$$

Figure 4.10: CPT rules for a noop variable.

Table 4.9 illustrates the CPT of the action variable (noop-atHarmony-0) in Figure 4.6 whose precondition is (atHarmony-0). There are two other actions that produce (atHarmony-0), (MoveHC-0) and (MoveHD-0), which must be considered in the CPT. In our example, when (MoveHC-0) and (MoveHD-0) are false, and (atHarmony-0) is true, then (noop-atHarmony-0) is true. Otherwise, (noop-atHarmony-0) is false.

Table 4.9: Conditional probability table of a noop variable (noop-atHarmony-0) with precondition (atHarmony-0).

MoveHC-0	MoveHD-0	atHarmony-0	Pr(noop-atHarmony-0=True)	Pr(noop-atHarmony-0=False)
True	True	True	0	1
True	True	False	0	1
True	False	True	0	1
True	False	False	0	1
False	True	True	0	1
False	True	False	0	1
False	False	True	1	0
False	False	False	0	1

Finally, the CPT of an observation variable is generated using the probability information given by the *observable* statement in the goal recognition problem definition. Figure 4.11 shows the rule for defining the CPT of an observable variable obs-A for an action A.

$$A \longrightarrow \text{obs-}A \implies pr(\text{obs-}A) \text{ is } \begin{cases} \textit{true positive} \text{ if } A = 1 \\ \\ \textit{false positive} \text{ if } A = 0 \end{cases}$$

Figure 4.11: CPT rules for an observation variable.

Table 4.10 shows the CPT of the observation variable (obs-MoveDU-1) in Figure 4.6, which is observed at time *1* with a probability of 0.8 of being observed if true, and a probability of 0.1 of being observed if false.

Once we generate the BN of the problem, we can query the posterior probability of each variable given all evidence variables – that is, the actions in the observed sequence. In our example, we know by intuition that the probability of (atDestiny-1) should be higher than the probability of (atHarmony-1) and (atColumbus-1); and that the probability of (atUnity-2) should be higher than the

Table 4.10: Conditional probability table of an observation variable (obs-MoveDU-1) of the observed action (MoveDU-1).

MoveDU-1	Pr(obs-MoveDU-1=True)	Pr(obs-MoveDU-1=False)
True	0.8	0.1
False	0.2	0.9

probability of (atDestiny-2) because actions (MoveHD-0) and (MoveDU-1) are observed with probabilities 1 and 0.8 respectively. In particular, when we query variables (atHarmony-1), (atColumbus-1), and (atDestiny-1), we get the following posterior probability:

$$\begin{aligned}
&\Pr(\text{atHarmony-1=True} \mid \text{evidence}) = 0\\
&\Pr(\text{atColumbus-1=True} \mid \text{evidence}) = 0\\
&\Pr(\text{atDestiny-1=True} \mid \text{evidence}) = 1\\
&\Pr(\text{atDestiny-2=True} \mid \text{evidence}) = 0.015\\
&\Pr(\text{atUnity-2=True} \mid \text{evidence}) = 0.985
\end{aligned}$$

If, for instance, we considered the case where the robot only observed action (MoveHD-0), then (atDestiny-2) and (atUnity-2) would have a 0.5 posterior probability of being true. This is because there is not enough information or evidence to infer the astronaut behavior, and, therefore, both locations would have the same probability of being true.

We can use these posterior probabilities values to approximate the new cost and Interaction information. In particular, we follow the algorithm described in Section 3.2, where the computation begins at level zero of the plan graph and proceeds sequentially to higher levels. At level zero, the cost of each proposition is zero as well as the Interaction between each pair of propositions. The cost of an action is the cost of achieving its preconditions plus the Interaction between all pairs of preconditions. This is computed using Equation 3.10. The Interaction between a pair of actions is infinity if the two actions are mutually exclusive, otherwise it will depend on the relationships between their preconditions. This is computed using Equation 3.11. Now we need to compute the cost of propositions, which we defined as the minimum cost among all the actions that achieve the proposition. However, it is possible that a costly action has a high probability of being true given the sequence of uncertain observed actions. For this reason, we make use of the posterior probabilities given by the BN to make a choice among all the actions that produce the proposition. In particular, the chosen action is the one that minimizes the ratio between action cost and probability. That is:

$$\operatorname*{argmin}_{a \in \mathcal{A}_x} \left\{ \frac{cost(a) + \mathcal{C}ost(a)}{pr(a)} \right\} \tag{4.3}$$

where $pr(a)$ is the posterior probability of action a, computed in the BN.

Once we have chosen the action that minimizes the expression in Equation 4.3, we can compute the cost of the proposition as in Equation 3.10. This tends to prefer actions with low cost and high probability.

Similarly, to compute the Interaction between two propositions we choose the pair of actions that minimize the ratio between action cost and probability. That is:

$$\operatorname*{argmin}_{a \in \mathcal{A}_x,\, b \in \mathcal{A}_y} \left\{ \frac{cost(a) + \mathcal{C}ost(a) + cost(b) + \mathcal{C}ost(b) + I(a,b)}{pr(a)\, pr(b)} \right\} - cost(x) - cost(y) \tag{4.4}$$

where $pr(a)$ and $pr(b)$ are the posterior probabilities of actions a and b, computed in the BN. Once we have chosen the pair of actions that minimizes the ratio between action cost and probability, we can compute the Interaction between two propositions. It will be infinity if all pair of actions are mutually exclusive. Otherwise, the Interaction is computed as in Equation 3.11.

Taking these decision rules into consideration, we can build a plan graph and propagate cost and Interaction information. The construction process finishes when two consecutive propositions layers are identical and there is quiescence in cost and Interaction for all propositions and actions in the plan graph. On completion, each possible goal proposition has an estimated cost of being achieved, considering the observed sequence.

Figure 4.12 shows the high-level algorithm used to solve a goal recognition problem with uncertain observations, namely uFGR, which may be summarized in the following steps:

1. Build a plan graph for the problem P (domain plus initial conditions) and propagate cost and Interaction information through this plan graph according to the technique described in Section 3.2.

2. For each (possibly conjunctive) goal $G \in \mathcal{G}$ estimate the $Cost(G)$ from the plan graph using Equation 3.15.

3. Build a BN, using the technique previously described. Use this BN to estimate probability for actions and propositions in the plan graph.

4. Compute new cost and Interaction estimates for this pruned plan graph, considering the posterior probabilities given by the BN using Equations 3.11.

5. For each (possibly conjunctive) goal $G \in \mathcal{G}$:

   a. Estimate the $Cost(G|O)$ from the cost and Interaction estimates in the pruned plan graph, again using Equations 3.6, 3.7, 3.10, and 3.11.

   b. Compute $\Delta(G, O)$ using Equation 2.11, and compute the probability $Pr(G|O)$ for the goal given the observations using Equation 2.12.

uFGR $(P, O, \mathcal{G})$

O	$\equiv$	a sequence of observed actions
$\mathcal{G}$	$\equiv$	a set of possible goals or hypothesis
g	$\equiv$	a goal $g \in \mathcal{G}$
pg	$\equiv$	a plan graph with cost and Interaction
BN	$\equiv$	a pruned Bayesian network
PR	$\equiv$	a probability distribution over $\mathcal{G}$

```
1. $pg \leftarrow$ BUILDPLANGRAPH($P$, $\mathcal{G}$)
2. for each $g \in \mathcal{G}$
   COMPUTECOST($G$, $pg$)
3. BN $\leftarrow$ BUILDBAYESIANNETWROK($pg$, $O$)
4. UPDATECOSTPLANGRAPH(BN)
5. for each $g \in \mathcal{G}$
   a. COMPUTECOST($G$, $pruned$–$pg$)
   b. PR $\leftarrow$ PR $\cup$ COMPUTEPROBABILITY($G$)
6. return PR
```

Figure 4.12: The uFGR pseudo-algorithm.

4.4 Conclusions

This chapter presented a fast technique for goal recognition (FGR). This technique computes cost estimates using a plan graph, and uses this information to infer probability estimates for the possible goals. This technique provides fast, high quality solutions when the percentage of observed actions is relatively high, and degrades as the percentage of observed actions decreases. In terms of speed, this approach yields results two orders of magnitude faster than Ramirez approach using HSP_f^*, and an order of magnitude faster than Ramirez approach using greedy LAMA. The solutions are comparable or better in quality than those produced by greedy LAMA. This technique can be used with or without information about the times for observed actions. The produced ranking could be used to select the highest probability goals, and then feed this set to the Ramírez approach to provide further refinement to the probabilities. While this does reduce the amount of computation time required by the Ramírez approach, in our experiments the ranking only improved occasionally, and the computation time was still large for the more difficult domains and problems. When the top goals are fed to the Ramírez approach with the LAMA planner, the overall ranking occasionally improves for higher time limits. For low time limits, it doesn't improve or decreases the solution quality.

We also presented a real-world application for goal recognition that involves free-flying robots tracking ISS crew member's daily routines to maintain a safe, clean, and healthy environment aboard the International Space Station. We show that current state-of-the-art optimal goal recognition approaches are not effective to solve this problem, and suboptimal and heuristic goal recognition approaches provide low accuracy solutions. However, we showed that the h^I and h^+ heuristics provide high quality Q results with low spreads. Specifically, when each goal in the hypotheses set involves a single astronaut, then the h^+ heuristic performs better than the h^I heuristic because of the low degree of interference among subgoals. However, when the hypotheses set involves more than one astronaut, and the astronauts' tasks interfere with each other, the h^I heuristic works better than the h^+ heuristic because of the use of the Interaction information.

Finally, we outlined an approach to solve goal recognition problems with uncertain observations that uses the Ramirez framework and combines a Bayesian network and a plan graph to find a solution. This approach has not yet been empirically validated, in part due to the extreme computational cost of comparison with the optimal approach.

Chapter 5

The h^I Family of Heuristics in Probabilistic Planning

Converting probability information into costs makes possible the use of the h^I heuristic to guide a deterministic planner towards low-cost (high-probability) plans. For this reason, in this chapter we present work that involves the use of probability information in a heuristic function to guide deterministic planning search towards high-probability plans. These plans can be used in a system that handles unexpected outcomes by runtime replanning. These plans can also be incrementally augmented with contingency branches for critical action outcomes. This chapter first introduces an overview of the approach and describes the details of each of the involved techniques. Finally, it shows an empirical study.

5.1 Probabilistic planning through Determinization and Replanning using h^I

Determinization consists of transforming a probabilistic planning domain into a deterministic planning domain. Therefore, the Determinization and Replanning method uses deterministic planners to solve a probabilistic planning problem, and runtime replanning to deal with unexpected states. The FF-Replan planner has been the pioneer in this paradigm, but it does not make use of the probabilistic information in the domain description, which may result in frequent replanning. To overcome this issue, Jiménez, Coles, and Smith developed an approach that follows the same line as FF-Replan, but turns the probability information into costs. Then, they search for a lowest cost plan using a deterministic optimizing planner.

The fact of converting probability information into costs makes possible the use of the h^I heuristic introduced in Chapter 3 to guide a deterministic planner. For this reason, in this section we present work that adopts the Determinization and Replanning approach with the aim of producing plans that are less likely to get stuck in dead-ends states. We do that by adopting the h^I heuristic for the purpose of quickly search towards high-probability plans. This approach has been implemented in the Parallel Integrated Planning and Scheduling System (PIPSS) (Plaza et al., 2008). Figure 5.1 shows the architecture of the new planner, namely PIPSS^I (E-Martín et al., 2014), and highlights the key features of the system: *PPDDL-PDDL Conversion*, *Plan Graph Estimator*, and *Heuristic Computation* modules. Given a PPDDL problem, the system initially transforms the probabilistic problem into a deterministic one using the technique explained in the next section. Then, the *Analysis & Processing* module processes and loads all the information given in the domain, and encodes the strings as numbers to decrease the computation time. The system builds a plan graph estimator as described in Section 3.2. After this step, the search process starts. The system performs an A* search when it is looking for a solution. For each state, the plan graph is updated and a relaxed plan is created to estimate the cost (probability) of achieving the goals from that state. This estimation is called a *Completion Cost Estimate* (CCE).

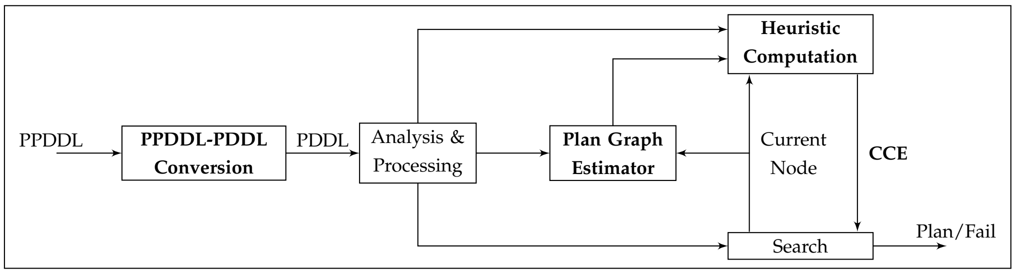


Figure 5.1: PIPSS^I System Architecture.

In the next section, we describe the conversion technique from PPDDL to PDDL that we use. Then, we present the heuristic technique that we apply. Finally, we present an extended example of how our technique works.

5.1.1 Conversion from PPDDL to PDDL

To convert from PPDDL to PDDL we follow the approach of Jiménez, Coles, and Smith (2006). In general, the process consists of generating a deterministic action for each probabilistic effect of a probabilistic action. For each new action created, the probability of its outcomes is transformed into an additive cost equal to the negative logarithm of the probability:

$$C_i = -Ln(P_i) \tag{5.1}$$

More precisely, if a is a probabilistic action with outcomes $o_1, ..., o_i$ with probabilities $p_1, ..., p_i$ respectively, we create a new deterministic action for each outcome. Each deterministic action a_i has all the preconditions of a. If the outcome o_i is conditional, a_i will also have additional preconditions corresponding to the conditions of o_i. The effects of a_i are the effects in the outcome o_i, and the cost of a_i is given by equation (5.1). Figure 5.2 shows an example of the conversion strategy. Figure 5.2(a) shows the probabilistic action (drive) that has two outcomes leading to two deterministic actions. The most likely action outcome is that the car successfully drives from one position to another. However, in the second outcome the car reaches the destination, but it gets a flat tire. The conversion process builds the two deterministic actions shown in Figures 5.2(b) and 5.2(c), one for each possible outcome, and generates the following additive costs $\mathcal{C}ost$(drive-1) $= -\ln(0.6) \approx 0.511$ and $\mathcal{C}ost$(drive-2) $= -\ln(0.4) \approx 0.916$.

5.1.2 Heuristic search guidance

This section describes the probabilistic plan graph heuristic that guides the planner toward lower cost (higher probability) plans. In order to do that, the planner performs an A* search where the evaluation function is the sum of the cost from the starting node to the current node n plus an estimation of the cost of going from the current node to the goals, which we call the Completion Cost Estimate (CCE). That is:

$$f(n) = cost(n) + \text{CCE}(n) \tag{5.2}$$

The CCE is computed using the plan graph estimator. For each state in the search the plan graph is updated and we compute a relaxed plan to estimate the probability of achieving the goals from that state. The relaxed plan is computed by performing regression on the plan graph developed in Chapter 3, where we propagate cost and Interaction information to compute more accurate estimates of cost. This algorithm makes use of the cost and Interaction information to make better choices for actions. In particular, the algorithm orders the goals at each level according to cost, and chooses the operator used to achieve each goal based on cost. More precisely:

- Arrange goals: the goals at each level are arranged from the highest cost to the lowest. In this way, we begin analyzing the most expensive (least prob-

```
(:action drive
 :parameters (?trk - truck ?from - location ?to - location ?pkg - package)
 :precondition (and (connected ?from ?to) (at ?from ?p) (in ?pkg ?p)
                    (verified ?pkg ?p) (not (flattire)))
 :effect (and (not(at ?from ?p)) (at ?to ?p) (probabilistic 0.4 (flattire))))
```

(a) Probabilistic action in PPDDL

```
(:action drive-1
 :parameters (?trk - truck ?from - location ?to - location ?pkg - package)
 :precondition (and (connected ?from ?to) (at ?from ?p) (in ?pkg ?p)
                    (verified ?pkg ?p) (not (flattire)))
 :effect (and (not (at ?from ?p)) (at ?to ?p) (increase (risk) 0.511)))
```

(b) Deterministic action a

```
(:action drive-2
 :parameters (?trk - truck ?from - location ?to - location ?pkg - package)
 :precondition (and (connected ?from ?to) (at ?from ?p) (in ?pkg ?p)
                    (verified ?pkg ?p) (not (flattire)))
 :effect (and (not (at ?from ?p)) (at ?to ?p) (flattire)
              (increase (risk) 0.916)))
```

(c) Deterministic action b

Figure 5.2: Determinization of a probabilistic action.

able) proposition to try to solve the most difficult goals first. In addition, as we solve *more expensive* goals, other *cheaper* goals might be also solved.

- Choose operators: if there is more than one operator that achieves a particular goal at a level, we choose the operator with the lowest cost (highest probability) given the other operators that have already been chosen at that level. Suppose that O is the set of operators selected in level l, and $\mathcal{A}_g$ is the set of actions that achieve the current goal g at level l. The operator we choose is:

$$\operatorname*{argmin}_{a \in \mathcal{A}_g} \left\{ cost(a) + \mathcal{C}ost(a) + \sum_{b \in O} I(a,b) \right\} \tag{5.3}$$

The intent here is to greedily choose a more synergistic and less costly set of operators.

Figure 5.3 shows the high-level algorithm used in the relaxed plan regression phase.

Function COSTESTIMATE (*G*,*l*)

G_l	$\equiv$	the set of goals at level *l* in the relaxed plan graph
g	$\equiv$	a goal proposition
l	$\equiv$	number of levels in the relaxed plan graph
A_l	$\equiv$	the set of actions at level *l* for an specific goal
a	$\equiv$	an action
O_l	$\equiv$	the set of operators selected at level *l*
π	$\equiv$	the set of actions selected
CCE	$\equiv$	the completion cost estimate of the current node

```
1. while $l > 0$
2.     $O_l \leftarrow \emptyset$
3.     $G_{l-1} \leftarrow \emptyset$
4.     while $G_l \neq \emptyset$
5.         $g \leftarrow \operatorname{argmax}_{g \in G_l} \{Cost(g)\}$
6.         $A_l \leftarrow \{a : g \in eff^+(a)\}$
7.         $a \leftarrow \operatorname{argmin}_{a \in A_l} \left\{ Cost(a) + Cost(g|a) + \sum_{b \in O_l} I(a,b) \right\}$
8.         $O_l \leftarrow \cup \{a\}$
9.         $\pi \leftarrow \pi \cup \{a\}$
10.        $G_{l-1} \leftarrow G_{l-1} \cup prec(a)$
11.        $G_l \leftarrow G_l - \{g\}$
12.    $l \leftarrow l - 1$
13. CCE(currentNode) $\leftarrow \sum_{i=1..n} Cost(\pi_i)$
14. return CCE
```

Figure 5.3: The relaxed plan regression pseudo-algorithm.

An extended example

Consider the Logistics problem shown in Figure 2.10 where there is a package and a truck at location a, and the package needs to be delivered to location c. The package can be loaded on the truck. The truck can drive between locations when it does not have a flat tire and it is verified that the package is inside the truck. In addition, a tire may go flat during the drive with a probability of 0.4. A flat tire can be changed, if the truck is at locations a or d where there is a spare tire. Finally, before unloading the package from the truck, it must be scanned.

To solve this problem, our system gets the PPDDL Logistics domain shown in Figure 2.9 as input. The probabilistic domain is then translated into a deterministic one using the determinization technique explained in Section 5.1.1. In other words, those probabilistic actions in the domain description are converted into deterministic actions. In our example, action *drive* is a probabilistic action that, as is shown in Figure 5.2, is split into the following two deterministic actions: *drive-1*, where a truck successfully drives from one location to another with an approximate cost of $-\ln(0.6) \approx 0.51$, and *drive-2*, where a truck successfully drive from one location to another, but gets a flat tire with an approximate cost of $-\ln(0.4) \approx 0.91$. In addition, action *verify* splits into two deterministic actions: *verify-1*, where the verification of the package is successfully done with an approximate cost of $-\ln(0.8) \approx 0.22$, and *verify-2*, where the action does not verify the package with an approximate cost of $-\ln(0.2) \approx 1.61$, although, it is not considered for planning purposes because it does not have any effects. The rest of the actions in the domain description are deterministic, and have an implicit probability of 1 and, therefore, a cost equal to $-\ln(1) = 0$.

After the determinization of the PPDDL domain, our system builds a plan graph estimator as described in Section 3.2. To illustrate that, consider that the current state consists of propositions (at a trk), (in trk pkg), (verified pkg trk), (scanned pkg trk), (¬flattire), and (spare d). In other words, the truck is at location a and does not have a flat tire, the package is already in the truck, verified, and scanned, and there is a spare tire at location d. Figure 5.4 shows a partial plan graph for this problem, where every operator and action in the fragment has an associated cost value.

Using Equation 3.6, we compute the cost for actions (scan pkg trk) and (drive-1 a d trk) at level 0 as:

$$\begin{aligned}
cost(\text{scan pkg trk}) &= \left\{ \begin{array}{l} cost(\text{in trk pkg}) + cost(\text{verified pkg trk}) + \\ I(\text{in trk pkg, verified pkg trk}) \end{array} \right\} \\
&= 0 + 0.22 + 0 = 0.22
\end{aligned}$$

$$\begin{aligned}
cost(\text{drive-1 a d trk}) &= \left\{ \begin{array}{l} cost(\text{at a trk}) + cost(\neg\text{flattire}) + cost(\text{in trk pkg}) + \\ cost(\text{verified pkg trk}) + I(\text{at a trk}, \neg\text{flattire}) + \\ I(\text{at a trk, in trk pkg}) + I(\text{at a trk, verified pkg trk}) + \\ I(\neg\text{flattire, in trk pkg}) + I(\neg\text{flattire, verified pkg trk}) + \\ I(\text{in trk pkg, verified pkg trk}) \end{array} \right\} \\
&= 0 + 0.22 + 0 = 0.22
\end{aligned}$$

Using Equation 3.7, we compute the Interaction between actions (scan pkg trk) and (drive-1 a d trk) at level 0 as:

$$
\begin{aligned}
I(\text{scan pkg trk, drive-1 trk a d}) &= \left\{ \begin{array}{l} 0 - cost(\text{in trk pkg}) - \\ cost(\text{verified pkg trk}) - \\ I(\text{in trk pkg, verified pkg trk}) \end{array} \right\} \\
&= 0 - 0 - 0.22 - 0 = -0.22
\end{aligned}
$$

The fact that $I(\text{scan pkg trk, drive-1 trk a d}) = -0.22$ means that there is some degree of synergy between both actions that comes from the fact that the two actions have two common preconditions, (in trk pkg) and (verified pkg trk).

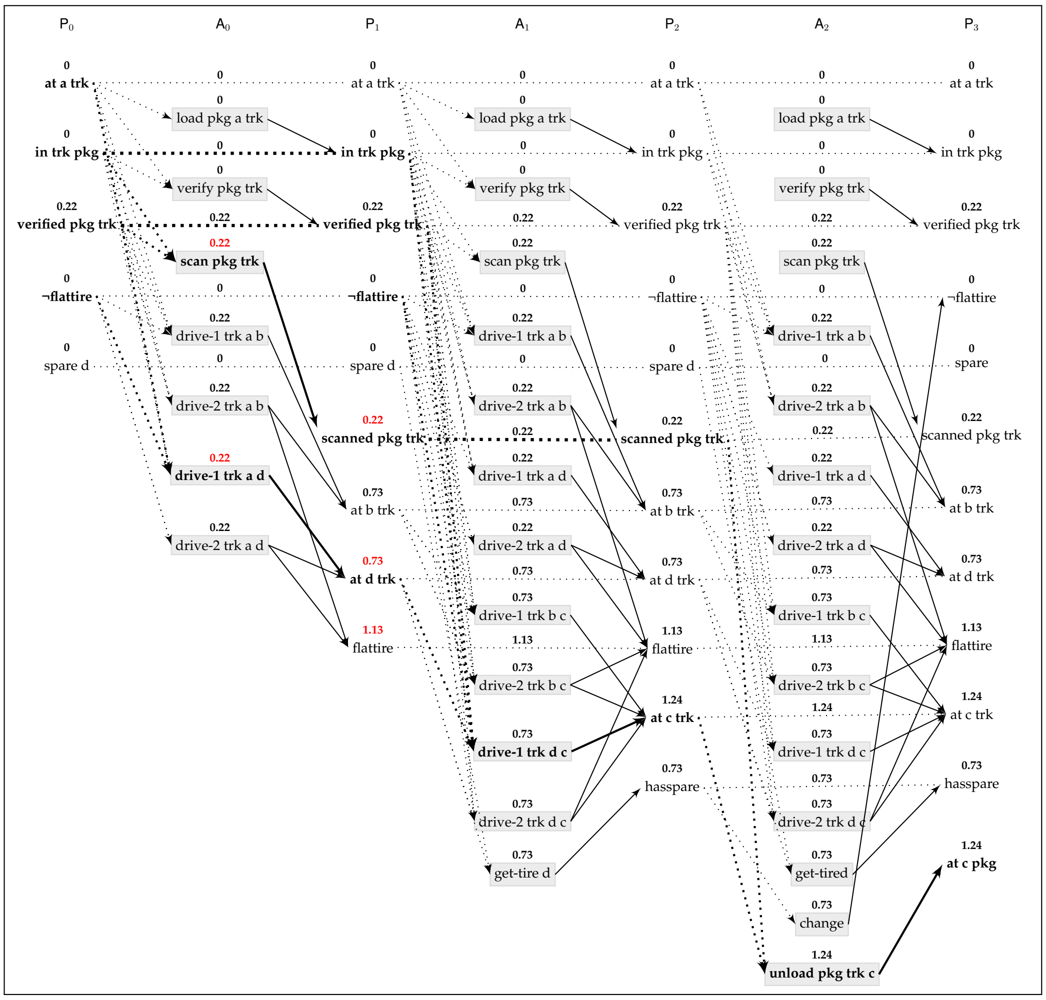


Figure 5.4: Fragment of a plan graph with cost and Interaction information.

Using Equation 3.10, we can compute the cost of propositions (at d trk), (flattire), and (scanned pkg trk) at level 1 as:

$$\begin{aligned}
cost(\text{at d trk}) &= \min\left\{\begin{array}{l} cost(\text{drive-1 trk a d}) + \mathcal{C}ost(\text{drive-1 trk a d}), \\ cost(\text{drive-2 trk a d}) + \mathcal{C}ost(\text{drive-2 trk a d}) \end{array}\right\} \\
&= \min\{0.22 + 0.51,\ 0.22 + 0.91\} \\
&= \min\{0.73,\ 1.13\} = 0.73
\end{aligned}$$

$$\begin{aligned}
cost(\text{flattire}) &= \min\left\{\begin{array}{l} cost(\text{drive-2 trk a b}) + \mathcal{C}ost(\text{drive-2 trk a b}), \\ cost(\text{drive-2 trk a d}) + \mathcal{C}ost(\text{drive-2 trk a d}) \end{array}\right\} \\
&= \min\{0.22 + 0.91,\ 0.22 + 0.91\} \\
&= \min\{1.13,\ 1.13\} = 1.13
\end{aligned}$$

$$\begin{aligned}
cost(\text{scanned pkg trk}) &= cost(\text{scan pkg trk}) + \mathcal{C}ost(\text{scan pkg trk}) \\
&= 0.22 + 0 = 0.22
\end{aligned}$$

Using Equation 3.11, we can compute the Interaction between propositions (at d trk) and (scanned pkg trk) as:

$$\begin{aligned}
I(\text{at d trk, scanned pkg trk}) &= \min\left\{\begin{array}{l} cost(\text{drive-1 trk a d} \wedge \text{scan pkg trk}), \\ cost(\text{drive-2 trk a d} \wedge \text{scan pkg trk}) \end{array}\right\} \\
&\quad - cost(\text{at d trk}) - cost(\text{scanned pkg trk})
\end{aligned}$$

where:

$$\begin{aligned}
cost(\text{drive-1 trk a d} \wedge \text{scan pkg trk}) &= \left\{\begin{array}{l} cost(\text{drive-1 trk a d}) + \\ \mathcal{C}ost(\text{drive-1 trk a d}) + \\ cost(\text{scan pkg trk}) + \\ \mathcal{C}ost(\text{scan pkg trk}) + \\ I(\text{drive-1 trk a d, scan pkg trk}) \end{array}\right\} \\
&= 0.22 + 0.51 + 0 + 0.22 - 0.22 = 0.73
\end{aligned}$$

$$\begin{aligned}
cost(\text{drive-2 trk a d} \wedge \text{scan pkg trk}) &= \left\{\begin{array}{l} cost(\text{drive-2 trk a d}) + \\ \mathcal{C}ost(\text{drive-2 trk a d}) + \\ cost(\text{scan pkg trk}) + \\ \mathcal{C}ost(\text{scan pkg trk}) + \\ I(\text{drive-2 trk a d, scan pkg trk}) \end{array}\right\} \\
&= 0.22 + 0.91 + 0 + 0.22 - 0.22 = 1.13
\end{aligned}$$

Therefore:

$$\begin{aligned} I(\text{at d trk, scanned pkg trk}) &= \min\{0.73,\ 1.13\} \\ &\quad -cost(\text{at d trk}) - cost(\text{scanned pkg trk}) \\ &= 0.73 - 0.73 - 0.22 = -0.22 \end{aligned}$$

The fact that $I(\text{at d trk, scanned pkg trk}) = -0.22$ means that there is synergy between having the package scanned and having it at location b. This synergy comes from the fact that actions (drive trk a d) and (scan pkg trk) have two common preconditions (in trk pkg) and (verified pkg trk).

Once the plan graph is built until quiescence, the search process starts. The system performs an A* search when it is looking for a solution. For each state, the plan graph is updated and a relaxed plan is created to estimate the CCE. To illustrate that, consider the plan graph in Figure 5.4. The relaxed plan construction starts at level 3 and performs a backwards search starting from the goal state until the initial state is reached. In the current example, the goal state is composed of proposition (at c pkg) with an estimated cost of 1.24. The system selects the cheapest action at level 2 that achieves it, which in this example is (unload pkg trk c) since it is the only one that achieves it. Its preconditions (at c trk) and (scanned pkg trk) form the new set of subgoals at level 2.

At level 2, the procedure first deals with (at c trk), which has the higher cost. It is achieved by actions in $\mathcal{A}_{\text{(at c trk)}}$ at level 1. In order to know which is the best choice, we perform the following calculations:

$$\underset{a \in \mathcal{A}_{\text{(at c trk)}}}{\text{argmin}} \quad \{cost(a) + \mathcal{C}ost(a)\}$$

$$\begin{aligned} cost(\text{drive-1 b c}) + \mathcal{C}ost(\text{drive-1 b c}) &= 1.24 + 0.51 = 1.75 \\ cost(\text{drive-2 b c}) + \mathcal{C}ost(\text{drive-2 b c}) &= 1.24 + 0.91 = 2.15 \\ cost(\text{drive-1 d c}) + \mathcal{C}ost(\text{drive-1 d c}) &= 1.24 + 0.51 = 1.75 \\ cost(\text{drive-2 d c}) + \mathcal{C}ost(\text{drive-2 d c}) &= 1.24 + 0.91 = 2.15 \end{aligned}$$

therefore:

$$\underset{a \in \mathcal{A}_{\text{(at c trk)}}}{\text{argmin}} \quad \{cost(a) + \mathcal{C}ost(a)\} = \text{drive-1 d c}$$

In this case, the action chosen is (drive-1 d c) because has the lowest cost. (We could have also selected (drive-1 b c) since it has the same cost as (drive-1 d c).) In this case, action (drive-1 d c) sets propositions (¬flattire), (at d trk), (in trk pkg), and (verified pkg trk) as subgoals at level 1.

Next subgoal, (scanned pkg trk), it is achieved by actions in $\mathcal{A}_{\text{(scanned pkg trk)}}$ at level 2. In order to know which is the best choice, the cost of achieving (scanned pkg trk) must be computed for actions in $\mathcal{A}_{\text{(scanned pkg trk)}}$ assuming (drive-1 d c). Considering the action cost values shown in Figure 5.4 and the following Interaction values:

$$I(\text{scan pkg trk, drive-1 d c}) = -0.22$$
$$I(\text{noop-scanned pkg trk, drive-1 d c}) = -0.22$$

the best choice is calculated as follows:

$$\operatorname*{argmin}_{a\in\mathcal{A}_{\text{(scanned pkg trk)}}} \{cost(a) + \mathcal{C}ost(a) + I(a, \text{drive-1 d c})\}$$

$$\left\{\begin{array}{l} cost(\text{noop-scanned pkg trk}) + \mathcal{C}ost(\text{scan pkg trk}) + \\ I(\text{noop-scanned pkg trk, drive-1 d c}) = \\ 0.22 + 0 - 0.22 = 0 \end{array}\right\}$$

$$\left\{\begin{array}{l} cost(\text{scan pkg trk}) + \mathcal{C}ost(\text{scan pkg trk}) + \\ I(\text{scan pkg trk, drive-1 d c}) = 0.22 + 0 - 0.22 = 0 \end{array}\right\}$$

therefore

$$\begin{aligned} \operatorname*{argmin}_{a\in\mathcal{A}_{\text{(scanned pkg trk)}}} & \{cost(a) + \mathcal{C}ost(a) + I(a, \text{drive-1 d c})\} \\ &= \text{(noop-scanned pkg trk)} \end{aligned}$$

Either (noop-scanned pkg trk) or (scan pkg trk) can be chosen. In this case, we pick action (noop-scanned pkg trk), which sets proposition (scanned pkg trk) as subgoals at level 1.

The same technique is applied for the rest of the layers until the initial state is reached. At level 1, the procedure first deals with subgoal (at d trk), followed by (scanned pkg trk), (verified pkg trk), (in trk pkg), and (¬flattire), with estimated costs 1.13, 0.22, 0.22, 0, and 0 respectively. In this way, starting with (at d trk), it is achieved by actions in $\mathcal{A}_{\text{(at d trk)}}$ at level 1. In order to know which is the best choice, we perform the following calculations:

$$\operatorname*{argmin}_{a\in\mathcal{A}_{\text{(at c trk)}}} \{cost(a) + \mathcal{C}ost(a)\}$$

$$cost(\text{drive-1 a d}) + \mathcal{C}ost(\text{drive-1 a d}) = 0.73 + 0.51 = 1.24$$
$$cost(\text{drive-2 a d}) + \mathcal{C}ost(\text{drive-2 a d}) = 0.73 + 0.91 = 1.64$$

therefore:

$$\operatorname*{argmin}_{a\in\mathcal{A}_{\text{(at a trk)}}} \{cost(a) + \mathcal{C}ost(a)\} = \text{drive-1 a d}$$

In this case, the selected action is (drive-1 trk a d) because it has the lowest cost. Action (drive-1 trk a d) sets propositions (at a trk), (¬flattire), (in trk pkg) and (verified pkg trk) as subgoals at level 0.

Next subgoal, (scanned pkg trk), it is achieved by action (scan pkg trk), which is selected because it is the only achiever of (scanned pkg trk) at level 0. Action (scan pkg trk) sets propositions (in trk pkg) and (verified pkg trk) as subgoals at level 0, but it was already set by (drive-1 trk a d).

Next subgoal, (verified pkg trk), it is achieved by actions in $\mathcal{A}_{\text{(verified pkg trk)}}$ at level 1. The cost of achieving (verified pkg trk) must be computed for actions in $\mathcal{A}_{\text{(verified pkg trk)}}$ assuming operators (drive-1 trk a d) and (scan pkg trk), and the following Interaction values:

$$
\begin{aligned}
&I(\text{verify pkg trk, drive-1 trk a d}) = \infty\\
&I(\text{verify pkg trk, scan pkg trk}) = -0.22\\
&I(\text{noop-verified pkg trk, scan pkg trk}) = -0.22\\
&I(\text{noop-verified pkg trk, drive-1 trk a d}) = -0.22
\end{aligned}
$$

That is:

$$
\begin{aligned}
\underset{a\in\mathcal{A}_{\text{(verified pkg trk)}}}{\text{argmin}} \quad & \left\{ \begin{array}{l} cost(a) + \mathcal{C}ost(a) + \\ I(a, \text{scan pkg trk}) + I(a, \text{drive-1 trk a d}) \end{array} \right\} \\
& \left\{ \begin{array}{l} cost(\text{verify pkg trk}) + \mathcal{C}ost(\text{verify pkg trk}) + \\ I(\text{verify pkg trk, scan pkg trk}) + \\ I(\text{verify pkg trk, drive-1 trk a d}) = \\ 0 + 0.22 - 22 + \infty = \infty \end{array} \right\} \\
& \left\{ \begin{array}{l} cost(\text{noop-verified pkg trk}) + \\ \mathcal{C}ost(\text{noop-verified pkg trk}) + \\ I(\text{noop-verified pkg trk, scan pkg trk}) + \\ I(\text{noop-verified pkg trk, drive trk a d}) = \\ 0.22 + 0 - 022 - 0.22 = -0.22 \end{array} \right\}
\end{aligned}
$$

therefore:

$$
\begin{aligned}
\underset{a\in\mathcal{A}_{\text{(verified pkg trk)}}}{\text{argmin}} \quad & \left\{ \begin{array}{l} cost(a) + \mathcal{C}ost(\text{verified pkg trk}|a) + \\ I(a, \text{scan pkg trk}) + I(a, \text{drive-1 trk a d}) \end{array} \right\} \\
= \quad & \text{noop-verified pkg trk}
\end{aligned}
$$

In this case, the selected action is (noop-verified pkg trk) because it has the lowest cost when (drive-1 trk a d) and (scan pkg trk) have already been chosen. Action (noop-verified pkg trk) sets proposition (verified pkg trk), but it was already set by (drive-1 trk a d).

Next subgoal, (in trk pkg), it is achieved by (load pkg a trk) and (noop-in trk pkg) at level 0. The cost of achieving (in trk pkg) must be computed for both of these assuming (scan pkg trk), (drive-1 trk a d), and (noop-verify pkg trk). In this case, the selected action is (noop-in trk pkg) because it is not mutually exclusive with any of the already chosen actions, while (load pkg a trk) is mutex with all of them.

The last subgoal, (¬flattire), it is only achieved by (¬flattire), which has no mutual exclusions with (scan pkg trk), (drive-1 trk a d), (noop-verify pkg trk), and (noop-in trk pkg). Therefore, it is selected.

Once the plan is extracted, we compute the heuristic estimation, which is the sum of the costs of every action outcome selected in the relaxed plan regression algorithm. In this way, we are considering the cost and Interaction information that has been computed previously in the relaxed plan extraction. Continuing with the example, suppose that the plan solution selected is composed of operators A and B, the CCE value would be:

$$\begin{aligned} \text{CCE}(n) &= \left\{ \begin{array}{l} \mathcal{C}ost(\text{drive-1 a d}) + \mathcal{C}ost(\text{scan pkg trk}) + \\ \mathcal{C}ost(\text{drive-1 d c}) + \mathcal{C}ost(\text{unload pkg trk c}) \end{array} \right\} \\ &= 0.22 + 0.22 + 0.73 + 1.24 = 2.41 \end{aligned}$$

This indicates that the estimated cost to reach the goal from that state is 2.41, i.e., a probability of $\exp\{-2.41\} = 0.089$.

5.2 Experimental evaluation

We have conducted an experimental evaluation on IPPC-06 (Bonet and Given, 2006) and IPPC-08 (Buffet and Bryce, 2008) fully-observable-probabilistic planning (FOP) domains. We have also conducted it on the *probabilistically interesting domains* introduced by Little and Thiebaux (2007), which consist of a number of very simple problems that explore the issue of probabilistic planning versus replanning. These problems lead to dead-ends. The domains vary by: 1) the number of dead-end states where the goal is unreachable, 2) the degree to which the probability of reaching a dead-end state can be reduced through the choice of actions, 3) the number of distinct trajectories from which the goal can be reached from the initial state, and 4) the presence of mutually exclusive choices that prevent alternative courses of actions.

The test consists of running the planner and using the resulting plan in the MDP Simulator (Younes et al., 2005). The planner and the simulator communicate by exchanging messages. The simulator first sends the planner the initial state. Then, the Interaction between planner and simulation consists of the planner sending an action and the simulator sending the next state to the planner. Since the simulator is stochastic, each plan is run many times to get an expected value. The experiments were conducted on an Pentium dual core processor at 2.4 GHz running Linux. For the rest of the planners, given that we were not able to obtain and run them ourselves, data are collected from work done by Yoon, Ruml, Benton, and Do (2010).

Below is a brief description of the domains used for the experimental evaluation.

IPPC-06

For IPPC-06 we are concerned with the probabilistic planning domains having full observability and probabilistic action outcomes (FOP track):

- Blocksworld: similar to the classical Blocksworld, but with additional actions. A gripper can hold a block or a tower of blocks or be empty. When trying to perform an action, there is a chance of dropping a block on the table.

- Exploding-Blocksworld: a dead-end version of the Blocksworld domain described above where additionally the blocks can explode. The explosion may affect the table or other blocks.

- Elevators: this domain consists of a set of coins arranged on different levels. To collect them, the elevators can move among the levels. The movements can fail if the elevator falls down the shaft and finishes on a lower level.

- Random: this domain is randomly generated with random preconditions and effects. A special reset-action can lead any state to the initial state, making it dead-end free.

- Tireworld: in this domain a car has to move between two locations. When the car drives a segment of the route, there is the chance of getting a flat tire. When this occurs the tire must be replaced. However, spare tires are not available in all locations.

- Zenotravel: this domain has actions to embark and disembark passengers from an aircraft that can fly at two alternative speeds between locations. The actions have a probability of failing without causing any effects. So, actions must sometimes be repeated.

IPPC-08

For IPPC-08 we are again concerned with the probabilistic planning domains having full observability and probabilistic action outcomes (FOP track):

- Blocksworld: similar to the IPPC-06 Blocksworld Domain.
- Exploding-Blocksworld: similar to the IPPC-06 Exploding-Blocksworld Domain.
- Triangle-tireworld: similar to the IPPC-06 Tire World Domain but with slight differences in the definition to permit short but dangerous paths.
- Rectangle-tireworld: this domain is inspired by Triangle-tireworld, except that there are no spare tires available. However, the domain defines an action that when the car gets a flat tire it can still go everywhere but at low probability.
- Zenotravel: similar to the IPPC-06 Zenotravel Domain.

Probabilistically Interesting Domains

Little and Thiébaux (Little and Thiébaux, 2007) have created a number of very simple problems that explore the issue of probabilistic planning versus replanning. These problems lead to dead-ends. The domains vary by 1) the number of dead-end states where the goal is unreachable, 2) the degree to which the probability of reaching a dead-end state can be reduced through the choice of actions, 3) the number of distinct trajectories from which the goal can be reached from the initial state, and 4) the presence of mutually exclusive choices that prevent alternative courses of actions.

- Climb: this domain consists of a person who is stuck on a roof because the ladder they used to climb up has fallen down. There are two options to get down: climbing down without the ladder but with a certain risk of injury or death, or calling for help from someone below to bring the ladder and then climb down with the ladder, which has no risk.
- River: in this domain a person on one side of the river needs to cross to the other side. There are three ways to achieve the goal with different chances of success.
- Tire1 & Tire10: domains based on the Tireworld domain, with different levels of difficulty to reach the final location without getting a flat tire.

The following planners have been used for the experimental evaluation:

- FF-Replan (Yoon et al., 2007): an online probabilistic planner that converts the probabilistic domain definitions into a deterministic domain using all-outcomes determinization. It then uses FF to compute a solution plan. During the execution, each time the planner leads to a state that is not expected, the planner searches for a new plan from the current unexpected state to the goal. FF-Replan won the 2004 International Probabilistic Planning Competition.
- FFH (Yoon et al., 2008): an FF-Replan planner that handles unexpected states by an online anticipatory strategy based on hindsight optimization.
- FFH$^+$ (Yoon et al., 2010): an improved FFH that uses methods to detect potentially useful actions and to reuse relevant plans. These methods allow the planner to reduce its computational cost.
- FPG (Buffet and Aberdeen, 2009): considers the planning problem as an optimization problem, and solves it using stochastic gradient ascent through the OLpomdp Algorithm (Baxter and Bartlett, 1999). The search is performed in policy space. This planner won the 2006 International Probabilistic Planning Competition.
- RFF (Teichteil-Königsbuch et al., 2010): uses the determinized problem generated by FF-Replan and then computes a policy by generating successive execution paths leading to the goal from the initial states by using FF. The generated policy has a low probability of failing during execution. This probability is computed by using a Monte-Carlo simulation. This planner won the 2008 International Probabilistic Planning Competition.

We compare these with four variants of the PIPSS planner:

- PIPSS^c: a PIPSS planner where the propagation of cost information through the plan graph does not consider Interaction estimates. During execution, the planner does not perform any further action when an unexpected state occurs.
- PIPSS^I: a PIPSS planner where the propagation of cost information through the plan graph considers Interaction estimates. Like PIPSS^c, during execution the planner does not perform any further action when an unexpected state occurs.
- PIPSS^c_r: a PIPSS planner where the propagation of cost information through the plan graph does not consider Interaction estimates. To deal with unexpected states at execution time, the planner does runtime replanning.

- PIPSS^I_r: a PIPSS planner where the propagation of cost information through the plan graph considers Interaction estimates. Like PIPSS^c_r, to deal with unexpected states at execution time, the planner does runtime replanning.

This section is divided into four subsections. The first one shows the tests performed with the different versions of our PIPSS planner. We explain the benefits of considering Interaction estimates and runtime replanning, and the CPU time for the four variants of PIPSS introduced. The other three subsections correspond to the set of domains used for experimental purposes. For these three, we show a table that represent the number of successful rounds. The results have been compared to those shown in (Yoon et al., 2010), given that we were not able to obtain and run them ourselves.

Test of the PIPSS planner versions

As mentioned above, the tests are performed in a simulation environment. Since PIPSS^c and PIPSS^I do not perform replanning when unexpected outcomes happen, we have evaluated the following strategies to consider a failure during runtime:

- *state* approach: fail when the state is not as expected. That is, the state given by the simulator is different from the state that the planner predicted during the search process.

- *prec* approach: fail when the state does not satisfy regressed preconditions. That is, the state given by the simulator does not satisfy the necessary preconditions to continue the remaining plan.

To illustrate the difference between these approaches consider the simple probabilistic Logistics domain, where there is a truck *trk* located at *warehouse*, and has a package *pkg* inside that has been already verified and has to be delivered to *store*. In order to accomplish this, the package needs to be scanned before it can be delivered, and the truck needs to be at the store. A tire may go flat during the trip. However, if this event happens, the truck reaches the store anyway. Figure 5.5 shows the plan generated by the planner. When the simulation starts, the planner sends the action (drive warehouse store) to the simulator. The performance of this action may give either a flat tire or not. If the outcome causes (flattire) and the simulation is working using *state*, then the process will stop and consider it as a failure. This is because the state was not that expected by the planner. On the other hand, using *prec* we can check that even though the car gets a flat tire, the preconditions for the remaining plan (scanned pkg) and (at store trk) are covered because the (flattire) fact does not affect the remaining plan. Thus, the plan can continue without causing a failure in the simulation.

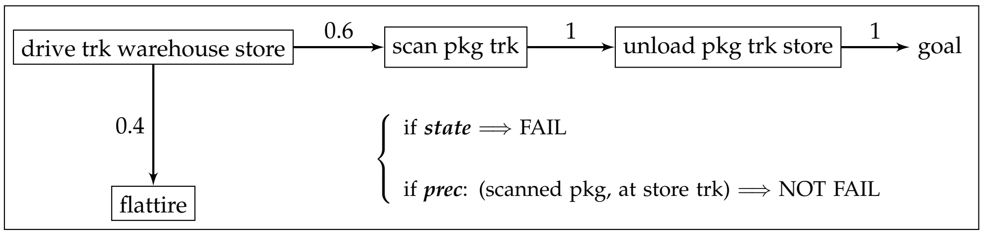

drive trk warehouse store
0.6
scan pkg trk
1
unload pkg trk store
1
goal
0.4
flattire
if state ⟹ FAIL
if prec: (scanned pkg, at store trk) ⟹ NOT FAIL

Figure 5.5: Example that shows the difference between the *state* and *prec* simulation approaches.

For all the planners, 500 trials per problem were performed with a total time limit of 500 minutes for the 500 trials. There are 15 problems for each domain. So, the maximum number of successful rounds for each domain is $15 \times 500 = 7500$. However, Triangle-tireworld domain on IPPC-08 only has 10 problems so that the total rounds in this case is $10 \times 500 = 5000$.

Tables 5.1, 5.2 and 5.3 show the number of successful rounds for the different versions of PIPSS in each domain. Results show us that the versions with a higher number of successful rounds are those that consider runtime replanning: PIPSS$_r^c$ and PIPSS$_r^I$. According to these tables, we can also confirm that failing during execution when the current state does not satisfy regressed preconditions (*prec*) works better than failing if the given state is not in the plan (*state*). Figure 5.6 shows the number of unsuccessful rounds that PIPSS$_s^I$ and PIPSS$_p^I$ found while performing the previous test. We can observe that using *prec*, the number of failures is usually lower than performing *state* in the domains Blocksworld, Exploding-Blocksworld, Elevator, Random and Triangle-Tireworld, where the plan's actions have fewer inter-dependences. This means that, if the system is doing runtime replanning, the replanning times will be lower. Because of this, we have performed PIPSS$_r^c$ and PIPSS$_r^I$ versions by applying the approach in which the planner performs replanning only when the state does not satisfy regressed preconditions. By using this strategy, we can deduce in advance if the remaining plan could be completely performed so that it would avoid replanning calls.

When we compare PIPSS$_r^c$ and PIPSS$_r^I$, PIPSS$_r^I$ holds a slight advantage overall. In general, in those domains where there are several mutex or non-mutex ways from which the goal can be reached from the initial state, Interaction estimates should find a plan with higher probability. In Figure 5.7, the generated plans for the two approaches have nearly identical probability except in one prob-

Table 5.1: Total number of successful rounds on the IPPC-06 using different versions of the PIPSS system.

	IPPC-06					
DOMAINS	PIPSS^c_s	PIPSS^c_p	PIPSS^I_s	PIPSS^I_p	PIPSS^c_r	PIPSS^I_r
Blocksworld	140	133	147	168	2762	**2951**
Exploding-Blocksworld	2386	4139	2357	**4137**	4038	4018
Elevators	4349	4414	4183	4235	6455	**6524**
Random	0	592	0	415	1508	**1511**
Tireworld	4393	4372	4392	4391	5548	**5614**
Zenotravel	500	500	500	500	**2285**	2207
TOTAL	11768	14150	11579	13846	22596	**22825**

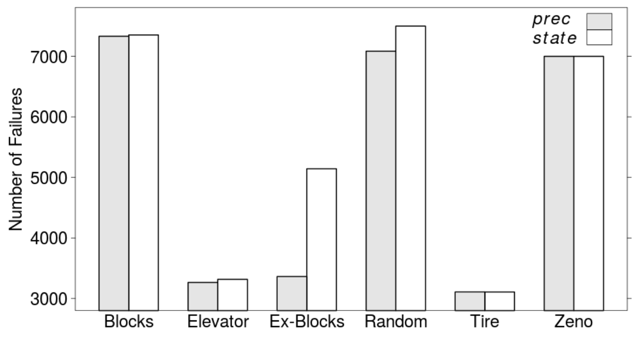
Table 5.2: Total number of successful rounds on the IPPC-08 using different versions of the PIPSS system.

	IPPC-08					
DOMAINS	PIPSS^c_s	PIPSS^c_p	PIPSS^I_s	PIPSS^I_p	PIPSS^c_r	PIPSS^I_r
Blocksworld	134	106	112	132	2358	**2493**
Exploding-Blocksworld	1461	3021	1469	**3083**	3001	3025
Triangle-Tireworld	335	327	**363**	340	338	343
Rectangle-Tireworld	288	0	**346**	0	314	320
Zenotravel	0	0	0	0	1730	**2000**
TOTAL	2218	3454	2290	3769	7741	**8181**

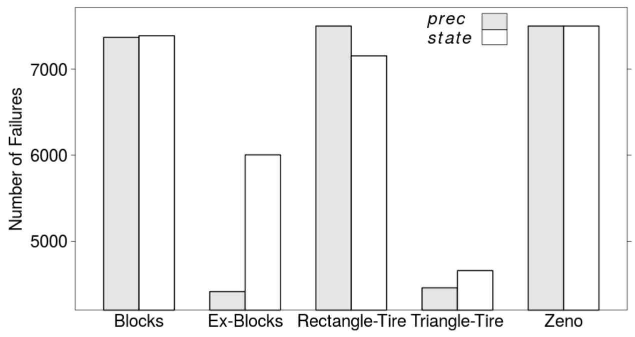
Table 5.3: Total number of successful rounds on Probabilistically Interesting Domains using different versions of the PIPSS system.

	PROBABILISTICALLY INTERESTING DOMAINS					
DOMAINS	PIPSS^c_s	PIPSS^c_p	PIPSS^I_s	PIPSS^I_p	PIPSS^c_r	PIPSS^I_r
Climb	**500**	**500**	**500**	**500**	**500**	**500**
River	257	248	255	271	254	**263**
Tire1	**266**	253	255	263	251	**266**
Tire10	0	0	0	0	0	0
TOTAL	1023	1001	1010	**1034**	1005	1029

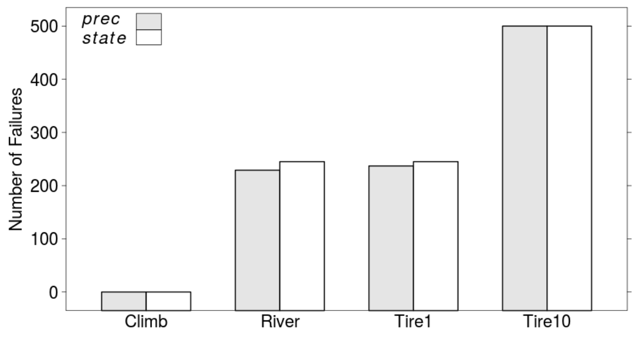
lem where the plan generated by PIPSS^I_r has significantly higher probability. For instance, in Figure 5.7(a) we can see that the probability generated by PIPSS^I_r in problem number six is higher than that found by PIPSS^c_r. In the case of the Rectangle-tireworld domain (Figure 5.7(c)), PIPSS^I_r generates a plan with higher probability than PIPSS^c_r for problem number three. In the case of the Elevator domain (Figure 5.7(b)), PIPSS$_r^I$ generates a plan with higher probability than PIPSS$_r^c$ in problem number ten.



(a) IPPC-06 Domains



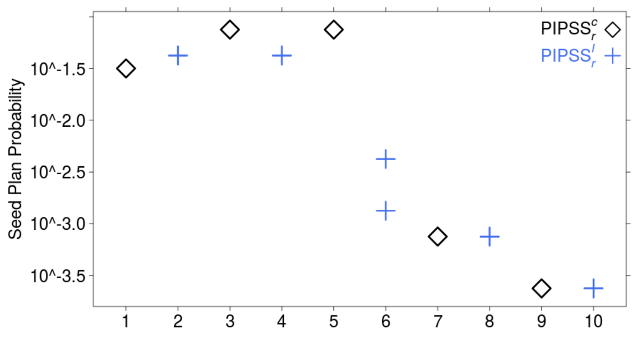
(b) IPPC-08 Domains



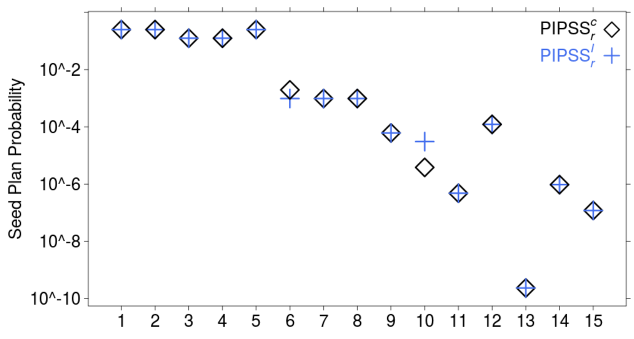
(c) Probabilistically Interesting Domains

Figure 5.6: Comparison between *state* and *prec* simulation approaches in number of failed rounds.

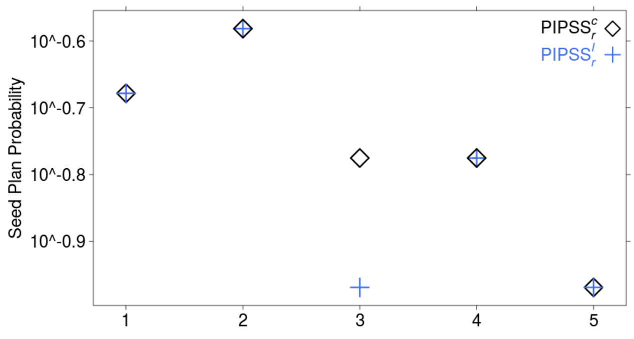
For Interaction alone, without any replanning, the number of problems with higher probability solutions is not large, but for the combination of Interaction estimates and runtime replanning the system's behavior improves considerably. In



(a) Blocksworld IPPC-06 Domain



(b) Elevator IPPC-06 Domain



(c) Rectangle-Tireworld IPPC-08 Domain

Figure 5.7: Comparison between the plan probability obtained by PIPSS^c and PIPSS^I in different domains.

this case, although the advantage of PIPSS^I over PIPSS^c is small for any individual planning episode, it compounds over the course of many replanning cycles, resulting in a more substantial advantage for PIPSS$_r^I$ over PIPSS$_r^c$.

Figure 5.8 shows the distribution of the time required to find the plan for PIPSS^c and PIPSS^I over all of the problems. The curves are very similar with a

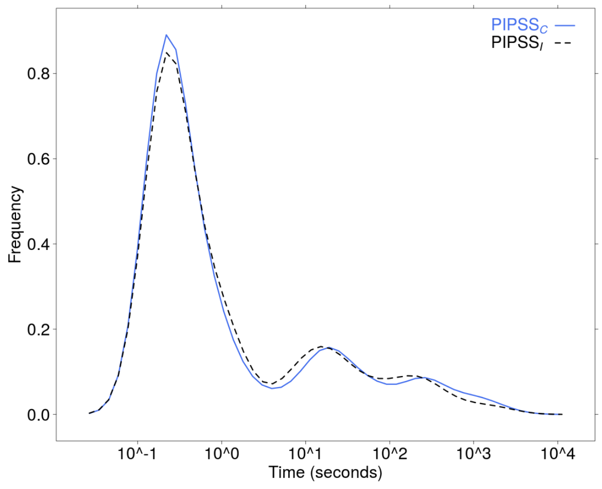


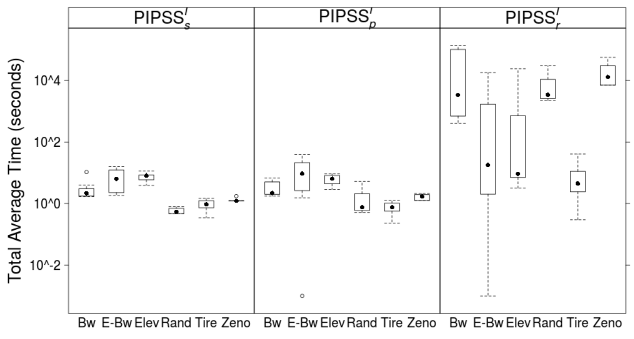
Figure 5.8: Distribution of Time for generating the plans.

distinct large peak and two other smaller but noticeable peaks. Most problems are solved in less than 5 seconds; however, the second peak shows that there are cases in which the planner requires 6–100 seconds. These cases correspond to the most difficult problems in Blocksworld-08, a single case in Blocksworld IPPC-06, and a few cases in Exploding-Blocksworld IPPC-06, Elevator and Rectangle-Tireworld. The third peak corresponds to the hardest problems in Exploding-Blocksworld IPCC-08 and Elevator, and with a single case in Random, Rectangle-Tireworld and Zenotravel IPPC-08. Problems whose time exceeds 1800 seconds are instances in which the planner does not find a solution within the allotted search time.

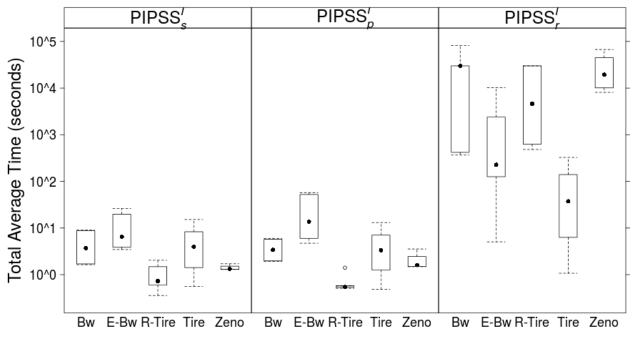
Figure 5.9 shows the total average time for simulation per domain while performing the previous test with the variants PIPSS^I_s, PIPSS^I_p and PIPSS^I_r. In general, we can observe that using the approach *prec* the time is slightly higher than applying *state* in some of the domains. However, as we have shown in Tables 5.1, 5.2 and 5.3, by performing *prec* the planner has a higher number of successful rounds. According to the graph, we also see a significant increase in the total time for PIPSS^I_r. This is largely due to the runtime replanning that this variant performs. However, the difference in number of successful rounds between the variants that do not use runtime replanning and the ones that do, is extremely large. So we think that the additional time is a worthwhile penalty.

The 2006 IPPC

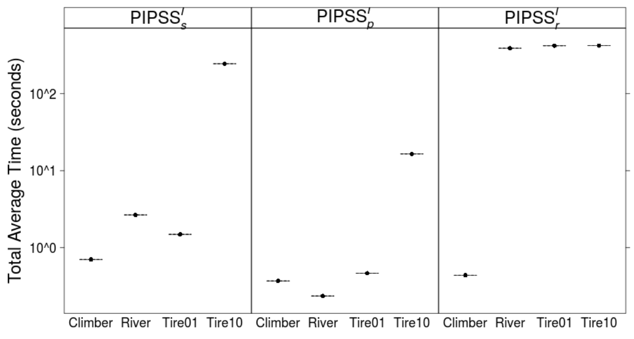
This subsection is concerned with the IPPC-06, and specifically with the fully observable probabilistic domains (FOP), which were described at the beginning



(a) IPPC-06 Domains



(b) IPPC-08 Domains



(c) Probabilistically Interesting Domains

Figure 5.9: Comparison between the total time required by PIPSS^I_s, PIPSS^I_p and PIPSS^I_r.

of the section. For all the planners, 30 trials per problem were performed (as in the competition) with a total time limit of 30 minutes for the 30 trials. There are 15 problems for each domain. So, the maximum number of successful rounds for each domain is $15 \times 30 = 450$.

Table 5.4 shows the number of successful rounds for FFH, FFH$^+$, FPG, PIPSS^c_r

and PIPSS$_r^I$ planners in each domain. PIPSS$_r^I$ gets good results in three of the five domains. The highest success rates are obtained in the domains Exploding-Blocksworld, Elevators, and Tireworld that have dead-end states even though PIPSS cannot recover from dead-end states. In fact, PIPSS$_r^I$ is the planner that achieves the highest rate in the Elevator domain. This is evidence that we are generating relatively low-cost (high-probability) plans. However, in domains like Blocksworld or Zenotravel, PIPSS performs poorly. Although we get good results in problems with dead-end states, we were somewhat surprised that in problems with no dead-ends (like the Blocksworld domain) we obtain worse results, since domains with no dead-ends are well suited to the replanning technique (a planner using a replanning strategy always finds a solution in a domain without dead-ends). Tables 5.5 shows information about the behavior of PIPSS$_r^I$ in the Blocksworld domain. For the smaller problems in this domain, PIPSS works perfectly by achieving all the rounds successfully. However, as the complexity of the problem increases, the performance of PIPSS is worse. We note that, the number of replanning calls during a simulation is related to the length of the plan generated. The greater the length of the plan, the greater the chance of failure. According to the data shown in Table 5.5, for the most difficult problems the replanning technique requires a significant amount of time to get a new plan. One reason for the large amount of time consumed for PIPSS is that it is being overwhelmed by the number of determinized ground actions in the domain. To corroborate this, Figure 5.10 shows the number of instantiated actions for each problem in each domain. The non-dead-end domains such as Blocksworld, Random or Zenotravel are those with the highest number of instantiated actions, which corresponded to the domains with low rates. In the Blocksworld domain, PIPSS only finds solutions for the first ten problems because for the rest of the problems the planner does not find a solution within the given time limit. We found additional evidence for this by performing the test for problems six and nine with the time limit increased by a factor of ten. This resulted in 100% successful rounds. We performed the same test for problems seven and eight getting 10 and 14 successful rounds. The total amount of time needed to achieve all the successful rounds for these two problems was 824 and 549 minutes respectively. We have found the same behavior in Zenotravel and Random domains; however, in these cases the planner only finds a solution for the first five problems within the given time limit.

All of this suggests that the performance of PIPSS could be improved by paying greater attention to minimizing the number of ground actions and improving the efficiency of plan graph construction and calculations.

Table 5.4: Total number of successful rounds on the IPPC-06.

	PLANNERS				
DOMAINS	FFH	FFH+	FPG	PIPSS$_r^c$	PIPSS$_r^I$
Blocksworld	256	**335**	283	160	162
Exploding-Blocksworld	205	**265**	193	233	239
Elevators	214	292	342	379	**396**
Random	301	**357**	292	83	62
Tireworld	343	**364**	337	342	360
Zenotravel	0	**310**	121	115	123
TOTAL	1319	**1923**	1568	1312	1342

Table 5.5: Execution Information Blocksworld IPPC-06 Domain using PIPSS$_r^I$.

BLOCKSWORLD					
Problem	Length Seed Plan	Time Seed Plan (seconds)	Replanning Calls	Average Replanning Time (seconds)	Successful Rounds
1	16	0.198	188	0.282	30
2	14	0.197	137	0.254	30
3	10	0.194	110	0.270	30
4	14	0.24	117	0.281	30
5	10	0.222	129	0.256	30
6	24	1.344	82	22.18	7
7	32	10.45	11	176.7	0
8	24	0.776	16	113.8	0
9	20	0.398	78	23.84	5
10	28	1.476	6	365.2	0

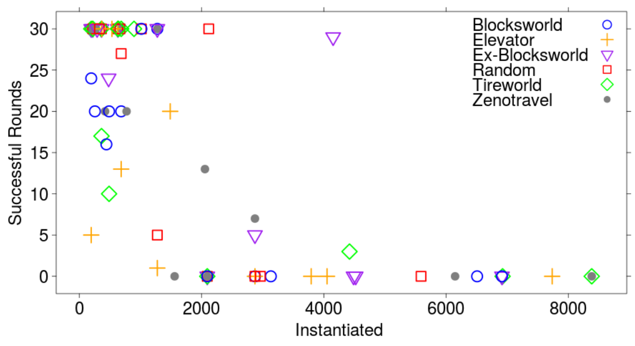


Figure 5.10: Number of instantiated actions in each domain of IPPC-06.

The 2008 IPPC

This subsection is concerned with the IPPC-08, and specifically with the fully observable probabilistic domains (FOP), which were described at the beginning of the section. For all the planners, 30 trials per problem were performed with a total time limit of 30 minutes for the 30 trials. There are 15 problems for each domain. So, the maximum number of successful rounds for each domain is $15 \times 30 = 450$. The results in Table 5.6 show that again PIPSS has a low success rate in those domains without dead-end states like Blocksworld. As is shown in Table 5.7, the behavior of our planner in this Blocksworld domain is the same as the previous section. The time consumed by replanning in the most difficult problems exhausted the given time. We can also confirm by Figure 5.11, that the number of determinized ground actions for this domain is large, and this fact influences the runtime replanning. In the Exploding-Blocksworld domain, PIPSS obtains better results than the winning RFF planner, although it has a low number of successful rounds compared to FFH$^+$. In the 2-Tireworld domain, PIPSS does poorly because this domain leads to a high number of dead-end states that it cannot recover from. PIPSS is unable to solve problems in SysAdmin-SLP and Search and Rescue domains from the 2008 IPPC, because it cannot parse PDDL `imply` and `exists` expressions.

Table 5.6: Total number of successful rounds on the IPPC-08.

	PLANNERS				
DOMAINS	FFH	FFH$^+$	RFF	PIPSS$_r^c$	PIPSS$_r^I$
Blocksworld	185	270	**364**	101	141
Exploding-Blocksworld	131	**214**	58	174	171
2-Tireworld	**420**	**420**	382	17	21
TOTAL	736	**904**	804	292	333

Table 5.7: Execution Information Blocksworld IPPC-08 Domain.

BLOCKSWORLD					
Problem	Length Seed Plan	Time Seed Plan (seconds)	Replanning Calls	Average Replanning Time (seconds)	Successful Rounds
1	12	0.213	115	0.245	30
2	12	0.208	123	0.247	30
3	12	0.216	151	0.234	30
4	12	0.198	0	0	30
5	26	19.66	65	28.45	6
6	26	19.15	55	33.93	5
7	26	19.33	51	35.35	6
8	26	22.05	61	29.70	4
9	28	1.307	4	442.9	0
10	28	1.276	4	437.5	0

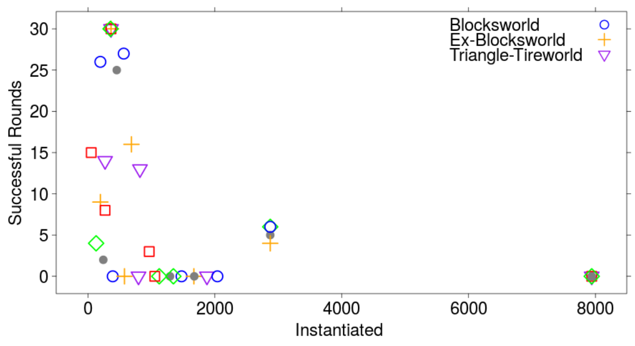


Figure 5.11: Number of instantiated actions in each domain of IPPC-08.

Probabilistically Interesting Domains

This subsection is concerned with the Probabilistically Interesting Domains, which were described at the beginning of the section. For all the planners, 30 trials per problem were performed with a total time limit of 30 minutes for the 30 trials. There is one problem for each domain so, the maximum number of successful rounds for each domain is 30. Runtime replanning does not result in any improvement in these domains due to the way that they are designed. However, we can see in Table 5.8 that both PIPSS$_r^c$ and PIPSS$_r^I$ get good results in almost all problems. In the Climb domain, both variants of PIPSS achieve the maximum number of successful rounds.This subsection is concerned with the Compared to FF-Replan, which only performs replanning as PIPSS does, we are performing better due to the higher probability plans. In the River domain, PIPSS$_r^I$ achieves the highest rate. However, for the Tireworld domains as the number of tires increases, PIPSS does not work as well due to the inability to deal with dead-end states in advance. Nevertheless, the result achieved by PIPSS in the other domains demonstrates the power of the heuristic.

5.3 Conclusions

This chapter presents an alternative for probabilistic planning under the Determinization and Replanning paradigm. This alternative uses a heuristic function that makes use of the probability information given in the domain definition to propagate cost and Interaction information through a plan graph. This heuristic estimator is used to guide the search toward high-probability plans. The resulting plans are used in a system that handles unexpected outcomes by runtime replanning.

According to the results of the 2006 IPPC and 2008 IPPC, the combination of

Table 5.8: Total number of successful rounds on Probabilistically Interesting Benchmarks.

	PLANNERS					
DOMAINS	FF-Replan	FFH	FFH^{+}	FPG	PIPSS$_r^c$	PIPSS$_r^I$
Climb	19	**30**	**30**	**30**	**30**	**30**
River	20	20	20	20	16	**23**
Tire1	15	**30**	**30**	**30**	16	21
Tire10	0	6	**30**	0	0	0
TOTAL	54	86	**110**	80	62	74

deterministic planning and replanning seems to work well. Although our planner does not deal with dead-end states in advance, the results dealing with probabilistic planning problems have high success rates in several cases. This is evidence that we are generating relatively high-probability plans. Unfortunately, the system does not get good results under some large domains with no dead-end states because of performance issues. A natural enhancement involves optimizing plan graph generation and calculation.

Of course, runtime replanning is not enough to deal with cases where there are dead-end states. That is why in the next chapter we present work that involves analyzing the generated plans to find potential points of failure that can be identified as recoverable or unrecoverable. Recoverable failures will be left in the plan and will be repaired through replanning at execution time. For each unrecoverable failure, we attempt to improve the chances of recovery, by adding precautionary steps such as taking along extra supplies or tools that would allow recovery if the failure occurs.

Chapter 6

Incremental Contingency Planning for Recovering from Uncertain Outcomes

This chapter presents a framework to improve probability of success of plans while retain scalability of the probabilistic solver presented in the previous chapter. Firstly, it introduces the idea of Precautionary Planning, on which our work is based. Then, it describes our framework for recovering uncertain outcomes. Finally, it presents an experimental evaluation.

6.1 Introduction

Incremental contingency planning is a framework that considers all potential failures in a plan and attempts to avoid them (Dearden et al., 2003). The classical approach to incremental contingency planning constructs a seed plan and incrementally adds contingency branches to the plan in order to improve the overall probability of it. This approach is practical in simple domains, but it is less useful when there are a large number of uncertainty sources. In those cases, the planner should focus on those outcomes that are unrecoverable or have low probability of recovery. Precautionary planning (Foss et al., 2007) is a form of incremental contingency planning. This interleaved planning and execution framework takes advantage of the speed of replanning, but only considers the unrecoverable outcomes in the plan and attempts to avoid them.

In this chapter, we present work inspired by precautionary planning. We describe an approach to incrementally generating contingency branches to deal with uncertain outcomes. The main idea is to first generate a high-probability non-branching seed plan, which is then augmented with contingency branches to handle the most critical outcomes of the actions in the plan. Any remaining outcomes are handled by runtime replanning. Figure 6.1 shows the architecture of our approach. It first generates a non-branching seed plan that is analyzed to find potential unexpected outcomes, which we generate by using PIPSS^I, the progressive heuristic planner introduced in Chapter 5. For the most critical outcomes, an attempt will be made to improve the chances of recovery by adding (1) a conformant solution that achieves the goal by using a different path, or (2) precautionary steps that allow recovery if the failure occurs . Both strategies will increase the overall probability of the plan. The process is repeated until (1) the resulting contingent plan achieves at least a given probability threshold, (2) the available time is exhausted, or (3) a certain number of branches are added. By critical outcomes, we mean that they are both likely and with poor chances of recovering.

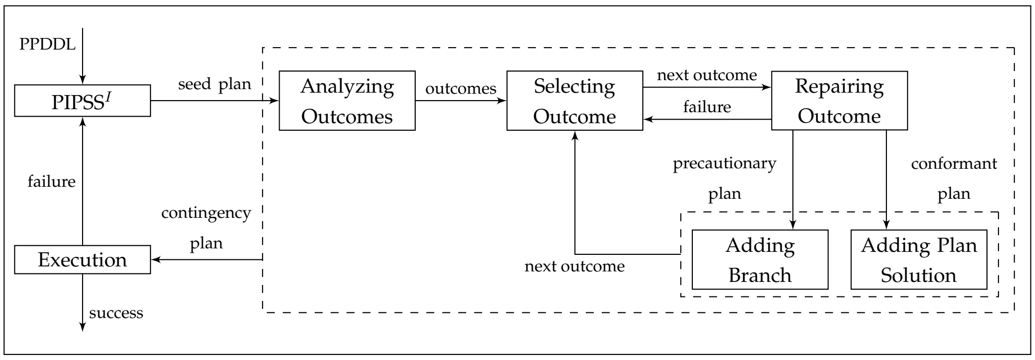


Figure 6.1: Precautionary Planning Architecture in the PIPSS^I System.

In the next section, we introduce our incremental approach. In Section 6.2, we define the heuristic function used to identify points of failure that potentially improve the total probability of the plan. In Section 6.3, we detail the different techniques we can apply to improve the chances of recovery if the failure occurs. In Section 6.4, we present an empirical study of these new techniques within PIPSS^I and compare with some other probabilistic planners.

6.2 Recognizing outcomes

Once a non-branching seed plan has been generated, we want to analyze all its potential unexpected outcomes, and estimate how much utility could be gained by inserting a branch at each given point of failure. A way to do this is by computing an estimate of how much the total probability could be potentially increased by using precautionary planning. We call this estimation *Gain*, and it is basically an estimate of how easy is to recover from a branch, if it happens. This measure is based on the CCE estimations and the h^I heuristic. A higher Gain, a higher chance of recovery.

To illustrate this approach consider that we are given the probabilistic planning problem shown in Figure 2.10, where there is a package *pkg* and a truck *trk* at location *a*, and the package needs to be delivered to location *c*. The truck can move between different locations, and it may have a flat tire during the trip with 0.4 probability. Assume that the *pkg* is inside the truck and it has been verified already. Figure 6.2 shows the non-branching seed plan generated by the planner, which has an estimated probability of reaching the goal equal to $\exp\{-\text{CCE(seedPlan)}\} = \exp\{-1.427115\} = 0.24$. Action (drive trk a b) has an alternative outcome o_1 with probability 0.4 and an estimated probability of reaching the goal equal to $\exp\{-\text{CCE}(o_1)\} = \exp\{\infty\} = 0$, which means that there is no chance of completing the objective, if this outcome actually happens – the tire goes flat and the truck cannot reach the goal. Action (drive trk b c) has an alternative outcome, o_2, with probability 0.4 and an estimated probability of reaching the goal equal to $\exp\{-\text{CCE}(o_2)\} = \exp\{0\} = 1$ because even though the tire goes flat, the truck still arrives to location c, and the remainder of the plan succeeds.

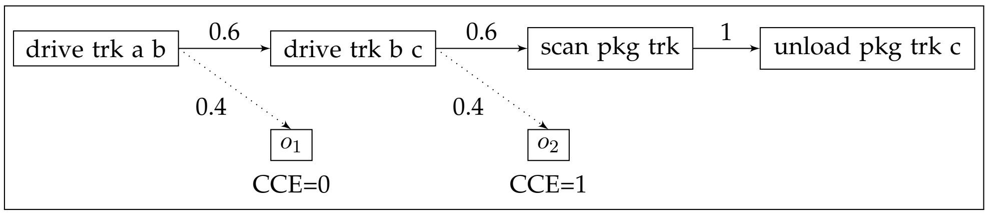


Figure 6.2: Example of a non-branching seed plan with potential outcomes to be repaired

If no contingency planning is performed and just rely on replanning when o_2 or o_1 occurs, we would get the following total probability:

$$\text{Pr} = 0.6(0.6)(1)\ +\ 0.4(0)\ +\ 0.6(0.4)(1)\ =\ 0.6$$

If contingency planning works perfectly for branch o_1, namely o_1', we assume a CCE(o_1') = 0.24. That is, the value of CCE(seedPlan) because this is the best value that we can get. This is the case where the truck gets a spare tire before driving from location *a* to location *b*, and we change it when o_1 occurs. Considering this assumption, the total probability that we would get is:

$$\text{Pr}(o_1) = 0.6(0.6) + 0.4(0.24) + 0.6(0.4)(1) = 0.696$$

In the case of outcome o_2, we already estimate that contingency planning works perfectly since its CCE is equal to 1. If we added a contingency branch for o_2, the probability of success of the plan would increase. However, it would not do better than just use the seed plan and rely on replanning.

For an alternative outcome (or branch) x of action a, the optimistic possible gain from precautionary planning will be the difference between the *estimated cost with repair* and the *estimated cost without repair*. We compute the latest using the CCE estimation as describes in Chapter 5. That is, the probability of reaching the goal from that state. On the other hand, to compute the *estimated cost with repair*, we propagate cost and Interaction in the plan graph only considering the outcome x and not taking into account other action outcomes. By doing this, we force outcome x to be in the plan, and, therefore, the new cost and Interaction information can be used to compute the probability estimated of reaching the goals when outcome x happens. We call this estimation *Probability Estimated* (PE) and is computed using Equation 3.15. More formally, for a branch x the gain is a measure of how much the total plan probability could potentially be increased by precautionary planning and is computed by the difference between the PE of branch x' when it actually happens and the CCE of x:

$$\text{Gain}(x) = \exp\{-\text{PE}(x')\} - \exp\{-\text{CCE}(x)\} \tag{6.1}$$

Following the previous example, for branches o_1 and o_2, the gains from precautionary planning are:

$$\begin{aligned}\text{Gain}(o_1') &= \exp\{-\text{PE}(o_1')\} - \exp\{-\text{CCE}(o_1)\} = 0.36 - 0 = 0.36\\ \text{Gain}(o_2') &= \exp\{-\text{PE}(o_2')\} - \exp\{- < \text{CCE}(o_2)\} = 0.36 - 1 = -0.64\end{aligned}$$

This means that by repairing branch o_1, the total plan probability will improve more than through branch o_2. Therefore, we would prefer to recover o_1 since it seems that it is possible to gain more probability mass, and o_2 might be recoverable by using replanning.

The calculation of gains allows us to create a ranking to start the recovery of alternative paths.

6.3 Repairing outcomes

Given the recognized undesirable outcomes ranking, the next step is to repair the plan in order to increase the overall probability of success. For each outcome, the idea is to look for the best improvement. In the next subsections, we present three methods to do that. The first method is called *Confrontation*, which tries to find a plan that avoids the problematic action's outcome when its execution depends on a condition. The second method is called *Precautionary Steps*, which adds precautionary actions before the problematic action to increase the probability of recovery it in case it happens. The third method is called *Conformant Planning*, which increases the total probability by adding conformant steps to the contingency plan solution.

6.3.1 Confrontation

A probabilistic outcome of an action may be subject to different conditions. In our example, it might be that for the action (unload pkg trk c), proposition ¬(at c pkg) occurs when, for instance, the store in c where the package needs to be delivered is closed. Confrontation on this condition will avoid ¬(at c pkg) by ensuring that the store is open before the start of driving. Figure 6.3 shows the high-level algorithm used.

Function CONFRONTATION (*a,o,p*)

a	≡	action causing the failure
c	≡	condition on the unrecoverable outcome of *a* occurs
o	≡	problem operators set
p	≡	PDDL problem espacification
g	≡	set of goals
plan	≡	new plan solution

```
1. $a' \leftarrow$ copy($a$)
2. $prec(a') \leftarrow prec(a') \cup (\neg c)$
3. $eff(a') \leftarrow eff(a') \cup$ (unique-effect)
4. $o \leftarrow \{o\} \cup a'$
5. $g \leftarrow \{g\} \cup$ (unique-effect)
6. $plan \leftarrow$ deterministicPlanner($o$, $p$)
7. return $plan$
```

Figure 6.3: The confrontation pseudo-algorithm.

The overall idea is to find a new plan that avoids or reduces the probability of getting to that branch, and then replace the old seed plan with the new plan. More precisely, suppose that *a* is the action in the seed plan with an unrecoverable outcome conditioned by *c*. We force the planner to find a new seed plan that achieves ¬*c* to prevent the failure from occurring. The way that we do that is by creating a new version of the action *a*, *a′*, that keeps its original preconditions plus a new additional precondition $\neg c$, and its original effects plus an additional unique effect. The unique effect is also added to the set of goals. We then add the new action to the set of operators and call the deterministic planner to find a plan for the goals. If a new plan is found and it has higher probability than the old seed plan, the new plan replaces the old seed plan.

In our example, suppose that the package *pkg* needs to be delivered in a store *s* at location *c*. The action of delivering the package, *unload*, has a conditional effect (not-close s), which will deliver the package if the store is not close. Figure 6.4 shows the new action *unload'*. It includes ¬(not-close s), the negation of the conditional effect, in its preconditions, and the proposition (unique-effect) in its effects. In addition, the new problem is defined and includes the proposition (unique-effect) in the goal set. The deterministic planner would return a plan with action (unload′) on it to guarantee that the store is open before the star of the driving.

```
(:action unload
 :parameters (?pkg - package ?t - truck ?l - location)
 :precondition (and (at ?l ?t) (scanned ?pkg ?t))
 :effect (and (not (in ?pkg ?t)) (at ?l ?pkg) (when (and (not-close s) (delivered pkg c))))
```

```
(:action unload'
 :parameters (?pkg - package ?t - truck ?l - location)
 :precondition (and (at ?l ?t) (scanned ?pkg ?t) (not (not-close s)))
 :effect (and (not (in ?pkg ?t)) (at ?l ?pkg) (delivered pkg c) (unique-effect)))
```

```
(define (problem logistics-p01)
 (:domain logistics)
 (:objects a b c d e - location trk - truck pkg - package
 (:init (connected a b) (connected a d) (connected b c) (connected d e)
        (connected e c) (at a trk) (at a pkg) (spare d) (spare e) (not (flattire)))
 (:goal (and (delivered pkg c) (unique-effect)))
```

Figure 6.4: Confrontation: new action and new problem definitions.

6.3.2 Precautionary Steps

Precautionary Steps consists of repairing an undesirable action's outcome by adding precautionary actions to the plan before the problematic action. This method improves the chance of recovery, if the seed plan fails, and makes possible to reach the goal when the unexpected outcome of the problematic action happens during runtime. Figure 6.5 shows the high-level algorithm used. The idea is to force the planner to find a plan that uses each alternative outcome, but does not lose any precondition needed to reach the goal when the action has the desired outcome. In other words, for each alternative outcome o of action a, the method:

1. Divides the initial seed plan into two parts: a *prefix*, which contains all ac-

tions preceding a, and a *suffix*, which contains all actions following a.

2. Creates a new action a' that keeps its original preconditions and effects.
3. Analyzes the causal structure of the suffix to collect all the preconditions needed by the suffix, and adds them to the set of preconditions of a'.
4. Adds a unique effect to a', which is added to the goal state to force a' into the plan.
5. Adds a' to the set of operators and calls the deterministic planner to find a plan for the new goal state. If a plan is found, the prefix is replaced with the prefix of the new plan and the suffix is added to it. Otherwise, we classify the outcome as *unrecoverable*.

Function PRECAUTIONARYSTEP (*a,o,p*)

a	≡	action causing the failure
o	≡	problem operators set
p	≡	PDDL problem especification
g	≡	set of goals
suffix	≡	plan containing the action's seed plan following *a*
newPlan	≡	precautionary plan
plan	≡	plan solution

```
1. $a' \leftarrow$ copy($a$)
2. $preconditions(a') \leftarrow preconditions(a) \cup$ causalStructure($suffix$)
3. $effects(a') \leftarrow effects(a) \cup unique\text{-}effect$
4. $o \leftarrow \{o\} \cup a'$
5. $g \leftarrow \{g\} \cup unique\text{-}effect$
6. $newPlan \leftarrow$ deterministicPlanner($o,p$)
7. $plan \leftarrow$ addBranch($newPlan,suffix$)
8. return $plan$
```

Figure 6.5: The precautionary steps pseudo-algorithm.

Returning to our example, assume that we are repairing outcome o_1. Figure 6.6 shows the new action created to repair o_1. It includes the proposition (unique-effect) in its effects. Its preconditions set remains the same because it already has all the preconditions necessary to enable the suffix. In addition, the new problem definition includes the proposition (unique-effect) in the goals set. The deterministic planner returns a new plan that has the precautionary action (get-tire), which increases chances of recovery in case the unexpected outcome o_1 occurs.

```
 (:action drive'
  :parameters (?from - location ?to - location ?p - person)
  :precondition (and (at ?from) (road ?from ?to) (not (flattire)))
  :effect (and (at ?to) (not (at ?from)) (flattire) (unique-effect)))
```

```
(define (problem logistics-p01)
 (:domain logistics)
 (:objects a b c d e - location trk - truck pkg - package
 (:init (connected a b) (connected a d) (connected b c)
        (connected d e) (connected e c) (at a trk) (at a pkg)
        (spare d) (spare e) (not (flattire)))
 (:goal (and (at c pkg) (unique-effect))))
```

Figure 6.6: Precautionary steps: new action and new problem definitions.

Figure 6.7 shows the contingency plan once outcome o_1 have been repaired by adding a revised suffix that includes the action (get-tire), and a contingency branch where the tire is changed after the outcome happens and the car gets a flat tire. On the other hand, o_2 does not need to be repair since it can be handled by runtime replanning.

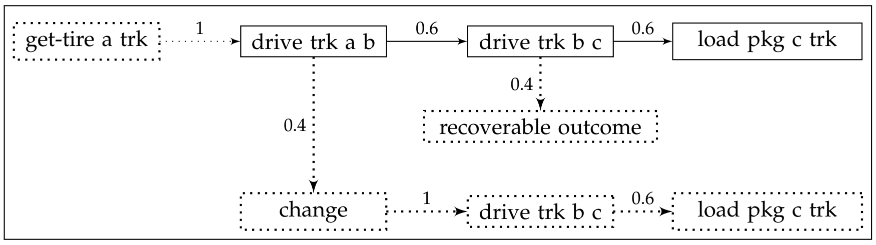


Figure 6.7: Recovering action outcomes by adding a precautionary step for alternative outcome o_1.

6.3.3 Conformant Plan

It is possible that there are several plans that reach the goal, which are not initially generated because they have lower probability. In some cases, one or more of these plans may be executable with the original seed plan and will raise the probability of the plan. Conformant plans may happen when the Precautionary Steps method is applied. This is the case where the plan that is generated contains action a' (the one forced to be in the plan), but it is only in the plan to achieve the unique effect. If this does not happen naturally during Precautionary Steps, then we can search for a conformant plan by calling the deterministic planner with the problematic action disabled.

As an example of this technique, consider the unrecoverable outcome of action (drive trk b c) shown in Figure 6.8. We can increase the overall probability of reaching the goal by simultaneously sending a second truck *trk2* to pick the package up. During execution time, both plans would be executed simultaneously. However, since the conformant plan generated might interfere with actions in the tail of the contingency plan, we need to find all the potential *execution conditions* and consider them during the execution of the plan. An execution condition is a proposition that determines which plan keeps being executed. If the execution condition is true, then the execution continues with the contingency plan. Otherwise, the execution continues with the conformant plan. In our example, only one of the trucks can pick the package up at location c. Therefore, during execution time, we need to consider the execution condition (at c trk), to disable either the conformant plan, if the proposition becomes true, or the contingency plan, if the proposition becomes false.

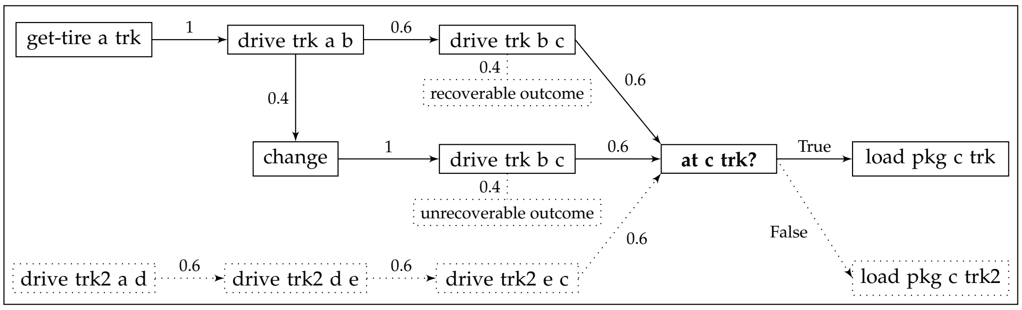


Figure 6.8: Recovering action outcomes by adding a conformant plan for the unrecoverable outcome of action (drive trk b c).

It may happen that the resulted conformant plan requires a suffix different to the suffix of the seed plan in order to be compatible with the seed plan. This revised suffix may decrease the probability of the seed plan. This is the case where, for instance, the truck $trk2$ in the conformant plan is a large truck that requires a driver with a specific license. The logistic company only has a driver with that license, and it was first assigned to drive truck trk. As a consequence, the revised suffix would content the necessary actions that (1) assign that driver to $trk2$ and (2) assign a new driver to trk. The actions in the revised suffix may have some probability of failure (for instance, the driver gets sick and cannot drive), and as a consequence of that, the overall probability of the seed plan may decrease.

6.4 Experimental evaluation

We have conducted an experimental evaluation on IPPC-06 (Bonet and Given, 2006) and IPPC-08 (Buffet and Bryce, 2008) fully-observable-probabilistic planning (FOP) domains, as well as on the *probabilistically interesting domains* introduced by Little and Thiebaux (Little and Thiébaux, 2007). The test consists of running the planner and using the resulting plan in the MDP Simulator (Younes et al., 2005). The planner and the simulator communicate by exchanging messages. The simulator first sends the planner the initial state. Then, the Interaction between planner and simulation consists of the planner sending an action and the simulator sending the next state to the planner.

FF-Replan, FHH, FHH$^+$, FPG, RFF planners described in Chapter 5 have been used for the experimental evaluation. We compare these with two variants of the PIPSS planner:

- PIPSS^I_r: a PIPSS planner where the propagation of cost information through the plan graph considers Interaction estimates. To deal with unexpected states at execution time, the planner does runtime replanning.
- C-PIPSS^I_r: a PIPSS planner where the propagation of cost information through the plan graph considers Interaction estimates. To deal with unexpected states, the plan solution has been incrementally augmented with confrontation, precautionary steps, and conformant plans. To deal with unexpected states at execution time, the planner does runtime replanning.

The experiments were conducted on a Pentium dual core processor at 2.4 GHz running Linux. For the rest of the planners, given that we were not able to obtain and run them ourselves, data are collected from work done by Yoon, Ruml, Benton, and Do (2010).

The 2006 IPPC

For the IPPC-06, we are concerned with the fully observable probabilistic domains (FOP), which were described in Chapter 5. For all the planners, 30 trials per problem were performed with a total time limit of 30 minutes for the 30 trials. There are 15 problems for each domain. So, the maximum number of successful rounds for each domain is $15 \times 30 = 450$.

Table 6.1 shows the number of successful rounds for FFH, FFH$^+$, FPG, PIPSS^I_r, and C-PIPSS^I_r planners in each domain. C-PIPSS^I_r gets good results in two of the three domains. The highest success rates are obtained in Exploding-Blocksworld

Table 6.1: Total number of successful rounds on the IPPC-06 using Incremental Contingency Planning.

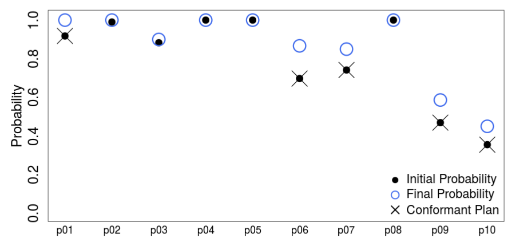
	PLANNERS				
DOMAINS	FFH	FFH$^+$	FPG	PIPSS$_r^I$	C-PIPSS$_r^I$
Exploding-Blocksworld	205	**265**	193	239	262
Elevators	214	292	342	**396**	382
Tireworld	343	**364**	337	360	356
TOTAL	762	921	872	995	**1000**

and Tireworld domains. In fact, C-PIPSS$_r^I$ is the planner that achieves the highest rate in the Exploding-Blocksworld domain. We expected that C-PIPSS$_r^I$ would perform better than PIPSS$_r^I$. However, in the Elevator domain, C-PIPSS$_r^I$ performs much poorer than PIPSS$_r^I$, and it is only slightly better in the Tireworld domain. Figure 6.9 shows data about the plan solution after applying Incremental Contingency Planning to the initial non-branching seed plan. The left column presents a plot for each domain in the IPPC-06 that shows the increase in probability after repairing the plan outcomes, and if a conformant branch has been added on the plan solution. The right column presents a scatter plot for each domain in the IPPC-06, where each dot in the plot represent the relationship between the *total number of outcomes in the plan* and the number of *recoverable outcomes*, and the *total number of outcomes in the plan* and the number of *unrecoverable outcomes*.

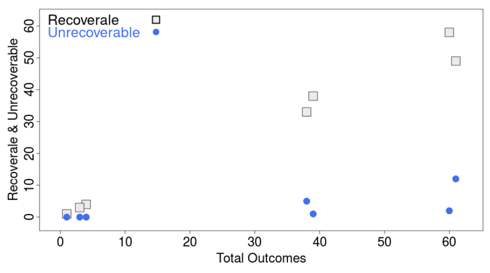
For the Exploding-Blocksworld domain, from Figure 6.9(a) we can see that the overall probability of the plan increases in all the problems. The reason for that is the high number of recoverable outcomes, which is shown in Figure 6.9(b). Therefore, the performance of C-PIPSS$_r^I$ is better than the performance of PIPSS$_r^I$.

For the Elevator domain, from Figure 6.9(c) we can see that the overall probability of the plan does not increase in any of the problems, but a conformant plan is added to each of them. Figure 6.9(d) shows that the number of unrecoverable outcomes is higher than the number of recoverable outcomes. This is why the overall probability does not increase. Although each problem has a conformant plan added, this additional plan has lower probability than the initial plan. Therefore, during execution the chances of failure are higher. However, the success rate of C-PIPSS$_r^I$ is higher than FFH and close to FFH$^+$.

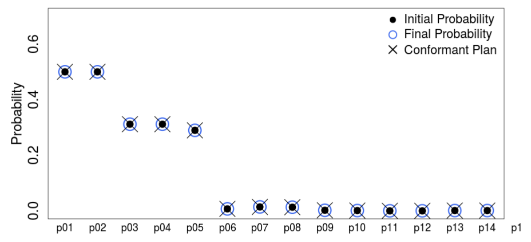
For the Tireworld domain, from Figure 6.9(e) we can see that the overall probability of the plan increases in half of the problems, and a conformant plan is added to each of them. Figure 6.9(d) shows that the number of recoverable outcomes is higher than the number of unrecoverable outcomes. However, C-PIPSS$_r^I$ performs just a bit better than PIPSS$_r^I$, where we expected a better performance.



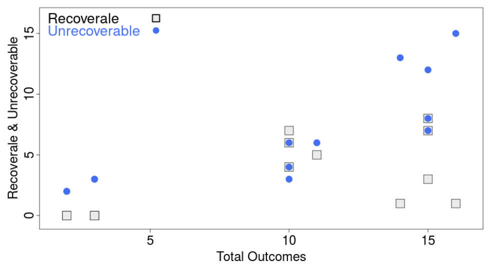
(a) Probability difference in the Exploding-Blocksworld Domain



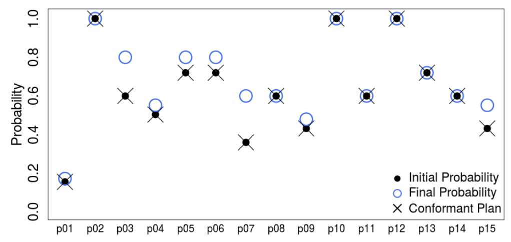
(b) Recoverable and unrecoverable outcomes in the Exploding-Blocksworld Domain



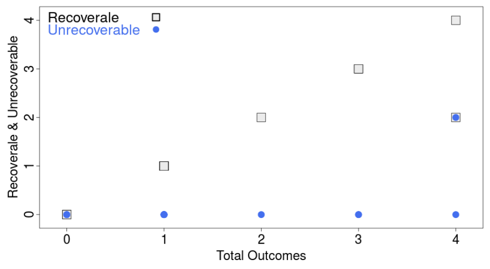
(c) Probability difference in the Elevator Domain



(d) Recoverable and unrecoverable outcomes in the Elevator Domain



(e) Probability difference in the Tireworld Domain



(f) Recoverable and unrecoverable outcomes in the Tireworld Domain

Figure 6.9: Comparison between initial and final plan probability (left column), and recoverable and unrecoverable plan outcomes (right column) of IPPC-06.

The success rate of C-PIPSS$_r^I$ is higher than FFH and FPG. FFH$^+$ performs slightly better.

The 2008 IPPC

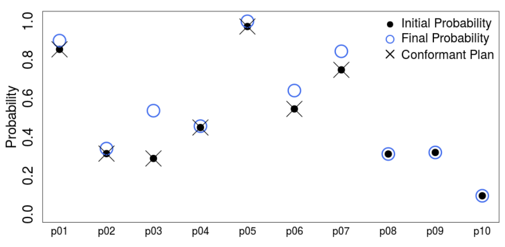
For the IPPC-08, we are again concerned with the fully observable probabilistic domains (FOP), which were described in Chapter 5. For all the planners, 30 trials per problem were performed with a total time limit of 30 minutes for the 30 trials.

There are 15 problems for each domain. So, the maximum number of successful rounds for each domain is $15 \times 30 = 450$.

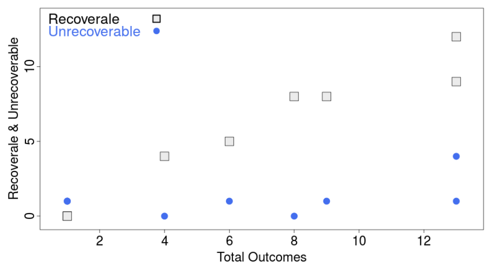
Table 6.2: Total number of successful rounds on the IPPC-08 using Incremental Contingency Planning.

	PLANNERS				
DOMAINS	FFH	FFH$^+$	RFF	PIPSS$_r^I$	C-PIPSS$_r^I$
Exploding-Blocksworld	131	**214**	58	171	176
2-Tireworld	**420**	**420**	382	21	68
TOTAL	551	**634**	440	192	244

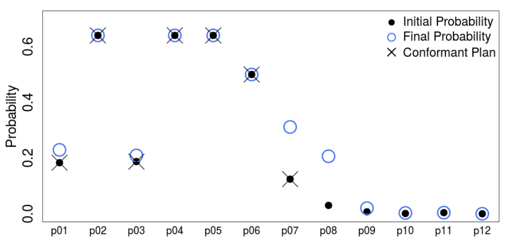
Table 6.2 shows the number of successful rounds for FFH, FFH$^+$, FPG, PIPSS$_r^I$, and C-PIPSS$_r^I$ planners in each domain. C-PIPSS$_r^I$ has a lower success rate than we expected. For the Exploding-Blocksworld domain, the success rate of C-PIPSS$_r^I$ does not improve, even though from Figure 6.10(a) we can see the overall probability increase for some of the problems. For the 2-Tireworld domain, the success rate of C-PIPSS$_r^I$ increases considerably compare to PIPSS$_r^I$, but it is still very low.



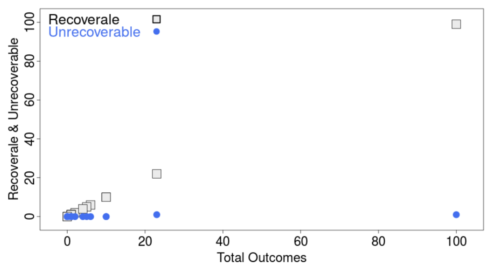
(a) Probability difference in the Exploding-Blocksworld Domain



(b) Recoverable and unrecoverable outcomes in the Exploding-Blocksworld Domain



(c) Probability difference in the 2-Tireworld Domain



(d) Recoverable and unrecoverable outcomes in the 2-Tireworld Domain

Figure 6.10: Comparison between initial and final plan probability, and recoverable and unrecoverable plan outcomes of IPPC-08.

Probabilistically Interesting Domains

Table 6.3 shows the number of successful rounds for FFH, FFH$^+$, FPG, PIPSS$_r^I$, and C-PIPSS$_r^I$ planners in each domain. From Figure 6.11 we can see that surprisingly there is no improvement in the overall probability for any of the tested domains. PIPSS$_r^I$, and C-PIPSS$_r^I$ have relatively high success rates in the Climb and River domain. However, we expected some improvement in the Tire1 and Tire10 domains after Incremental Contingency Planning was applied.

Table 6.3: Total number of successful rounds on the Probabilistically Interesting Benchmarks.

	Planners					
Domains	FF-Replan	FFH	FFH$^+$	FPG	PIPSS$_r^I$	C-PIPSS$_r^I$
Climb	19	**30**	**30**	**30**	**30**	**30**
River	20	20	20	20	**23**	20
Tire1	15	**30**	**30**	**30**	21	19
Tire10	0	6	**30**	0	0	0
Total	54	86	**110**	80	74	69

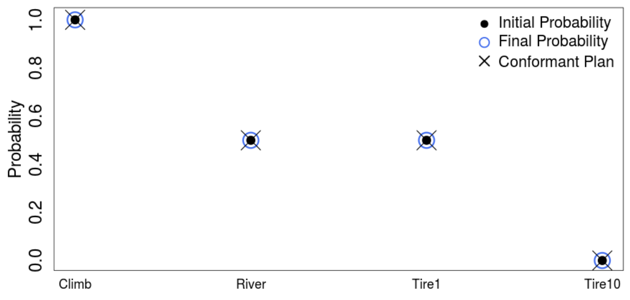


Figure 6.11: Probability difference on the Probabilistically Interesting Benchmarks.

6.5 Conclusions

In general, Incremental Precautionary Planning provides little additional benefit. In a few domains, Incremental Precautionary Planning can help; the success rates are higher, which means that the planner has been able to reach the goal in a larger percentage of problems. However, we expected that the combination of Incremental Precautionary Planning and runtime replanning would increase the success rate for all the tested domains. Our hypothesis for the bad performance of our framework is the classical all-outcomes determinization approach.

Our approach consists of generating a deterministic planning domain from a probabilistic planning domain, turning the probability information into costs. These costs are used in a heuristic function that propagates cost and Interaction information through a plan graph. This heuristic estimator is used to guide the search toward high-probability non-branching seed plans. The resulting plans are then analyzed to find potential points of failure that can be identified as recoverable or unrecoverable. Recoverable failures will be left in the plan and will be repaired through replanning at execution time. For each unrecoverable failure, we attempt to incrementally improve the chances of recovery by applying confrontation, adding precautionary steps, and adding conformant plans. The final plan is a contingency plan that has a higher probability of success during execution time. However, we observed that the cost and Interaction information underestimates the actual cost of propositions and actions in the plan graph, and therefore, the probability of each state in the search space.

To illustrate this, consider the simple probabilistic Logistics domain defined in Figure 2.10. Assume we use all-outcomes determinization, then the probabilistic domain results in a deterministic domain with two deterministic actions created from the probabilistic action *drive*. Figure 6.12 shows that determinization process. The most likely outcome of the action implies that the car successfully drives between locations with probability 0.6. This results in action *drive-1*. For the other outcome, the car achieves the destination, but it gets a flat tire with a probability of 0.4. This results in action *drive-2*.

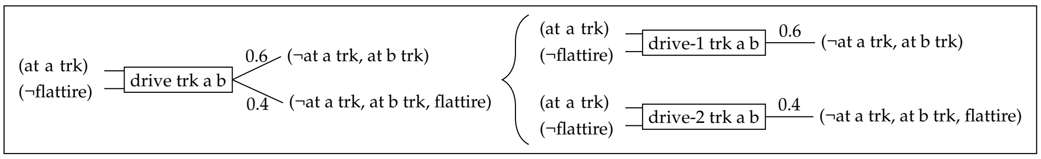


Figure 6.12: Example of all-outcomes determinization of a probabilistic action.

Figure 6.13 shows the plan solution that is generated by our planner. If we compute the probability of the resulting state after performing (drive-1 trk a b), the probability of (at b trk) and ¬(flattire) is 0.6. As a result, the probability of performing (drive b c) is equal to the product of the probability of its preconditions. That is, $pr(\text{at b trk})\,pr(\neg\text{flattire}) = 0.6(0.6) = 0.36$. As a consequence, the probability of (at c trk) is the product of the probability of performing (drive b c) and the probability of achieving (at c trk) through (drive b c). That is, $0.36(0.6) = 0.216$. This means that the probability of success of the plan is 0.216.

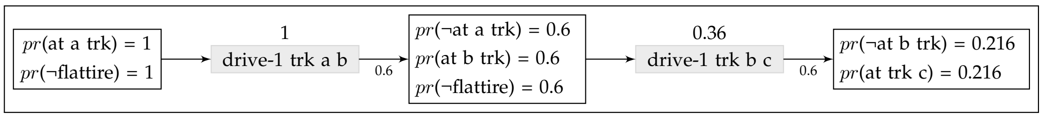


Figure 6.13: Plan solution probability using Determinization.

However, both outcomes of the probabilistic action (drive trk a b) result in the car at the next location. This means that, the probability of achieving (at b trk) is dependent of the outcome. The all-outcomes determinization technique does not consider the dependence between propositions and outcomes since it does not consider the overall probability of those propositions that are common in all the outcomes. Therefore, by considering propositions individually instead of as a result of an action's outcome, we can compute more accurate probability estimates.

To illustrate this, consider the example in Figure 6.14 that shows the same plan solution as before. In this case, the probability for each state is computed by considering probabilistic actions and propositions individually. Therefore, after (drive trk a b) is performed, there is a probabilistic state where ¬(at a trk) and (at b trk) have a probability of 1 (since both propositions are in both outcomes of the action), ¬(flattire) has a probability of 0.6, and (flattire) has a probability of 0.4. As a consequence of this, the probability of (drive trk b c) is 0.6, instead of 0.36 for the all-outcomes determinization. This results in (at c trk) having a probability of 0.6. Consequently, the probability of success of the plan is 0.6 where before was 0.216.

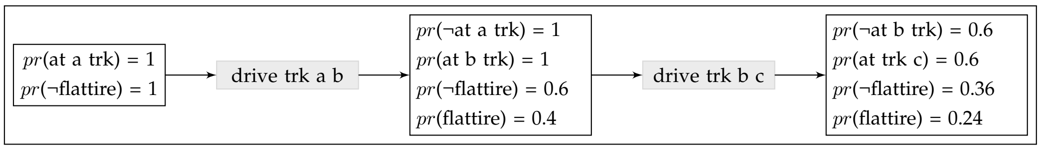


Figure 6.14: Plan solution probability considering the overall probability of propositions across action outcomes.

The underestimation that is caused by the all-outcomes determinization technique is, therefore, harmful to our cost propagation, yielding seed plans that do not have high probability of success. To illustrate this, consider the simple probabilistic Logistic problem in Figure 2.10. It has two possible paths that achieve the goal:

$$\begin{aligned}\pi_1 &= \{\text{(drive trk a b), (drive trk b c)}\}\\ \pi_2 &= \{\text{(drive trk a d), (drive trk d e), (drive trk e c)}\}\end{aligned}$$

As previously explained, π_1 has a probability of 0.6 of reaching the goal. On the other hand, π_2 has a probability of 1 of reaching the goal since both locations d and e have a spare tire. Therefore, if the truck got a flat tire in any of those locations, it would be able to change the tire and successfully continuing the drive.

The fact that we are not generating high probability of success plans means that the contingency branches added to the initial seed plan have low probability of success. For this reason, the number of successful rounds does not improve after applying Incremental Contingency Planning. In addition, the conformant plans added to the initial seed plan have even lower probability of success, which reduces the chances of achieving the goal.

Therefore, in the next chapter we present a new approach to compute estimates of probability , which considers the overall probability of each proposition across all of the action's outcomes, and the dependence between those propositions in the different outcomes. This new technique benefits the probability propagation in plan graphs and generates plans with higher probability of success.

Chapter 7

Probability Estimates without Determinization

In Chapter 5, we used action determinization and probability propagation in a plan graph to compute probability estimates, which are then used to guide the search towards high-probability plans. Unfortunately, this did not work as well as we expected. The reason is that multiple outcomes of an action may have the same proposition, so the actual probability of a proposition given an action is higher than the probability we can get from a single determinized action. To overcome this issue, in this chapter we introduce a technique to compute estimates of probability without determinization that are then used to guide the search towards high probability of success plans.

7.1 Introduction

All-outcomes determinization generates a deterministic action for each outcome of a probabilistic action. Considering each individual outcome might underestimate costs of propositions. In other words, for any single outcome of an action, the probability of a proposition may be lower than considering the probability across all the outcomes. As a result, action determinization as we did in Chapter 5 may mislead the planner into picking the wrong outcome or action. To illustrate this, consider the simple action A shown in Figure 7.1, which has three outcomes: outcome o_1 that produces x and y with a probability 0.3; outcome o_2 that produces x with a probability 0.3; and outcome o_3 that produces z with a probability 0.4. Outcomes o_1 and o_2 have a common proposition x, while outcome o_3 produces a different proposition that does not occur in any other outcome. Suppose that the three outcomes lead the planner to the goal with equal probability. The outcome

o_3 has a probability of 0.4, which is higher than the probability of either o_1 or o_2. Therefore, the planner may rely on outcome o_3. However, the combination of outcomes o_1 and o_2 will lead to a higher probability, and, therefore, it results in a better plan. (The true probability of x is a combination of outcomes o_1 and o_2.) If instead of considering each individual outcome, it would sum everything up, it would overestimate costs because it is not considering the dependence between propositions in the different outcomes.

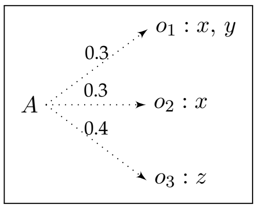


Figure 7.1: Example of a probabilistic action and its outcomes.

To overcome this issue we could use a determinization approach in which we created a new deterministic action for each possible proposition combination across all of the action's outcomes. To illustrate, consider the probabilistic action A in Figure 7.2. We could create deterministic actions A_1 for proposition x with probability 0.6 because x is in outcomes o_1 and o_2; A_2 for proposition y with probability 0.3 because y is only in outcome o_1; A_3 for proposition z with probability 0.4 because z is only in outcome o_3; and A_4 for the pair of propositions (x, y) from outcome o_1 with probability 0.3. There is no outcome that contains y and z or all three x, y, and z, so these possibilities do not need to be considered. These new deterministic actions would be mutually exclusive, and we can use them in probability propagation as we did in Chapter 5. However, this would significantly increase the number of actions in the plan graph and, therefore, increase propagation time.

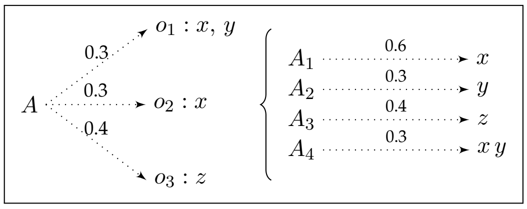


Figure 7.2: Example of a potential determinization technique.

For these reasons, we present a new way to estimate probabilities that we call *Probability Estimates without Determinization* (PEWD), which does not rely on determinization. This novel approach uses the probabilistic problem without transforming it into a deterministic one. Given a PPDDL problem, we initially process and load all the information given in the domain. Then, we build a probabilistic plan graph estimator as will be described in Section 7.4. This propagation technique is moderately different from the technique presented in Chapter 5 because it considers the dependence among propositions in action outcomes to avoid the reliance on individual outcomes. After this step, the search process starts. The system performs an A* search when it is looking for a solution. This search is not as usual since it is performed in a space of probabilistic states. For each probabilistic state, the plan graph is updated and the system estimates the probability of achieving the goals from that state. This estimation is called a *Completion Probability Estimate* (CPE). The result is more accurate estimates of probability that help to guide the search towards high probability of success plans.

The next section explains the Interaction concept in terms of probability. Section 7.3 describes the search in the space of probabilistic states. Section 7.4 describes the probabilistic plan graph heuristic used to guide the probabilistic search towards high-probability plans. Finally, Section 7.5 presents an empirical study of this technique and compares with some other probabilistic planners.

7.2 Translation from Cost Interaction to Probability Interaction

In Chapter 3, we define cost Interaction, I, between two elements as the cost of the conjunction minus the individual costs. I can be directly translated into probability Interaction, I_p, as the probability of the conjunction divided by the individual probabilities. I_p is a value that represents how more or less likely it is that two propositions or actions are established together instead of independently. Formally, the optimal Interaction, I_p^*, considers n-ary Interaction relationships among propositions and among actions in the plan graph. It is defined as:

$$I_p^*(p_0, p_1, ..., p_n) = \frac{pr^*(p_0 \wedge p_1 \wedge ... \wedge p_n)}{pr^*(p_0)\, pr^*(p_1) \ldots pr^*(p_n)} \tag{7.1}$$

where the term $pr^*(p_0 \wedge p_1 \wedge ... \wedge p_n)$ is the maximum probability among all the possible plans that achieve all the members in the set. Computing I_p^* would be computationally prohibitive. As a result, we limit the calculation of these values to pairs of propositions and pairs of actions in each level of a plan graph. In other words, binary Interaction:

$$I_p^*(p, q) = \frac{pr(p \wedge q)}{pr(p)\, pr(q)} \tag{7.2}$$

For this definition, I_p has the following features:

$$I^*_p(p,q) \text{ is } \begin{cases} > 1 & \text{if } p \text{ and } q \text{ are } \textit{synergistic} \\ = 1 & \text{if } p \text{ and } q \text{ are } \textit{independent} \\ < 1 & \text{if } p \text{ and } q \textit{ interfere} \\ = 0 & \text{if } p \text{ and } q \text{ are } \textit{mutually exclusive} \end{cases}$$

I_p provides information about the degree of interference or synergy between pairs of propositions and pairs of actions in a plan graph. When $0 < I_p(p,\, q) < 1$ it means that there is some interference between the best plans for achieving p and q, so it is less likely to achieve them both than to achieve them independently. In the extreme case, $I = 0$, the propositions or actions are mutually exclusive. Similarly, when $I_p(p,\, q) > 1$ the two elements are synergistic, which means that the probability of establishing both p and q is higher than the product of the probabilities for establishing the two independently. However, this probability cannot be higher than the probability of establishing the most difficult of p and q. As a result, $I_p(p,\, q)$ is a positive number ranging between zero and $1/\max\{pr(p),\, pr(q)\}$. That is:

$$I_p(p,q) \,\leq\, \min\left\{\frac{pr(p)}{pr(p)\,pr(q)}, \frac{pr(q)}{pr(p)\,pr(q)}\right\} \,=\, \frac{1}{\max\{pr(p),\, pr(q)\}}$$

This upper bound will be discussed in more detail in Section 7.4.2.

7.3 Search in the space of probabilistic states

A probabilistic state s is a set of propositions where propositions have a probability of being true and there is Interaction between them. This means that, for each proposition p in s we need to compute the probability of p being true in that particular probabilistic state s. In addition, there might be Interaction between propositions in s. Therefore, we also need to compute the Interaction between each pair of propositions. This state s serves as a compact approximation for the probability distribution over the individual states in the Markov space. More formally, a probabilistic state s consists of a set of propositions with individual probabilities $\mathcal{P}r(x)$ together with a probability Interaction $I_p(x,y)$ for all pairs x and y in s.

The following subsections describe in detail how to compute the probability and Interaction information in a probabilistic state.

7.3.1 Calculating probabilities for a probabilistic state

Consider a probabilistic state s and let $s\prime$ be the new state after attempting to perform action a in s. The probability of a proposition $x\prime$ in $s\prime$ is given by the probability of getting the proposition when the action succeeds plus the probability of getting the proposition when the action fails. That is:

$$\begin{aligned}
\mathcal{P}r(x\prime) &= pr(x\prime|a)\,pr(a) + pr(x\prime|\neg a)\,pr(\neg a) \\
&= pr(x\prime|a)\,pr(a) + pr(x|\neg a)\,pr(\neg a) \quad (x \text{ was true before } a \text{ given } \neg a) \\
&= pr(x\prime|a)\,pr(a) + pr(x)\,pr(\neg a|x) \qquad (\text{applying Baye's rules}) \\
&= pr(x\prime|a)\,pr(a) + pr(x)\,(1 - pr(a|x)) \\
&= pr(x\prime|a)\,pr(a) + pr(x)\,(1 - pr(\mathcal{P}_a|x)) \\
&= pr(x\prime|a)\,pr(a) + pr(x) - pr(x)\,pr(\mathcal{P}_a|x) \\
&= \underbrace{pr(x\prime|a)\,pr(a)}_{T_1} + \underbrace{pr(x) - pr(x \wedge \mathcal{P}_a)}_{T_2} \qquad (7.3)
\end{aligned}$$

The first term T_1 can be rewritten in terms of the action's outcomes as:

$$pr(a)\,pr(x\prime\,|\,a) \;=\; pr(a) \sum_{o \in \mathcal{O}(a)} pr(o)\,pr(x\prime\,|\,o, a)$$

where the conditional probability of x given an outcome o of action a is defined as:

$$pr(x|o,a)\ is \begin{cases} 1 & \text{if } (x \in o) \\ 0 & \text{if } (\neg x \in o) \\ pr(x|\mathcal{P}_a) & \text{if } (x, \neg x \notin o) \end{cases}$$

In other words, if the outcome o produces the proposition x, then the conditional probability is 1 (the outcome is considered). If o produces $\neg x$, then the conditional probability is 0 (the outcome is not considered). Finally, if o does not produce either x or $\neg x$, then it is necessary to compute the probability that x persists through a, which depends on the probability of x given the preconditions of a:

$$pr(x|\mathcal{P}_a)\ is \begin{cases} 1 & \text{if } (x \in \mathcal{P}_a) \\ 0 & \text{if } (\neg x \in \mathcal{P}_a) \\ pr(x) \prod_{p_i \in \mathcal{P}_a} I_p(x, p_i) & \text{if } (x, \neg x \notin \mathcal{P}_a) \end{cases}$$

In other words, if the proposition x belongs to the action's preconditions, the conditional probability is 1 (the outcome is considered). If $\neg x$ belongs to the action's preconditions, the conditional probability is 0 (the outcome is not considered). If x and $\neg x$ do not belong to the action's preconditions, then it is necessary to compute the probability that x holds given the preconditions of a, which is the probability of x times the Interaction of x with the preconditions of a.

The second term T_2 in Equation 7.3 computes the probability of the proposition assuming that the action fails. The first term in T_2 refers to the probability of the proposition before the action is applied. The second term in T_2 refers to the probability that x is consistent with the preconditions of a, which is given as:

$$pr(x \wedge \mathcal{P}_a) \;\; is \;\; \begin{cases} pr(a) & \text{if } (x \in \mathcal{P}_a) \\ \\ pr(a)\, pr(x) \prod\limits_{p \in \mathcal{P}_a} I_p(p, x) & \text{if } (x \notin \mathcal{P}_a) \end{cases}$$

In other words, if the proposition x belongs to the action's preconditions, the term reduces to the probability of the action. Otherwise, it is necessary to consider the relationship between x and the action's preconditions.

Figure 7.3 shows a simple probabilistic Tire domain and problem where there is a truck at location *a* that needs to drive to location *c*. The truck can drive between locations when it does not have a flat tire since a tire may go flat during the drive with a probability of 0.4. In addition, there is a spare tire at location *d*. A flat tire can be changed, if the truck has picked up a spare tire.

Figure 7.4 shows the transition process from a probabilistic state S_0 to a probabilistic state S_1 for the simple probabilistic Logistics problem in Figure 7.3. The probabilistic state S_0 is the initial state, where each proposition has probability equal to 1. The probabilistic state S_1 is the result of applying (drive trk a b) to S_0. The probability of (at b trk) in S_1 after applying (drive trk a d) is:

$$\begin{aligned} \mathcal{P}r(\text{(at b trk)}') &= pr(\text{(at b trk)}'|\text{drive trk a d})\, pr(\text{drive trk a d}) + pr(\text{at b trk}) - \\ & \quad\; pr(\text{at a trk} \wedge \neg\text{flattire} \wedge \text{at b trk}) \end{aligned}$$

where $pr(\text{at b trk})$ and $pr(\text{at a trk}\wedge\neg\text{flattire}\wedge\text{at b trk})$ are equal to 0 since (at b trk) is not present in S_0. Therefore, $pr(\text{at b trk})$ reduces to $pr(\text{(at b trk)}'|\text{drive trk a d})$ times $pr(\text{drive trk a d})$, where:

$$\begin{aligned} pr(\text{(at b trk)}'|\text{drive trk a d}) &= pr(o_1)\, pr(\text{(at b trk)}' \mid o_1, \text{drive trk a d}) + \\ & \quad\; pr(o_2)\, pr(\text{(at b trk)}' \mid o_2, \text{drive trk a d}) \\ &= 0.6\,(1) + 0.4\,(1) = 1 \end{aligned}$$

and:

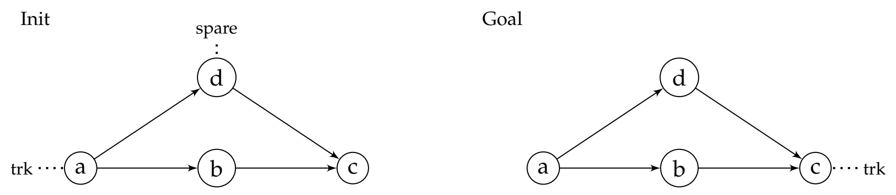
$$\begin{aligned} pr(\text{drive trk a d}) &= pr(\text{at a trk})\, pr(\neg\text{flattire})\, I_p(\text{at a trk}, \neg\text{flattire}) \\ &= 1\,(1)\,(1) = 1 \end{aligned}$$

```
(:action drive
 :parameters (?trk - truck ?from - location ?to - location)
 :precondition (and (connected ?from ?to) (at ?from ?trk) (not (flattire)))
 :effect (and (not (at ?from ?trk)) (at ?to ?trk)
         (probabilistic 0.4 (flattire))))

(:action get-tire
 :parameters (?trk - truck ?l - location)
 :precondition (and (at ?l ?trk) (spare ?l))
 :effect (and (not (spare ?l)) (hasspare)))

(:action change
 :precondition (hasspare)
 :effect (and (not (hasspare)) (not (flattire))))
```

(a) Logistics domain



(b) Logistics problem

Figure 7.3: A PPDDL domain and problem description on the Tire domain.

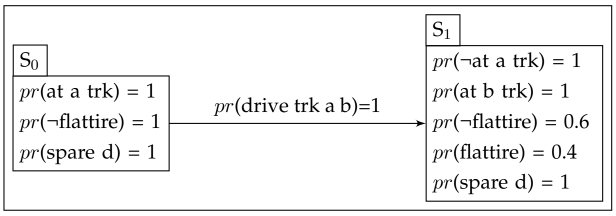


Figure 7.4: Example of the transition from a probabilistic state to another probabilistic state.

So the result is equal to 1 since both outcomes produce the proposition. As a result:

$$\mathcal{P}r((\text{at b trk})') \quad = \quad pr(\text{at b trk}|\text{drive trk a d})\, pr(\text{drive trk a d}) = 1\,(1) = 1$$

This means that, after performing (drive trk a d), (at b trk) has probability 1 regardless of the action's outcomes.

Another example is to calculate the probability of ¬(flattire) in S_2 after applying (drive trk a d). This is calculated as follows:

$$\begin{aligned}\mathcal{P}r(\neg\text{flattire'}) &= pr(\neg\text{flattire'}|\text{drive trk a d})\,pr(\text{drive trk a d}) + pr(\neg\text{flattire}) - \\ &\quad pr(\text{at a trk} \wedge \neg\text{flattire})\end{aligned}$$

where $pr(\neg\text{flattire})$ is given in S_0 and is equal to 1; $pr(\text{at a trk}\wedge\neg\text{flattire'})$ is equal to 1 since ¬(flattire) belongs to the preconditions of (drive trk a d); $pr(\text{drive trk a d})$ was previously computed and is equal to 1; and $pr(\neg\text{flattire'}|\text{drive trk a d})$ is computed as:

$$\begin{aligned}pr(\neg\text{flattire'}|\text{drive trk a d}) &= pr(o_1)\,pr(\neg\text{flattire'} \mid o_1, \text{drive trk a d}) + \\ &\quad pr(o_2)\,pr(\neg\text{flattire'} \mid o_2, \text{drive trk a d}) \\ &= 0.6\,(1) + 0.4\ = 0.6\end{aligned}$$

where o_2 produces (flattire), the counterpart proposition of ¬(flattire), and therefore, o_2 is not considered. For o_1, it does not produce either ¬(flattire) or (flattire) so it is necessary to compute how ¬(flattire) interferes with (drive trk a d) preconditions at the previous state. In this example, ¬(flattire) belongs to the (drive trk a d) preconditions. Therefore, the probability is:

$$\begin{aligned}pr(\mathcal{P}_{(\text{drive trk a d})} \wedge \neg\text{flattire}) &= pr(\text{at a trk} \wedge \neg\text{flattire}) \\ &= pr(\text{at a trk})\,pr(\neg\text{flattire})\,I_p(\text{at a trk}, \neg\text{flattire}) \\ &= 1\,(1)\,(1) = 1\end{aligned}$$

As a result:

$$\begin{aligned}\mathcal{P}r(\neg\text{flattire'}) &= pr(\neg\text{flattire'}|\text{drive trk a d})\,pr(\text{drive trk a d}) + pr(\neg\text{flattire}) - \\ &\quad pr(\text{at a trk} \wedge \neg\text{flattire}) = 0.6\,(1) + 1 - 1 = 0.6\end{aligned}$$

This means that, after performing (drive trk a d), ¬(flattire) has probability 0.6 of being true.

7.3.2 Calculating Interaction information for a probabilistic state

Considering the new probabilistic state $s\prime$, the Interaction for each pair of propositions in $s\prime$ is:

$$I_p(x\prime, y\prime) = \frac{pr(x\prime \wedge y\prime)}{pr(x\prime)\,pr(y\prime)} \ \leq\ \frac{1}{\max\{pr(x), p(y)\}}$$

where the conjunction probability of $x\prime$ and $y\prime$ is given by the probability of getting both when the action succeeds plus the probability of getting both when the action fails. That is:

$$\begin{aligned}
pr(x\prime \wedge y\prime) &= pr(x\prime \wedge y\prime|a)\,pr(a) + pr(x \wedge y|\neg a)\,pr(\neg a) \\
&= pr(x\prime \wedge y\prime|a)\,pr(a) + pr(x \wedge y|\neg a)\,pr(\neg a) \\
&= pr(x\prime \wedge y\prime|a)\,pr(a) + pr(x \wedge y)\,pr(\neg a|x \wedge y) \\
&= pr(x\prime \wedge y\prime|a)\,pr(a) + pr(x \wedge y)\,(1 - pr(a|x \wedge y)) \\
&= pr(x\prime \wedge y\prime|a)\,pr(a) + pr(x \wedge y) - pr(x \wedge y)\,pr(a|x \wedge y) \\
&= pr(x\prime \wedge y\prime|a)\,pr(a) + pr(x \wedge y) - pr(x \wedge y)\,pr(x \wedge y \wedge a) \\
&= \underbrace{pr(x\prime \wedge y\prime|a)\,pr(a)}_{T_1} + \underbrace{pr(x \wedge y) - pr(x \wedge y \wedge \mathcal{P}_a)}_{T_2} \qquad (7.4)
\end{aligned}$$

The term T_1 in Equation 7.4 can be rewritten in terms of the action's outcomes as:

$$pr(x\prime \wedge y\prime|a)\,pr(a) = pr(a) \sum_{o \in \mathcal{O}a} pr(o)\,pr(x\prime \wedge y\prime|o,a)$$

where the conditional probability of $x\prime$ and $y\prime$ given an outcome o of action a is given as:

$$pr(x\prime \wedge y\prime|o,a)\ is \begin{cases} 1 & \text{if } (x, y \in o) \\ 0 & \text{if } (\neg x \in o \ \vee \ \neg y \in o) \\ pr(y|\mathcal{P}_a) & \text{if } (x \in o \ \wedge \ y, \neg y \notin o) \\ pr(x|\mathcal{P}_a) & \text{if } (x, \neg x \notin o \ \wedge \ y \in o) \\ pr(x \wedge y|\mathcal{P}_a) & \text{if } (x, \neg x, y, \neg y \notin o) \end{cases}$$

In other words, if the outcome o produces propositions x and y, then the conditional probability is 1 (the outcome is considered). If o produces $\neg x$ or $\neg y$, then the conditional probability is 0 (the outcome is not considered). If o produces x and does not produce y or $\neg y$, then it is necessary to compute the probability that y persists through a, which depends on the probability of y given the preconditions of a. If o produces y and does not produce x or $\neg x$, then it is necessary to compute the probability that x persists through a, which depends on the probability of x given the preconditions of a. If o does not produce x, $\neg x$, y, or $\neg y$, then it is necessary to compute the probability that x and y both persist through a, which depends on the probability of x and y given the preconditions of a, which is given as:

$$pr(x \wedge y|a) \; is \; \begin{cases} 1 & \text{if } (x, y \in \mathcal{P}_a) \\ 0 & \text{if } (\neg x \in \mathcal{P}_a \;\vee\; \neg y \in \mathcal{P}_a) \\ pr(x) \prod\limits_{p \in \mathcal{P}_a} I_p(x,p) & \text{if } (x \notin \mathcal{P}_a \;\wedge\; y \in \mathcal{P}_a) \\ pr(y) \prod\limits_{p \in \mathcal{P}_a} I_p(y,p) & \text{if } (x \in \mathcal{P}_a \;\wedge\; y \notin \mathcal{P}_a) \\ pr(x)pr(y) \prod\limits_{p \in \mathcal{P}_a} I_p(x,p)\, I_p(y,p) & \text{if } (x, \neg x, y, \neg y \notin \mathcal{P}_a) \end{cases}$$

In other words, if propositions x and y belong to the action's preconditions, the conditional probability is 1 (the outcome is considered). If $\neg x$ or $\neg y$ belongs to the action's preconditions, the conditional probability is 0 (the outcome is not considered). If x does not belong to the action's preconditions, but y does, it is necessary to compute the relation at the previous probabilistic state between x and the action's preconditions as in Equation 7.5. If y does not belong to the action's preconditions, but x does, it is necessary to compute the relation at the previous probabilistic state between y and the action's preconditions as in Equation 7.5. If x, $\neg x$, y, and $\neg y$ do not belong to the action's preconditions, it is necessary to compute the relation at the previous probabilistic state between x and y, and the action's preconditions as in Equation 7.5.

The term T_2 in Equation 7.4 computes the probability of the conjunction x and y assuming that the action fails. The first term in T_2 refers to the conjunction probability of x and y before a is applied. The second term in T_2 refers to the probability that x and y are consistent with the preconditions of a, which is given as:

$$pr(x \wedge y \wedge \mathcal{P}_a) \; is \; \begin{cases} pr(a) & \text{if } (x, y \in \mathcal{P}_a) \\ pr(a)\, pr(x) \prod\limits_{p \in \mathcal{P}_a} I_p(p,x) & \text{if } (x \notin \mathcal{P}_a \;\wedge\; y \in \mathcal{P}_a) \\ pr(a)\, pr(y) \prod\limits_{p \in \mathcal{P}_a} I_p(p,y) & \text{if } (x \in \mathcal{P}_a \;\wedge\; y \notin \mathcal{P}_a) \\ pr(a)\, pr(x)\, pr(y) \prod\limits_{p \in \mathcal{P}_a} I_p(p,x)\, I_p(p,y) & \text{if } (x, y \notin \mathcal{P}_a) \end{cases}$$

In other words, if propositions x and y belong to the action's preconditions, the term reduces to the probability of the action. If x does not belong to the action's preconditions, but y does, it is necessary to compute the Interaction between x and the action's preconditions as in Equation 7.5. If y does not belong to the action's preconditions, but x does, it is necessary to compute the Interaction between y and the action's preconditions as in Equation 7.5. If neither x nor

y belong to the action's preconditions, it is necessary to compute the Interaction between x and y and the action's preconditions as in Equation 7.5.

Returning to the current example, consider the calculation of the Interaction between (at b trk) and ¬(flattire) at S$_2$ after performing (drive trk a d). In this case, the Interaction is:

$$\begin{aligned}
I_p(\text{(at b trk)}', \neg\text{flattire}') &= pr(\text{(at b trk)}' \wedge \neg\text{flattire}' | \text{drive trk a b})\, pr(\text{drive trk a b}) + \\
&\quad pr(\text{(at b trk)}' \wedge \neg\text{flattire}' | \neg\text{drive trk a b})\, pr(\neg\text{drive trk a b}) \\
&= pr(\text{(at b trk)}' \wedge \neg\text{flattire}' | \text{drive trk a b})\, pr(\text{drive trk a b}) + \\
&\quad pr(\text{at b trk} \wedge \neg\text{flattire}) - pr(\text{at b trk} \wedge \neg\text{flattire} \wedge \text{at a trk})
\end{aligned}$$

where $pr(\text{at b trk} \wedge \neg\text{flattire})$ and $pr(\text{at b trk} \wedge \neg\text{flattire} \wedge \text{at a trk})$ are equal to 0 since (at b trk) is not present in the previous state S$_0$. Therefore, the Interaction between (at b trk) and ¬(flattire) reduces to the conjunction probability of (at b trk) and ¬(flattire) given (drive trk a d) times the probability of (drive trk a d). That is:

$$\begin{aligned}
pr(\text{(at b trk)}' \wedge \neg\text{flattire}' | \text{drive trk a b}) &= pr(\text{(at b trk)}' \wedge \neg\text{flattire}' | \, o_1, \text{drive trk a b}) + \\
&\quad pr(\text{(at b trk)}' \wedge \neg\text{flattire}' | \, o_2, \text{drive trk a b}) \\
&= 0.6(1) + 0.4(0) = 0.6
\end{aligned}$$

where o_2 produces (at b trk) and (flattire). Therefore, o_2 is not considered – the conditional probability is 0. For o_1, it produces (at b trk), but it does not produces either ¬(flattire) or (flattire) so it is necessary to compute how ¬(flattire) interferes with (drive trk a b) preconditions at the previous state. In this example, ¬(flattire) belongs to (drive trk a b) preconditions. Therefore, as before, the probability is 1.

As a result:

$$\begin{aligned}
I_p(\text{(at b trk)}', \neg\text{flattire}') &= pr(\text{(at b trk)}' \wedge \neg\text{flattire}' | \text{drive trk a b})\, pr(\text{drive trk a b}) \\
&= 0.6\,(1)\, = 0.6
\end{aligned}$$

An Interaction value of 0.6 means that there is some kind of interference between propositions. This interference comes from the fact that one of the action's outcomes, o_2, produces (flattire).

7.4 Probability Estimates without Determinization (PEWD)

In the previous section, we described the concept of probabilistic state and how to compute them. In searching for a plan, we also need a heuristic estimate to help the planner decide what state to expand next. In order to do this, we need an estimate of how likely the state is of leading to the goals. In this section, we describe an approach to computing more accurate estimates of probability that allows the planner to search towards non-branching seed plans with high probability of success. We first describe how we do this propagation in the probability computation considering the overall probability of each proposition across all of the action's outcomes and the dependencies between those propositions in the different outcomes. Then, we propose a heuristic function that makes use of this probability propagation to guide a planner towards high probability of success plans.

7.4.1 Computing probability and Interaction

As we did in Chapter 5, probability and Interaction information can be estimated using a plan graph. Just as with the propagation of cost Interaction, the computation of probability and Interaction information begins at level zero of the plan graph and assumes that (1) the probability of each proposition at this level is 1, and (2) the Interaction between each pair of propositions at this level is 1. Again, neither of these assumptions are essential, but they are adopted in this work for simplicity.

Computing action probability and Interaction

The probability and Interaction information of a proposition's layer at a given level of the plan graph is used to compute the probability and the Interaction information for the subsequent actions' layer. In particular, considering an action a at level l with a set of preconditions $\mathcal{P}_a$, the estimation of how likely it is to execute an action is the product of achieving all its preconditions times the Interaction between all pairs of propositions:

$$pr(a) \approx \prod_{x \in \mathcal{P}_a} pr(x) \prod_{\substack{(x_i, x_j) \in \mathcal{P}_a \\ j > i}} I_p(x_i, x_j) \;\leq\; \max\; pr(x) \tag{7.5}$$

The Interaction between two actions a and b at level l, with sets of preconditions $\mathcal{P}_a$ and $\mathcal{P}_b$ is defined as:

$$I_p(a,\, b)\;\; is \begin{cases} 0 & \text{if } a \text{ and } b \text{ are mutex by inconsistent effects or interference} \\ \frac{pr(a \wedge b)}{pr(a)\, pr(b)} & \text{otherwise} \end{cases} \tag{7.6}$$

If the actions are mutex by inconsistent effects or interference, then the Interaction is zero. Otherwise, $pr(a \wedge b)$ is $pr(\mathcal{P}_a \cup \mathcal{P}_b)$, i.e., the probability of the union of their preconditions. This is approximated as in Equation 7.5 by:

$$pr(\mathcal{P}_a \cup \mathcal{P}_b) \approx \prod_{x \in \mathcal{P}_a \cup \mathcal{P}_b} pr(x) \prod_{\substack{(x_i, x_j) \in \mathcal{P}_a \cup \mathcal{P}_b \\ j > i}} I_p(x_i, x_j)$$

where the probability of performing two actions a and b will be the product of the probability of achieving all their preconditions times the Interaction between all pairs of preconditions.

The Interaction above can be simplified. To illustrate that, consider a simple problem with operator A that has preconditions x, y, and t, and operator B that has preconditions x, y, and z. Assuming that A and B are not mutually exclusive, the Interaction between actions A and B will be:

$$I_p(A, B) = \frac{pr(\mathcal{P}_A \cup \mathcal{P}_B)}{pr(A)\, pr(B)} \tag{7.7}$$

where:

$$\begin{aligned}
pr(\mathcal{P}_A \cup \mathcal{P}_B) &= pr(t)\, pr(x)\, pr(y)\, pr(z)\, I_p(t,x)\, I_p(t,y)\, I_p(t,z)\, I_p(x,y)\, I_p(x,z)\, I_p(y,z) \\
pr(A) &= pr(t)\, pr(x)\, pr(y)\, I_p(t,x)\, I_p(t,y)\, I_p(x,y) \\
pr(B) &= pr(x)\, pr(y)\, pr(z)\, I_p(x,y)\, I_p(x,z)\, I_p(y,z)
\end{aligned}$$

Therefore, the Interaction between A and B can be rewritten as:

$$\begin{aligned}
I_p(A, B) &= \frac{\cancel{pr(t)}\cancel{pr(x)}\cancel{pr(y)}\cancel{pr(z)}\cancel{I_p(t,x)}\cancel{I_p(t,y)}I_p(t,z)\cancel{I_p(x,y)}\cancel{I_p(x,z)}\cancel{I_p(y,z)}}{\cancel{pr(t)}\cancel{pr(x)}\cancel{pr(y)}\cancel{I_p(t,x)}\cancel{I_p(t,y)}\cancel{I_p(x,y)}\cancel{pr(z)}pr(x)\, pr(y)\cancel{I_p(z,x)}\cancel{I_p(z,y)}I_p(x,y)} \\
&= \frac{I_p(t,z)}{pr(x)\, pr(y)\, I_p(x,y)}
\end{aligned}$$

In general:

$$I_p(a, b) \simeq \frac{\displaystyle\prod_{\substack{x_i \in \mathcal{P}_a - \mathcal{P}_b \\ x_j \in \mathcal{P}_b - \mathcal{P}_a}} I_p(x_i, x_j)}{\displaystyle\prod_{x \in \mathcal{P}_a \cap \mathcal{P}_b} pr(x) \prod_{\substack{(x_i, x_j) \in \mathcal{P}_a \cap \mathcal{P}_b \\ j > i}} I_p(x_i, x_j)} \tag{7.8}$$

where the numerator is the Interaction between unique preconditions for each action, and the denominator is the probability of common preconditions and the Interaction between them.

Computing proposition probability

The next step consists of estimating the probability of the propositions at the next level. In this calculation, all the possible actions at the previous level that achieve each proposition need to be taken into account. Just as before, we are choosing the action that maximizes the probability, but we are considering the action as a whole. The probability of a proposition is the maximum probability among all the actions that produce the proposition, but this is where the calculation differs from before. We must consider all outcomes of the action that contribute to the proposition. More formally, for a proposition x at level l, achieved by actions $\mathcal{A}_x$ at the preceding level, the probability is calculated as:

$$pr(x) = \max_{a \in \mathcal{A}(x)} \left\{ pr(a) \sum_{o \in \mathcal{OA}(a,x)} pr(o)\, pr(x \mid o, a) \right\} \tag{7.9}$$

where $\mathcal{OA}(a, x)$ is the set of outcomes of action a that produce x. Therefore, the second term in the equation gives information about the total probability of x given the action a. This information is given by the conditional probability of x given o, which is defined as:

$$pr(x \mid o, a) \;\; \text{is} \;\; \begin{cases} 1 & \text{if } (x \in o) \\ 0 & \text{if } (\neg x \in o) \\ pr(x \mid \mathcal{P}_a) & \text{if } (x, \neg x \notin o) \end{cases} \tag{7.10}$$

where $\mathcal{P}_a$ is the set of preconditions of a. If the outcome o produces the proposition x, then the conditional probability is 1 (the outcome is considered). If o produces $\neg x$ (deletes x), then the conditional probability is 0 (the outcome is not considered). Finally, if o does not produce either x or $\neg x$, then we need to compute the probability that x persists through the action. This requires considering the relationship between x and the action's preconditions at the previous level. If x belongs to the action's preconditions, then the conditional probability is 1 (x is necessary for the action and the outcome is considered). If $\neg x$ belongs to the action's preconditions, the conditional probability is 0 (x is inconsistent with the action so the outcome is not considered). If x or $\neg x$ do not belong to the action's preconditions, then it is necessary to consider whether the proposition was there in the previous layer given the preconditions of the action. Formally:

$$pr(x \mid \mathcal{P}_a) \ is \begin{cases} 1 & \text{if } (x \in \mathcal{P}_a) \\ 0 & \text{if } (\neg x \in \mathcal{P}_a) \\ pr(x) \prod\limits_{p \in \mathcal{P}_a} I_p(x,p) & \text{if } (x, \neg x \notin \mathcal{P}_a) \end{cases} \tag{7.11}$$

Figure 7.5 shows a partial plan graph for the problem in Figure 7.3. The numbers above the propositions and actions are the probabilities associated with each one computed during the probability propagation process (those highlighted are the probabilities that we will compute).

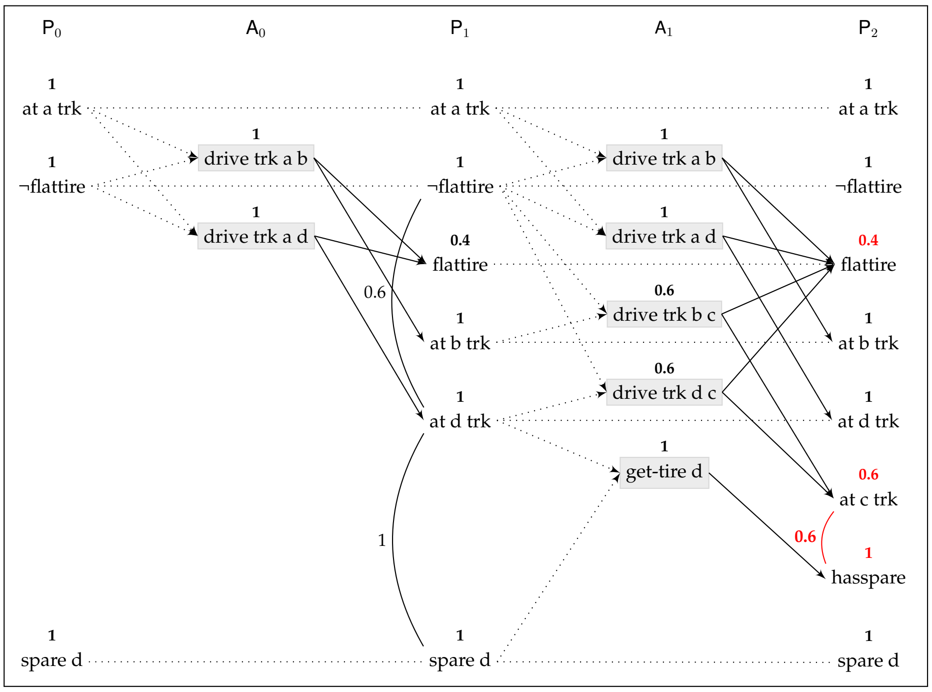


Figure 7.5: A partial plan graph with probability values of propositions and actions.

The probability of proposition (at c trk) at level 2 is (o_1 is the outcome of action *drive* that achieves the location without a flat tire, and o_2 is the outcome that gets the flat tire):

$$\begin{aligned} pr(\text{at c trk}) &= \max\{pr(\text{drive d c}), pr(\text{drive b c})\} \\ &= \max\left\{ \begin{array}{l} pr(\text{drive b c})\,[\,pr(o_1)\,pr(\text{at c trk}, o_1) + pr(o_2)\,pr(\text{at c trk}, o_2)\,], \\ pr(\text{drive d c})\,[\,pr(o_1)\,pr(\text{at c trk}, o_1) + pr(o_2)\,pr(\text{at c trk}, o_2)\,] \end{array} \right\} \\ &= \max\{0.6\,[\,0.6(1) + 0.4(1)\,], 0.6\,[\,0.6(1) + 0.4(1)\,]\} = 0.6 \end{aligned}$$

where (drive b c) and (drive d c) actions produce (at c trk), and both of their outcomes o_1 and o_2 reach it.

The probability of proposition (flattire) at level 2 is:

$$\begin{aligned}
pr(\text{flattire}) &= \max\,\{pr(\text{drive a b}), pr(\text{drive a d})\, pr(\text{drive b c})\, pr(\text{drive d c})\,\} \\
&= \max \left\{ \begin{array}{l} pr(\text{drive a b})\,[\,pr(o_1)pr(\text{flattire}, o_1) + pr(o_2)pr(\text{flattire}, o_2)\,], \\ pr(\text{drive a d})\,[\,pr(o_1)pr(\text{flattire}, o_1) + pr(o_2)pr(\text{flattire}, o_2)\,], \\ pr(\text{drive b c})\,[\,pr(o_1)pr(\text{flattire}, o_1) + pr(o_2)pr(\text{flattire}, o_2)\,], \\ pr(\text{drive d c})\,[\,pr(o_1)pr(\text{flattire}, o_1) + pr(o_2)pr(\text{flattire}, o_2)\,] \end{array} \right\} \\
&= \max \left\{ \begin{array}{ll} 1\,[\,0.6 + 0.4(1)\,], & 1\,[\,0.6 + 0.4(1)\,], \\ 0.6\,[\,0.6 + 0.4(1)\,], & 0.6\,[\,0.6 + 0.4(1)\,] \end{array} \right\} = 0.4
\end{aligned}$$

where actions (drive a b), (drive a d), (drive b c), and (drive d c) produce (flattire), and their outcome o_2 reaches it. For the case of outcome o_1, neither (flattire) nor its counterpart ¬(flattire) belongs to o_1. Therefore, it is necessary to determine the probability that (flattire) will persist, which requires considering the Interaction between (flattire) and each action precondition. For all the actions, ¬(flattire) belongs to the action's preconditions so the outcome is not considered.

Computing proposition Interaction

Finally, we compute the Interaction between propositions. In order to calculate the Interaction between two propositions x and y at a level l, we need to consider all the possible ways to achieve both propositions. In other words, all the actions that achieve the pair of propositions and the Interaction between them. Suppose that $\mathcal{A}_x$ and $\mathcal{A}_y$ are the sets of actions that achieve propositions x and y respectively at level l. The Interaction between x and y is then:

$$I_p(x, y) \approx \frac{\max \left\{ \begin{array}{l} \displaystyle\max_{\substack{a \in \mathcal{A}_x \cap \mathcal{A}_y \\ a \notin \text{noop}}} pr(a)\, pr(x \wedge y | a), \\[2em] \displaystyle\max_{\substack{a \in \mathcal{A}_x,\, b \in \mathcal{A}_y \\ a \notin \text{noop},\, b \notin \text{noop} \\ a \neq b}} pr(a \wedge b)\, pr(x \wedge y | a \wedge b), \\[2em] pr(x)\, pr(y)\, I_p(x, y) \end{array} \right\}}{pr(x)\, pr(y)} \qquad (7.12)$$

The first term in the $\max$ expression corresponds to those actions that accomplish both propositions x and y. It is computed as:

$$\max_{\substack{a\in\mathcal{A}_x\cap\mathcal{A}_y\\ a\notin \text{noop}}} \{pr(a)\,pr(x\wedge y|a)\} = \max_{\substack{a\in\mathcal{A}_x\cap\mathcal{A}_y\\ a\notin \text{noop}}} \left\{pr(a)\sum_{o\in\mathcal{O}_a} pr(o)\,pr(x\wedge y|o,a)\right\}$$

where $\mathcal{O}_a$ is the set of outcomes of action a. The conditional probability of x and y given an outcome o is given as:

$$pr(x\wedge y\,|\,o,a)\ is\ \begin{cases} 1 & \text{if } (x,y\in o)\\ 0 & \text{if } (\neg x\in o\ \vee\ \neg y\in o)\\ pr(x\,|\,\mathcal{P}_a) & \text{if } (y\in o\ \wedge\ x,\neg x\notin o)\\ pr(y\,|\,\mathcal{P}_a) & \text{if } (x\in o\ \wedge\ y,\neg y\notin o)\\ pr(x\wedge y\,|\,\mathcal{P}_a) & \text{if } (x,\neg x,y,\neg y\notin o)\end{cases}$$

In other words, if o produces x and y, the probability is 1 (the outcome is considered). If o produces $\neg x$ or $\neg y$, the probability is 0 (the outcome is not considered). If o produces x, but does not produces y or $\neg y$, then the probability is the probability that y persists through a, which depends on the probability of y given the preconditions of a. Likewise, if o produces y, but does not produce x or $\neg x$, then the probability is the probability that x persists through a, which depends on the probability of x given the preconditions of a. We compute these two using Equation 7.11. Finally, when o does not produce x, $\neg x$, y, or $\neg y$, then the probability is the probability that x and y both persist through a, which depends on the probability of x and y given the preconditions of a – in other words, it is the probability of x times the probability of y times the Interaction of x with the preconditions of a and b times the Interaction of y with the preconditions of a and b.

Similarly, the second term in the $\max$ expression corresponds to those actions that accomplish only one proposition each. It is given as:

$$\max_{\substack{a\in\mathcal{A}_x,\,b\in\mathcal{A}_y\\ a\notin\text{noop},\,b\notin\text{noop}}} \{pr(a\wedge b)pr(x\wedge y|a\wedge b)\}$$

which is equal to:

$$\max_{\substack{a\in\mathcal{A}_x,\,b\in\mathcal{A}_y\\ a\notin\text{noop},\,b\notin\text{noop}}} \left\{pr(a\wedge b)\sum_{o_i\in\mathcal{O}_a} pr(o_i)\,pr(x|o_i,a,b)\sum_{o_j\in\mathcal{O}_b} pr(o_j)\,pr(y|o_j,a,b)\right\}$$

where $\mathcal{O}_a$ is the set of outcomes of action a, and $\mathcal{O}_b$ is the set of outcomes of action b. The conditional probabilities $pr(x|o_i, a, b)$ and $pr(y|o_j, a, b)$ are given as:

$$pr(x|o,a,b)\ is \begin{cases} 1 & \text{if } (x \in o) \\ 0 & \text{if } (\neg x \in o) \\ pr(x \,|\, \mathcal{P}_a \wedge \mathcal{P}_b) & \text{if } (x, \neg x \notin o) \end{cases}$$

In other words, if o produces x, the probability is 1 (the outcome is considered). If o produces $\neg x$, the probability is 0 (the outcome is not considered). If o does not produces x and $\neg x$, then the probability is the probability that x persists through a and b, which depends on the probability of x before a given the preconditions of both a and b, which is:

$$pr(x \,|\, \mathcal{P}_a \wedge \mathcal{P}_b)\ is \begin{cases} 1 & \text{if } (x \in \mathcal{P}_a \cup \mathcal{P}_b) \\ 0 & \text{if } (\neg x \in \mathcal{P}_a \cup \mathcal{P}_b) \\ pr(x) \prod\limits_{p \in \mathcal{P}_a \cup \mathcal{P}_b} I_p(x,p) & \text{if } (x, \neg x \notin \mathcal{P}_a \cup \mathcal{P}_b) \end{cases}$$

In other words, if the proposition x belongs to the union of the actions' preconditions, then the conditional probability is 1 (the outcome is considered). If $\neg x$ belongs to the actions' preconditions, the conditional probability is 0 (the outcome is not considered). If x and $\neg x$ do not belong to the actions' preconditions, then the probability that x holds given the preconditions of a and b is the probability of x times the Interaction of x with the preconditions of a and b.

Finally, the third term in the $\max$ expression corresponds to the case where both propositions persist through the noop action. This is given as the product of the probability of each individual proposition at the previous level and the Interaction between them.

Returning to the current example, consider the calculation of the Interaction between (at c trk) and (¬flattire) at level 2 is:

$$
\begin{aligned}
I_p(\text{at c trk, }\neg\text{flattire}) &\approx \frac{\max\left\{\begin{array}{l} pr(\text{drive b c})\, pr(\text{at c trk, }\neg\text{flattire}|\text{drive b c}), \\ pr(\text{drive d c})\, pr(\text{at c trk, }\neg\text{flattire}|\text{drive d c}) \end{array}\right\}}{pr(\text{at c trk})\, pr(\neg\text{flattire})} \\
&\approx \frac{\max\left\{\begin{array}{l} pr(\text{drive b c}) \sum_{o\in(\text{drive b c})} pr(o)\, pr(\text{at c trk, }\neg\text{flattire}), \\ pr(\text{drive d c}) \sum_{o\in(\text{drive d c})} pr(o)\, pr(\text{at c trk, }\neg\text{flattire}) \end{array}\right\}}{pr(\text{at c trk})\, pr(\neg\text{flattire})} \\
&\approx \frac{\max\left\{\begin{array}{l} pr(\text{drive b c}) \left\{\begin{array}{l} pr(o_1)\, pr(\text{at c trk, }\neg\text{flattire}|o_1)+ \\ pr(o_2)\, pr(\text{at c trk, }\neg\text{flattire}|o_2) \end{array}\right\}, \\ \\ pr(\text{drive d c}) \left\{\begin{array}{l} pr(o_1)\, pr(\text{at c trk, }\neg\text{flattire}|o_1)+ \\ pr(o_2)\, pr(\text{at c trk, }\neg\text{flattire}|o_2) \end{array}\right\} \end{array}\right\}}{pr(\text{at c trk})\, pr(\text{flattire})}
\end{aligned}
$$

Actions (drive b c) and (drive d c) produce (at c trk) in both outcomes o_1 and o_2, but it does not produce (¬flattire). Therefore, for outcome o_1 it is necessary to determine the probability of (¬flattire), which requires considering the Interaction between (¬flattire) and each action precondition. For all the actions, ¬(flattire) belongs to the action's preconditions so the probability reduces to the probability of the action. Outcome o_2 it is not considered because it produces the proposition (flattire), the opposite of (¬flattire). The Interaction is then:

$$
\begin{aligned}
I_p(\text{at c trk, }\neg\text{flattire}) &\approx \frac{\max\left\{\begin{array}{l} pr(\text{drive b c})\, pr(o_1)\, pr(\text{at c trk, }\neg\text{flattire}|o_1), \\ pr(\text{drive d c})\, pr(o_1)\, pr(\text{at c trk, }\neg\text{flattire}|o_1) \end{array}\right\}}{pr(\text{at c trk})\, pr(\text{flattire})} \\
&\approx \frac{\max\{0.6(0.4)(0.6),\, 0.6(0.4)(0.6)\}}{0.6(0.4)} = \frac{0.6(0.4)(0.6)}{0.6(0.4)} = 0.6
\end{aligned}
$$

The fact that I_p(at c pkg, ¬flattire) $= 0.6$ means that there is interference between having the package at location c and not having a flat tire, which comes from the fact that action (drive b c) has (flattire) as effect.

7.4.2 Upper bounds on probability and Interaction

Because the probabilities in Equations 7.5 and 7.9 are estimated based on binary Interaction, the resulting calculations can sometimes overestimate probability and Interaction. Therefore, we can improve estimates by considering upper bounds on *action probability*, *action Interaction*, and *propositions Interaction*. The reason for these upper bounds are shown in next subsections.

Upper bounds on action probability

The probability of an operator a should be at most the minimum probability among all its preconditions, if they are completely synergistic. That is:

$$pr(a) \leq \min_{x \in \mathcal{P}_a} pr(x) \tag{7.13}$$

To illustrate this, consider a simple problem with the following two operators with probability 1:

$$\begin{aligned} A &: p \rightarrow x,\, y,\, z \\ B &: x,\, y,\, z \rightarrow t \end{aligned} \tag{7.14}$$

Figure 7.6 shows a partial planning graph for the operators described in Equation 7.14. The number above the propositions and actions refers to the estimated probability computed for each proposition and action during the probability propagation process. The number next to red edges refers to the Interaction value between the pair of propositions that each edge connects. The example in the figure starts at level $i-1$ with propositions layer P_{i-1}, which contains a single proposition p with $pr = 0.5$. Therefore, the estimated probability of action A is:

$$pr(A) = pr(p) = 0.5$$

The next step is to compute the estimated probabilities of propositions at level P_i, which have the following values:

$$\begin{aligned} pr(x) &= pr(A)\, pr(x|A) &= 0.5(1) &= 0.5 \\ pr(y) &= pr(A)\, pr(y|A) &= 0.5(1) &= 0.5 \\ pr(z) &= pr(A)\, pr(z|A) &= 0.5(1) &= 0.5 \end{aligned}$$

The computation of the estimated probability of a proposition is followed by the Interaction computation between pairs of propositions. The Interaction val-

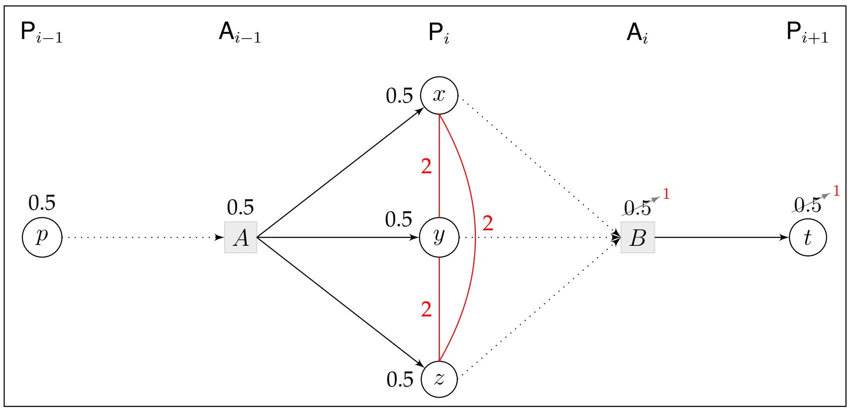


Figure 7.6: Example of upper bounds on action probability during the propagation of probability and Interaction information in a plan graph.

ues are:

$$I_p(x,y) = \frac{pr(A)\,pr(x \wedge y|A)}{pr(x)\,pr(y)} = \frac{0.5(1)}{0.5(0.5)} = 2$$

$$I_p(x,z) = \frac{pr(A)\,pr(x \wedge z|A)}{pr(x)\,pr(z)} = \frac{0.5(1)}{0.5(0.5)} = 2$$

$$I_p(z,y) = \frac{pr(A)\,pr(y \wedge z|A)}{pr(y)\,pr(z)} = \frac{0.5(1)}{0.5(0.5)} = 2$$

The next step is to compute the probability of operator B at layer A_i. B has x, y, and z as preconditions. The estimated probability of action B is the product of the estimated probability of each of its preconditions combined with the Interaction between them, which is:

$$pr(B) = pr(x)\,pr(y)\,pr(z)\,I_p(x,y)\,I_p(x,z)\,I_p(y,z) = 0.5(0.5)(0.5)(2)(2)(2) = 1$$

In this case, the binary Interaction information is overestimating the probability of B, which should be less or equal to the minimum probability among preconditions of B (if they are synergistic among all of them). The true probability of B should consider the Interaction among x, y, and z, which is:

$$I_p^*(x,y,z) = \frac{pr^*(A)\,pr^*(x \wedge y \wedge z|A)}{pr^*(x)\,pr^*(y)\,pr^*(z)} = \frac{0.5(1)}{0.5(0.5)(0.5)} = 4$$

Therefore, the true probability of B is:

$$pr^*(B) = pr^*(x)\,pr^*(y)\,pr^*(z)\,I_p^*(x,y,z) = 0.5(0.5)(0.5)(4) = 0.5$$

In this case, by enforcing the upper bounds from Equation 7.13 we get that the probability of B is at most 0.5, which is a more accurate estimate.

Upper bounds on action and proposition Interaction

The approximated binary Interaction between two elements should always be bounded by $1/\max\{pr(x),\, pr(y)\}$. That is:

$$I_p^*(x,y) \;=\; \frac{pr^*(x \wedge y)}{pr^*(x)\, pr^*(y)} \;\leq\; \frac{\min\{\, pr^*(x), pr^*(y)\}}{pr^*(x)\, pr^*(y)}$$

$$\text{If } \min\{pr(x), pr(y)\} = pr(x), \;\text{ then } I_p(x,y) = \frac{\cancel{pr(x)}}{\cancel{pr(x)}pr(y)} = \frac{1}{pr(y)}$$

$$\text{If } \min\{pr(x), pr(y)\} = pr(y), \;\text{ then } I_p(x,y) = \frac{\cancel{pr(y)}}{pr(x)\cancel{pr(y)}} = \frac{1}{pr(x)}$$

Therefore:

$$I_p(x,y) \;\leq\; \frac{1}{\max\{pr(x),\, pr(y)\}} \tag{7.15}$$

Considering these bounds during the Interaction computation results in more accurate probability estimates. To illustrate this, consider a simple problem with the following two operators with probability 1:

$$\begin{aligned} A &: \; x \rightarrow p \\ B &: \; y,\, z \rightarrow q \end{aligned} \tag{7.16}$$

Figure 7.7 shows a partial planning graph for the operators described in Equation 7.16. The number above the propositions and actions refers to the estimated probability computed for each proposition and action during the probability propagation process. The number next to the edges refers to the Interaction value between the pair of propositions that each edge connects. The example in the figure starts at level i with propositions layer P_i, which contains propositions x, y, and z, with probabilities 0.56, 0.75, and 0.56 respectively, and $I(x,y) = 1.33$, $I(x,z) = 1.78$, and $I(y,z) = 1$. The estimated probability of actions A and B is:

$$\begin{aligned} pr(A) &= pr(x) \;=\; 0.56 \\ pr(B) &= pr(y)\, pr(z)\, I(y,z) \;=\; 0.75\,(0.56)\,(1) \;=\; 0.42 \end{aligned}$$

The next step is to compute the estimated Interaction of actions at level A_i. Using Equation 7.8, we compute the Interaction between actions A and B, which is:

$$I(A,B) \;=\; I(x,y)\, I(x,z) \;=\; 1.33\,(1.78) \;=\; 2.36$$

In this case, according to the bounds in Equation 7.15 we know that the Interaction between A and B should not be higher than:

$$\frac{1}{\max\{pr(A)\, pr(B)\}} = \frac{1}{\max\{0.56,\, 0.42\}} = \frac{1}{0.56} = 1.78$$

Therefore, considering these bounds during the Interaction computation results in more accurate probability estimates.

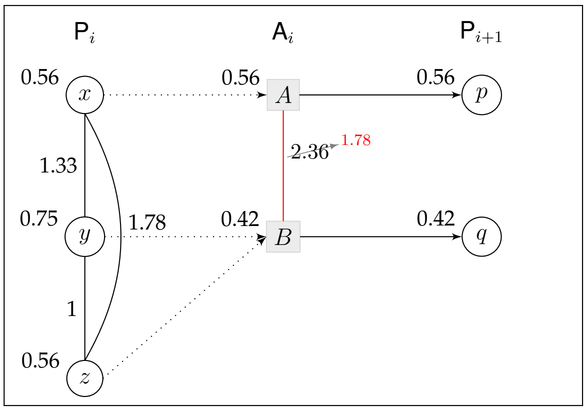


Figure 7.7: Example of upper bounds on action Interaction during the propagation of probability and Interaction information in a plan graph.

7.4.3 Probabilistic heuristic estimator

Using Equations 7.5, 7.6, 7.9, and 7.12 we can build a plan graph and propagate probability and Interaction information. The construction process finishes when two consecutive propositions layers are identical and there is quiescence in probability and Interaction for all propositions and actions in the plan graph. On completion, each possible goal proposition has an estimated probability of being achieved, and there is an Interaction estimation between each pair of goal propositions. Therefore, using the probability and Interaction information computed in the probabilistic plan graph we can compute an estimated probability of achieving a (possibly conjunctive) goal $G = \{\, g_1, ..., g_n\}$ from a particular state n, which we call the Completion Probability Estimate (CPE), as:

$$\text{CPE}(n) \approx \prod_{g \in G} pr(g) \prod_{\substack{(g_i, g_j) \in G \\ j > i}} I_p(g_i, g_j) \;\leq\; \min_{g \in G} pr(g) \tag{7.17}$$

Figure 7.8 shows the high-level algorithm for computing the CPE used to compute the probability estimation of reaching the goal from a particular state, which may be summarized in the following steps:

1. For each proposition p in the probabilistic state S compute the probability of p in S using Equation 7.3.
2. For each each pair of propositions p and q in the probabilistic state S compute the Interaction between p and q in S using Equation 7.4.
3. Initialize the probabilistic plan graph with the probability and Interaction information of the current state and compute the new probability and Interaction estimates using Equations 7.5, 7.6, 7.9, and 7.12 .
4. Compute the CPE of the current state S by estimating the probability of G from the probability and Interaction estimates in the updated probabilistic plan graph using Equation 7.17.

Function PROBABILITYESTIMATE (s)

s	$\equiv$	the current probabilistic state
p	$\equiv$	a proposition $p \in s$
q	$\equiv$	a proposition $q \in s$
G	$\equiv$	the set of goals
g	$\equiv$	a goal proposition
CPE	$\equiv$	the completion probability estimate

```
1. for each $p \in s$
     $pr_s(p) \leftarrow$ COMPUTEPROBABILITY($p$)
2. for each $(p, q) \in s$
     $I_{p_s}(p, q) \leftarrow$ COMPUTEINTERACTION($p$, $q$)
3. UPDATEPRPLANGRAPH($s$)
4. CPE($s$) $\leftarrow \prod_{g \in G} pr(g)$
5. return CPE
```

Figure 7.8: The CPE calculation pseudo-algorithm.

An extended example

Consider the progress of the probabilistic search process shown in Figure 7.9 that finds a path for the simple probabilistic Logistic problem in Figure 7.3. S_0 is the initial state. Actions (drive trk a b) and (drive trk a d) are the applicable actions in S_0, and generate the probabilistic states S_1 and S_2 respectively. The path to the goal through (drive trk a d) and state S_2 has a higher probability than the path through (drive trk a b) and state S_1 because of the fact that location d has a spare tire, while location b does not. The heuristic function for each state is:

$$\begin{aligned}\text{CPE}(S_1) &= 0.6\\ \text{CPE}(S_2) &= 1\end{aligned}$$

Therefore, the next node to be expanded is S_2 where (drive trk d c) and (get-tire d) are the applicable actions, and generate states S_3 and S_4 respectively.

The path to the goal through (get-tire d) has a higher probability than the path through (drive trk d c). The fact that $pr(\neg\text{flattire}) = 0.6$ at S_2 harms the probability of (drive trk d c). On the other hand, the spare tire at location d benefits the probability of (get-tire d). The heuristic function for each state is:

$$\begin{aligned}\text{CPE}(S_3) &= 0.6\\ \text{CPE}(S_4) &= 1\end{aligned}$$

Therefore, the next node to be expanded is S_4 where (drive d c) and (change) are the applicable actions, and generate states S_5 and S_6 respectively. Again, $pr(\neg\text{flattire}) = 0.6$ at S_4 harms the probability of (drive trk d c), while the spare tire at location d benefits the probability of (change). The heuristic function for each state is:

$$\begin{aligned}\text{CPE}(S_5) &= 0.6\\ \text{CPE}(S_6) &= 1\end{aligned}$$

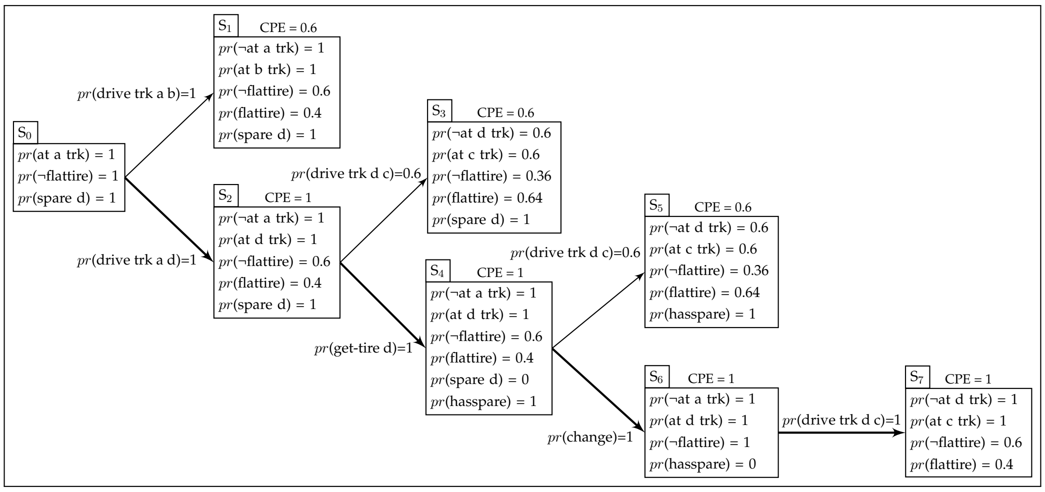


Figure 7.9: Search progress using Proposition Determinization solving a simple probabilistic Logistics problem.

Therefore, the next node to be expanded is S_6 where (drive d c) is the single applicable action, and generate the state S_7. It is important to note that $pr(\neg\text{flattire})$ in S_6 increases from 0.6 to 1 after applying (change). Therefore, $pr(\text{drive d c})$ also increases to 1. S_7 contains the goal state and it has the highest probability of success. As a consequence, it is not necessary to keep exploring the search space.

For this particular problem, the heuristic leads to a maximum search probability, and finds the following plan solution with the highest probability of success:

$$\pi = \{\text{(drive trk a d) (get-tire d) (change) (drive d c)}\}$$

7.5 Experimental evaluation

We have conducted an experimental evaluation on IPPC-06 (Bonet and Given, 2006) and IPPC-08 (Buffet and Bryce, 2008) fully-observable-probabilistic planning (FOP) domains, as well as on the *probabilistically interesting domains* introduced by Little and Thiebaux (2007). The test consists of running the planner and using the resulting plan in the MDP Simulator (Younes et al., 2005). The planner and the simulator communicate by exchanging messages. The simulator first sends the planner the initial state. Then, the Interaction between planner and simulation consists of the planner sending an action and the simulator sending the next state to the planner.

FF-Replan, FHH, FHH$^+$, FPG, RFF, and PIPSS$_r^I$ planners described in Chapter 5, and the C-PIPSS$_r^I$ planner described in Chapter 6 have been used for the experimental evaluation. We compare these with our approach, namely PIPSS^{IP}. Figure 7.10 shows the PIPSS^{IP} architecture and highlights the key features of the system: *Probabilistic Plan Graph Estimator*, *Probabilistic State Information Update*, and *Heuristic Computation* modules. Given a PPDDL problem, the system initially processes and loads all the information given in the domain, and encodes the strings as numbers to decrease the computation time in the *Analysis & Processing* module. Then, the system builds a probabilistic plan graph estimator as is described in Section 7.4. The system performs an A* search when it is looking for a solution. This search is not as usual since it is performed in a space of probabilistic states. For each probabilistic state, the plan graph is updated and the system estimates the probability of achieving the goals from that state or *Completion Probability Estimate* (CPE).

We use two variants of the PIPSS^{IP} planner:

- PIPSS^{IP}: a PIPSS planner that uses probabilistic states rather than action determinization, where the propagation of probability through the plan graph

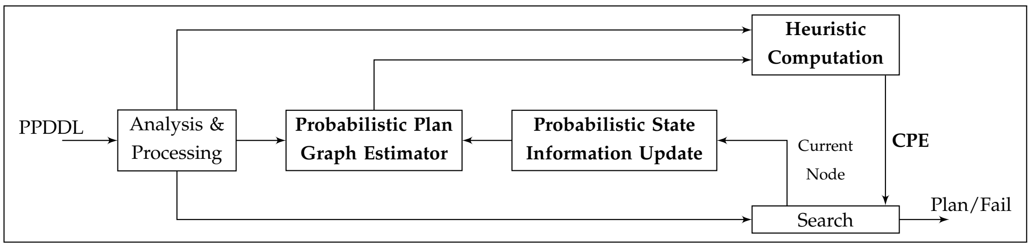


Figure 7.10: PIPSS^{IP} System Architecture.

considers Interaction estimates. During execution, the planner does not perform any further action when an unexpected state occurs.

- PIPSS$_r^{IP}$: a PIPSS planner that uses probabilistic states rather than action determinization, where the propagation of probability through the plan graph considers Interaction estimates. To deal with unexpected states at execution time, the planner does runtime replanning.

The experiments were conducted on a Pentium dual core processor at 2.4 GHz running Linux. For the rest of the planners, given that we were not able to obtain and run them ourselves, data are collected from work done by Yoon, Ruml, Benton, and Do (2010).

We have chosen those domains where simple replanning fails because some of the actions' outcomes yield dead-end states. Thus, we can evaluate if our novel PEWD approach is guiding the search towards high probability of success plans. Below is a brief description of the domains used for the experimental evaluation.

- Exploding-Blocksworld: a dead-end version of the Blocksworld domain described above where additionally the blocks can explode. The explosion may affect the table or other blocks.

- Triangle-tireworld: similar to the IPPC-06 Tire World Domain but with slight differences in the definition to permit short but dangerous paths.

- Tireworld: in this domain a car has to move between two locations. When the car drives a segment of the route, there is the chance of getting a flat tire. When this occurs the tire must be replaced. However, spare tires are not available in all locations.

- Climb: this domain consists of a person who is stuck on a roof because the ladder they used to climb up has fallen down. There are two options to get down: climbing down without the ladder but with a certain risk of injury or death, or calling for help from someone below to bring the ladder and then climb down with the ladder, which has no risk.

- River: in this domain a person on one side of the river needs to cross to the other side. There are three ways to achieve the goal with different chances of success.
- Tire1 & Tire10: domains based on the Tireworld domain, with different levels of difficulty to reach the final location without getting a flat tire.

Table 7.1 shows the number of successful rounds for FFH, FFH$^+$, FPG, PIPSS^I_r, C-PIPSS^I_r, PIPSS^{IP}, and PIPSS$^{IP}_r$ planners in each domain. For all the planners, 30 trials per problem were performed (as in the competition) with a total limit of 30 minutes for the 30 trials. Exploding-Blocksworld-06, Exploding-Blocksworld-08, and Tireworld domains have 15 problems for each domain. So, the maximum number of successful rounds for each domain is $15 \times 30 = 450$. However, Triangle-tireworld domain has 10 problems so that the total rounds in this case is $10 \times 30 = 300$. Climb, River, Tire1, and Tire10 domains have one problem for each domain, so the maximum number of successful rounds for each domain is 30.

Table 7.1: Total number of successful rounds using PEWD.

	PLANNERS						
DOMAINS	FFH	FFH$^+$	FPG	PIPSS^I_r	C-PIPSS^I_r	PIPSS^{IP}	PIPSS$^{IP}_r$
Exploding-Blocksworld-06	205	265	193	239	**266**	132	158
Tireworld-06	343	364	337	360	362	352	**365**
Climb	**30**	**30**	**30**	**30**	**30**	**30**	**30**
River	20	20	20	**23**	21	18	20
Tire1	**30**	**30**	**30**	21	18	**30**	**30**
Tire10	6	**30**	0	0	0	0	0
TOTAL	624	**739**	610	663	697	562	603
	FFH	FFH$^+$	RFF	PIPSS^I_r	C-PIPSS^I_r	PIPSS^{IP}	PIPSS$^{IP}_r$
Exploding-Blocksworld-08	131	**214**	58	171	170	85	103
Triangle-Tireworld-08	420	420	382	21	67	210	210
TOTAL	551	**634**	440	192	237	295	313

For the Exploding-Blocksworld-06 domain, C-PIPSS^I_r gets the highest rate of successful rounds closely followed by FFH$^+$, PIPSS^I_r, FFH, FPG, and finally PIPSS^{IP} and PIPSS$^{IP}_r$. There is the same trend for the Exploding-Blocksworld-08 domain, where FFH$^+$ stands out against the rest of the planners, followed by PIPSS^I_r, C-PIPSS^I_r, FFH, PIPSS^{IP}, and PIPSS$^{IP}_r$. The planner with the lowest rate of successful rounds is RFF, the competition winner. For the Tireworld domain, all the approaches get a similar number of successful rounds, where PIPSS^{IP} has the highest rate. For the Triangle-Tireworld domain, FFH^{+} and FFH get the highest rate followed by RFF. PIPSS^{IP} and PIPSS$_r^{IP}$ perform much better than PIPSS$_r^{I}$ and C-PIPSS$_r^{I}$. This is because PEWD is finding plans that avoid dead-end states, and thus manages to solve more rounds. Climb, River, Tire1, and Tire10 are problems with dead-ends and a small likelihood of simple paths. Results shows that all the approaches solve all the rounds for the Climb domain. For the River domain, PIPSS$_r^{I}$ achieves the highest rate of successful rounds. However, the other approaches are very close. For the Tire1 domain, all the approaches solve all the problems, except PIPSS$_r^{I}$ and C-PIPSS$_r^{I}$. This fact shows that PEWD is finding high probability of success plans. For the Tire10 domain, FFH and FHH^{+} are the only planners that solve the problem and are able to complete 6 and 30 rounds respectively.

With regard to the difference in performance between PIPSS^{IP} and PIPSS$_r^{IP}$, the successful rate is only slightly higher in PIPSS$_r^{IP}$ that performs replanning compare to PIPSS^{IP} that does not. This means that runtime replanning does not make a big difference because the technique is generating high probability of success plans.

It appears that PIPSS$_r^{I}$ and C-PIPSS$_r^{I}$ perform much better than PIPSS^{IP} and PIPSS$_r^{IP}$ in most of the domains. The issue here is that PIPSS$_r^{I}$ and C-PIPSS$_r^{I}$ scale much better that PIPSS^{IP} and PIPSS$_r^{IP}$ in term of the amount of time taken to solve the problem. PIPSS^{IP} and PIPSS$_r^{IP}$ were unable to solve all the problems for the hardest domains such as Blocksword and Tire because they run out of time due to the complexity caused by the update of the plan graph for each probabilistic state. In particular, for the Exploding-Blocksworld-06, PIPSS^{IP} solves only 40% of the problems, while PIPSS$_r^{I}$ solves 66% of them. For this reason, the number of successful rounds for both approaches is lower than for PIPSS$_r^{I}$ and C-PIPSS$_r^{I}$. For the Tireworld-06 domain, PIPSS^{IP} solves the 86% of the problems, while PIPSS$_r^{I}$ solves all of them. However, it still gets a high number of successful rounds, which is evidence that we are generating high probability of success plans. For the Triangle-Tireworld-08, PIPSS^{IP} and PIPSS$_r^{IP}$ only solve 46% of the problems, compared to PIPSS$_r^{I}$ and C-PIPSS$_r^{I}$ that solve 66% of them.

In order to confirm that this is a problem of efficiency and, therefore, the PEWD technique is generating high probability of success plans, we have given PIPSS^{IP} and PIPSS$_r^{IP}$ unlimited amount of time to solve problems for the Blocksword and Tireworld domains. Table 7.2 shows the results of this test. We compare the number of successful rounds for PIPSS^{IP} and PIPSS$_r^{IP}$ given 30 minutes against the number of successful rounds for their counterparts uPIPSS^{IP} and

uPIPSS$_r^{IP}$ respectively given unlimited amount of time.

Table 7.2: Total number of successful rounds using PEWD given unlimited amount of time.

	PLANNERS			
DOMAINS	PIPSS^{IP}	PIPSS$_r^{IP}$	uPIPSS^{IP}	uPIPSS$_r^{IP}$
Exploding-Blocksworld-06	132	158	180	180
Exploding-Blocksworld-08	85	103	156	161
Tireworld-06	352	365	391	423
Triangle-Tireworld-08	210	210	300	300
TOTAL	779	836	1027	1064

For the Exploding-Blocksworld-06 domain, uPIPSS^I and uPIPSS$_r^I$ are able to solve almost 60% of the problems versus 40% given 30 minutes. The remainder of the problems are not solved because uPIPSS^I and uPIPSS$_r^I$ run out of memory. Regardless of this, the number of successful rounds increases for both uPIPSS^I and uPIPSS$_r^I$. For the Exploding-Blocksworld-08 domain, uPIPSS^I and uPIPSS$_r^I$ are able to solve almost 66% of the problems versus 46% given 30 minutes. Again, the remainder of the problems are not solved because uPIPSS^I and uPIPSS$_r^I$ run out of memory, and the number of successful rounds increases for both uPIPSS^I and uPIPSS$_r^I$. For the Tireworld domain, uPIPSS^{IP} and uPIPSS$_r^{IP}$ solve all the problems. As a consequence, the number of successful rounds increases considerably. In particular, uPIPSS$_r^{IP}$ gets 423 of 450 successful rounds. For the Triangle-Tireworld domain, uPIPSS^{IP} and uPIPSS$_r^{IP}$ solve all the problems and gets the highest rate of successful rounds. All of this suggest that PEWD is finding plans that avoid dead-end states, and its performance could be dramatically improved by improving the efficiency of the PEWD technique.

7.6 Conclusions

In this chapter, we investigated a way to compute estimates of probability without action determinization for probabilistic planning. This technique uses the PPDDL domain definition as is and performs search in the space of probabilistic states. The probability information provided in the domain definition is used to propagate probability and Interaction information through a plan graph. This propagation technique considers the overall probability of each proposition across all of the action's outcomes and the dependencies between those propositions in the different outcomes. The resulting probabilities are then used to compute a heuristic function that guides the search toward high probability of success plans. The resulting plans are used in a system that handles unexpected outcomes by runtime replanning.

According to the results, the approach suffers from poor scalability for large domains. However, the results dealing with probabilistic planning problems have high success rates considering the number of solved problems. This is evidence that we are generating relatively high probability of success plans.

This appears to be a promising technique, but more effort is required to improve efficiency and memory usage of the probability plan graph computation in order to improve scalability.

Chapter 8

Conclusions and Future Work

This chapter summarizes the main conclusions of this dissertation and briefly considers some ways in which the work presented here may be extended.

8.1 Conclusions

Heuristic search is the dominant technique in solving automated planning problems. Improvements in the performance of heuristic search comes from two factors: improvements in search algorithms, and improvements in heuristic functions. This dissertation focused on the later, motivated by the need for a non-admissible domain-independent heuristic that quickly computes more accurate estimates of cost and more accurate estimates of probability.

We improved accuracy in cost estimation by developing a novel heuristic framework based on cost propagation in a plan graph. This approach, namely h^I, computes more accurate estimates because it makes use of Interaction information during cost propagation. Effectively, the use of Interaction information captures the degree of dependence between pairs of propositions and pairs of actions in a plan graph. It can be used across different automated planning areas, such as deterministic and probabilistic planning, and planning-based goal recognition. In particular, computing more accurate estimates of cost has a significant impact in solving goal recognition problems. A major weakness to this approach appears to be the high computational overhead in classical planning given by the plan graph update at each state. It seems to be possible that efforts to improve the efficiency of the plan graph computation would make this technique more competitive, particularly for problems where low-cost plans are important. However, for doing goal recognition, the accuracy of this heuristic pays off because it allows planning to be avoided altogether. We presented the ISS-CAD domain, a real time goal recognition problem concerned with maintenance tasks that astronauts must conduct for the Environmental Control and Life Support System aboard de International Space Station. We showed that our FGR approach is practical for this real time goal recognition problem. The reason for this is that the Interaction information helps to compute more accurate estimates of cost when the degree of interference and/or synergy among subgoals is high. Additionally, we outlined an approach to solve goal recognition problems with uncertain observations that combines a Bayesian network and a plan graph to find a solution.

We improved accuracy in probability estimation by developing PEWD, a novel probability estimation technique for probabilistic planning. This technique is based on propagation of probability and Interaction information in a plan graph without action determinization. In particular, this technique considers probabilistic actions as a whole, which results in the consideration of the overall probability of each proposition across all of the action's outcomes and the dependencies between those propositions in the different outcomes. As a result, this technique generates high probability of success seed plans. Again, the main weakness of this technique appears to be the high computational overhead when the plan graph is updated at each state during the planning search. Additional efforts to improve the time and space efficiency of this heuristic computation are needed to make this technique practical for larger domains.

8.2 Future work

The work described in this dissertation suggests a number of extensions and directions for future work. Here, we review some possibilities.

Improving Efficiency in Propagation of Probability in a Plan Graph

In Chapter 7, we introduced a heuristic function that computes more accurate estimates of probability. This heuristic is used to guide heuristic search for probabilistic planning. Results show evidence of the generation of high probability of success seed plans. However, the computational overhead of this technique is high. It would clearly be valuable to investigate techniques for improving both the space and time efficiency of the propagation. Some possible ways to do this are (1) minimizing the propagation by propagating only significant changes, (2) performing a lazy evaluation, and (3) using more sophisticated data structures.

Temporal Goal Recognition

In Chapter 4, we introduced our FGR goal recognition system that computes cost estimates using a plan graph, and uses this information to infer probability estimates for the possible goals. Simple goal recognition problems consist of a classical planning domain and problem, a set of possible goals, and a sequence of observed actions. An important avenue of further research is to investigate goal recognition problems that involve durative actions and concurrency. This is the case where the agent is doing a sequence of tasks that involve the use of temporal constraints.

Ramirez formulation of planning-based goal recognition solves two planning problems for each possible goal. If we applied this formulation to temporal goal recognition problems, it would require solving two temporal planning problems for each possible goal. This is likely to be computationally prohibitive since the best temporal planner is much slower than a classical planner. An approximate solution would be a better option to solve temporal planning problems. It seems to be possible to easily extend FGR to deal with temporal goal recognition. FGR is based on a plan graph, which is a structure that has been adapted to manage temporal planning (Smith and Weld, 1999; Garrido et al., 2002; Long and Fox, 2003). A temporal plan graph assumes the time in which actions happen. It is easy to adapt the FGR approach to use the time at which observations take place. This makes FGR a relatively easy approach to adapt to temporal goal recognition.

Uncertain Observations in Goal Recognition

In Chapter 4, we presented a theoretical approach to solve goal recognition problems with uncertain observations. In particular, we make use of Ramirez formulation and introduce two techniques to solve goal recognition problems with uncertain observations. The first technique computes optimal solutions. It uses an optimal planner to solve the problem. The second technique computes approximate solutions. It uses a hybrid solution that combines a Bayesian network and a plan graph to solve the problem. While we have done some preliminary evaluation of this approach with the optimal technique, the computation time is prohibitive (days to weeks for each solution). We could instead use our approximate FGR approach for solving these problems. This would allow us to further investigate the accuracy of our heuristic-based hybrid approach.

Oversubscription in Probabilistic Planning

In Chapter 2, we defined a classical planning problem as $\Pi = < P, O, I, G >$ where P is a set of predicates; O is a set of actions where each action has the form $< prec(O), add(O), del(O) >$; $I \subseteq P$ is a set of initial state predicates; and $G \subseteq P$ is a set of goal state predicates. In over-subscription planning, a problem has the same definition where goals $g \in G$ also have a utility value $V(g) \geqslant 0$ that represents how much the goal is worth to the user, and actions $a \in O$ have an associated cost $Cost(a) \geqslant 0$ that represents how costly it is to execute it. Oversubscription planning aims to achieve a subset of goals that maximize the trade-off between the maximum achievable benefit and the cost of the plan solution. Nigenda and Kambhapati (2005) developed a technique that makes use of cost propagation in a plan graph and mutex analysis for goal set selection. Do, Benton, van den Briel and Kambhapati (2007) developed a heuristic technique that uses a combination of cost propagation over a relaxed planning graph and Integer Programming to capture goal achievement cost and goal utility. However, none of these works consider probabilistic actions.

A possible avenue of further research is to investigate probabilistic oversubscription planning problems that involve PPDDL problems where action outcomes are probabilistic. In Chapters 3 and 7 we show that it is easy to propagate cost and probability in a plan graph. The challenge here is to estimate utility information in an uncertain environment since cost and probability are not independent of each other. The lowest cost approach might have low probability of success; the highest probability approach might have high cost.

Appendix A

Extended Experimental Results for Goal Recognition

A.1 Random Observations

Table A.1: Goal recognition with random observations in 5s

Domain		Blocks					Campus					Grid					Intrusion					Kitchen					Logistics				
Approach	$\%O$	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10
	T	1080.6	729.06	529.04	405.1	405.14	1.78	1.11	0.55	0.28	0.28	224.67	144.5	79.83	41.47	37.42	588.34	414.3	214.2	4.25	4.22	694.67	243.5	60.65	33.3	33.31	44.93	42.02	18.94	9.18	9.18
HSP^*_f	Q	1	1	0.86	0.86	0.86	1	1	1	1	1	1	0.2	0.66	0.86	0.73	1	1	1	1	1	1	1	0.86	0.86	0.86	1	0.93	0.66	0.8	0.8
	S	1.06	1.13	3.73	9.33	9.33	1	1	1	1	1	1	1.06	1.93	3.6	3.93	1	1	1.06	4.46	4.6	1	1	1.2	1.26	1.26	1.06	1.06	2.26	3.6	3.6
	T	531.26	446.26	379.81	357.94	357.94	1.22	0.8	0.5	0.32	0.32	114.12	79.76	52.92	39.20	38.8	447.41	281.11	151.37	3.58	3.55	480.39	171.08	49.62	37.93	37.92	36.26	32.46	14.8	7.04	7.04
$g\text{HSP}^*_f$	Q	1	1	1	1	1	1	1	1	1	1	1	0.2	0.8	0.93	0.8	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.06	1.13	4.06	11.46	11.46	1	1	1	1	1	1	1.2	2.26	3.86	4.2	1	1	1.06	4.46	4.6	1	1	1.33	1.4	1.4	1	1.13	2.26	3.6	3.6
	T	0.19	0.2	0.18	0.18	0.17	1.73	1.43	0.82	0.53	0.52	1.11	1.1	0.93	0.84	0.84	0.16	0.41	0.12	0.12	0.12	0.12	0.12	0.11	0.11	0.11	0.17	0.17	0.17	0.16	0.16
	Q	1	1	1	1	1	1	1	1	1	1	0.73	0.66	0.73	0.73	0.73	1	1	1	1	1	0.26	0.26	0.26	0.26	0.26	1	1	1	1	1
	S	20.26	20.26	20.26	20.26	20.26	1	1	1	1	1	5.33	5.33	5.33	5.33	5.33	16.66	16.06	16.66	16.66	16.66	1	1	1	1	1	15	15	15	15	15
LAMA	Q_{20}	1	1	1	1	1	1	1	1	1	1	0.73	0.66	0.73	0.73	0.73	1	0.93	1	1	1	0.73	0.73	0.73	0.73	0.73	1	1	1	1	1
	Q_{50}	1	1	1	1	1	1	1	1	1	1	0.73	0.66	0.73	0.73	0.73	1	0.93	1	1	1	1	1	1	1	1	1	1	1	1	1
	Solve	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	T	0.19	0.19	0.19	0.18	0.19	0.58	0.57	0.41	0.38	0.38	0.86	0.77	0.69	0.66	0.66	0.15	0.15	0.12	0.11	0.12	0.11	0.1	0.1	0.1	0.1	0.17	0.17	0.17	0.16	0.17
	Q	1	1	1	1	1	1	1	1	0.8	0.8	0.73	0.66	0.73	0.73	0.73	1	1.06	1	1	1	0.26	0.26	0.26	0.26	0.26	1	1	1	1	1
	S	20.26	20.26	20.26	20.26	20.26	1	1	1	1	1	5.33	5.33	5.33	5.33	5.33	16.66	16.73	16.66	16.66	16.66	1	1	1	1	1	15	15	15	15	15
LAMA_G	Q_{20}	1	1	1	1	1	1	1	1	0.8	0.8	0.73	0.66	0.73	0.73	0.73	1	1	1	1	1	0.73	0.73	0.73	0.73	0.73	1	1	1	1	1
	Q_{50}	1	1	1	1	1	1	1	1	1	1	0.73	0.66	0.73	0.73	0.73	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Solve	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table A.2: Goal recognition with random observations in 10s

Domain		Blocks					Campus					Grid					Intrusion					Kitchen					Logistics				
Approach	$\%O$	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10
	T	1080.6	729.06	529.04	405.1	405.14	1.78	1.11	0.55	0.28	0.28	224.67	144.5	79.83	41.47	37.42	588.34	414.3	214.2	4.25	4.22	694.67	243.5	60.65	33.3	33.31	44.93	42.02	18.94	9.18	9.18
HSP^*_f	Q	1	1	0.86	0.86	0.86	1	1	1	1	1	1	0.2	0.66	0.86	0.73	1	1	1	1	1	1	1	0.86	0.86	0.86	1	0.93	0.66	0.8	0.8
	S	1.06	1.13	3.73	9.33	9.33	1	1	1	1	1	1	1.06	1.93	3.6	3.93	1	1	1.06	4.46	4.6	1	1	1.2	1.26	1.26	1.06	1.06	2.26	3.6	3.6
	T	531.26	446.26	379.81	357.94	357.94	1.22	0.8	0.5	0.32	0.32	114.12	79.76	52.92	39.20	38.8	447.41	281.11	151.37	3.58	3.55	480.39	171.08	49.62	37.93	37.92	36.26	32.46	14.8	7.04	7.04
$g\text{HSP}^*_f$	Q	1	1	1	1	1	1	1	1	1	1	1	0.2	0.8	0.93	0.8	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.06	1.13	4.06	11.46	11.46	1	1	1	1	1	1	1.2	2.26	3.86	4.2	1	1	1.06	4.46	4.6	1	1	1.33	1.4	1.4	1	1.13	2.26	3.6	3.6
	T	0.19	0.2	0.18	0.18	0.18	2.68	1.43	0.82	0.53	0.52	6.30	6.35	5.54	5.21	5.23	2.09	2.01	1.88	1.07	1.04	3.32	3.19	2.49	2.34	2.34	0.17	0.17	0.17	0.17	0.17
	Q	1	1	1	1	1	1	1	1	1	1	0.53	0.4	0.6	0.73	0.66	0.8	0.8	0.8	0.8	0.8	1	1	1	1	1	1	1	1	1	1
	S	20.26	20.26	20.26	20.26	20.26	1	1	1	1	1	3.33	3.33	3.4	3.53	3.6	15	15	15	15	15	1.53	1	1.33	1.4	1.4	15	15	15	15	15
LAMA	Q_{20}	1	1	1	1	1	1	1	1	1	1	0.6	0.46	0.6	0.73	0.66	0.8	0.8	0.8	0.8	0.8	1	1	1	1	1	1	1	1	1	1
	Q_{50}	1	1	1	1	1	1	1	1	1	1	0.6	0.66	0.6	0.73	0.66	0.86	0.93	1	1	1	1	1	1	1	1	1	1	1	1	1
	Solve	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	T	0.19	0.19	0.19	0.19	0.18	0.59	0.55	0.4	0.37	0.37	4.86	4.67	4.23	4.03	4.02	0.49	0.37	0.37	0.36	0.36	0.42	0.36	0.32	0.32	0.32	0.17	0.17	0.18	0.17	0.17
	Q	1	1	1	1	1	1	1	1	0.8	0.8	0.33	0.4	0.53	0.53	0.6	0.8	0.8	0.8	0.8	0.8	1	1	1	1	1	1	1	1	1	1
	S	20.26	20.26	20.26	20.26	20.26	1	1	1	1	1	2.33	2.33	2.4	2.66	2.6	15	14.73	15	15	15	1.53	1.26	1.33	1.4	1.4	15	15	15	15	15
LAMA_G	Q_{20}	1	1	1	1	1	1	1	1	0.8	0.8	0.4	0.6	0.53	0.53	0.6	0.8	0.73	0.8	0.8	0.8	1	1	1	1	1	1	1	1	1	1
	Q_{50}	1	1	1	1	1	1	1	1	1	1	0.6	0.8	0.73	0.8	0.8	0.86	0.8	0.86	1	1	1	1	1	1	1	1	1	1	1	1
	Solve	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table A.3: Goal recognition with random observations in 20s

Domain		Blocks					Campus					Grid					Intrusion					Kitchen					Logistics				
Approach	%O	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10
HSP^*_f	T	1080.6	729.06	529.04	405.1	405.14	1.78	1.11	0.55	0.28	0.28	224.67	144.5	79.83	41.47	37.42	588.34	414.3	214.2	4.25	4.22	694.67	243.5	60.65	33.3	33.31	44.93	42.02	18.94	9.18	9.18
	Q	1	1	0.86	0.86	0.86	1	1	1	1	1	1	0.2	0.66	0.86	0.73	1	1	1	1	1	1	1	0.86	0.86	0.86	1	0.93	0.66	0.8	0.8
	S	1.06	1.13	3.73	9.33	9.33	1	1	1	1	1	1	1.06	1.93	3.6	3.93	1	1	1.06	4.46	4.6	1	1	1.2	1.26	1.26	1.06	1.06	2.26	3.6	3.6
$g\text{HSP}^*_f$	T	531.26	446.26	379.81	357.94	357.94	1.22	0.8	0.5	0.32	0.32	114.12	79.76	52.92	39.20	38.8	447.41	281.11	151.37	3.58	3.55	480.39	171.08	49.62	37.93	37.92	36.26	32.46	14.8	7.04	7.04
	Q	1	1	1	1	1	1	1	1	1	1	1	0.2	0.8	0.93	0.8	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.06	1.13	4.06	11.46	11.46	1	1	1	1	1	1	1.2	2.26	3.86	4.2	1	1	1.06	4.46	4.6	1	1	1.33	1.4	1.4	1	1.13	2.26	3.6	3.6
LAMA	T	12.96	12.96	12.96	12.94	12.94	2.71	1.45	0.82	0.52	0.53	13.94	13.19	11.66	10.38	10.36	13.56	13.27	12.86	6.55	6.47	8.11	7.98	6.48	6.33	6.33	10.26	10.05	8.39	5.96	5.96
	Q	0.4	0.4	0.4	0.4	0.4	1	1	1	1	1	0.4	0.46	0.6	0.66	0.73	0.46	0.46	0.46	0.46	0.46	1	1	1	1	1	0.53	0.53	0.53	0.53	0.53
	S	7.66	7.66	7.66	7.66	7.66	1	1	1	1	1	2	1.93	1.8	2.53	2.53	8.86	8.26	7.66	8.33	8.46	1.53	1	1.33	1.4	1.4	8.0	8.0	8.0	8.0	8.0
	Q_{20}	1	1	1	1	1	1	1	1	1	1	0.4	0.66	0.66	0.73	0.8	0.46	0.46	0.46	0.46	0.46	1	1	1	1	1	0.53	0.53	0.53	0.53	0.53
	Q_{50}	1	1	1	1	1	1	1	1	1	1	0.8	0.86	0.86	0.86	0.86	0.46	0.46	0.46	0.46	0.46	1	1	1	1	1	0.66	0.66	0.73	0.86	0.86
	d	0.08	0.073	0.062	0.063	0.063	0.066	0	0	0	0	0.194	0.11	0.092	0.028	0.026	0.058	0.057	0.054	0.034	0.034	0	0	0	0	0	0.114	0.108	0.09	0.067	0.067
	Solve	0.1	0.1	0.1	0.1	0.1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
LAMA$_G$	T	8.55	8.56	8.55	8.56	8.55	0.59	0.57	0.4	0.38	0.38	10.43	9.38	8.46	7.51	7.63	2.27	1.87	1.63	1.54	1.55	0.42	0.36	0.32	0.31	0.31	2.89	2.69	2.51	2.44	2.43
	Q	0.33	0.33	0.33	0.33	0.33	1	1	1	0.8	0.8	0.26	0.46	0.46	0.6	0.66	0.4	0.4	0.4	0.4	0.4	1	1	1	1	1	0.53	0.53	0.53	0.53	0.53
	S	7.66	7.66	7.66	7.66	7.66	1	1	1	1	1	1.66	1.66	1.73	2.13	2.13	10	8.8	7	7.66	7.8	1.53	1	1.33	1.4	1.4	8.0	8.0	8.0	8.0	8.0
	Q_{20}	0.8	0.8	0.8	0.8	0.8	1	1	1	0.8	0.8	0.53	0.66	0.66	0.73	0.8	0.4	0.4	0.4	0.4	0.4	1	1	1	1	1	0.53	0.53	0.53	0.53	0.53
	Q_{50}	1	1	1	1	1	1	1	1	1	1	0.8	0.86	0.8	0.86	0.86	0.4	0.4	0.4	1	1	1	1	1	1	1	0.93	0.86	0.73	0.6	0.6
	d	0.081	0.075	0.061	0.052	0.052	0.066	5.3×10^{-5}	0.02	0.19	0.19	0.201	0.14	0.095	0.04	0.034	0.06	0.06	0.058	0.034	0.034	8.6×10^{-4}	1×10^{-3}	5×10^{-3}	4×10^{-3}	4×10^{-3}	0.112	0.107	0.09	0.067	0.067
	Solve	0.1	0.1	0.1	0.1	0.1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table A.4: Goal recognition with random observations in 60s

Domain		Blocks					Campus					Grid					Intrusion					Kitchen					Logistics				
Approach	%O	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10
HSP^*_f	T	1080.6	729.06	529.04	405.1	405.14	1.78	1.11	0.55	0.28	0.28	224.67	144.5	79.83	41.47	37.42	588.34	414.3	214.2	4.25	4.22	694.67	243.5	60.65	33.3	33.31	44.93	42.02	18.94	9.18	9.18
	Q	1	1	0.86	0.86	0.86	1	1	1	1	1	1	0.2	0.66	0.86	0.73	1	1	1	1	1	1	1	0.86	0.86	0.86	1	0.93	0.66	0.8	0.8
	S	1.06	1.13	3.73	9.33	9.33	1	1	1	1	1	1	1.06	1.93	3.6	3.93	1	1	1.06	4.46	4.6	1	1	1.2	1.26	1.26	1.06	1.06	2.26	3.6	3.6
$g\text{HSP}^*_f$	T	531.26	446.26	379.81	357.94	357.94	1.22	0.8	0.5	0.32	0.32	114.12	79.76	52.92	39.20	38.8	447.41	281.11	151.37	3.58	3.55	480.39	171.08	49.62	37.93	37.92	36.26	32.46	14.8	7.04	7.04
	Q	1	1	1	1	1	1	1	1	1	1	1	0.2	0.8	0.93	0.8	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.06	1.13	4.06	11.46	11.46	1	1	1	1	1	1	1.2	2.26	3.86	4.2	1	1	1.06	4.46	4.6	1	1	1.33	1.4	1.4	1	1.13	2.26	3.6	3.6
LAMA	T	54.36	54.36	54.42	54.43	54.45	2.7	1.42	0.83	0.53	0.53	36.33	31.65	24.49	17.37	16.30	44.89	44.48	37.99	12.72	12.63	26.33	26.20	20.45	20.3	20.3	45.17	41.71	26.53	11.38	11.39
	Q	0.06	0.06	0.06	0.06	0.06	1	1	1	1	1	0.46	0.33	0.66	0.93	0.8	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.26	1.26	1.26	1.26	1.26	1	1	1	1	1	1.73	1.73	1.73	3.86	4.2	1.6	1	1.06	4.46	4.6	1.53	1	1.33	1.4	1.4	1	1.13	2.26	3.6	3.6
	Q_{20}	0.13	0.13	0.13	0.13	0.13	1	1	1	1	1	0.66	0.53	0.66	1	0.86	1	1	1	1	1	1	1	1	1	1	1	1	0.93	1	1
	Q_{50}	0.73	0.73	0.73	0.73	0.73	1	1	1	1	1	0.86	0.8	0.86	1	0.86	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	d	0.085	0.081	0.057	0.04	0.04	0.066	0	0	0	0	0.136	0.1	0.036	0	0	1×10^{-6}	9×10^{-6}	1.9×10^{-5}	0	0	0	0	0	0	0	0	2×10^{-6}	0	0	0
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
LAMA$_G$	T	39.63	39.62	39.61	39.63	39.62	0.6	0.58	0.4	0.39	0.38	28.15	22.42	16.30	9.98	9.43	3.32	2.63	2.2	2.08	2.08	0.42	0.36	0.32	0.31	0.31	5.47	4.95	4.50	4.33	4.36
	Q	0.06	0.06	0.06	0.06	0.06	1	1	1	0.8	0.8	0.46	0.4	0.66	0.86	0.73	1	1	1	1	1	1	1	1	1	1	1	0.8	0.4	0.46	0.46
	S	1.4	1.4	1.4	1.4	1.4	1	1	1	1	1	1.66	1.66	1.4	3.4	3.73	16.66	10.4	1.13	4.46	4.6	1.53	1	1.33	1.4	1.4	1	1.2	1.8	2.93	2.93
	Q_{20}	0.26	0.26	0.26	0.26	0.26	1	1	1	0.8	0.8	0.6	0.53	0.8	1	0.86	1	1	1	1	1	1	1	1	1	1	1	1	0.66	0.6	0.6
	Q_{50}	0.73	0.73	0.73	0.73	0.73	1	1	1	1	1	0.8	0.86	0.86	1	0.86	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.86	0.86
	d	0.085	0.081	0.058	0.042	0.042	0.066	5.3×10^{-5}	0.02	0.2	0.2	0.127	0.111	0.042	0.015	0.015	1.2×10^{-5}	1×10^{-3}	5×10^{-3}	0	0	8.6×10^{-4}	1×10^{-3}	5×10^{-3}	4×10^{-3}	4×10^{-3}	1×10^{-3}	0.045	0.07	0.063	0.063
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table A.5: Goal recognition with random observations in 120s

Domain		Blocks					Campus					Grid					Intrusion					Kitchen					Logistics				
Approach	%O	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10
HSP^*_f	T	1080.6	729.06	529.04	405.1	405.14	1.78	1.11	0.55	0.28	0.28	224.67	144.5	79.83	41.47	37.42	588.34	414.3	214.2	4.25	4.22	694.67	243.5	60.65	33.3	33.31	44.93	42.02	18.94	9.18	9.18
	Q	1	1	0.86	0.86	0.86	1	1	1	1	1	1	0.2	0.66	0.86	0.73	1	1	1	1	1	1	1	0.86	0.86	0.86	1	0.93	0.66	0.8	0.8
	S	1.06	1.13	3.73	9.33	9.33	1	1	1	1	1	1	1.06	1.93	3.6	3.93	1	1	1.06	4.46	4.6	1	1	1.2	1.26	1.26	1.06	1.06	2.26	3.6	3.6
$g\text{HSP}^*_f$	T	531.26	446.26	379.81	357.94	357.94	1.22	0.8	0.5	0.32	0.32	114.12	79.76	52.92	39.20	38.8	447.41	281.11	151.37	3.58	3.55	480.39	171.08	49.62	37.93	37.92	36.26	32.46	14.8	7.04	7.04
	Q	1	1	1	1	1	1	1	1	1	1	1	0.2	0.8	0.93	0.8	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.06	1.13	4.06	11.46	11.46	1	1	1	1	1	1	1.2	2.26	3.86	4.2	1	1	1.06	4.46	4.6	1	1	1.33	1.4	1.4	1	1.13	2.26	3.6	3.6
LAMA	T	120.41	120.39	120.38	120.39	120.38	2.69	1.44	0.83	0.52	0.53	61.95	50.22	35.49	18.71	16.50	92.85	89.01	64.97	17.57	17.48	52.14	51.83	40.4	40.25	40.25	87.71	75.02	33.38	11.4	11.4
	Q	0.06	0.06	0.06	0.06	0.06	1	1	1	1	1	0.73	0.33	0.8	0.93	0.8	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.26	1.26	1.26	1.26	1.26	1	1	1	1	1	1.6	1.46	2.33	3.86	4.2	1.6	1	1.06	4.46	4.6	1.13	1	1.33	1.4	1.4	1	1.13	2.26	3.6	3.6
	Q_{20}	0.13	0.13	0.13	0.13	0.13	1	1	1	1	1	0.73	0.53	0.86	1	0.86	1	1	1	1	1	1	1	1	1	1	1	1	0.93	1	1
	Q_{50}	0.73	0.73	0.73	0.73	0.73	1	1	1	1	1	1	0.86	0.93	1	0.86	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	d	0.085	0.081	0.057	0.04	0.04	0.066	0	0	0	0	0.075	0.072	0.01	0	0	1×10^{-6}	5×10^{-6}	2×10^{-6}	0	0	0	0	0	0	0	0	0	0	0	0
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
LAMA$_G$	T	87.34	87.33	87.34	87.41	87.33	0.58	0.57	0.39	0.38	0.38	45.96	34.43	20.26	10.01	9.80	3.32	2.63	2.22	2.08	2.08	0.42	0.36	0.32	0.32	0.31	5.47	4.98	4.54	4.36	4.37
	Q	0.06	0.06	0.06	0.06	0.06	1	1	1	0.8	0.8	0.6	0.26	0.73	0.86	0.73	1	1	1	1	1	1	1	1	1	1	1	0.8	0.4	0.46	0.46
	S	1.4	1.4	1.4	1.4	1.4	1	1	1	1	1	1.6	1.4	2.13	3.4	3.73	16.66	10.4	1.13	4.46	4.6	1.53	1	1.33	1.4	1.4	1	1.2	1.8	2.93	2.93
	Q_{20}	0.26	0.26	0.26	0.26	0.26	1	1	1	0.8	0.8	0.8	0.6	0.86	1	0.86	1	1	1	1	1	1	1	1	1	1	1	1	0.66	0.6	0.6
	Q_{50}	0.73	0.73	0.73	0.73	0.73	1	1	1	1	1	1	1	0.93	1	0.86	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.86	0.86
	d	0.085	0.081	0.058	0.042	0.042	0.066	4.9×10^{-5}	0.02	0.19	0.19	0.073	0.068	0.02	0.015	0.015	6×10^{-6}	1×10^{-3}	5×10^{-3}	0	0	8.6×10^{-4}	1×10^{-3}	5×10^{-3}	4×10^{-3}	4×10^{-3}	1×10^{-3}	0.045	0.07	0.063	0.063
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table A.6: Goal recognition with random observations in 240s

Domain		Blocks					Campus					Grid					Intrusion					Kitchen					Logistics				
Approach	$\%O$	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10
HSP^*_f	T	1080.6	729.06	529.04	405.1	405.14	1.78	1.11	0.55	0.28	0.28	224.67	144.5	79.83	41.47	37.42	588.34	414.3	214.2	4.25	4.22	694.67	243.5	60.65	33.3	33.31	44.93	42.02	18.94	9.18	9.18
	Q	1	1	0.86	0.86	0.86	1	1	1	1	1	1	0.2	0.66	0.86	0.73	1	1	1	1	1	1	1	0.86	0.86	0.86	1	0.93	0.66	0.8	0.8
	S	1.06	1.13	3.73	9.33	9.33	1	1	1	1	1	1	1.06	1.93	3.6	3.93	1	1	1.06	4.46	4.6	1	1	1.2	1.26	1.26	1.06	1.06	2.26	3.6	3.6
$g\text{HSP}^*_f$	T	531.26	446.26	379.81	357.94	357.94	1.22	0.8	0.5	0.32	0.32	114.12	79.76	52.92	39.20	38.8	447.41	281.11	151.37	3.58	3.55	480.39	171.08	49.62	37.93	37.92	36.26	32.46	14.8	7.04	7.04
	Q	1	1	1	1	1	1	1	1	1	1	1	0.2	0.8	0.93	0.8	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.06	1.13	4.06	11.46	11.46	1	1	1	1	1	1	1.2	2.26	3.86	4.2	1	1	1.06	4.46	4.6	1	1	1.33	1.4	1.4	1	1.13	2.26	3.6	3.6
LAMA	T	228.49	228.61	228.51	228.62	228.66	2.71	1.45	0.81	0.54	0.52	104.28	81.43	47.80	20.06	16.53	184.06	173.21	96.9	25.56	25.47	98.11	93.35	72.98	71.31	71.34	166.61	132.48	40.23	11.39	11.39
	Q	0.06	0.06	0.06	0.06	0.06	1	1	1	1	1	0.66	0.2	0.8	0.93	0.8	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.26	1.26	1.26	1.26	1.26	1	1	1	1	1	1.26	1.13	2.33	3.86	4.2	1.6	1	1.06	4.46	4.6	1	1	1.33	1.4	1.4	1	1.13	2.26	3.6	3.6
	Q_{20}	0.13	0.13	0.13	0.13	0.13	1	1	1	1	1	0.86	0.66	0.86	1	0.86	1	1	1	1	1	1	1	1	1	1	1	1	0.93	1	1
	Q_{50}	0.73	0.73	0.73	0.73	0.73	1	1	1	1	1	1	1	0.93	1	0.86	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	d	0.085	0.081	0.057	0.04	0.04	0.066	0	0	0	0	0.055	7.1×10^{-3}	6.5×10^{-3}	0	0	1×10^{-6}	2×10^{-6}	2×10^{-6}	0	0	0	0	0	0	0	0	0	0	0	0
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
LAMA_G	T	184.15	184.14	184.16	184.17	184.17	0.59	0.59	0.39	0.37	0.38	63.59	40.33	21.2	10.11	9.38	3.32	2.62	2.21	2.08	2.07	0.42	0.36	0.32	0.31	0.32	5.46	4.99	4.54	4.38	4.35
	Q	0.06	0.06	0.06	0.06	0.06	1	1	1	0.8	0.8	0.73	0.26	0.73	0.86	0.73	1	1	1	1	1	1	1	1	1	1	1	0.8	0.4	0.46	0.46
	S	1.4	1.4	1.4	1.4	1.4	1	1	1	1	1	1.33	1.8	2	3.4	3.73	16.66	12.93	1.2	4.46	4.6	1.53	1	1.33	1.4	1.4	1	1.2	1.8	2.93	2.93
	Q_{20}	0.26	0.26	0.26	0.26	0.26	1	1	1	0.8	0.8	0.86	0.6	0.93	1	0.86	1	1	1	1	1	1	1	1	1	1	1	1	0.66	0.6	0.6
	Q_{50}	0.73	0.73	0.73	0.73	0.73	1	1	1	1	1	1	1	0.93	1	0.86	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.86	0.86
	d	0.085	0.081	0.058	0.042	0.042	0.066	4.9×10^{-5}	0.02	0.2	0.2	0.05	0.055	0.012	0.015	0.015	6×10^{-6}	1×10^{-3}	7×10^{-3}	0	0	8.6×10^{-4}	1×10^{-3}	5×10^{-3}	4×10^{-3}	4×10^{-3}	1×10^{-3}	0.045	0.07	0.063	0.063
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table A.7: Goal recognition with random observations in 360s

Domain		Blocks					Campus					Grid					Intrusion					Kitchen					Logistics				
Approach	$\%O$	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10
HSP^*_f	T	1080.6	729.06	529.04	405.1	405.14	1.78	1.11	0.55	0.28	0.28	224.67	144.5	79.83	41.47	37.42	588.34	414.3	214.2	4.25	4.22	694.67	243.5	60.65	33.3	33.31	44.93	42.02	18.94	9.18	9.18
	Q	1	1	0.86	0.86	0.86	1	1	1	1	1	1	0.2	0.66	0.86	0.73	1	1	1	1	1	1	1	0.86	0.86	0.86	1	0.93	0.66	0.8	0.8
	S	1.06	1.13	3.73	9.33	9.33	1	1	1	1	1	1	1.06	1.93	3.6	3.93	1	1	1.06	4.46	4.6	1	1	1.2	1.26	1.26	1.06	1.06	2.26	3.6	3.6
$g\text{HSP}^*_f$	T	531.26	446.26	379.81	357.94	357.94	1.22	0.8	0.5	0.32	0.32	114.12	79.76	52.92	39.20	38.8	447.41	281.11	151.37	3.58	3.55	480.39	171.08	49.62	37.93	37.92	36.26	32.46	14.8	7.04	7.04
	Q	1	1	1	1	1	1	1	1	1	1	1	0.2	0.8	0.93	0.8	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.06	1.13	4.06	11.46	11.46	1	1	1	1	1	1	1.2	2.26	3.86	4.2	1	1	1.06	4.46	4.6	1	1	1.33	1.4	1.4	1	1.13	2.26	3.6	3.6
LAMA	T	333.97	333.6	333.87	334.09	334.03	2.69	1.42	0.84	0.53	0.53	133.33	104.96	52.11	20.11	16.42	274.74	252.25	118.13	33.53	33.45	131.92	117.14	90.25	83.24	83.35	240.34	180.41	44.67	11.39	11.39
	Q	0.06	0.06	0.06	0.06	0.06	1	1	1	1	1	0.8	0.26	0.8	0.93	0.8	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.26	1.26	1.26	1.26	1.26	1	1	1	1	1	1.2	1.2	2.26	3.86	4.2	1.6	1	1.06	4.46	4.6	1	1	1.33	1.4	1.4	1	1.13	2.26	3.6	3.6
	Q_{20}	0.13	0.13	0.13	0.13	0.13	1	1	1	1	1	1	0.66	0.93	1	0.86	1	1	1	1	1	1	1	1	1	1	1	1	0.93	1	1
	Q_{50}	0.73	0.73	0.73	0.73	0.73	1	1	1	1	1	1	1	0.93	1	0.86	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	d	0.085	0.081	0.057	0.04	0.04	0.066	0	0	0	0	0.04	0	0	0	0	0	0	2×10^{-6}	0	0	0	0	0	0	0	0	0	0	0	0
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
LAMA_G	T	280.91	280.9	280.9	280.91	280.9	0.6	0.56	0.4	0.38	0.38	73.46	40.7	21.16	9.82	9.83	3.32	2.62	2.21	2.08	2.08	0.41	0.36	0.32	0.31	0.32	5.47	4.99	4.54	4.36	4.35
	Q	0.06	0.06	0.06	0.06	0.06	1	1	1	0.8	0.8	0.86	0.33	0.73	0.86	0.73	1	1	1	1	1	1	1	1	1	1	1	0.8	0.4	0.46	0.46
	S	1.4	1.4	1.4	1.4	1.4	1	1	1	1	1	1.2	1.8	2	3.4	3.73	16.66	10.4	1.2	4.46	4.6	1.53	1	1.33	1.4	1.4	1	1.2	1.8	2.93	2.93
	Q_{20}	0.26	0.26	0.26	0.26	0.26	1	1	1	0.8	0.8	0.93	0.66	0.93	1	0.86	1	1	1	1	1	1	1	1	1	1	1	1	0.66	0.6	0.6
	Q_{50}	0.73	0.73	0.73	0.73	0.73	1	1	1	1	1	1	1	0.93	1	0.86	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.86	0.86
	d	0.085	0.081	0.058	0.042	0.042	0.066	5.3×10^{-5}	0.02	0.19	0.19	0.022	0.029	0.012	0.015	0.015	8×10^{-6}	1×10^{-3}	7×10^{-3}	0	0	8.6×10^{-4}	1×10^{-3}	5×10^{-3}	4×10^{-3}	4×10^{-3}	1×10^{-3}	0.045	0.07	0.063	0.063
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table A.8: Goal recognition with random observations in 1800s

Domain		Blocks					Campus					Grid					Intrusion					Kitchen					Logistics				
Approach	$\%O$	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10
HSP^*_f	T	1080.6	729.06	529.04	405.1	405.14	1.78	1.11	0.55	0.28	0.28	224.67	144.5	79.83	41.47	37.42	588.34	414.3	214.2	4.25	4.22	694.67	243.5	60.65	33.3	33.31	44.93	42.02	18.94	9.18	9.18
	Q	1	1	0.86	0.86	0.86	1	1	1	1	1	1	0.2	0.66	0.86	0.73	1	1	1	1	1	1	1	0.86	0.86	0.86	1	0.93	0.66	0.8	0.8
	S	1.06	1.13	3.73	9.33	9.33	1	1	1	1	1	1	1.06	1.93	3.6	3.93	1	1	1.06	4.46	4.6	1	1	1.2	1.26	1.26	1.06	1.06	2.26	3.6	3.6
$g\text{HSP}^*_f$	T	531.26	446.26	379.81	357.94	357.94	1.22	0.8	0.5	0.32	0.32	114.12	79.76	52.92	39.20	38.8	447.41	281.11	151.37	3.58	3.55	480.39	171.08	49.62	37.93	37.92	36.26	32.46	14.8	7.04	7.04
	Q	1	1	1	1	1	1	1	1	1	1	1	0.2	0.8	0.93	0.8	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.06	1.13	4.06	11.46	11.46	1	1	1	1	1	1	1.2	2.26	3.86	4.2	1	1	1.06	4.46	4.6	1	1	1.33	1.4	1.4	1	1.13	2.26	3.6	3.6
LAMA	T	1603.24	1522.96	1260.6	1077.62	1082.15	2.74	1.43	0.84	0.52	0.52	294.75	208.70	62.12	20.38	16.52	1303.14	778.67	200.47	76.59	75.54	359.71	221.22	116.66	92.17	92.23	772.76	401.79	50.26	11.39	11.39
	Q	0.33	0.8	0.8	0.93	1	1	1	1	1	1	1	0.26	0.8	0.93	0.8	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1	1.13	3.86	10.4	10.93	1	1	1	1	1	1	1.2	2.26	3.86	4.2	1	1	1.06	4.46	4.6	1	1	1.33	1.4	1.4	1	1.13	2.26	3.6	3.6
	Q_{20}	1	1	1	1	1	1	1	1	1	1	1	0.66	0.93	1	0.86	1	1	1	1	1	1	1	1	1	1	1	1	0.93	1	1
	Q_{50}	1	1	1	1	1	1	1	1	1	1	1	1	0.93	1	0.86	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	d	0.042	0.017	9×10^{-3}	1×10^{-3}	2×10^{-3}	0.066	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
LAMA_G	T	849.08	840.76	814.95	803.04	809.1	0.57	0.56	0.4	0.38	0.38	81.36	40.27	19.26	10.16	9.76	3.33	2.62	2.21	2.07	2.08	0.42	0.36	0.32	0.32	0.32	5.43	4.95	4.54	4.37	4.36
	Q	1	0.8	0.73	0.66	0.46	1	1	1	0.8	0.8	1	0.33	0.73	0.86	0.73	1	1	1	1	1	1	1	1	1	1	1	0.8	0.4	0.46	0.46
	S	2.26	1.2	3	6.2	4.2	1	1	1	1	1	1	1.8	2	3.4	3.73	15.4	11.66	1.2	4.46	4.6	1.53	1	1.33	1.4	1.4	1	1.2	1.8	2.93	2.93
	Q_{20}	1	1	1	1	1	1	1	1	0.8	0.8	1	0.66	0.93	1	0.86	1	1	1	1	1	1	1	1	1	1	1	1	0.66	0.6	0.6
	Q_{50}	1	1	1	1	1	1	1	1	1	1	1	1	0.93	1	0.86	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.86	0.86
	d	5×10^{-3}	0.02	0.024	0.014	0.015	0.066	4.9×10^{-5}	0.02	0.19	0.19	6.9×10^{-5}	0.029	0.012	0.015	0.015	6×10^{-6}	1×10^{-3}	7×10^{-3}	0	0	8.6×10^{-4}	1×10^{-3}	5×10^{-3}	4×10^{-3}	4×10^{-3}	1×10^{-3}	0.045	0.07	0.063	0.063
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table A.9: Goal recognition with random observations for relaxed plan heuristics

Domain		Blocks					Campus					Grid					Intrusion					Kitchen					Logistics				
Approach	$\%O$	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10
HSP^*_f	T	1080.6	729.06	529.04	405.1	405.14	1.78	1.11	0.55	0.28	0.28	224.67	144.5	79.83	41.47	37.42	588.34	414.3	214.2	4.25	4.22	694.67	243.5	60.65	33.3	33.31	44.93	42.02	18.94	9.18	9.18
	Q	1	1	0.86	0.86	0.86	1	1	1	1	1	1	0.2	0.66	0.86	0.73	1	1	1	1	1	1	1	0.86	0.86	0.86	1	0.93	0.66	0.8	0.8
	S	1.06	1.13	3.73	9.33	9.33	1	1	1	1	1	1	1.06	1.93	3.6	3.93	1	1	1.06	4.46	4.6	1	1	1.2	1.26	1.26	1.06	1.06	2.26	3.6	3.6
$g\text{HSP}^*_f$	T	531.26	446.26	379.81	357.94	357.94	1.22	0.8	0.5	0.32	0.32	114.12	79.76	52.92	39.20	38.8	447.41	281.11	151.37	3.58	3.55	480.39	171.08	49.62	37.93	37.92	36.26	32.46	14.8	7.04	7.04
	Q	1	1	1	1	1	1	1	1	1	1	1	0.2	0.8	0.93	0.8	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.06	1.13	4.06	11.46	11.46	1	1	1	1	1	1	1.2	2.26	3.86	4.2	1	1	1.06	4.46	4.6	1	1	1.33	1.4	1.4	1	1.13	2.26	3.6	3.6
h^a_s	T	0.44	0.39	0.37	0.36	0.36	0.05	0.04	0.03	0.03	0.03	0.42	0.33	0.26	0.24	0.24	0.46	0.32	0.27	0.25	0.25	0.06	0.05	0.04	0.04	0.04	0.34	0.32	0.3	0.29	0.3
	Q	1	1	1	1	1	1	1	0.86	0.93	0.93	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.93	0.93
	S	20.26	20.26	20.26	20.26	20.26	2	2	1.8	1.86	1.86	6.66	6.66	6.6	6.4	6.4	16.66	16.06	16.66	16.66	16.66	3	3	3	3	3	15	15	14.8	14.66	14.66
	Q_{20}	1	1	1	1	1	1	1	0.86	0.93	0.93	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.93	0.93
	Q_{50}	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.93	0.93
	d	0.086	0.08	0.055	0.031	0.031	0.466	0.466	0.495	0.347	0.347	0.252	0.238	0.287	0.132	0.126	0.117	0.117	0.114	0.071	0.07	0.414	0.407	0.274	0.216	0.216	0.115	0.11	0.093	0.068	0.068
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\text{FGR}^I_{rp}\text{T}$	T	1	1.07	1.28	1.43	1.43	0.27	0.29	0.33	0.38	0.38	107.06	109.03	112.48	114.42	115.41	0.88	0.77	0.56	0.31	0.31	0.26	0.24	0.2	0.16	0.16	0.88	0.99	1.15	1.24	1.25
	Q	0.86	0.93	0.46	0.46	0.46	1	1	1	0.93	0.93	1	0.13	0.8	0.73	0.53	1	1	1	1	1	1	0.26	0.26	0.26	0.26	1	0.86	0.53	0.33	0.33
	S	2.33	3.66	2.33	5.26	5.26	1	1	1	1.13	1.13	1	1.2	1.6	2.66	2.73	1	8.73	4.26	12.6	12.6	1	1.53	1	1	1	1	1.33	2.06	2.33	2.33
	Q_{20}	0.86	0.93	0.66	0.66	0.66	1	1	1	0.93	0.93	1	0.13	0.8	0.73	0.53	1	1	1	1	1	1	0.26	0.26	0.26	0.26	1	0.93	0.73	0.8	0.8
	Q_{50}	0.93	1	0.73	0.86	0.86	1	1	1	0.93	0.93	1	0.13	0.8	0.73	0.53	1	1	1	1	1	1	1	1	1	1	1	0.93	0.73	0.86	0.86
	d	0.017	0.025	0.038	0.033	0.033	0.066	2.2×10^{-5}	2.2×10^{-3}	0.112	0.112	1.35×10^{-4}	1.94×10^{-4}	0.017	0.068	0.062	0	0.06	0.054	0.057	0.055	4.25×10^{-4}	0.45	0.24	0.213	0.213	4.88×10^{-4}	0.026	0.061	0.05	0.05
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\text{FGR}^I_{rp}\text{ET}$	T	6.01	5.43	4.16	3.69	3.68	0.31	0.33	0.39	0.42	0.42	118.91	119.85	122.44	124.18	124.99	1.29	1.18	0.99	0.73	0.73	0.28	0.26	0.23	0.19	0.19	7.61	3.70	2.82	2.83	2.83
	Q	0.93	0.73	0.46	0.46	0.46	1	1	1	0.93	0.93	1	0.13	0.8	0.73	0.53	1	1	1	1	1	1	0.26	0.26	0.26	0.26	1	0.66	0.53	0.33	0.33
	S	3.66	2.46	2.33	5.26	5.26	1	1	1	1.13	1.13	1	1.2	1.6	2.66	2.73	1	8.73	4.26	12.6	12.6	1	1.53	1	1	1	1	1.26	2.06	2.26	2.26
	Q_{20}	0.93	0.73	0.66	0.66	0.66	1	1	1	0.93	0.93	1	0.13	0.8	0.73	0.53	1	1	1	1	1	1	0.26	0.26	0.26	0.26	1	0.73	0.73	0.8	0.8
	Q_{50}	1	0.86	0.73	0.86	0.86	1	1	1	0.93	0.93	1	0.13	0.8	0.73	0.53	1	1	1	1	1	1	1	1	1	1	1	1	0.73	0.86	0.86
	d	0.016	0.035	0.038	0.033	0.033	0.066	2.4×10^{-5}	3.4×10^{-3}	0.106	0.106	1.35×10^{-4}	1.94×10^{-4}	0.017	0.068	0.062	0	0.06	0.054	0.057	0.055	4.25×10^{-4}	0.45	0.24	0.213	0.213	7.58×10^{-4}	0.038	0.061	0.044	0.044
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\text{FGR}^+_{rp}\text{T}$	T	0.76	0.7	0.7	0.76	0.76	0.23	0.23	0.25	0.28	0.28	33.30	33.37	34.41	35.62	35.73	0.88	0.77	0.56	0.29	0.29	0.25	0.23	0.2	0.16	0.16	0.8	0.54	0.46	0.49	0.49
	Q	1	0.86	0.66	0.4	0.4	1	1	0.93	0.93	0.93	1	0.13	0.8	0.8	0.6	1	1	1	1	1	1	1	1	1	1	1	0.6	0.2	0.4	0.4
	S	1	7.53	6.86	4.6	4.6	1	1.33	1.13	1.46	1.46	1	1.26	2.13	2.8	2.86	1	8.73	4.26	12.6	12.6	1	1.93	2.2	2.33	2.33	1	3.86	1.46	2.2	2.2
	Q_{20}	1	0.86	0.73	0.53	0.53	1	1	0.93	0.93	0.93	1	0.13	0.8	0.8	0.6	1	1	1	1	1	1	1	1	1	1	1	0.66	0.26	0.6	0.6
	Q_{50}	1	0.86	0.73	0.66	0.66	1	1	0.93	0.93	0.93	1	0.13	0.8	0.8	0.6	1	1	1	1	1	1	1	1	1	1	1	0.8	0.4	0.73	0.73
	d	4.4×10^{-3}	0.049	0.05	0.044	0.044	0.066	0.166	0.152	0.214	0.214	1.35×10^{-4}	4.2×10^{-3}	0.013	0.071	0.065	0	0.06	0.054	0.057	0.055	3×10^{-4}	0.262	0.175	0.138	0.138	1.9×10^{-3}	0.058	0.087	0.06	0.06
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\text{FGR}^+_{rp}\text{ET}$	T	6.43	5.18	2.71	2.37	2.36	0.28	0.27	0.31	0.32	0.32	139.34	74.26	49.58	46.59	45.97	1.28	1.18	0.98	0.72	0.72	0.28	0.26	0.22	0.18	0.18	6.91	5.81	2.32	2.04	2.04
	Q	1	0.73	0.53	0.4	0.4	1	1	1	0.93	0.93	1	0.13	0.8	0.8	0.6	1	1	1	1	1	1	1	1	1	1	1	0.6	0.33	0.4	0.4
	S	1.06	1.13	3.13	4.6	4.6	1	1	1.06	1.46	1.46	1	1.26	2.26	2.8	2.86	1	8.73	4.26	12.6	12.6	1	1.93	2.2	2.33	2.33	1	2	1.66	2.2	2.2
	Q_{20}	1	0.73	0.73	0.53	0.53	1	1	1	0.93	0.93	1	0.13	0.8	0.8	0.6	1	1	1	1	1	1	1	1	1	1	1	0.73	0.46	0.6	0.6
	Q_{50}	1	0.8	0.86	0.66	0.66	1	1	1	0.93	0.93	1	0.13	0.8	0.8	0.6	1	1	1	1	1	1	1	1	1	1	1	0.8	0.73	0.73	0.73
	d	1.1×10^{-3}	0.03	0.04	0.044	0.044	0.066	2.4×10^{-5}	0.052	0.214	0.214	1.35×10^{-4}	4.2×10^{-3}	0.023	0.072	0.066	0	0.06	0.054	0.057	0.055	3×10^{-4}	0.262	0.175	0.138	0.138	2.6×10^{-3}	0.047	0.074	0.06	0.06
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\text{FGR}^{FF}_{rp}\text{T}$	T	0.65	0.56	0.42	0.34	0.34	0.21	0.19	0.18	0.17	0.17	17.98	17.81	17.48	17.01	16.95	0.44	0.38	0.29	0.15	0.15	0.13	0.11	0.1	0.08	0.08	0.67	0.41	0.25	0.17	0.17
	Q	1	0.86	0.73	0.13	0.13	1	1	0.66	0.53	0.53	1	0.2	0.8	0.73	0.53	1	0.73	0.13	0.13	0.13	0.53	0.53	0.53	0.26	0.26	0.8	0.33	0.06	0.33	0.33
	S	1	10	9.06	3.33	3.33	1	1.73	1.13	1.13	1.13	1	1.73	2.33	2.6	2.73	16.66	12.4	2.46	3.06	3.06	2.06	1.53	1.13	1	1	1.2	2	1.46	3	3
	Q_{20}	1	0.86	0.73	0.46	0.46	1	1	0.66	0.53	0.53	1	0.2	0.8	0.73	0.53	1	0.73	0.13	0.13	0.13	0.53	0.53	0.53	0.26	0.26	0.8	0.53	0.2	0.33	0.33
	Q_{50}	1	0.86	0.73	0.53	0.53	1	1	0.66	0.53	0.53	1	0.2	0.8	0.8	0.53	1	0.73	0.6	0.6	0.6	1	1	1	1	1	0.93	0.6	0.4	0.66	0.66
	d	4.4×10^{-3}	0.061	0.048	0.053	0.053	0.066	0.333	0.355	0.332	0.332	7×10^{-5}	0.025	0.064	0.068	0.07	0.117	0.115	0.12	0.072	0.071	0.435	0.382	0.181	0.23	0.23	0.05	0.095	0.105	0.068	0.068
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\text{FGR}^{FF}_{rp}\text{ET}$	T	8.54	6.13	3.31	1.14	1.14	0.27	0.26	0.21	0.17	0.17	135.78	113.56	64.66	34.46	25.10	0.49	0.43	0.33	0.19	0.19	0.14	0.12	0.1	0.09	0.09	3.75	4.38	3.3	0.26	0.26
	Q	0.06	0.13	0.53	0.33	0.33	0.66	0.73	0.53	0.53	0.53	0.73	0.6	0.73	0.86	0.6	1	0.73	0.13	0.13	0.13	0.53	0.53	0.53	0.26	0.26	0.33	0.8	0.66	0.33	0.33
	S	1.66	1.73	3.86	3.66	3.66	1.2	1.4	1.2	1.13	1.13	2.8	4.46	5.53	4.73	4.26	16.66	12.4	2.46	3.06	3.06	2.06	1.53	1.13	1	1	2.46	9.66	10.66	3	3
	Q_{20}	0.26	0.26	0.53	0.53	0.53	0.66	0.73	0.53	0.53	0.53	0.8	0.6	0.73	0.86	0.6	1	0.73	0.13	0.13	0.13	0.53	0.53	0.53	0.26	0.26	0.4	0.8	0.73	0.33	0.33
	Q_{50}	0.6	0.66	0.6	0.53	0.53	0.66	0.73	0.53	0.53	0.53	0.93	0.6	0.73	0.86	0.6	1	0.73	0.6	0.6	0.6	1	1	1	1	1	0.46	0.8	0.86	0.66	0.66
	d	0.083	0.078	0.066	0.056	0.056	0.438	0.466	0.41	0.332	0.332	0.224	0.233	0.226	0.155	0.152	0.117	0.115	0.12	0.072	0.071	0.435	0.382	0.181	0.23	0.23	0.114	0.108	0.091	0.068	0.068
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FGR^I_{rp}	T	1	1.02	1.26	1.45	1.44	0.27	0.29	0.35	0.39	0.39	107.30	109.75	110.72	116.48	113.75	0.88	0.49	0.21	0.21	0.2	0.26	0.19	0.14	0.13	0.13	0.88	1	1.19	1.26	1.26
	Q	0.86	0.93	0.53	0.4	0.4	1	0.93	1	0.73	0.73	1	0.4	0.8	0.73	0.53	1	1	0.93	1	1	1	0.26	0.26	0.26	0.26	1	0.66	0.33	0.53	0.53
	S	2.33	4.93	3.2	4.66	4.66	1	1.53	1.53	1.06	1.06	1	3	2.26	2.6	2.66	1	7.46	2.4	4.6	4.6	1	1.53	1	1	1	1	1.66	2.53	2.86	2.86
	Q_{20}	0.86	0.93	0.73	0.6	0.6	1	0.93	1	0.73	0.73	1	0.4	0.8	0.73	0.53	1	1	0.93	1	1	1	0.26	0.26	0.26	0.26	1	0.8	0.53	0.86	0.86
	Q_{50}	0.93	0.93	0.8	0.86	0.86	1	0.93	1	0.73	0.73	1	0.4	0.8	0.73	0.53	1	1	1	1	1	1	1	1	1	1	1	0.86	0.66	0.86	0.86
	d	0.017	0.024	0.04	0.024	0.024	0.066	0.3	0.27	0.233	0.233	1.36×10^{-4}	0.041	0.09	0.068	0.063	0	0.053	0.042	0.024	0.02	4.25×10^{-4}	0.45	0.19	0.166	0.166	4.88×10^{-4}	0.051	0.08	0.026	0.026
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\text{FGR}^I_{rp}\text{E}$	T	5.97	4.44	3.59	3.68	3.67	0.31	0.36	0.41	0.43	0.43	118.46	120.98	118.37	122.9	123.36	1.28	0.9	0.63	0.63	0.64	0.28	0.22	0.16	0.16	0.16	7.61	3.12	2.8	2.82	2.83
	Q	0.93	0.73	0.53	0.4	0.4	1	1	1	0.73	0.73	1	0.4	0.93	0.73	0.53	1	1	0.93	1	1	1	0.26	0.26	0.26	0.26	1	0.73	0.26	0.53	0.53
	S	3.66	2.4	3.2	4.66	4.66	1	1.53	1.53	1.06	1.06	1	3	3.60	2.6	2.66	1	7.46	2.4	4.6	4.6	1	1.53	1	1	1	1	1.66	1.8	2.86	2.86
	Q_{20}	0.93	0.8	0.73	0.6	0.6	1	1	1	0.73	0.73	1	0.4	0.93	0.73	0.53	1	1	0.93	1	1	1	0.26	0.26	0.26	0.26	1	0.8	0.46	0.86	0.86
	Q_{50}	1	0.86	0.8	0.86	0.86	1	1	1	0.73	0.73	1	0.4	0.93	0.73	0.53	1	1	1	1	1	1	1	1	1	1	1	0.86	0.6	0.86	0.86
	d	0.016	0.034	0.04	0.024	0.024	0.066	0.266	0.269	0.233	0.233	1.36×10^{-4}	0.041	0.098	0.068	0.063	0	0.053	0.042	0.024	0.02	4.25×10^{-4}	0.45	0.19	0.166	0.166	7.58×10^{-4}	0.051	0.073	0.026	0.026
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FGR^+_{rp}	T	1.33	0.7	0.64	0.78	0.78	0.31	0.26	0.25	0.26	0.26	33.11	33.44	33.10	35.66	35.64	0.89	0.51	0.19	0.19	0.19	0.29	0.21	0.13	0.12	0.13	0.97	0.43	0.45	0.5	0.5
	Q	1	1	0.73	0.33	0.33	1	1	1	0.86	0.86	1	0.4	0.93	0.8	0.6	1	1	0.93	1	1	1	1	1	1	1	1	0.53	0.4	0.66	0.66
	S	1	11.33	7	4.13	4.13	1	1.8	1.53	1.46	1.46	1	3.06	3.66	2.73	2.8	1	7.46	2.4	4.6	4.6	1	1.93	1.46	1.4	1.4	1	5.8	2.66	4	4
	Q_{20}	1	1	0.8	0.46	0.46	1	1	1	0.86	0.86	1	0.4	0.93	0.8	0.6	1	1	0.93	1	1	1	1	1	1	1	1	0.6	0.4	0.66	0.66
	Q_{50}	1	1	0.86	0.6	0.6	1	1	1	0.86	0.86	1	0.4	0.93	0.8	0.6	1	1	1	1	1	1	1	1	1	1	1	0.66	0.53	0.8	0.8
	d	4.4×10^{-3}	0.047	0.041	0.044	0.044	0.066	0.366	0.305	0.267	0.267	1.35×10^{-4}	0.045	0.095	0.073	0.067	0	0.053	0.042	0.024	0.02	3×10^{-4}	0.262	0.068	0.014	0.014	1.9×10^{-3}	0.083	0.08	0.052	0.052
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\text{FGR}^+_{rp}\text{E}$	T	6.45	4.79	2.17	2.37	2.36	0.28	0.28	0.3	0.29	0.29	138.84	70.40	46.41	46.69	45.92	1.28	0.97	0.61	0.62	0.62	0.28	0.24	0.16	0.15	0.15	6.95	5.88	2.03	2.06	2.07
	Q	1	0.8	0.53	0.33	0.33	1	1	1	0.86	0.86	1	0.4	0.93	0.8	0.6	1	1	0.93	1	1	1	1	1	1	1	1	0.6	0.4	0.66	0.66
	S	1.06	1.13	3	4.13	4.13	1	1.6	1.53	1.46	1.46	1	3.06	3.66	2.73	2.8	1	7.46	2.4	4.6	4.6	1	1.93	1.46	1.4	1.4	1	3	2.06	4	4
	Q_{20}	1	0.8	0.66	0.46	0.46	1	1	1	0.86	0.86	1	0.4	0.93	0.8	0.6	1	1	0.93	1	1	1	1	1	1	1	1	0.73	0.4	0.66	0.66
	Q_{50}	1	0.8	0.86	0.6	0.6	1	1	1	0.86	0.86	1	0.4	0.93	0.8	0.6	1	1	1	1	1	1	1	1	1	1	1	0.8	0.46	0.8	0.8
	d	1.1×10^{-3}	0.03	0.043	0.044	0.044	0.066	0.3	0.305	0.267	0.267	1.35×10^{-4}	0.045	0.095	0.073	0.067	0	0.053	0.042	0.024	0.02	3×10^{-4}	0.262	0.068	0.014	0.014	2.6×10^{-3}	0.061	0.081	0.052	0.052
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FGR^{FF}_{rp}	T	0.65	0.54	0.44	0.41	0.44	0.21	0.2	0.19	0.19	0.18	18.21	18.27	17.04	17.19	17.07	0.44	0.26	0.1	0.09	0.08	0.13	0.1	0.08	0.06	0.08	0.67	0.35	0.19	0.27	0.19
	Q	1	1	0.73	0.06	0.06	1	1	0.66	0.53	0.53	1	0.26	0.8	0.73	0.53	1	0.66	0.26	0.2	0.2	0.53	0.53	0.4	0.33	0.33	0.8	0.4	0.33	0.33	0.33
	S	1	13.8	9.00	2.26	2.26	1	1.8	1.4	1.13	1.13	1	1.73	2.73	2.73	2.8	16.66	9.93	1.33	2.66	2.66	2.06	1.53	1.26	1	1	1.2	3.86	1.6	2.86	2.86
	Q_{20}	1	1	0.73	0.33	0.33	1	1	0.66	0.53	0.53	1	0.26	0.8	0.73	0.53	1	0.66	0.6	0.26	0.26	0.53	0.53	0.4	0.33	0.33	0.8	0.46	0.4	0.33	0.33
	Q_{50}	1	1	0.73	0.46	0.46	1	1	0.66	0.53	0.53	1	0.26	0.8	0.73	0.53	1	0.73	0.73	0.66	0.66	1	1	0.8	0.86	0.86	0.93	0.53	0.6	0.8	0.8
	d	4.4×10^{-3}	0.06	0.045	0.053	0.053	0.066	0.366	0.404	0.332	0.332	7×10^{-5}	0.031	0.086	0.088	0.083	0.117	0.112	0.093	0.07	0.068	0.435	0.382	0.248	0.198	0.198	0.05	0.103	0.091	0.067	0.067
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\text{FGR}^{FF}_{rp}\text{E}$	T	8.55	5.26	2.57	0.96	0.96	0.27	0.26	0.23	0.18	0.17	136.55	118.74	64.55	34.63	24.82	0.49	0.34	0.14	0.13	0.13	0.14	0.11	0.07	0.07	0.07	3.77	4.72	2.25	0.26	0.26
	Q	0.06	0.13	0.46	0.33	0.33	0.66	0.93	0.73	0.53	0.53	0.2	0.46	0.66	0.66	0.4	1	0.66	0.26	0.2	0.2	0.53	0.53	0.4	0.33	0.33	0.33	0.86	0.73	0.33	0.33
	S	1.66	1.46	3.4	3.53	3.53	1.2	1.6	1.53	1.13	1.13	1	2.53	4.8	4	3.60	16.66	9.93	1.33	2.66	2.66	2.06	1.53	1.26	1	1	2.46	10	8.93	2.86	2.86
	Q_{20}	0.26	0.26	0.46	0.53	0.53	0.66	0.93	0.73	0.53	0.53	0.2	0.46	0.66	0.66	0.4	1	0.66	0.6	0.26	0.26	0.53	0.53	0.4	0.33	0.33	0.4	0.86	0.8	0.33	0.33
	Q_{50}	0.6	0.6	0.53	0.53	0.53	0.66	0.93	0.73	0.53	0.53	0.6	0.53	0.66	0.73	0.46	1	0.73	0.73	0.66	0.66	1	1	0.8	0.86	0.86	0.46	0.86	0.93	0.8	0.8
	d	0.083	0.082	0.066	0.056	0.056	0.438	0.466	0.436	0.332	0.332	0.23	0.238	0.232	0.173	0.17	0.117	0.112	0.093	0.07	0.068	0.435	0.382	0.248	0.198	0.198	0.114	0.11	0.088	0.067	0.067
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table A.10: Goal recognition with random observations in FGR given the time-step

Domain		Blocks					Campus					Grid					Intrusion					Kitchen					Logistics				
Approach	$\%O$	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10
HSP^*_f	T	1080.6	729.06	529.04	405.1	405.14	1.78	1.11	0.55	0.28	0.28	224.67	144.5	79.83	41.47	37.42	588.34	414.3	214.2	4.25	4.22	694.67	243.5	60.65	33.3	33.31	44.93	42.02	18.94	9.18	9.18
	Q	1	1	0.86	0.86	0.86	1	1	1	1	1	1	0.2	0.66	0.86	0.73	1	1	1	1	1	1	1	0.86	0.86	0.86	1	0.93	0.66	0.8	0.8
	S	1.06	1.13	3.73	9.33	9.33	1	1	1	1	1	1	1.06	1.93	3.6	3.93	1	1	1.06	4.46	4.6	1	1	1.2	1.26	1.26	1.06	1.06	2.26	3.6	3.6
$g\text{HSP}^*_f$	T	531.26	446.26	379.81	357.94	357.94	1.22	0.8	0.5	0.32	0.32	114.12	79.76	52.92	39.20	38.8	447.41	281.11	151.37	3.58	3.55	480.39	171.08	49.62	37.93	37.92	36.26	32.46	14.8	7.04	7.04
	Q	1	1	1	1	1	1	1	1	1	1	1	0.2	0.8	0.93	0.8	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.06	1.13	4.06	11.46	11.46	1	1	1	1	1	1	1.2	2.26	3.86	4.2	1	1	1.06	4.46	4.6	1	1	1.33	1.4	1.4	1	1.13	2.26	3.6	3.6
FGR^IT	T	0.94	1.07	1.28	1.42	1.43	0.27	0.29	0.33	0.38	0.38	106.84	108.53	112.21	114.01	114.71	0.89	0.77	0.56	0.31	0.31	0.26	0.24	0.2	0.16	0.16	0.88	0.99	1.14	1.25	1.25
	Q	1	0.93	0.53	0.13	0.13	1	1	1	0.93	0.93	1	0.2	0.86	0.93	0.86	1	1	0.93	0.93	0.93	1	1	1	1	1	1	0.86	0.73	0.6	0.6
	S	2.33	2.33	1.53	1.8	1.8	1	1	1	1.06	1.06	1	1.93	3.2	4.2	4.2	1	1	1	4.4	4.53	1	1	1.2	1.26	1.26	1	1.13	1.73	2.8	2.8
	Q_{20}	1	0.93	0.66	0.53	0.53	1	1	1	0.93	0.93	1	0.2	0.86	0.93	0.86	1	1	1	0.93	0.93	1	1	1	1	1	1	0.93	0.8	0.73	0.73
	Q_{50}	1	1	0.8	0.86	0.86	1	1	1	0.93	0.93	1	0.2	0.86	0.93	0.86	1	1	1	1	1	1	1	1	1	1	1	0.93	0.86	0.86	0.86
	d	4.4×10^{-3}	0.016	0.035	0.022	0.022	0.066	2.4×10^{-5}	2.5×10^{-3}	0.098	0.098	4×10^{-3}	0.077	0.071	0.056	0.048	1.6×10^{-5}	2.7×10^{-3}	0.048	0.014	0.014	1.14×10^{-4}	0.011	0.03	0.058	0.058	5.8×10^{-4}	0.022	0.061	0.046	0.046
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
FGR^IET	T	12.59	8.94	4.6	3.68	3.67	0.39	0.39	0.4	0.42	0.42	127.22	125.25	124.59	123.92	124.81	1.29	1.18	0.99	0.74	0.74	0.28	0.26	0.23	0.19	0.19	8.51	4.36	3	2.83	2.85
	Q	0.93	0.73	0.46	0.13	0.13	1	1	1	0.93	0.93	1	0.2	0.86	0.93	0.86	1	1	0.93	0.93	0.93	1	1	1	1	1	0.86	0.66	0.66	0.6	0.6
	S	1.06	1.13	1.86	1.73	1.73	1	1	1	1.06	1.06	1	2	3.4	4.2	4.2	1	1	1	4.4	4.53	1	1	1.2	1.26	1.26	1.13	1.4	1.8	2.8	2.8
	Q_{20}	1	0.93	0.66	0.4	0.4	1	1	1	0.93	0.93	1	0.2	0.86	0.93	0.86	1	1	1	0.93	0.93	1	1	1	1	1	0.86	0.8	0.8	0.73	0.73
	Q_{50}	1	1	0.86	0.8	0.8	1	1	1	0.93	0.93	1	0.26	0.86	0.93	0.86	1	1	1	1	1	1	1	1	1	1	0.93	1	0.86	0.86	0.86
	d	7.4×10^{-3}	0.04	0.03	0.015	0.015	0.066	5.6×10^{-5}	6.4×10^{-3}	0.098	0.098	4×10^{-3}	0.078	0.085	0.056	0.048	1.6×10^{-5}	2.7×10^{-3}	0.048	0.014	0.014	1.14×10^{-4}	0.011	0.03	0.058	0.058	0.017	0.054	0.068	0.045	0.045
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FGR^+T	T	0.76	0.7	0.7	0.76	0.76	0.23	0.23	0.25	0.28	0.28	33.24	33.36	34.31	35.67	36.06	0.88	0.77	0.56	0.29	0.29	0.25	0.23	0.2	0.16	0.16	0.8	0.55	0.46	0.49	0.49
	Q	1	0.86	0.6	0.46	0.46	1	1	0.86	0.93	0.93	1	0.13	0.8	0.93	0.73	1	1	1	1	0.93	1	1	1	1	1	1	0.66	0.46	0.53	0.53
	S	1	7.53	5.2	2.13	2.13	1	1.33	1.06	1.13	1.13	1	1.4	2.86	3.8	4	1	1.13	1.13	4.06	3.93	1	1	1.33	1.4	1.4	1	2	1.86	3	3
	Q_{20}	1	0.86	0.66	0.66	0.66	1	1	0.86	0.93	0.93	1	0.13	0.8	0.93	0.73	1	1	1	1	0.93	1	1	1	1	1	1	0.73	0.46	0.6	0.6
	Q_{50}	1	0.86	0.66	0.73	0.73	1	1	0.86	0.93	0.93	1	0.13	0.8	0.93	0.73	1	1	1	1	1	1	1	1	1	1	1	0.86	0.66	0.73	0.73
	d	4.4×10^{-3}	0.049	0.054	0.043	0.043	0.066	0.166	0.17	0.158	0.158	1.2×10^{-3}	0.031	0.056	0.057	0.053	0.011	0.045	0.074	0.021	0.018	1×10^{-6}	0.011	2×10^{-6}	7.4×10^{-3}	7.4×10^{-3}	7.5×10^{-4}	0.042	0.075	0.053	0.053
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FGR^+ET	T	6.43	5.18	2.7	2.36	2.36	0.27	0.27	0.3	0.31	0.31	139.24	74.07	49.56	46.73	46.23	1.28	1.18	0.98	0.72	0.72	0.28	0.25	0.22	0.18	0.18	6.90	5.81	2.31	2.04	2.04
	Q	1	0.86	0.53	0.46	0.46	1	1	0.93	0.93	0.93	1	0.26	0.8	0.93	0.73	1	1	1	1	0.93	1	1	1	1	1	0.86	0.73	0.53	0.53	0.53
	S	1.06	1.13	1.53	2.13	2.13	1	1	1	1.13	1.13	1.66	2.13	3.26	4	4.06	1	1.13	1.13	4.06	3.93	1	1	1.33	1.4	1.4	1.13	1.26	2	2.86	2.86
	Q_{20}	1	0.93	0.73	0.66	0.66	1	1	0.93	0.93	0.93	1	0.26	0.8	0.93	0.73	1	1	1	1	0.93	1	1	1	1	1	0.86	1	0.6	0.6	0.6
	Q_{50}	1	0.93	0.86	0.73	0.73	1	1	0.93	0.93	0.93	1	0.26	0.8	0.93	0.73	1	1	1	1	1	1	1	1	1	1	1	1	0.86	0.73	0.73
	d	0.012	0.031	0.041	0.043	0.043	0.066	2.4×10^{-5}	0.073	0.158	0.158	0.06	0.117	0.082	0.064	0.057	0.011	0.045	0.074	0.021	0.018	1×10^{-6}	0.011	2×10^{-6}	7.4×10^{-3}	7.4×10^{-3}	0.016	0.04	0.068	0.053	0.053
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table A.11: Goal recognition with random observations in FGR not given the time-step

Domain		Blocks					Campus					Grid					Intrusion					Kitchen					Logistics				
Approach	$\%O$	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10
HSP^*_f	T	1080.6	729.06	529.04	405.1	405.14	1.78	1.11	0.55	0.28	0.28	224.67	144.5	79.83	41.47	37.42	588.34	414.3	214.2	4.25	4.22	694.67	243.5	60.65	33.3	33.31	44.93	42.02	18.94	9.18	9.18
	Q	1	1	0.86	0.86	0.86	1	1	1	1	1	1	0.2	0.66	0.86	0.73	1	1	1	1	1	1	1	0.86	0.86	0.86	1	0.93	0.66	0.8	0.8
	S	1.06	1.13	3.73	9.33	9.33	1	1	1	1	1	1	1.06	1.93	3.6	3.93	1	1	1.06	4.46	4.6	1	1	1.2	1.26	1.26	1.06	1.06	2.26	3.6	3.6
$g\text{HSP}^*_f$	T	531.26	446.26	379.81	357.94	357.94	1.22	0.8	0.5	0.32	0.32	114.12	79.76	52.92	39.20	38.8	447.41	281.11	151.37	3.58	3.55	480.39	171.08	49.62	37.93	37.92	36.26	32.46	14.8	7.04	7.04
	Q	1	1	1	1	1	1	1	1	1	1	1	0.2	0.8	0.93	0.8	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.06	1.13	4.06	11.46	11.46	1	1	1	1	1	1	1.2	2.26	3.86	4.2	1	1	1.06	4.46	4.6	1	1	1.33	1.4	1.4	1	1.13	2.26	3.6	3.6
FGR^I	T	1	1.03	1.25	1.44	1.45	0.27	0.29	0.35	0.39	0.39	107.95	109.83	110.98	115.92	115.77	0.89	0.49	0.21	0.21	0.21	0.26	0.19	0.14	0.13	0.13	0.88	1.01	1.19	1.26	1.26
	Q	1	0.93	0.46	0.13	0.13	1	1	1	0.93	0.93	1	0.26	0.86	0.93	0.86	1	1	0.93	0.93	0.93	1	1	1	1	1	1	0.86	0.53	0.6	0.6
	S	1.06	6.13	2.33	1.73	1.73	1	1.6	1.53	1.13	1.13	1	2.06	3.8	3.93	3.93	1	1	1	4.4	4.53	1	1	1.2	1.26	1.26	1	1.26	1.6	2.46	2.46
	Q_{20}	1	0.93	0.6	0.46	0.46	1	1	1	0.93	0.93	1	0.26	0.86	0.93	0.86	1	1	1	0.93	0.93	1	1	1	1	1	1	0.93	0.66	0.73	0.73
	Q_{50}	1	1	0.8	0.8	0.8	1	1	1	0.93	0.93	1	0.26	0.86	0.93	0.86	1	1	1	1	1	1	1	1	1	1	1	0.93	0.8	0.86	0.86
	d	1.1×10^{-3}	0.023	0.035	0.017	0.017	0.066	0.266	0.271	0.09	0.09	4×10^{-3}	0.071	0.121	0.049	0.041	1.6×10^{-5}	2.7×10^{-3}	0.048	0.014	0.014	1.14×10^{-4}	0.011	0.03	0.058	0.058	5.8×10^{-4}	0.025	0.073	0.043	0.043
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FGR^IE	T	12.55	8.31	6.09	3.67	3.67	0.39	0.44	0.43	0.43	0.43	127.80	130.82	121.87	126.25	125.93	1.28	0.9	0.63	0.63	0.63	0.28	0.22	0.16	0.16	0.16	8.51	3.57	3.08	2.82	2.82
	Q	0.93	0.73	0.46	0.13	0.13	1	1	1	0.93	0.93	1	0.26	0.86	0.93	0.86	1	1	0.93	0.93	0.93	1	1	1	1	1	0.86	0.66	0.6	0.6	0.6
	S	1.06	1.2	2.53	1.66	1.66	1	1	1.06	1.13	1.13	1	2.2	3.93	3.93	3.93	1	1	1	4.4	4.53	1	1	1.2	1.26	1.26	1.13	1.26	1.73	2.46	2.46
	Q_{20}	1	0.93	0.66	0.33	0.33	1	1	1	0.93	0.93	1	0.26	0.86	0.93	0.86	1	1	1	0.93	0.93	1	1	1	1	1	0.86	0.8	0.66	0.73	0.73
	Q_{50}	1	1	0.86	0.8	0.8	1	1	1	0.93	0.93	1	0.26	0.86	0.93	0.86	1	1	1	1	1	1	1	1	1	1	0.93	1	0.73	0.86	0.86
	d	7.4×10^{-3}	0.041	0.031	0.014	0.014	0.066	1.1×10^{-4}	0.042	0.09	0.09	4×10^{-3}	0.071	0.121	0.049	0.041	1.6×10^{-5}	2.7×10^{-3}	0.048	0.014	0.014	1.14×10^{-4}	0.011	0.03	0.058	0.058	0.017	0.043	0.072	0.042	0.042
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FGR^+	T	0.76	0.6	0.64	0.78	0.78	0.23	0.22	0.25	0.26	0.26	33.02	33.47	33.25	35.71	35.91	0.88	0.49	0.19	0.19	0.19	0.25	0.19	0.13	0.12	0.12	0.8	0.4	0.44	0.5	0.5
	Q	1	1	0.73	0.46	0.46	1	1	1	0.93	0.93	1	0.2	0.8	0.93	0.73	1	1	1	1	0.93	1	1	1	1	1	1	0.53	0.46	0.6	0.6
	S	1	11.33	6.6	2.06	2.06	1	1.66	1.46	1.26	1.26	1	1.53	3.33	3.66	3.86	1	1.13	1.13	4.06	3.93	1	1	1.33	1.4	1.4	1	3	2.13	2.93	2.93
	Q_{20}	1	1	0.86	0.6	0.6	1	1	1	0.93	0.93	1	0.2	0.8	0.93	0.73	1	1	1	1	0.93	1	1	1	1	1	1	0.6	0.6	0.66	0.66
	Q_{50}	1	1	0.86	0.73	0.73	1	1	1	0.93	0.93	1	0.2	0.8	0.93	0.73	1	1	1	1	1	1	1	1	1	1	1	0.66	0.73	0.8	0.8
	d	4.4×10^{-3}	0.047	0.045	0.046	0.046	0.066	0.3	0.235	0.2	0.2	1.2×10^{-3}	0.04	0.113	0.055	0.05	0.011	0.045	0.074	0.021	0.018	1×10^{-6}	0.011	2×10^{-6}	7.4×10^{-3}	7.4×10^{-3}	7.5×10^{-4}	0.071	0.077	0.052	0.052
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FGR^+E	T	6.46	4.65	2.16	2.52	2.49	0.27	0.28	0.29	0.32	0.29	139.00	70.46	46.53	46.78	46.20	1.28	0.99	0.61	0.68	0.62	0.28	0.24	0.16	0.39	0.15	6.93	5.94	2.02	2.24	2.16
	Q	1	0.8	0.53	0.46	0.46	1	1	1	0.93	0.93	0.86	0.26	0.66	0.8	0.73	1	1	1	1	0.93	1	1	1	1	1	0.86	0.73	0.53	0.6	0.6
	S	1.06	1.13	2.66	2.06	2.06	1	1.4	1.26	1.26	1.26	1.6	2.13	2.93	2.8	2.93	1	1.13	1.13	4.06	3.93	1	1	1.33	1.4	1.4	1.13	1.26	1.8	2.93	2.93
	Q_{20}	1	0.86	0.8	0.6	0.6	1	1	1	0.93	0.93	0.86	0.26	0.66	0.8	0.73	1	1	1	1	0.93	1	1	1	1	1	0.86	1	0.6	0.66	0.66
	Q_{50}	1	0.93	0.8	0.73	0.73	1	1	1	0.93	0.93	0.93	0.26	0.66	0.93	0.73	1	1	1	1	1	1	1	1	1	1	1	1	0.86	0.8	0.8
	d	0.012	0.035	0.043	0.046	0.046	0.066	0.2	0.135	0.2	0.2	0.056	0.102	0.112	0.057	0.05	0.011	0.045	0.074	0.021	0.018	1×10^{-6}	0.011	2×10^{-6}	7.4×10^{-3}	7.4×10^{-3}	0.016	0.04	0.071	0.052	0.052
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

A.2 Sequential Observations

Table A.12: Goal recognition with sequential observations in 5s

Domain		Blocks					Campus					Grid					Intrusion					Kitchen					Logistics				
Approach	%O	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10
HSP$^*_f$	T	908.52	677.66	612.36	432.56	321.31	1.77	1.03	0.66	0.45	0.27	259.53	164.18	121.34	87.53	33.94	591.93	147.18	40.68	7.21	1.38	694.32	178.74	105.86	57.87	37.56	63.54	51.24	20.34	9.46	9.47
	Q	1	1	1	0.66	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.46	0.33	0.33
	S	1.06	2.2	3.13	8.86	14.73	1	1	1	1	1.13	6.66	6.66	6.66	6.66	6.66	1	1	1.06	1.2	5	1	1	1	1.4	1.4	1	1.13	1.46	1.46	1.46
gHSP$^*_f$	T	531.02	427.78	402.37	369.21	358.32	0.94	0.58	0.41	0.31	0.23	119.23	82.18	64.6	50.97	40.64	450.05	113.91	29.56	4.56	1.24	256.74	64.89	52.46	38.93	32.49	36.3	32.48	14.82	7.05	7.04
	Q	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.06	2.2	3.13	11.86	14.73	1	1	1	1	1.13	6.66	6.66	6.66	6.66	6.66	1	1	1.06	1.2	5	1	1	1	1.4	1.66	1	1.13	2.26	3.6	3.6
LAMA	T	0.2	0.19	0.19	0.19	0.19	1.75	1.38	0.8	0.62	0.52	0.97	0.91	0.89	0.88	0.79	0.15	0.14	0.14	0.12	0.13	0.12	0.12	0.12	0.12	0.12	0.17	0.17	0.17	0.17	0.16
	Q	1	1	1	1	1	1	1	1	1	1	0.73	0.73	0.73	0.73	0.73	1	1	1	1	1	0.26	0.26	0.26	0.26	0.26	1	1	1	1	1
	S	20.26	20.26	20.26	20.26	20.26	1	1	1	1	1.13	5.33	5.33	5.33	5.33	5.33	16.66	16.66	16.66	16.66	16.66	1	1	1	1	1	15	15	15	15	15
	Q_{20}	1	1	1	1	1	1	1	1	1	1	0.73	0.73	0.73	0.73	0.73	1	1	1	1	1	0.73	0.73	0.73	0.73	0.73	1	1	1	1	1
	Q_{50}	1	1	1	1	1	1	1	1	1	1	0.73	0.73	0.73	0.73	0.73	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Solve	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LAMA$_G$	T	0.19	0.19	0.2	0.19	0.2	0.65	0.55	0.45	0.42	0.36	0.83	0.75	0.74	0.74	0.7	0.16	0.14	0.13	0.13	0.12	0.11	0.11	0.11	0.11	0.1	0.17	0.17	0.17	0.17	0.16
	Q	1	1	1	1	1	1	1	1	0.8	0.73	0.73	0.73	0.73	0.73	0.8	1	1	1	1	1	0.26	0.26	0.26	0.26	0.33	1	1	1	1	1
	S	20.26	20.26	20.26	20.26	20.26	1	1	1	1.06	1.06	5.33	5.33	5.33	5.33	5.46	16.66	16.66	16.66	16.66	16.66	1	1	1	1	1	15	15	15	15	15
	Q_{20}	1	1	1	1	1	1	1	1	0.8	0.73	0.73	0.73	0.73	0.73	0.73	1	1	1	1	1	0.73	0.73	0.73	0.73	0.73	1	1	1	1	1
	Q_{50}	1	1	1	1	1	1	1	1	1	1	0.73	0.73	0.73	0.73	0.73	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Solve	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table A.13: Goal recognition with sequential observations in 10s

Domain		Blocks					Campus					Grid					Intrusion					Kitchen					Logistics				
Approach	%O	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10
HSP$^*_f$	T	908.52	677.66	612.36	432.56	321.31	1.77	1.03	0.66	0.45	0.27	259.53	164.18	121.34	87.53	33.94	591.93	147.18	40.68	7.21	1.38	694.32	178.74	105.86	57.87	37.56	63.54	51.24	20.34	9.46	9.47
	Q	1	1	1	0.66	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.46	0.33	0.33
	S	1.06	2.2	3.13	8.86	14.73	1	1	1	1	1.13	6.66	6.66	6.66	6.66	6.66	1	1	1.06	1.2	5	1	1	1	1.4	1.4	1	1.13	1.46	1.46	1.46
gHSP$^*_f$	T	531.02	427.78	402.37	369.21	358.32	0.94	0.58	0.41	0.31	0.23	119.23	82.18	64.6	50.97	40.64	450.05	113.91	29.56	4.56	1.24	256.74	64.89	52.46	38.93	32.49	36.3	32.48	14.82	7.05	7.04
	Q	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.06	2.2	3.13	11.86	14.73	1	1	1	1	1.13	6.66	6.66	6.66	6.66	6.66	1	1	1.06	1.2	5	1	1	1	1.4	1.66	1	1.13	2.26	3.6	3.6
LAMA	T	0.2	0.19	0.19	0.2	0.19	2.72	1.4	0.8	0.62	0.52	5.94	5.68	5.62	5.52	5.10	2.10	2.04	2.02	1.82	0.95	3.33	3.17	2.47	2.35	2.33	0.17	0.17	0.17	0.16	0.17
	Q	1	1	1	1	1	1	1	1	1	1	0.53	0.6	0.66	0.6	0.66	0.8	0.8	0.8	0.8	0.8	1	1	1	1	1	1	1	1	1	1
	S	20.26	20.26	20.26	20.26	20.26	1	1	1	1	1.13	4.4	4.73	5	4.73	5	15	15	15	15	15	1.53	1	1	1.4	1.66	15	15	15	15	15
	Q_{20}	1	1	1	1	1	1	1	1	1	1	0.73	0.73	0.73	0.73	0.73	0.8	0.8	0.8	0.8	0.8	1	1	1	1	1	1	1	1	1	1
	Q_{50}	1	1	1	1	1	1	1	1	1	1	0.8	0.8	0.8	0.8	0.8	0.86	0.93	1	1	1	1	1	1	1	1	1	1	1	1	1
	Solve	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LAMA$_G$	T	0.2	0.19	0.19	0.19	0.18	0.61	0.55	0.44	0.42	0.37	4.93	4.65	4.52	4.83	4.53	0.5	0.43	0.4	0.38	0.36	0.43	0.36	0.34	0.33	0.44	0.17	0.17	0.16	0.17	0.17
	Q	1	1	1	1	1	1	1	1	0.8	0.73	0.53	0.6	0.66	0.53	0.73	0.8	0.8	0.8	0.8	0.8	1	1	1	1	1	1	1	1	1	1
	S	20.26	20.26	20.26	20.26	20.26	1	1	1	1.06	1.06	3.73	4.06	4.33	3.8	4.6	15	15	15	15	15	1.53	1.13	1	1.4	1.66	15	15	15	15	15
	Q_{20}	1	1	1	1	1	1	1	1	0.8	0.73	0.73	0.73	0.73	0.73	0.66	0.8	0.8	0.8	0.8	0.8	1	1	1	1	1	1	1	1	1	1
	Q_{50}	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.86	0.86	0.86	0.86	1	1	1	1	1	1	1	1	1	1	1
	Solve	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table A.14: Goal recognition with sequential observations in 20s

Domain		Blocks					Campus					Grid					Intrusion					Kitchen					Logistics				
Approach	%O	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10
HSP$^*_f$	T	908.52	677.66	612.36	432.56	321.31	1.77	1.03	0.66	0.45	0.27	259.52	164.18	121.34	87.53	33.94	591.93	147.18	40.68	7.21	1.38	694.32	178.74	105.86	57.87	37.56	63.54	51.24	20.34	9.46	9.47
	Q	1	1	1	0.66	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.46	0.33	0.33
	S	1.06	1.06	3.13	8.86	14.73	1	1	1	1	1.13	6.66	6.66	6.66	6.66	6.66	1	1	1.06	1.2	5	1	1	1	1.4	1.4	1	1.13	1.46	1.46	1.46
gHSP$^*_f$	T	531.02	427.78	402.37	369.21	358.32	0.94	0.58	0.41	0.31	0.23	119.23	82.18	6 4.6	50.97	40.64	450.05	113.91	29.56	4.56	1.24	256.74	64.89	52.46	38.93	32.49	36.3	32.48	14.82	7.05	7.04
	Q	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.06	2.2	3.13	11.86	14.73	1	1	1	1	1.13	6.66	6.66	6.66	6.66	6.66	1	1	1.06	1.2	5	1	1	1	1.4	1.66	1	1.13	2.26	3.6	3.6
LAMA	T	13.00	12.99	12.99	12.99	12.98	2.79	1.38	0.8	0.62	0.52	13.39	12.27	12.13	11.78	10.71	13.61	13.34	13.30	11.82	5.43	8.12	7.96	6.46	6.34	6.32	10.31	10.1	8.42	6.02	6.01
	Q	0.66	0.66	0.66	0.66	0.66	1	1	1	1	1	0.66	0.66	0.66	0.73	0.73	0.46	0.46	0.4	0.4	0.46	1	1	1	1	1	0.53	0.53	0.53	0.53	0.53
	S	16.6	16.6	16.6	16.6	16.6	1	1	1	1	1.13	4	4.06	4.06	4.33	4.2	8.86	8.26	7.73	7.66	8.26	1.53	1	1	1.4	1.66	8	8	8	8	8
	Q_{20}	0.66	0.66	0.66	0.66	0.66	1	1	1	1	1	1	1	1	1	1	0.46	0.46	0.46	0.46	0.46	1	1	1	1	1	0.53	0.53	0.53	0.53	0.53
	Q_{50}	0.66	0.66	0.66	0.66	0.66	1	1	1	1	1	1	1	1	1	1	0.46	0.46	0.46	0.46	0.46	1	1	1	1	1	0.66	0.66	0.73	0.86	0.86
	d	0.086	0.078	0.064	0.03	0.02	0	0	0	0	0	0.012	0.01	0.01	8×10^{-3}	3×10^{-3}	0.058	0.057	0.066	0.058	0.024	1.8×10^{-4}	0	2×10^{-6}	0	0	0.114	0.108	0.09	0.067	0.067
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
LAMA$_G$	T	8.58	8.57	8.57	8.58	8.87	0.63	0.52	0.45	0.42	0.37	10.41	9.29	9.1	8.78	8.37	2.31	1.89	1.76	1.65	1.52	0.42	0.36	0.34	0.33	0.7	2.93	2.72	2.54	2.46	2.46
	Q	0.4	0.4	0.4	0.4	0.4	1	1	1	0.8	0.73	0.66	0.73	0.73	0.73	0.8	0.4	0.4	0.4	0.4	0.46	1	1	1	1	1	0.53	0.53	0.53	0.53	0.53
	S	13.00	13.00	13.00	13.00	13.6	1	1	1	1.06	1.06	3.73	4.06	4.33	4.33	4.8	10	8.20	7.66	7.46	7.20	1.53	1.13	1	1.4	1.66	8	8	8	8	7.53
	Q_{20}	0.4	0.4	0.4	0.4	0.4	1	1	1	0.8	0.73	1	1	1	1	0.93	0.4	0.4	0.4	0.4	0.46	1	1	1	1	1	0.53	0.53	0.53	0.53	0.6
	Q_{50}	0.4	0.4	0.4	0.4	0.4	1	1	1	1	1	1	1	1	1	0.93	0.4	0.4	0.4	0.53	1	1	1	1	1	1	0.93	0.86	0.73	0.6	0.66
	d	0.087	0.078	0.066	0.031	0.021	3×10^{-6}	4.8×10^{-4}	5.9×10^{-3}	0.138	0.23	0.014	0.01	8×10^{-3}	8×10^{-3}	2×10^{-3}	0.06	0.06	0.072	0.076	0.024	1×10^{-3}	7.4×10^{-4}	5×10^{-3}	0.037	0	0.112	0.024	0.09	0.067	0.067
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table A.15: Goal recognition with sequential observations in 60s

Domain		Blocks					Campus					Grid					Intrusion					Kitchen					Logistics				
Approach	$\%O$	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10
HSP^*_f	T	908.52	677.66	612.36	432.56	321.31	1.77	1.03	0.66	0.45	0.27	259.52	164.18	121.34	87.53	33.94	591.93	147.18	40.68	7.21	1.38	694.32	178.74	105.86	57.87	37.56	63.54	51.24	20.34	9.46	9.47
	Q	1	1	1	0.66	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.46	0.33	0.33
	S	1.06	1.06	3.13	8.86	14.73	1	1	1	1	1.13	6.66	6.66	6.66	6.66	6.66	1	1	1.06	1.2	5	1	1	1	1.4	1.4	1	1.13	1.46	1.46	1.46
$g\text{HSP}^*_f$	T	531.02	427.78	402.37	369.21	358.32	0.94	0.58	0.41	0.31	0.23	119.23	82.18	6 4.6	50.97	40.64	450.05	113.91	29.56	4.56	1.24	256.74	64.89	52.46	38.93	32.49	36.3	32.48	14.82	7.05	7.04
	Q	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.06	2.2	3.13	11.86	14.73	1	1	1	1	1.13	6.66	6.66	6.66	6.66	6.66	1	1	1.06	1.2	5	1	1	1	1.4	1.66	1	1.13	2.26	3.6	3.6
LAMA	T	54.37	54.41	54.42	54.41	54.39	2.76	1.38	0.8	0.61	0.52	33.51	29.81	27.29	23.22	18.28	44.92	44.40	41.49	25.47	10.06	26.34	22.03	20.43	20.31	20.29	45.19	41.66	26.55	11.43	11.44
	Q	0.06	0.06	0.06	0.06	0.06	1	1	1	1	1	0.73	0.73	0.8	0.8	0.93	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.26	1.26	1.26	1.26	1.26	1	1	1	1	1.13	4.33	4.4	4.73	4.86	6.06	1.6	1.6	1.46	1.26	5	1.13	1	1	1.4	1.66	1	1.13	2.26	3.6	3.6
	Q_{20}	0.13	0.13	0.13	0.13	0.13	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	1	1
	Q_{50}	0.73	0.73	0.73	0.73	0.73	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	d	0.085	0.076	0.062	0.038	0.035	0	0	0	0	0	8×10^{-3}	6×10^{-3}	4.9×10^{-3}	2.1×10^{-3}	5.3×10^{-4}	1×10^{-6}	1.8×10^{-4}	0.013	8.1×10^{-3}	0	1.8×10^{-4}	0	0	0	0	0	2×10^{-6}	0	0	0
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
LAMA_G	T	39.63	39.61	39.62	39.62	40.43	0.61	0.52	0.44	0.41	0.36	27.58	23.69	20.54	16.25	11.68	3.35	2.63	2.39	2.19	2.02	0.42	0.35	0.34	0.32	1.63	5.53	5.03	4.58	4.38	4.49
	Q	0.06	0.06	0.06	0.06	0.06	1	1	1	0.8	0.73	0.73	0.73	0.8	0.8	1	1	0.93	1	1	1	1	1	1	1	1	1	0.8	0.4	0.46	0.46
	S	1.4	1.4	1.4	1.4	2.53	1	1	1	1.06	1.06	4.26	4.4	4.66	4.8	6.66	16.66	10	1.93	2.26	5	1.53	1.13	1	1.4	1.66	1	1.2	1.8	2.93	2.93
	Q_{20}	0.26	0.26	0.26	0.26	0.26	1	1	1	0.8	0.73	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.66	0.6	0.66
	Q_{50}	0.73	0.73	0.73	0.73	0.73	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.86	0.86
	d	0.085	0.076	0.063	0.04	0.038	3×10^{-6}	4.8×10^{-4}	5.9×10^{-3}	0.138	0.23	0.01	6×10^{-3}	6×10^{-3}	2×10^{-3}	0	6×10^{-6}	0.016	0.023	0.036	0	1×10^{-3}	7.4×10^{-4}	5×10^{-3}	0.037	0	1.3×10^{-3}	0.045	0.07	0.063	0.063
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table A.16: Goal recognition with sequential observations in 120s

Domain		Blocks					Campus					Grid					Intrusion					Kitchen					Logistics				
Approach	$\%O$	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10
HSP^*_f	T	908.52	677.66	612.36	432.56	321.31	1.77	1.03	0.66	0.45	0.27	259.52	164.18	121.34	87.53	33.94	591.93	147.18	40.68	7.21	1.38	694.32	178.74	105.86	57.87	37.56	63.54	51.24	20.34	9.46	9.47
	Q	1	1	1	0.66	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.46	0.33	0.33
	S	1.06	1.06	3.13	8.86	14.73	1	1	1	1	1.13	6.66	6.66	6.66	6.66	6.66	1	1	1.06	1.2	5	1	1	1	1.4	1.4	1	1.13	1.46	1.46	1.46
$g\text{HSP}^*_f$	T	531.02	427.78	402.37	369.21	358.32	0.94	0.58	0.41	0.31	0.23	119.23	82.18	6 4.6	50.97	40.64	450.05	113.91	29.56	4.56	1.24	256.74	64.89	52.46	38.93	32.49	36.3	32.48	14.82	7.05	7.04
	Q	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.06	2.2	3.13	11.86	14.73	1	1	1	1	1.13	6.66	6.66	6.66	6.66	6.66	1	1	1.06	1.2	5	1	1	1	1.4	1.66	1	1.13	2.26	3.6	3.6
LAMA	T	120.3	120.37	120.37	120.36	120.36	2.76	1.39	0.8	0.61	0.51	50.31	41.70	35.87	30.22	21.08	92.85	88.06	73.91	33.42	14.72	52.14	41.97	40.38	40.26	40.24	87.67	75.07	33.38	11.42	11.43
	Q	0.06	0.06	0.06	0.06	0.06	1	1	1	1	1	0.8	0.8	0.8	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.26	1.26	1.26	1.26	1.26	1	1	1	1	1.13	4.66	4.73	4.86	6.66	6.66	1.6	1	1.06	1.2	5	1	1	1	1.4	1.66	1	1.13	2.26	3.6	3.6
	Q_{20}	0.13	0.13	0.13	0.13	0.13	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	1	1
	Q_{50}	0.73	0.73	0.73	0.73	0.73	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	d	0.085	0.076	0.062	0.038	0.035	0	0	0	0	0	6×10^{-3}	4.9×10^{-3}	2.1×10^{-3}	0	0	1×10^{-6}	9.1×10^{-5}	6.9×10^{-3}	2.2×10^{-3}	0	1.8×10^{-4}	0	0	0	0	0	0	0	0	0
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
LAMA_G	T	87.32	87.32	87.38	87.39	89.14	0.61	0.5	0.44	0.4	0.36	45.81	35.12	28.69	19.70	12.14	3.34	2.63	2.39	2.19	2.03	0.42	0.36	0.33	0.32	2.34	5.53	5.02	4.57	4.39	4.48
	Q	0.06	0.06	0.06	0.06	0.06	1	1	1	0.8	0.73	0.8	0.8	0.86	1	1	1	1	0.93	1	1	1	1	1	1	1	1	0.8	0.4	0.46	0.46
	S	1.4	1.4	1.4	1.4	2.53	1	1	1	1.06	1.06	4.66	4.8	5.4	6.66	6.66	16.66	10	2	2.33	5	1.53	1.13	1	1.4	1.66	1	1.2	1.8	2.93	2.93
	Q_{20}	0.26	0.26	0.26	0.26	0.26	1	1	1	0.8	0.73	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.66	0.6	0.66
	Q_{50}	0.73	0.73	0.73	0.73	0.73	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.86	0.86
	d	0.085	0.076	0.063	0.04	0.038	3×10^{-6}	4.8×10^{-4}	5.9×10^{-3}	0.138	0.23	6×10^{-3}	2×10^{-3}	2.2×10^{-3}	0	0	6×10^{-6}	0.013	0.033	0.041	0	1×10^{-3}	7.4×10^{-4}	5×10^{-3}	0.037	0	1.3×10^{-3}	0.045	0.07	0.063	0.063
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table A.17: Goal recognition with sequential observations in 240s

Domain		Blocks					Campus					Grid					Intrusion					Kitchen					Logistics				
Approach	$\%O$	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10
HSP^*_f	T	908.52	677.66	612.36	432.56	321.31	1.77	1.03	0.66	0.45	0.27	259.52	164.18	121.34	87.53	33.94	591.93	147.18	40.68	7.21	1.38	694.32	178.74	105.86	57.87	37.56	63.54	51.24	20.34	9.46	9.47
	Q	1	1	1	0.66	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.46	0.33	0.33
	S	1.06	1.06	3.13	8.86	14.73	1	1	1	1	1.13	6.66	6.66	6.66	6.66	6.66	1	1	1.06	1.2	5	1	1	1	1.4	1.4	1	1.13	1.46	1.46	1.46
$g\text{HSP}^*_f$	T	531.02	427.78	402.37	369.21	358.32	0.94	0.58	0.41	0.31	0.23	119.23	82.18	6 4.6	50.97	40.64	450.05	113.91	29.56	4.56	1.24	256.74	64.89	52.46	38.93	32.49	36.3	32.48	14.82	7.05	7.04
	Q	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.06	2.2	3.13	11.86	14.73	1	1	1	1	1.13	6.66	6.66	6.66	6.66	6.66	1	1	1.06	1.2	5	1	1	1	1.4	1.66	1	1.13	2.26	3.6	3.6
LAMA	T	228.43	228.58	228.47	228.53	228.55	2.75	1.38	0.8	0.62	0.52	75.62	63.24	47.92	33.23	20.25	18 4.6	166.57	115.08	41.45	22.7	98.3	77.16	73.57	73.53	72.23	166.7	132.6	40.21	11.45	11.43
	Q	0.06	0.06	0.06	0.06	0.06	1	1	1	1	1	0.8	0.86	0.93	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.26	1.26	1.26	1.26	1.26	1	1	1	1	1.13	4.8	5.46	6.06	6.66	6.66	1.6	1	1.06	1.2	5	1	1	1	1.4	1.66	1	1.13	2.26	3.6	3.6
	Q_{20}	0.13	0.13	0.13	0.13	0.13	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	1	1
	Q_{50}	0.73	0.73	0.73	0.73	0.73	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	d	0.085	0.076	0.062	0.038	0.035	0	0	0	0	0	2×10^{-3}	1×10^{-3}	1×10^{-3}	0	0	1×10^{-6}	2.4×10^{-5}	6.9×10^{-3}	2.2×10^{-3}	0	1.8×10^{-4}	0	0	0	0	0	0	0	0	0
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
LAMA_G	T	184.13	184.14	184.15	184.15	187.55	0.64	0.53	0.43	0.42	0.36	67.60	48.2	33.28	19.78	12.24	3.36	2.63	2.39	2.19	2.02	0.42	0.36	0.34	0.32	2.7	5.55	5.04	4.58	4.4	4.48
	Q	0.06	0.06	0.06	0.06	0.06	1	1	1	0.8	0.73	0.73	0.86	0.93	1	1	1	0.93	1	1	1	1	1	1	1	1	1	0.8	0.4	0.46	0.46
	S	1.4	1.4	1.4	1.4	2.53	1	1	1	1.06	1.06	4.53	5.4	6.06	6.66	6.66	16.66	10	2.06	2.2	5	1.53	1.13	1	1.4	1.66	1	1.2	1.8	2.93	2.93
	Q_{20}	0.26	0.26	0.26	0.26	0.26	1	1	1	0.8	0.73	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.66	0.6	0.66
	Q_{50}	0.73	0.73	0.73	0.73	0.73	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.86	0.86
	d	0.085	0.076	0.063	0.04	0.038	3×10^{-6}	4.8×10^{-4}	5.9×10^{-3}	0.138	0.23	2×10^{-3}	3.8×10^{-3}	1×10^{-3}	0	0	6×10^{-6}	0.015	0.03	0.038	0	1×10^{-3}	7.4×10^{-4}	5×10^{-3}	0.037	0	1.3×10^{-3}	0.045	0.07	0.063	0.063
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table A.18: Goal recognition with sequential observations in 360s

Domain		Blocks					Campus					Grid					Intrusion					Kitchen					Logistics				
Approach	$\%O$	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10
	T	908.52	677.66	612.36	432.56	321.31	1.77	1.03	0.66	0.45	0.27	259.52	164.18	121.34	87.53	33.94	591.93	147.18	40.68	7.21	1.38	694.32	178.74	105.86	57.87	37.56	63.54	51.24	20.34	9.46	9.47
HSP^*_f	Q	1	1	1	0.66	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.46	0.33	0.33
	S	1.06	1.06	3.13	8.86	14.73	1	1	1	1	1.13	6.66	6.66	6.66	6.66	6.66	1	1	1.06	1.2	5	1	1	1	1.4	1.4	1	1.13	1.46	1.46	1.46
	T	531.02	427.78	402.37	369.21	358.32	0.94	0.58	0.41	0.31	0.23	119.23	82.18	6 4.6	50.97	40.64	450.05	113.91	29.56	4.56	1.24	256.74	64.89	52.46	38.93	32.49	36.3	32.48	14.82	7.05	7.04
$g\mathrm{HSP}^*_f$	Q	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.06	2.2	3.13	11.86	14.73	1	1	1	1	1.13	6.66	6.66	6.66	6.66	6.66	1	1	1.06	1.2	5	1	1	1	1.4	1.66	1	1.13	2.26	3.6	3.6
	T	333.81	333.69	333.95	333.84	334.02	2.75	1.39	0.8	0.62	0.52	86.76	73.93	56.12	34.16	20.28	274.73	233.95	140.99	49.58	30.68	131.92	95.17	93.54	93.33	86.64	240.42	180.3	44.66	11.45	11.43
	Q	0.06	0.06	0.06	0.06	0.06	1	1	1	1	1	0.93	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.26	1.26	1.26	1.26	1.26	1	1	1	1	1.13	6.06	6.66	6.66	6.66	6.66	1.6	1	1.06	1.2	5	1	1	1	1.4	1.66	1	1.13	2.26	3.6	3.6
LAMA	Q_{20}	0.13	0.13	0.13	0.13	0.13	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	1	1
	Q_{50}	0.73	0.73	0.73	0.73	0.73	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	d	0.085	0.076	0.062	0.038	0.035	0	0	0	0	0	1×10^{-3}	0	0	0	0	0	4×10^{-6}	1.7×10^{-3}	3.2×10^{-4}	0	1.8×10^{-4}	0	0	0	0	0	0	0	0	0
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	T	280.91	280.89	280.89	280.89	285.22	0.6	0.53	0.44	0.41	0.37	81.35	49.05	37.94	19.16	12.03	3.35	2.63	2.39	2.19	2.02	0.42	0.36	0.34	0.32	2.7	5.51	5.04	4.58	4.4	4.49
	Q	0.06	0.06	0.06	0.06	0	1	1	1	0.8	0.73	0.86	0.86	0.93	1	1	1	1	1	1	1	1	1	1	1	1	1	0.8	0.4	0.46	0.46
	S	1.4	1.4	1.4	1.4	1.46	1	1	1	1.06	1.06	5.46	5.46	6.06	6.66	6.66	16.66	11.26	2	2.46	5	1.53	1.13	1	1.4	1.66	1	1.2	1.8	2.93	2.93
LAMA_G	Q_{20}	0.26	0.26	0.26	0.26	0.26	1	1	1	0.8	0.73	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.66	0.6	0.66
	Q_{50}	0.73	0.73	0.73	0.73	0.73	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.86	0.86
	d	0.085	0.076	0.064	0.04	0.038	3×10^{-6}	4.8×10^{-4}	5.9×10^{-3}	0.138	0.23	1.6×10^{-3}	1×10^{-3}	1×10^{-3}	0	0	6×10^{-6}	0.017	0.027	0.043	0	1×10^{-3}	7.4×10^{-4}	5×10^{-3}	0.037	0	1.3×10^{-3}	0.045	0.07	0.063	0.063
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table A.19: Goal recognition with sequential observations in 1800s

Domain		Blocks					Campus					Grid					Intrusion					Kitchen					Logistics				
Approach	$\%O$	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10
	T	908.52	677.66	612.36	432.56	321.31	1.77	1.03	0.66	0.45	0.27	259.53	164.18	121.34	87.53	33.94	591.93	147.18	40.68	7.21	1.38	694.32	178.74	105.86	57.87	37.56	63.54	51.24	20.34	9.46	9.47
HSP^*_f	Q	1	1	1	0.66	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.46	0.33	0.33
	S	1.06	2.2	3.13	8.86	14.73	1	1	1	1	1.13	6.66	6.66	6.66	6.66	6.66	1	1	1.06	1.2	5	1	1	1	1.4	1.4	1	1.13	1.46	1.46	1.46
	T	531.02	427.78	402.37	369.21	358.32	0.94	0.58	0.41	0.31	0.23	119.23	82.18	64.6	50.97	40.64	450.05	113.91	29.56	4.56	1.24	256.74	64.89	52.46	38.93	32.49	36.3	32.48	14.82	7.05	7.04
$g\mathrm{HSP}^*_f$	Q	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.06	2.2	3.13	11.86	14.73	1	1	1	1	1.13	6.66	6.66	6.66	6.66	6.66	1	1	1.06	1.2	5	1	1	1	1.4	1.66	1	1.13	2.26	3.6	3.6
	T	1596.43	1471.80	1263.15	1085.34	1037.39	2.8	1.39	0.8	0.62	0.51	110.11	69.85	56.61	32.84	20.21	1302.52	551.54	213.17	104.16	55.12	357.29	164.37	133.43	107.19	97.92	772.77	400.97	50.44	11.43	11.45
	Q	0.73	0.73	0.86	0.93	0.93	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.06	2.13	2.86	11	13.73	1	1	1	1	1.13	6.66	6.66	6.66	6.66	6.66	1	1	1.06	1.2	5	1	1	1	1.4	1.66	1	1.13	2.26	3.6	3.6
LAMA	Q_{20}	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	1	1
	Q_{50}	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	d	0.026	0.021	5×10^{-3}	2.1×10^{-3}	1.1×10^{-3}	0	0	0	0	0	0	0	0	0	0	0	0	3.6×10^{-5}	0	0	1.8×10^{-4}	0	0	0	0	0	0	0	0	0
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	T	860.44	846.22	825.27	817.13	785.3	0.62	0.54	0.44	0.41	0.37	84.9	47.99	36.36	19.21	11.76	3.35	2.63	2.4	2.19	2.02	0.42	0.36	0.34	0.32	2.69	5.53	5.05	4.57	4.39	4.49
	Q	0.66	0.86	0.6	0.4	0.46	1	1	1	0.8	0.73	1	1	1	1	1	1	1	0.93	1	1	1	1	1	1	1	1	0.8	0.4	0.46	0.46
	S	1	1.53	1.6	3.73	5.86	1	1	1	1.06	1.06	6.66	6.66	6.66	6.66	6.66	16.66	10	1.86	2.33	5	1.53	1.13	1	1.4	1.66	1	1.2	1.8	2.93	2.93
LAMA_G	Q_{20}	1	1	1	1	1	1	1	1	0.8	0.73	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.66	0.6	0.66
	Q_{50}	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.86	0.86
	d	0.026	0.018	0.028	0.017	0.016	3×10^{-6}	4.8×10^{-4}	5.9×10^{-3}	0.138	0.23	0	0	0	0	0	8×10^{-6}	0.013	0.026	0.041	0	1×10^{-3}	7.4×10^{-4}	5×10^{-3}	0.037	0	1.3×10^{-3}	0.045	0.07	0.063	0.063
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table A.20: Goal recognition with sequential observations for relaxed plan heuristics

Domain		Blocks					Campus					Grid					Intrusion					Kitchen					Logistics				
Approach	$\%O$	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10
HSP_f^*	T	908.52	677.66	612.36	432.56	321.31	1.77	1.03	0.66	0.45	0.27	259.53	164.18	121.34	87.53	33.94	591.93	147.18	40.68	7.21	1.38	694.32	178.74	105.86	57.87	37.56	63.54	51.24	20.34	9.46	9.47
	Q	1	1	1	0.66	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.46	0.33	0.33
	S	1.06	2.2	3.13	8.86	14.73	1	1	1	1	1.13	6.66	6.66	6.66	6.66	6.66	1	1	1.06	1.2	5	1	1	1	1.4	1.4	1	1.13	1.46	1.46	1.46
$g\text{HSP}_f^*$	T	531.02	427.78	402.37	369.21	358.32	0.94	0.58	0.41	0.31	0.23	119.23	82.18	64.6	50.97	40.64	450.05	113.91	29.56	4.56	1.24	256.74	64.89	52.46	38.93	32.49	36.3	32.48	14.82	7.05	7.04
	Q	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.06	2.2	3.13	11.86	14.73	1	1	1	1	1.13	6.66	6.66	6.66	6.66	6.66	1	1	1.06	1.2	5	1	1	1	1.4	1.66	1	1.13	2.26	3.6	3.6
h_s^a	T	0.43	0.39	0.39	0.36	0.36	0.05	0.04	0.03	0.03	0.03	0.42	0.32	0.28	0.25	0.24	0.47	0.32	0.3	0.26	0.26	0.06	0.05	0.05	0.04	0.04	0.35	0.32	0.3	0.31	0.3
	Q	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.93	0.93
	S	20.26	20.26	20.26	20.26	20.26	2	2	2	2	2	6.66	6.66	6.66	6.66	6.66	16.66	16.06	16.66	16.66	16.66	3	3	3	3	3	15	15	14.8	14.66	14.66
	Q_{20}	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.93	0.93
	Q_{50}	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.93	0.93
	d	0.086	0.08	0.055	0.031	0.031	0.466	0.466	0.495	0.347	0.347	0.252	0.238	0.208	0.132	0.126	0.117	0.117	0.114	0.071	0.07	0.414	0.407	0.274	0.216	0.216	0.115	0.11	0.093	0.068	0.068
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\text{FGR}_{rp}^{I}\text{T}$	T	1	1.12	1.23	1.41	1.44	0.27	0.32	0.34	0.38	0.41	112.2	116.52	118.15	120.2	120.9	0.88	0.49	0.3	0.2	0.21	0.26	0.19	0.16	0.13	0.13	0.88	0.98	1.15	1.25	1.25
	Q	0.86	0.73	0.66	0.26	0.2	1	1	1	1	0.93	1	0.93	0.93	0.93	0.8	1	1	1	1	1	1	0.26	0.26	0.26	0.26	1	0.86	0.53	0.33	0.33
	S	2.33	3.86	3.26	4.33	5.66	1	1	1	1	1.13	1	1.6	1.53	2.4	3.4	1	5.46	9.06	4.8	5.13	1	1.53	1.53	1	1	1	1.33	2.06	2.33	2.33
	Q_{20}	0.86	0.73	0.66	0.46	0.46	1	1	1	1	0.93	1	1	0.93	0.93	0.8	1	1	1	1	1	1	0.26	0.26	0.26	0.26	1	0.93	0.73	0.8	0.8
	Q_{50}	0.93	1	1	0.93	1	1	1	1	1	0.93	1	1	1	1	0.8	1	1	1	1	1	1	1	1	1	1	1	0.93	0.73	0.86	0.86
	d	0.017	0.03	0.044	0.04	0.021	2×10^{-6}	1.8×10^{-4}	1.4×10^{-3}	0.024	0.133	0.25	0.21	0.193	0.143	0.053	0	0.042	0.085	0.072	4.4×10^{-3}	6×10^{-4}	0.436	0.361	0.164	0.183	4.88×10^{-4}	0.026	0.061	0.05	0.05
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\text{FGR}_{rp}^{I}\text{ET}$	T	6.00	4.4	3.64	3.64	3.61	0.31	0.36	0.39	0.42	0.44	125.25	126.61	127.25	129.48	129.93	1.29	0.9	0.72	0.63	0.63	0.28	0.22	0.18	0.16	0.16	7.61	3.72	2.81	2.83	2.83
	Q	0.93	0.73	0.6	0.26	0.2	1	1	1	1	0.93	1	0.93	0.93	0.93	0.8	1	1	1	1	1	1	0.26	0.26	0.26	0.26	1	0.66	0.53	0.33	0.33
	S	3.66	1.4	2.06	4.33	5.66	1	1	1	1	1.13	1.06	1.66	1.53	2.4	3.4	1	5.46	9.06	4.8	5.13	1	1.53	1.53	1	1	1	1.26	2.06	2.26	2.26
	Q_{20}	0.93	0.73	0.66	0.46	0.46	1	1	1	1	0.93	1	1	0.93	0.93	0.8	1	1	1	1	1	1	0.26	0.26	0.26	0.26	1	0.73	0.73	0.8	0.8
	Q_{50}	1	0.86	1	0.93	1	1	1	1	1	0.93	1	1	1	1	0.8	1	1	1	1	1	1	1	1	1	1	1	1	0.73	0.86	0.86
	d	0.016	0.026	0.046	0.04	0.021	2×10^{-6}	1.8×10^{-4}	1.5×10^{-3}	0.023	0.133	0.25	0.21	0.193	0.143	0.053	0	0.042	0.085	0.072	4.4×10^{-3}	6×10^{-4}	0.436	0.361	0.164	0.183	7.58×10^{-4}	0.038	0.061	0.044	0.044
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\text{FGR}_{rp}^{+}\text{T}$	T	0.76	0.59	0.61	0.75	0.78	0.23	0.24	0.26	0.25	0.27	33.03	33.42	34.14	35.44	36.03	0.88	0.49	0.3	0.19	0.19	0.25	0.19	0.15	0.12	0.12	0.8	0.54	0.46	0.49	0.49
	Q	1	1	0.73	0.2	0.13	1	1	1	1	1	1	0.93	0.93	0.93	0.86	1	1	1	1	1	1	1	1	1	1	1	0.6	0.2	0.4	0.4
	S	1	11.6	5.66	3.66	5.53	1	1.2	1	1.26	1.33	1	1.53	1.6	2.4	3.66	1	5.46	9.06	4.8	5.13	1	1.93	1.93	1.66	1.66	1	3.86	1.46	2.2	2.2
	Q_{20}	1	1	0.73	0.46	0.33	1	1	1	1	1	1	0.93	0.93	0.93	0.86	1	1	1	1	1	1	1	1	1	1	1	0.66	0.26	0.6	0.6
	Q_{50}	1	1	1	0.73	0.66	1	1	1	1	1	1	1	0.93	1	0.86	1	1	1	1	1	1	1	1	1	1	1	0.8	0.4	0.73	0.73
	d	4.4×10^{-3}	0.041	0.04	0.048	0.038	2×10^{-6}	0.1	3.6×10^{-3}	0.158	0.092	0.25	0.211	0.196	0.137	0.048	0	0.042	0.085	0.072	4.4×10^{-3}	1.6×10^{-4}	0.248	0.22	0.094	0	1.9×10^{-3}	0.058	0.087	0.06	0.06
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\text{FGR}_{rp}^{+}\text{ET}$	T	6.45	3.64	2.85	2.35	2.32	0.27	0.27	0.3	0.28	0.3	137.71	52.39	46.36	45.28	45.14	1.29	0.9	0.72	0.61	0.62	0.28	0.21	0.18	0.15	0.15	6.91	5.83	2.32	2.04	2.04
	Q	1	0.73	0.53	0.2	0.13	1	1	1	1	1	1	0.93	0.93	0.93	0.86	1	1	1	1	1	1	1	1	1	1	1	0.6	0.33	0.4	0.4
	S	1.06	1.4	2	3.6	5.53	1	1.2	1	1.26	1.33	1.13	1.53	1.4	2.4	3.66	1	5.46	9.06	4.8	5.13	1	1.93	1.93	1.66	1.66	1	2	1.66	2.2	2.2
	Q_{20}	1	0.73	0.6	0.46	0.33	1	1	1	1	1	1	1	0.93	0.93	0.86	1	1	1	1	1	1	1	1	1	1	1	0.73	0.46	0.6	0.6
	Q_{50}	1	0.93	0.73	0.73	0.66	1	1	1	1	1	1	1	0.93	1	0.86	1	1	1	1	1	1	1	1	1	1	1	0.8	0.73	0.73	0.73
	d	1.1×10^{-3}	0.03	0.042	0.05	0.038	2×10^{-6}	0.1	3.6×10^{-3}	0.158	0.092	0.25	0.211	0.187	0.135	0.048	0	0.042	0.085	0.072	4.4×10^{-3}	1.6×10^{-4}	0.248	0.22	0.094	0	2.6×10^{-3}	0.047	0.074	0.06	0.06
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\text{FGR}_{rp}^{FF}\text{T}$	T	0.65	0.41	0.36	0.33	0.33	0.21	0.17	0.17	0.17	0.17	18.01	17.22	16.99	16.73	16.53	0.44	0.26	0.17	0.08	0.08	0.13	0.09	0.08	0.06	0.06	0.67	0.41	0.25	0.17	0.17
	Q	1	1	0.8	0.2	0.06	1	1	0.73	0.4	0.4	1	1	0.93	0.93	0.8	1	0.73	0.13	0.13	0.13	0.53	0.53	0.53	0.53	0.53	0.8	0.33	0.06	0.33	0.33
	S	1	14	8.93	4.13	1.86	1	1.4	1.2	1	1	1	2.2	1.73	2.46	3.53	16.66	12.4	2.26	2.26	2.26	2.06	1.53	1.53	1	1	1.2	2	1.46	3	3
	Q_{20}	1	1	0.8	0.6	0.46	1	1	0.73	0.4	0.4	1	1	0.93	0.93	0.8	1	0.73	0.13	0.13	0.13	0.53	0.53	0.53	0.53	0.53	0.8	0.53	0.2	0.33	0.33
	Q_{50}	1	1	0.8	0.66	0.53	1	1	0.73	0.4	0.4	1	1	0.93	0.93	0.86	1	0.73	0.6	0.6	0.6	1	1	1	1	1	0.93	0.6	0.4	0.66	0.66
	d	4.4×10^{-3}	0.053	0.046	0.048	0.048	2×10^{-6}	0.2	0.238	0.375	0.218	0.25	0.197	0.198	0.14	0.054	0.117	0.115	0.113	0.105	0.051	0.435	0.373	0.331	0.103	0.155	0.05	0.095	0.105	0.068	0.068
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\text{FGR}_{rp}^{FF}\text{ET}$	T	8.59	4.62	3.09	1.09	0.45	0.27	0.23	0.21	0.17	0.17	136.64	74.08	46.45	22.60	18.48	0.49	0.31	0.21	0.13	0.13	0.14	0.1	0.08	0.07	0.07	3.75	4.38	3.29	0.26	0.26
	Q	0.06	0.13	0.2	0.2	0.06	0.66	0.66	0.66	0.4	0.4	0.86	0.86	0.86	0.86	0.8	1	0.73	0.13	0.13	0.13	0.53	0.53	0.53	0.53	0.53	0.33	0.8	0.66	0.33	0.33
	S	1.66	1.46	2.6	3.4	1.86	1.2	1.2	1.4	1	1	4.46	5.73	4.66	3.73	3.53	16.66	12.4	2.26	2.26	2.26	2.06	1.53	1.53	1	1	2.46	9.66	10.66	3	3
	Q_{20}	0.26	0.2	0.26	0.46	0.46	0.66	0.66	0.66	0.4	0.4	0.93	0.86	0.86	0.86	0.8	1	0.73	0.13	0.13	0.13	0.53	0.53	0.53	0.53	0.53	0.4	0.8	0.73	0.33	0.33
	Q_{50}	0.6	0.53	0.53	0.53	0.53	0.66	0.66	0.66	0.4	0.4	0.93	0.86	0.86	0.86	0.86	1	0.73	0.6	0.6	0.6	1	1	1	1	1	0.46	0.8	0.86	0.66	0.66
	d	0.083	0.08	0.071	0.051	0.048	0.438	0.466	0.404	0.375	0.218	0.054	0.017	0.031	0.047	0.054	0.117	0.115	0.113	0.105	0.051	0.435	0.373	0.331	0.103	0.155	0.114	0.108	0.091	0.068	0.068
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FGR_{rp}^{I}	T	1	1.12	1.22	1.41	1.45	0.27	0.32	0.34	0.38	0.41	112.45	117.33	118.13	119.76	120.72	0.88	0.49	0.3	0.2	0.2	0.26	0.19	0.16	0.13	0.13	0.88	1.01	1.19	1.26	1.27
	Q	0.86	0.73	0.66	0.26	0.2	1	1	1	1	0.93	1	0.93	0.93	0.93	0.8	1	1	1	1	1	1	0.26	0.26	0.26	0.26	1	0.66	0.33	0.53	0.53
	S	2.33	3.86	3.26	4.33	5.66	1	1	1	1	1.13	1	1.53	1.86	2.53	3.4	1	5.46	9.06	4.8	5.13	1	1.53	1.53	1	1	1	1.66	2.53	2.86	2.86
	Q_{20}	0.86	0.73	0.66	0.46	0.46	1	1	1	1	0.93	1	1	0.93	0.93	0.8	1	1	1	1	1	1	0.26	0.26	0.26	0.26	1	0.8	0.53	0.86	0.86
	Q_{50}	0.93	1	1	0.93	1	1	1	1	1	0.93	1	1	1	1	0.8	1	1	1	1	1	1	1	1	1	1	1	0.86	0.66	0.86	0.86
	d	0.017	0.03	0.044	0.04	0.021	2×10^{-6}	1.8×10^{-4}	1.4×10^{-3}	0.024	0.133	0.25	0.21	0.193	0.143	0.053	0	0.042	0.085	0.072	4.4×10^{-3}	6×10^{-4}	0.436	0.361	0.164	0.183	4.88×10^{-4}	0.051	0.08	0.026	0.026
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\text{FGR}_{rp}^{I}\text{E}$	T	6.04	4.35	3.63	3.63	3.59	0.31	0.37	0.39	0.42	0.44	126.60	127.23	127.41	128.86	130.02	1.29	0.9	0.72	0.63	0.63	0.28	0.22	0.18	0.16	0.16	7.73	3.12	2.81	2.82	2.82
	Q	0.93	0.73	0.6	0.26	0.2	1	1	1	1	0.93	1	0.93	0.93	0.93	0.8	1	1	1	1	1	1	0.26	0.26	0.26	0.26	1	0.73	0.26	0.53	0.53
	S	3.66	1.4	2.06	4.33	5.66	1	1	1	1	1.13	1	1.53	1.86	2.53	3.4	1	5.46	9.06	4.8	5.13	1	1.53	1.53	1	1	1	1.66	1.8	2.86	2.86
	Q_{20}	0.93	0.73	0.66	0.46	0.46	1	1	1	1	0.93	1	1	0.93	0.93	0.8	1	1	1	1	1	1	0.26	0.26	0.26	0.26	1	0.8	0.46	0.86	0.86
	Q_{50}	1	0.86	1	0.93	1	1	1	1	1	0.93	1	1	1	1	0.8	1	1	1	1	1	1	1	1	1	1	1	0.86	0.6	0.86	0.86
	d	0.016	0.026	0.046	0.04	0.021	2×10^{-6}	1.8×10^{-4}	1.5×10^{-3}	0.023	0.133	0.25	0.21	0.193	0.143	0.053	0	0.042	0.085	0.072	4.4×10^{-3}	6×10^{-4}	0.436	0.361	0.164	0.183	7.58×10^{-4}	0.051	0.073	0.026	0.026
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FGR_{rp}^{+}	T	0.85	0.6	0.61	0.75	0.78	0.23	0.24	0.27	0.25	0.27	33.27	33.74	34.36	35.50	36.11	0.93	0.49	0.3	0.19	0.19	0.26	0.19	0.15	0.12	0.12	0.9	0.4	0.45	0.5	0.5
	Q	1	1	0.73	0.2	0.13	1	1	1	1	1	1	0.93	0.93	0.93	0.86	1	1	1	1	1	1	1	1	1	1	1	0.53	0.4	0.66	0.66
	S	1	11.6	5.66	3.66	5.53	1	1.2	1	1.26	1.33	1	1.53	1.86	2.53	3.66	1	5.46	9.06	4.8	5.13	1	1.93	1.93	1.66	1.66	1	5.8	2.66	4	4
	Q_{20}	1	1	0.73	0.46	0.33	1	1	1	1	1	1	0.93	0.93	0.93	0.86	1	1	1	1	1	1	1	1	1	1	1	0.6	0.4	0.66	0.66
	Q_{50}	1	1	1	0.73	0.66	1	1	1	1	1	1	1	0.93	1	0.86	1	1	1	1	1	1	1	1	1	1	1	0.66	0.53	0.8	0.8
	d	4.4×10^{-3}	0.041	0.04	0.048	0.038	2×10^{-6}	0.1	3.6×10^{-3}	0.158	0.092	0.25	0.211	0.196	0.137	0.048	0	0.042	0.085	0.072	4.4×10^{-3}	1.6×10^{-4}	0.248	0.22	0.094	0	1.9×10^{-3}	0.083	0.08	0.052	0.052
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\text{FGR}_{rp}^{+}\text{E}$	T	6.46	3.66	2.85	2.35	2.31	0.29	0.27	0.3	0.28	0.3	138.71	52.95	46.69	45.37	45.32	1.31	0.9	0.72	0.61	0.62	0.28	0.21	0.18	0.15	0.15	7.59	5.22	2.03	2.07	2.07
	Q	1	0.73	0.53	0.2	0.13	1	1	1	1	1	1	0.93	0.93	0.93	0.86	1	1	1	1	1	1	1	1	1	1	1	0.6	0.4	0.66	0.66
	S	1.06	1.4	2	3.6	5.53	1	1.2	1	1.26	1.33	1	1.53	1.86	2.53	3.66	1	5.46	9.06	4.8	5.13	1	1.93	1.93	1.66	1.66	1	3	2.06	4	4
	Q_{20}	1	0.73	0.6	0.46	0.33	1	1	1	1	1	1	1	0.93	0.93	0.86	1	1	1	1	1	1	1	1	1	1	1	0.73	0.4	0.66	0.66
	Q_{50}	1	0.93	0.73	0.73	0.66	1	1	1	1	1	1	1	0.93	1	0.86	1	1	1	1	1	1	1	1	1	1	1	0.8	0.46	0.8	0.8
	d	1.1×10^{-3}	0.03	0.042	0.05	0.038	2×10^{-6}	0.1	3.6×10^{-3}	0.158	0.092	0.25	0.211	0.187	0.135	0.048	0	0.042	0.085	0.072	4.4×10^{-3}	1.6×10^{-4}	0.248	0.22	0.094	0	2.6×10^{-3}	0.061	0.081	0.052	0.052
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FGR_{rp}^{FF}	T	0.65	0.42	0.41	0.35	0.39	0.21	0.18	0.25	0.18	0.2	18.33	17.31	17.03	16.84	16.58	0.44	0.29	0.18	0.09	0.09	0.13	0.11	0.08	0.07	0.07	0.67	0.31	0.19	0.22	0.26
	Q	1	1	0.8	0.2	0.06	1	1	0.73	0.4	0.4	1	1	0.93	0.93	0.8	1	0.73	0.13	0.13	0.13	0.53	0.53	0.53	0.53	0.53	0.8	0.4	0.33	0.33	0.33
	S	1	14	8.93	4.13	1.86	1	1.4	1.2	1	1	1	2.2	1.73	2.46	3.53	16.66	12.4	2.26	2.26	2.26	2.06	1.53	1.53	1	1	1.2	3.86	1.6	2.86	2.86
	Q_{20}	1	1	0.8	0.6	0.46	1	1	0.73	0.4	0.4	1	1	0.93	0.93	0.8	1	0.73	0.13	0.13	0.13	0.53	0.53	0.53	0.53	0.53	0.8	0.46	0.4	0.33	0.33
	Q_{50}	1	1	0.8	0.66	0.53	1	1	0.73	0.4	0.4	1	1	0.93	0.93	0.86	1	0.73	0.6	0.6	0.6	1	1	1	1	1	0.93	0.53	0.6	0.8	0.8
	d	4.4×10^{-3}	0.053	0.046	0.048	0.048	2×10^{-6}	0.2	0.238	0.375	0.218	0.25	0.197	0.198	0.14	0.054	0.117	0.115	0.113	0.105	0.051	0.435	0.373	0.331	0.103	0.155	0.05	0.103	0.091	0.067	0.067
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\text{FGR}_{rp}^{FF}\text{E}$	T	8.58	5.17	3.09	1.1	0.45	0.27	0.25	0.21	0.17	0.17	136.89	74.76	46.55	22.72	18.56	0.49	0.31	0.22	0.13	0.13	0.14	0.1	0.08	0.07	0.07	3.78	4.79	2.26	0.26	0.26
	Q	0.06	0.13	0.2	0.2	0.06	0.66	0.66	0.66	0.4	0.4	0.33	0.26	0.46	0.66	0.8	1	0.73	0.13	0.13	0.13	0.53	0.53	0.53	0.53	0.53	0.33	0.86	0.73	0.33	0.33
	S	1.66	1.46	2.6	3.4	1.86	1.2	1.2	1.4	1	1	1.66	1.53	1.8	2.53	3.53	16.66	12.4	2.26	2.26	2.26	2.06	1.53	1.53	1	1	2.46	10	8.93	2.86	2.86
	Q_{20}	0.26	0.2	0.26	0.46	0.46	0.66	0.66	0.66	0.4	0.4	0.4	0.33	0.53	0.66	0.8	1	0.73	0.13	0.13	0.13	0.53	0.53	0.53	0.53	0.53	0.4	0.86	0.8	0.33	0.33
	Q_{50}	0.6	0.53	0.53	0.53	0.53	0.66	0.66	0.66	0.4	0.4	0.66	0.6	0.66	0.8	0.86	1	0.73	0.6	0.6	0.6	1	1	1	1	1	0.46	0.86	0.93	0.8	0.8
	d	0.083	0.08	0.071	0.051	0.048	0.438	0.466	0.404	0.375	0.218	0.067	0.04	0.047	0.054	0.054	0.117	0.115	0.113	0.105	0.051	0.435	0.373	0.331	0.103	0.155	0.114	0.11	0.088	0.067	0.067
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table A.21: Goal recognition with sequential observations in FGR given the time-step

Domain		Blocks					Campus					Grid					Intrusion					Kitchen					Logistics				
Approach	%*O*	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10
HSP$^*_f$	*T*	908.52	677.66	612.36	432.56	321.31	1.77	1.03	0.66	0.45	0.27	259.53	164.18	121.34	87.53	33.94	591.93	147.18	40.68	7.21	1.38	694.32	178.74	105.86	57.87	37.56	63.54	51.24	20.34	9.46	9.47
	Q	1	1	1	0.66	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.46	0.33	0.33
	S	1.06	2.2	3.13	8.86	14.73	1	1	1	1	1.13	6.66	6.66	6.66	6.66	6.66	1	1	1.06	1.2	5	1	1	1	1.4	1.4	1	1.13	1.46	1.46	1.46
*g*HSP$^*_f$	*T*	531.02	427.78	402.37	369.21	358.32	0.94	0.58	0.41	0.31	0.23	119.23	82.18	64.6	50.97	40.64	450.05	113.91	29.56	4.56	1.24	256.74	64.89	52.46	38.93	32.49	36.3	32.48	14.82	7.05	7.04
	Q	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.06	2.2	3.13	11.86	14.73	1	1	1	1	1.13	6.66	6.66	6.66	6.66	6.66	1	1	1.06	1.2	5	1	1	1	1.4	1.66	1	1.13	2.26	3.6	3.6
FGR^{I}T	*T*	1	1.12	1.22	1.41	1.44	0.27	0.32	0.34	0.38	0.41	112.26	117.05	118.43	120.41	121.15	0.88	0.49	0.3	0.2	0.2	0.26	0.19	0.16	0.13	0.13	0.88	0.99	1.15	1.25	1.25
	Q	1	0.86	0.8	0.2	0.06	1	1	1	1	1	1	1	1	0.86	1	1	0.93	0.93	1	1	1	1	1	1	0.73	1	0.86	0.73	0.6	0.6
	S	1.06	3.73	2.46	2.06	1.46	1	1	1	1	1.13	1	1.93	2.13	2.73	4.46	1	1	1	1.2	5	1	1	1	1.4	1.4	1	1.13	1.73	2.8	2.8
	Q_{20}	1	0.93	0.93	0.46	0.33	1	1	1	1	1	1	1	1	0.86	1	1	1	0.93	1	1	1	1	1	1	0.73	1	0.93	0.8	0.73	0.73
	Q_{50}	1	0.93	0.93	0.8	0.66	1	1	1	1	1	1	1	1	0.93	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.86	0.86	0.86
	d	1.1×10^{-3}	0.024	0.028	0.021	0.015	2×10^{-6}	1.8×10^{-4}	1.4×10^{-3}	0.022	0.14	0.218	0.188	0.155	0.094	0.03	1.6×10^{-5}	4.3×10^{-3}	3×10^{-3}	1.64×10^{-4}	0	6.6×10^{-5}	6.1×10^{-3}	0.026	0.034	0.06	5.8×10^{-4}	0.022	0.061	0.046	0.046
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FGR^{I}ET	*T*	12.74	5.91	4.42	3.63	3.59	0.39	0.39	0.4	0.42	0.44	132.34	128.18	127.55	129.55	130.8	1.29	0.9	0.72	0.63	0.64	0.28	0.22	0.18	0.16	0.16	8.53	4.36	3.02	2.83	2.83
	Q	0.93	0.86	0.8	0.2	0.06	1	1	1	1	1	1	1	1	0.86	1	1	0.93	0.93	1	1	1	1	1	1	0.73	0.86	0.66	0.66	0.6	0.6
	S	1.06	1.4	1.33	2	1.4	1	1	1	1	1.13	1	1.93	2.13	2.73	4.46	1	1	1	1.2	5	1	1	1	1.4	1.4	1.13	1.4	1.8	2.8	2.8
	Q_{20}	1	0.93	0.93	0.4	0.13	1	1	1	1	1	1	1	1	0.86	1	1	1	0.93	1	1	1	1	1	1	0.73	0.86	0.8	0.8	0.73	0.73
	Q_{50}	1	0.93	1	0.86	0.66	1	1	1	1	1	1	1	1	0.93	1	1	1	1	1	1	1	1	1	1	1	0.93	1	0.86	0.86	0.86
	d	7.4×10^{-3}	0.023	0.021	0.014	0.011	1×10^{-6}	6.8×10^{-5}	6.18×10^{-4}	0.022	0.14	0.217	0.185	0.152	0.086	0.03	1.6×10^{-5}	4.3×10^{-3}	3×10^{-3}	1.64×10^{-4}	0	6.6×10^{-5}	6.1×10^{-3}	0.026	0.034	0.06	0.017	0.054	0.068	0.045	0.045
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FGR$^+$T	*T*	0.76	0.59	0.61	0.75	0.78	0.23	0.24	0.26	0.25	0.27	32.93	33.4	34.13	35.29	36.03	0.88	0.49	0.3	0.19	0.19	0.25	0.19	0.15	0.12	0.12	0.8	0.54	0.46	0.49	0.49
	Q	1	0.93	0.66	0.46	0.4	1	1	1	1	1	1	1	1	1	0.86	1	1	0.93	0.93	0.93	1	1	1	1	1	1	0.66	0.46	0.53	0.53
	S	1	11.53	5	1.66	1.6	1	1.2	1	1.2	1.2	1	2	2.86	4.4	4.2	1	1	1.2	1.06	3.66	1	1	1	1.4	1.66	1	2	1.86	3	3
	Q_{20}	1	0.93	0.86	0.8	0.66	1	1	1	1	1	1	1	1	1	0.86	1	1	1	1	1	1	1	1	1	1	1	0.73	0.46	0.6	0.6
	Q_{50}	1	1	0.86	0.86	0.86	1	1	1	1	1	1	1	1	1	0.86	1	1	1	1	1	1	1	1	1	1	1	0.86	0.66	0.73	0.73
	d	4.4×10^{-3}	0.045	0.042	0.05	0.055	2×10^{-6}	0.1	9.37×10^{-4}	0.126	0.093	0.25	0.2	0.176	0.101	0.056	0.011	0.034	0.043	0.055	0.018	1.79×10^{-4}	1×10^{-6}	1×10^{-6}	1×10^{-6}	0	7.5×10^{-4}	0.042	0.075	0.053	0.053
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FGR$^+$ET	*T*	6.42	3.64	2.84	2.35	2.31	0.27	0.27	0.3	0.28	0.3	141.52	52.44	46.32	45.30	45.28	1.28	0.9	0.72	0.61	0.62	0.28	0.21	0.18	0.15	0.15	6.92	5.82	2.31	2.04	2.04
	Q	1	0.73	0.53	0.46	0.4	1	1	1	1	1	1	1	1	1	0.86	1	1	0.93	0.93	0.93	1	1	1	1	1	0.86	0.73	0.53	0.53	0.53
	S	1.06	1.26	1.33	1.66	1.6	1	1.2	1	1.2	1.2	1.46	2.53	3.4	4.4	4.2	1	1	1.2	1.06	3.66	1	1	1	1.4	1.66	1.13	1.26	2	2.86	2.86
	Q_{20}	1	0.93	0.8	0.8	0.66	1	1	1	1	1	1	1	1	1	0.86	1	1	1	1	1	1	1	1	1	1	0.86	1	0.6	0.6	0.6
	Q_{50}	1	1	0.86	0.86	0.86	1	1	1	1	1	1	1	1	1	0.86	1	1	1	1	1	1	1	1	1	1	1	1	0.86	0.73	0.73
	d	0.012	0.034	0.043	0.045	0.055	2×10^{-6}	0.1	9.37×10^{-4}	0.126	0.093	0.186	0.186	0.15	0.088	0.056	0.011	0.034	0.043	0.055	0.018	1.79×10^{-4}	1×10^{-6}	1×10^{-6}	1×10^{-6}	0	0.016	0.04	0.068	0.053	0.053
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table A.22: Goal recognition with sequential observations in FGR not given the time-step

Domain		Blocks					Campus					Grid					Intrusion					Kitchen					Logistics				
Approach	%*O*	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10	100	70	50	30	10
HSP$^*_f$	*T*	908.52	677.66	612.36	432.56	321.31	1.77	1.03	0.66	0.45	0.27	259.53	164.18	121.34	87.53	33.94	591.93	147.18	40.68	7.21	1.38	694.32	178.74	105.86	57.87	37.56	63.54	51.24	20.34	9.46	9.47
	Q	1	1	1	0.66	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.46	0.33	0.33
	S	1.06	2.2	3.13	8.86	14.73	1	1	1	1	1.13	6.66	6.66	6.66	6.66	6.66	1	1	1.06	1.2	5	1	1	1	1.4	1.4	1	1.13	1.46	1.46	1.46
*g*HSP$^*_f$	*T*	531.02	427.78	402.37	369.21	358.32	0.94	0.58	0.41	0.31	0.23	119.23	82.18	64.6	50.97	40.64	450.05	113.91	29.56	4.56	1.24	256.74	64.89	52.46	38.93	32.49	36.3	32.48	14.82	7.05	7.04
	Q	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.66	0.8	0.8
	S	1.06	2.2	3.13	11.86	14.73	1	1	1	1	1.13	6.66	6.66	6.66	6.66	6.66	1	1	1.06	1.2	5	1	1	1	1.4	1.66	1	1.13	2.26	3.6	3.6
FGR^I	*T*	1	1.12	1.22	1.41	1.44	0.27	0.32	0.34	0.38	0.41	112.84	116.80	118.43	119.85	120.59	0.88	0.5	0.3	0.2	0.2	0.26	0.19	0.16	0.13	0.13	0.88	1.01	1.18	1.26	1.26
	Q	1	0.86	0.8	0.2	0.06	1	1	1	1	1	1	0.8	0.86	0.8	1	1	0.93	0.93	1	1	1	1	1	1	0.73	1	0.86	0.53	0.6	0.6
	S	1.06	3.73	2.46	2.06	1.46	1	1	1	1	1.13	1	1.4	1.53	2.4	4.46	1	1	1	1.2	5	1	1	1	1.4	1.4	1	1.26	1.6	2.46	2.46
	Q_{20}	1	0.93	0.93	0.46	0.33	1	1	1	1	1	1	0.93	0.86	0.8	1	1	1	0.93	1	1	1	1	1	1	0.73	1	0.93	0.66	0.73	0.73
	Q_{50}	1	0.93	0.93	0.8	0.66	1	1	1	1	1	1	1	1	0.93	1	1	1	1	1	1	1	1	1	1	1	1	0.93	0.8	0.86	0.86
	d	1.1×10^{-3}	0.024	0.028	0.021	0.015	2×10^{-6}	1.8×10^{-4}	1.4×10^{-3}	0.022	0.14	0.218	0.19	0.16	0.096	0.03	1.6×10^{-5}	4.3×10^{-3}	3×10^{-3}	1.64×10^{-4}	0	6.6×10^{-5}	6.1×10^{-3}	0.026	0.034	0.06	5.8×10^{-4}	0.025	0.073	0.043	0.043
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FGR^{I}E	*T*	12.6	5.98	4.4	3.63	3.58	0.39	0.39	0.4	0.42	0.44	133.03	128.05	127.42	129.38	129.83	1.28	0.91	0.72	0.63	0.63	0.28	0.21	0.18	0.16	0.16	8.51	3.58	3.08	2.82	2.82
	Q	0.93	0.86	0.8	0.2	0.06	1	1	1	1	1	1	0.8	0.86	0.8	1	1	0.93	0.93	1	1	1	1	1	1	0.73	0.86	0.66	0.6	0.6	0.6
	S	1.06	1.4	1.33	2	1.4	1	1	1	1	1.13	1	1.4	1.53	2.4	4.46	1	1	1	1.2	5	1	1	1	1.4	1.4	1.13	1.26	1.73	2.46	2.46
	Q_{20}	1	0.93	0.93	0.4	0.13	1	1	1	1	1	1	0.93	0.86	0.8	1	1	1	0.93	1	1	1	1	1	1	0.73	0.86	0.8	0.66	0.73	0.73
	Q_{50}	1	0.93	1	0.86	0.66	1	1	1	1	1	1	1	1	0.93	1	1	1	1	1	1	1	1	1	1	1	0.93	1	0.73	0.86	0.86
	d	7.4×10^{-3}	0.023	0.021	0.014	0.011	1×10^{-6}	6.8×10^{-5}	6.18×10^{-4}	0.022	0.14	0.217	0.186	0.156	0.09	0.03	1.6×10^{-5}	4.3×10^{-3}	3×10^{-3}	1.64×10^{-4}	0	6.6×10^{-5}	6.1×10^{-3}	0.026	0.034	0.06	0.017	0.043	0.072	0.042	0.042
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FGR$^+$	*T*	0.76	0.59	0.61	0.75	0.78	0.23	0.24	0.27	0.25	0.27	33.33	33.54	34.34	35.48	36.00	0.88	0.49	0.3	0.19	0.19	0.25	0.19	0.15	0.12	0.12	0.81	0.4	0.44	0.5	0.5
	Q	1	0.93	0.66	0.46	0.4	1	1	1	1	1	1	0.73	0.73	0.66	0.66	1	1	0.93	0.93	0.93	1	1	1	1	1	1	0.53	0.46	0.6	0.6
	S	1	11.53	5	1.66	1.6	1	1.2	1	1.2	1.2	1	1.53	1.66	2	2.2	1	1	1.2	1.06	3.66	1	1	1	1.4	1.66	1	3	2.13	2.93	2.93
	Q_{20}	1	0.93	0.86	0.8	0.66	1	1	1	1	1	1	0.73	0.8	0.66	0.66	1	1	1	1	1	1	1	1	1	1	1	0.6	0.6	0.66	0.66
	Q_{50}	1	1	0.86	0.86	0.86	1	1	1	1	1	1	0.93	0.86	0.86	0.73	1	1	1	1	1	1	1	1	1	1	1	0.66	0.73	0.8	0.8
	d	4.4×10^{-3}	0.045	0.042	0.05	0.055	2×10^{-6}	0.1	9.37×10^{-4}	0.126	0.093	0.25	0.2	0.181	0.108	0.064	0.011	0.034	0.043	0.055	0.018	1.79×10^{-4}	1×10^{-6}	1×10^{-6}	1×10^{-6}	0	7.5×10^{-4}	0.071	0.077	0.052	0.052
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FGR$^+$E	*T*	7.31	3.66	3.07	2.62	2.57	0.29	0.27	0.3	0.32	0.32	139.81	52.91	46.51	45.48	45.36	1.37	0.9	0.8	0.8	0.7	0.32	0.21	0.2	0.15	0.15	7.31	5.21	2.14	2.41	2.21
	Q	1	0.73	0.53	0.46	0.4	1	1	1	1	1	0.73	0.73	0.6	0.66	0.66	1	1	0.93	0.93	0.93	1	1	1	1	1	0.86	0.73	0.53	0.6	0.6
	S	1.06	1.26	1.33	1.66	1.6	1	1.2	1	1.2	1.2	1.2	1.66	1.73	2	2.2	1	1	1.2	1.06	3.66	1	1	1	1.4	1.66	1.13	1.26	1.8	2.93	2.93
	Q_{20}	1	0.93	0.8	0.8	0.66	1	1	1	1	1	0.73	0.73	0.6	0.66	0.66	1	1	1	1	1	1	1	1	1	1	0.86	1	0.6	0.66	0.66
	Q_{50}	1	1	0.86	0.86	0.86	1	1	1	1	1	1	0.93	0.86	0.86	0.73	1	1	1	1	1	1	1	1	1	1	1	1	0.86	0.8	0.8
	d	0.012	0.034	0.043	0.045	0.055	2×10^{-6}	0.1	9.37×10^{-4}	0.126	0.093	0.19	0.188	0.155	0.095	0.064	0.011	0.034	0.043	0.055	0.018	1.79×10^{-4}	1×10^{-6}	1×10^{-6}	1×10^{-6}	0	0.016	0.04	0.071	0.052	0.052
	Solve	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Appendix B

ISS Crew Activities Domain Definition

```
(define (domain ISS-CAD)
 (:requirements :strips :typing :negative-preconditions :action-costs)
 (:types crew module system component tool)
 (:constants cdr fe1 fe2 fe3 - crew
  Harmony Columbus Destiny Kibo Unity Leonardo Tranquility - module
  wrs ars stw galley she fdss epm mares hrf pfs ared cevis colbert altea
  padles - system
  food-warmer toilet filter distiller sensor cdl meemm ultrasound crew-laptop
  breathing-equipment torso sck laptop slammd helmet treadmill - component
  battery anemometer hygrometer tool-box headlight vacuum-cleaner utensils food
  cleaning-products holter dosimeter spectrometer catheter sphygmomanometer
  pulse-oximeter stethoscope electroencephalograph - tool
 (:predicates (connected ?m1 ?m2 - module)
  (at ?c - crew ?m - module)
  (taken ?t - tool ?c - crew)
  (available ?t - tool ?m - module)
  (taken-replacement ?t - component ?c - crew)
  (replacement-in ?t - component ?m - module)
  (on ?c - component ?s - system ?m - module)
  (enable ?c - component ?s - system ?m - module)
  (in ?c - component ?s - system ?m - module)
  (inspected ?cp - component ?s - system ?m - module ?c - crew)
  (replaced ?cp - component ?s - system ?m - module ?c - crew)
  (airflow-cleaned ?c - crew)
```

```
   (airflow-measured ?c - crew)
   (humidity-measured ?c - crew)
   (wrs-repaired ?c - crew)
   (wrs-inspected ?c - crew)
   (fdss-inspected ?m - module ?c - crew)
   (fdss-repaired ?m - module ?c - crew)
   (vacuumed ?m - module ?c - crew)
   (food-heated ?c - crew)
   (toilet-cleaned ?c - crew)
   (food-eaten ?c - crew)
   (heart-rate-measured ?c - crew)
   (blood-pressure-measured ?c - crew)
   (brain-activity-measured ?c - crew)
   (muscle-measured ?c - crew)
   (skeletal-measure ?c - crew)
   (tD-image ?c - crew)
   (mass-measured ?c - crew)
   (breath-stream-analyzed ?c - crew)
   (radiation-measured ?cr - crew)
   (high-pressure-calibrated ?cr - crew)
   (resistive-exercise-done ?c - crew)
   (aerobic-exercise-done ?c - crew)
   (brain-radiation-measured ?c - crew)
   (body-radiation-measured ?c - crew)
   (health-checked ?c - crew)
   (oxygen-measured ?c - crew)
   (brain-wavees-measured ?c - crew)
   (health-data-sent ?c - crew)

 (:functions (total-cost))

 (:action move
  :parameters (?c - crew ?m1 ?m2 - module)
  :precondition (and (at ?c ?m1) (connected ?m1 ?m2))
  :effect (and (not (at ?c ?m1)) (at ?c ?m2) (increase (total-cost) 1)))

 (:action get
  :parameters (?t - tool ?m - module ?c - crew)
 :precondition (and (available ?t ?m) (at ?c ?m))
 :effect (and (taken ?t ?c) (not (available ?t ?m)) (increase (total-cost) 20)))

(:action put-away
 :parameters (?t - tool ?m - module ?c - crew)
 :precondition (and (at ?c ?m) (taken ?t ?c) (not (available ?t ?m)))
 :effect (and (not (taken ?t ?c)) (available ?t ?m) (increase (total-cost) 20)))

(:action get-replacement
 :parameters (?t - component ?c - crew)
 :precondition (and (at ?c Leonardo) (replacement-in ?t Leonardo))
 :effect (and (taken-replacement ?t ?c) (increase (total-cost) 20)))

(:action power-up
 :parameters (?o - component ?s - system ?m - module ?c - crew)
 :precondition (and (at ?c ?m) (not (on ?o ?s ?m)) (in ?o ?s ?m))
 :effect (and (on ?o ?s ?m) (increase (total-cost) 10)))

(:action power-off
 :parameters (?o - component ?s - system ?m - module ?c - crew)
 :precondition (and (at ?c ?m) (on ?o ?s ?m) (in ?o ?s ?m))
 :effect (and (not (on ?o ?s ?m)) (increase (total-cost) 10)))

;; SPACECRAFT MAINTENANCE
(:action repair-component
 :parameters (?o - component ?s - system ?m - module ?c - crew)
 :precondition (and (taken headlight ?c) (taken tool-box ?c) (at ?c ?m)
                    (in ?o ?s ?m) (not (enable ?o ?s ?m)) (not (on ?o ?s ?m)))
 :effect (and (enable ?o ?s ?m) (increase (total-cost) 25)))

(:action replace-component
 :parameters (?o - component ?s - system ?m - module ?c - crew)
 :precondition (and (taken headlight ?c) (taken tool-box ?c) (at ?c ?m)
                    (taken-replacement ?o ?c) (in ?o ?s ?m) (not (on ?o ?s ?m)))
 :effect (and (replaced ?o ?s ?m ?c) (not(taken-replacement ?o ?c))
              (increase (total-cost) 20)))

(:action inspect-component
 :parameters (?o - component ?s - system ?m - module ?c - crew)
 :precondition (and (in ?o ?s ?m) (not (on ?o ?s ?m)) (at ?c ?m) (taken headlight ?c)
                    (taken tool-box ?c))
 :effect (and (inspected ?o ?s ?m ?c) (increase (total-cost) 10)))

;; LAB EXPERIMENTS
(:action body-oxygen-monitoring
 :parameters (?c - crew)
 :precondition (and (taken pulse-oximeter ?c))
 :effect (and (oxygen-measured ?c) (increase (total-cost) 10)))

(:action brain-monitoring
 :parameters (?c - crew)
 :precondition (and (taken electroencephalograph ?c))
 :effect (and (brain-wavees-measured ?c) (increase (total-cost) 10)))

(:action health-monitoring
 :parameters (?c - crew)
 :precondition (and (oxygen-measured ?c) (blood-pressure-measured ?c)
                    (heart-rate-measured ?c) (brain-wavees-measured ?c))
 :effect (and (health-checked ?c)))

(:action send-health-data
 :parameters (?c - crew)
 :precondition (and (at ?c Kibo) (on laptop padles Kibo) (health-checked ?c))
 :effect (and (health-data-sent ?c) (increase (total-cost) 30)))

;; EPM - European Physiology Module
(:action blood-heart-monitoring
 :parameters (?c - crew)
 :precondition (and (at ?c Columbus) (on cdl epm Columbus))
 :effect (and (blood-pressure-measured ?c) (heart-rate-measured ?c)
              (increase (total-cost) 50)))

(:action blood-heart-monitoring-holter
 :parameters (?c - crew)
 :precondition (and (taken holter ?c) (at ?c Kibo))
 :effect (and (heart-rate-measured ?c) (blood-pressure-measured ?c)
              (increase (total-cost) 50)))

(:action blood-heart-manual-monitoring
 :parameters (?c - crew)
 :precondition (and (taken stethoscope ?c) (taken sphygmomanometer ?c))
 :effect (and (blood-pressure-measured ?c) (increase (total-cost) 40)))

(:action brain-activity-measuring
 :parameters (?c - crew)
 :precondition (and (at ?c Columbus) (on meemm epm Columbus))
 :effect (and (brain-activity-measured ?c) (increase (total-cost) 10)))

;; MARES - Muscle Atrophy Research Exercise System
(:action microgravity-study
 :parameters (?c - crew)
 :precondition (and (at ?c Columbus) (on laptop mares Columbus))
 :effect (and (muscle-measured ?c) (skeletal-measure ?c) (increase (total-cost) 15)))

;; HRF - Human Resource Facilicty
(:action sonography
 :parameters (?c - crew)
 :precondition (and (at ?c Destiny) (on laptop hrf Destiny) (on ultrasound hrf Destiny))
 :effect (and (tD-image ?c) (increase (total-cost) 15)))

(:action dosimetric-mapping-HRF
 :parameters (?c - crew)
 :precondition (and (at ?c Destiny) (taken dosimeter ?c) (on laptop hrf Destiny))
 :effect (and (radiation-measured ?c) (increase (total-cost) 15)))

(:action high-pressure-calibration-D
 :parameters (?c - crew)
 :precondition (and (at ?c Destiny) (taken spectrometer ?c) (on laptop hrf Destiny)
                    (not (on sensor hrf Destiny)))
 :effect (and (high-pressure-calibrated ?c) (increase (total-cost) 20)))

(:action high-pressure-calibration-C
 :parameters (?c - crew)
 :precondition (and (at ?c Columbus) (taken spectrometer ?c) (on laptop pfs Columbus)
                   (not (on sensor pfs Columbus)))
 :effect (and (high-pressure-calibrated ?c) (increase (total-cost) 20)))

(:action breath-stream-analysis-D
 :parameters (?c - crew)
 :precondition (and (at ?c Destiny) (on laptop hrf Destiny) (on sensor hrf Destiny)
                   (taken catheter ?c) (high-pressure-calibrated ?c))
 :effect (and (breath-stream-analyzed ?c) (increase (total-cost) 15)))

(:action breath-stream-analysis-C
 :parameters (?c - crew)
 :precondition (and (at ?c Columbus) (on laptop pfs Columbus) (on sensor pfs Columbus)
                   (taken catheter ?c) (high-pressure-calibrated ?c))
 :effect (and (breath-stream-analyzed ?c) (increase (total-cost) 15)))

(:action measure-mass
 :parameters (?c - crew)
 :precondition (and (at ?c Destiny) (on laptop hrf Destiny) (on slammd hrf Destiny))
 :effect (and (mass-measured ?c) (increase (total-cost) 15)))

;; CEVIS - Cycle Ergometer with Vibration Isolation System
(:action aerobic-exercise
 :parameters (?c - crew)
 :precondition (and (at ?c Destiny) (on laptop cevis Destiny))
 :effect (and (aerobic-exercise-done ?c) (increase (total-cost) 10)))

;; COLBERT - Combined Operational Load Bearing External Resistive Exercise Treadmill
(:action resistive-exercise
 :parameters (?c - crew)
 :precondition (and (at ?c Destiny) (on laptop colbert Destiny)
                   (on treadmill colbert Destiny))
 :effect (and (resistive-exercise-done ?c) (increase (total-cost) 10)))

;; ALTEA - Anomalous Long Term Effects in Astronaut's Central Nervous System
(:action dosimetric-mapping-AD
 :parameters (?c - crew)
 :precondition (and (at ?c Destiny) (on laptop altea Destiny))
 :effect (and (radiation-measured ?c) (increase (total-cost) 10)))
```

```
(:action dosimetric-mapping-AC
 :parameters (?c - crew)
 :precondition (and (at ?c Columbus) (on laptop altea Columbus))
 :effect (and (radiation-measured ?c) (increase (total-cost) 10)))

(:action measure-brain-radiation-AD
 :parameters (?c - crew)
 :precondition (and (at ?c Destiny) (on helmet altea Destiny)
                    (on laptop altea Destiny))
 :effect (and (brain-radiation-measured ?c) (brain-activity-measured ?c)
              (increase (total-cost) 10)))

(:action measure-brain-radiation-AC
 :parameters (?c - crew)
 :precondition (and (at ?c Columbus) (on laptop altea Columbus)
                    (on helmet altea Columbus))
 :effect (and (brain-radiation-measured ?c) (brain-activity-measured ?c)
              (increase (total-cost) 10)))

;; PADLES - PAssive Dosimeter for Lifescience Experiments in Space
(:action dosimetric-mapping-P
 :parameters (?c - crew)
 :precondition (and (at ?c Kibo) (on laptop padles Kibo))
 :effect (and (radiation-measured ?c) (increase (total-cost) 15)))

(:action measure-body-radiation-P
 :parameters (?c - crew)
 :precondition (and (at ?c Kibo) (taken dosimeter ?c) (on laptop padles Kibo))
 :effect (and (body-radiation-measured ?c) (increase (total-cost) 15)))

;; water control
(:action repair-wrs-distiller
 :parameters (?c - crew)
 :precondition (and (at ?c Tranquility) (on distiller wrs Tranquility)
                    (replaced distiller wrs Tranquility ?c))
 :effect (and (wrs-repaired ?c)))
```

```
(:action repair-wrs-filter
 :parameters (?c - crew)
 :precondition (and (at ?c Tranquility) (on filter wrs Tranquility)
                    (replaced filter wrs Tranquility ?c))
 :effect (and (wrs-repaired ?c)))

(:action wrs-inspection
 :parameters (?c - crew)
 :precondition (and (at ?c Tranquility) (inspected filter wrs Tranquility ?c)
                    (inspected distiller wrs Tranquility ?c))
 :effect (and (wrs-inspected ?c)))

;; airflow levels measure
(:action airflow-measuring
 :parameters (?c - crew ?m - module)
 :precondition (and (at ?c ?m) (taken battery ?c) (taken anemometer ?c))
 :effect (and (airflow-measured ?c) (increase (total-cost) 15)))

;; airflow cleaning
(:action airflow-cleaning
 :parameters (?c - crew ?m - module)
 :precondition (and (at ?c ?m) (airflow-measured ?c) (on filter ars ?m)
                    (replaced filter ars ?m ?c) )
 :effect (and (airflow-cleaned ?c) ))

;; fire detector system
(:action fdss-repair-sensor
 :parameters (?c - crew ?m - module)
 :precondition (and (at ?c ?m) (on sensor fdss ?m) (replaced sensor fdss ?m ?c))
 :effect (and (fdss-repaired ?m ?c)))

(:action fdss-repair-extinguisher
 :parameters (?c - crew ?m - module)
 :precondition (and (at ?c ?m) (replaced extinguisher fdss ?m ?c))
 :effect (and (fdss-repaired ?m ?c)))

(:action fdss-repair-breathing-equipment
 :parameters (?c - crew ?m - module)
 :precondition (and (at ?c ?m) (replaced breathing-equipment fdss ?m ?c))
 :effect (and (fdss-repaired ?m ?c)))

(:action fdss-inspection
 :parameters (?c - crew ?m - module)
 :precondition (and (at ?c ?m) (inspected sensor fdss ?m ?c)
                    (inspected extinguisher fdss ?m ?c)
                    (inspected breathing-equipment fdss ?m ?c))
 :effect (and (fdss-inspected ?m ?c)))

;; housekeeping activities
(:action food-warming
 :parameters (?c - crew)
 :precondition (and (at ?c Unity) (taken food ?c) (enable food-warmer galley Unity))
 :effect (and (food-heated ?c) (increase (total-cost) 20)))

(:action eating-snack
 :parameters (?c - crew)
 :precondition (and (at ?c Unity) (taken food ?c) (taken utensils ?c))
 :effect (and (food-eaten ?c) (increase (total-cost) 20)))

(:action eating-meal
 :parameters (?c - crew)
 :precondition (and (at ?c Unity) (food-heated ?c) (taken utensils ?c))
 :effect (and (food-eaten ?c) (increase (total-cost) 20)))

(:action vacuum-module
 :parameters (?c - crew ?m - module)
 :precondition (and (at ?c ?m) (taken vacuum-cleaner ?c))
 :effect (and (vacuumed ?m ?c) (increase (total-cost) 10)))



(:action toilet-cleaning
 :parameters (?c - crew)
 :precondition (and (at ?c Tranquility) (in toilet she Tranquility)
                    (taken cleaning-products ?c))
 :effect (and (toilet-cleaned ?c) (increase (total-cost) 15)))
```

Bibliography

Albore, A., Palacios, H., and Geffner, H. (2009). A translation-based approach to contingent planning. In *Proceedings of the Joint Conference on Artificial Intelligence*, Pasadena, CA, USA.

Albrecht, D. W., Zukerman, I., Nicholson, A. E., and Bud, A. (1997). Towards a bayesian model for keyhole plan recognition in large domains. In *Proceedings of the International Conference on User Modeling*, Chia Laguna, Sardinia.

Bäckström, C. and Nebel, B. (1995). Complexity results for SAS$^+$ planning. *Computational Intelligence*, 11(4):625–655.

Bagdigian, R. M. (2008). Environmental Control and Life Support System. www.nasa.gov.

Barto, A. G., Bradtke, S. J., and Singh, S. P. (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72:81–138.

Bauer, M. (1998). Acquisition of abstract plan descriptions for plan recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Madison, WI, USA.

Baxter, J. and Bartlett, P. L. (1999). Direct gradient-based reinforcement learning: I. gradient estimation algorithms. Technical report, Australian National University.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA.

Bernard, D. E., Dorais, G. A., Fry, C., Gamble Jr., E. B., Kanefsky, B., Kurien, J., Millar, W., Muscettola, N., Nayak, P. P., Pell, B., Rajan, K., Rouquette, N., Smith, B., and Williams, B. C. (1998). Desing of the remote agent experiment for spacecraft autonomy.

Blum, A. and Furst, M. (1997). Fast planning through planning graph analysis. *Artificial Intelligence*, 90:281–300.

Blum, A. and Langford, J. C. (1999). Probabilistic planning in the Graphplan framework. In *Proceedings of the European Conference on Planning*, Durham, UK.

Bonet, B. and Geffner, H. (2001). Planning as heuristic search. *Journal of Artificial Intelligence Research*, 129:5–33.

Bonet, B. and Geffner, H. (2003a). Labeled RTDP: improving the convergence of real-time dynamic programming. In *Proceedings of the International Conference of Automated Planning and Scheduling*, Trento, Italy.

Bonet, B. and Geffner, H. (2003b). VHPOP: versatile heuristic partial order planner. *Journal of Artificial Intelligence Research*, 20:405–430.

Bonet, B. and Geffner, H. (2005). mGPT: a probabilistic planner based on heuristic search. *Journal of Artificial Intelligence Research*, 24:933–944.

Bonet, B. and Given, R. (2006). International Probabilistic Planning Competition. http://www.ldc.usb.ve/˜bonet/ipc5.

Bonet, B., Loerincs, G., and Geffner, H. (1997). A robust and fast action selection mechanism for planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Providence, RI, USA.

Brafman, R. I. and Shani, G. (2012). A multi-path compilation approach to contingent planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Toronto, Canada.

Bresina, J. L. (2015). Activity planning for a lunar orbital mission. In *Proceedings of the Annual Conference on Innovative Applications of Artificial Intelligence*, Austin, TX, USA.

Brown, J. S., Burton, R. R., and Larkin, K. M. (1977). Representing and using procedural bugs for educational purposes. In *Proceedings of Annual Conference*, New York, NY, USA.

Bryce, D., Kambhampati, S., and Smith, D. E. (2006). Planning graph heuristics for belief space search. *Journal of Artificial Intelligence Research*, 26:35–99.

Bryce, D. and Smith, D. E. (2006). Using interaction to compute better probability estimates in plan graphs. In *Proceedings of the ICAPS-06 Workshop on Planning Under Uncertainty and Execution Control for Autonomous Systems*, The English Lake District, Cumbria, UK.

Buffet, O. and Aberdeen, D. (2009). The Factored Policy-Gradient planner. volume 173, pages 722–747.

Buffet, O. and Bryce, D. (2008). Proceedings of the International Planning Competition. http://ippc-2008.loria.fr/wiki/index.php/Main_Page.

Bui, H. (2003). A general model for on-line probabilistic plan recognition. In *Proceedings of International Joint Conference on Artificial Intelligence*, Acapulco, Mexico.

Castaño, B., E-Martín, Y., R-Moreno, M. D., and Usero, L. (2014). Sistema inteligente de detección y orientación de usuarios en bibliotecas. *Revista Espanõla de Documentación Científica*, 31(5):421–436.

Chen, Y., Huang, R., Xing, Z., and Zhang, W. (2009). Long-distance mutual exclusion for planning. *Artificial Intelligence*, 173:197–412.

Chen, Y., Huang, R., and Zhang, W. (2008). Fast planning by search in domain transition graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Chicago, IL, USA.

Chen, Y., Wah, B. W., and Hsu, C.-W. (2006). Temporal planning using subgoal partitioning and resolution in SGPlan. *Journal of Artificial Intelligence Research*, 26:323–369.

Coles, A., Coles, A., Fox, M., and Long, D. (2010). Forward-chaining partial-order planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, Toronto, Ontario, Canada.

Currie, K. and Tate, A. (1991). O-Plan: the open planning architecture. *Artificial Intelligence*, 51(1):49–86.

Dearden, R., Meuleau, N., Ramakrishnan, S., Smith, D. E., and Washington, R. (2003). Incremental Contingency Planning. In *Proceedings of ICAPS-03 Workshop on Planning under Uncertainty*, Trento, Italy.

Do, M. and Kambhampati, S. (2002). Planning graph-based heuristics for cost-sensitive temporal planning. In *Proceedings of the Conference on Artificial Intelligence Planning and Scheduling*, Toulouse, France.

Do, M. and Kambhampati, S. (2003). Sapa: a multi-objective metric temporal planner. *Journal of Artificial Intelligence*, 20:155–194.

Do, M. B., Benton, J., van den Briel, M., and Kambhampati, S. (2007). Planning with goal utility dependencies. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Hyderabad, India.

Drabble, B., Dalton, J., and Tate, A. (1997). Repairing plans on-the-fly. In *Proceedings of the NASA Workshop on Planning and Scheduling for Space*, Oxnard, CA, USA.

E-Martín, Y., R-Moreno, M. D., and Smith, D. E. (2014). Progressive heuristic search for probabilistic planing based on interaction estimates. *Expert Systems*, 31(5):421–436.

E-Martín, Y., R-Moreno, M. D., and Smith, D. E. (2015a). A Fast Goal Recognition Technique based on Interaction Estimates. *Proceedings of International Joint Conference on Artificial Intelligence*.

E-Martín, Y., R-Moreno, M. D., and Smith, D. E. (2015b). A Heuristic Estimator based on Cost Interaction. *In Proceedings of the ICAPS-15 Workshop on Heuristics and Search for Domain-independent Planning*.

E-Martín, Y., R-Moreno, M. D., and Smith, D. E. (2015c). Practical goal recognition for ISS crew activities. *Proceedings of the International Workshop of Planning and Scheduling for Space*.

Fikes, R. and Nilsson, N. J. (1971). STRIPS: a new approach to the application of theorem proving to problem solving. In *Proceedings of the International Joint Conference on Artificial Intelligence*, London, UK.

Foss, J., Onder, N., and Smith, D. E. (2007). Preventing unrecoverable failures through precautionary planning. In *Proceedings of the ICAPS'07 Workshop on Moving Planning and Scheduling Systems into the Real World*, Providence, RI, USA.

Fox, M., Long, D., and Magazzeni, D. (2011). Automatic construction of efficient multiple battery sage policies.

Garrido, A., Fox, M., and Long, D. (2002). Temporal planning with PDDL2.1. In *Proceedings of the European Conference on Artificial Intelligence*, Lyon, France.

Geib, C. W. and Goldman, R. P. (2009). A probabilistic plan recognition algorithm based on plan tree grammar. *Artificial Intelligence*, 84(1-2):57–112.

Gerevini, A., Saetti, A., and Serina, I. (2003). Planning through stochastic local search and temporal action graphs in LPG. *Journal of Artificial Intelligence Research*, 20:239–290.

Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated Planning: theory & practice*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Godefroid, P. and Kabanza, F. (1991). An efficient reactive planner for synthesizing reactive plans. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Anaheim, CA, USA.

Hansen, E. A. and Zilberstein, S. (2001). LAO*: a heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129(1-2):35–62.

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Science, and Cybernetics*, SSC-4(2):100–107.

Haslum, P. (2008). Additive and reversed relaxed reachability heuristics revisited. In *Proceedings of the International Planning Competition*.

Haslum, P. and Geffner, H. (2000). Admissible heuristic for optimal planning. In *Proceedings of the International Conference on Artificial Intelligence Planning and Scheduling*, Breckenridge, CO, USA.

Helmert, M. (2006). The Fast Downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246.

Helmert, M. (2008). Changes in PDDL 3.1. http://ipc.informatik.uni-freiburg.de/PddlExtension.

Helmert, M. (2009). Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence*, 173(5–6):503–535.

Helmert, M. and Domshlak, C. (2009). Landmarks, critical paths and abstractions: What's the difference anyway? In *Proceedings of the International Conference on Artificial Intelligence Planning and Scheduling*, Sydney, Australia.

Helmert, M., Haslum, P., and Hoffmann, J. (2008). Explicit-state abstraction: a new method for generating heuristic fnctions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Chicago, IL, USA.

Hoffmann, J. (2003). The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research*, 20:291–341.

Hoffmann, J. and Nebel, B. (2001). The FF planning system: fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302.

Hoffmann, J., Porteous, J., and Sebastia, L. (2004). Ordered Landmarks in planning. *Journal of Artificial Intelligence Research*, 22:215–278.

Howard, R. A. (1960). *Dynamic Programming and Markov Processes*. The MIT press, New York London, Cambridge, MA, USA.

Huang, R., Chen, Y., and Zhang, W. (2012). SAS$^+$ planning as satisfiability. *Journal of Artificial Intelligence Research*, 43:293–328.

Huber, M. J., Durfee, E. H., and Wellman, M. P. (1994). The automated mapping of plans for plan recognition. *Proceedings of Uncertainty in Artificial Intelligence*.

Jigui, S. and Minghao, Y. (2007). Recognizing the agent's goals incrementally: planning graph as a basis. *Frontiers of Computer Science in China*, 1(1):26–36.

Jiménez, S., Coles, A., and Smith, A. (2006). Planning in probabilistic domains using a deterministic numeric planner. In *Proceedings of the Workshop of the UK Planning and Scheduling Special Interest Group*, Nottingham, UK.

Joslin, D. and Pollack, M. E. (1994). Least-cost flaw repair: A plan refinement strategy for partial order planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Seattle, WA, USA.

Kalyaman, R. and Givan, R. (2008). LDFS with deterministic plan based subgoals. In *Proceedings of the International Planning Competition*.

Kautz, H. and Allen, J. F. (1986). Generalized plan recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Philadelphia, PA, USA.

Kautz, H., Fox, D., Etzioni, O., Borriello, G., and Arnstein, L. (2002). An overview of the assisted cognition project. In *Proceedings of the AAAI Workshop on Automation as Caregiver: The Role of Intelligent Technology in Elder Care*, Edmonton, Alberta, Canada.

Kautz, H. and Selman, B. (1992). Admissible heuristic for optimal planning. In *Proceedings of the European Conference on Artificial Intelligence*, Vienna, Austria.

Kautz, H. and Selman, B. (1996). Pushing the envelope: planning, propositional logic, and stochastic search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Portland, OR, USA.

Keller, T. and Helmert, M. (2013). Trial-based heuristic tree search for finite horizon MDPs. In *Proceedings of the International Conference on Artificial Intelligence Planning and Scheduling*, Rome, Italy.

Keren, S., Gal, A., and Karpas, E. (2014). Goal recognition desing. In *Proceedings of the International Conference on Artificial Intelligence Planning and Scheduling*, Portsmouth, NH, USA.

Keren, S., Gal, A., and Karpas, E. (2015). Goal recognition desing for non-optimal agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Austin, TX, USA.

Keyder, E. and Geffner, H. (2007). Heuristics for planning with action costs. In *Proceedings of the Spanish Artificial Intelligence Conference*, Salamanca, Spain.

Keyder, E. and Geffner, H. (2008a). Heuristics for planning with action costs revisited. In *Proceedings of the European Conference on Artificial Intelligence*, Patras, Greece.

Keyder, E. and Geffner, H. (2008b). The HMDP planner for planning with probabilities. In *Proceedings of the International Planning Competition*.

Keyder, E., Hoffmann, J., and Haslum, P. (2012). Semi-relaxed plan heuristics. In *Proceedings of the International Conference on Artificial Intelligence Planning and Scheduling*, Atibaia, Brazil.

Keyder, E., Richter, S., and Helmert, M. (2010). Sound and complete landmarks for And/Or graphs. In *Proceedings of the European Conference on Artificial Intelligence*, Lisbon, Portugal.

Khan, S., Decker, K., Gillis, W., and Schmidt, C. (2003). A multi-agent system-driven AI planing approach to biological pathway discovery.

Kingston, J., Shadbolt, N., and Tate, A. (1996). CommonKADS models for knowledge based planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Portland, OR, USA.

Kolobov, A., Mausam, and Weld, D. (2012). LRTDP vs. UCT for online probabilistic planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Toronto, Canada.

Kruskal, W. H. and Allen, W. W. (1952). Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260).

Lesh, N. and Etzioni, O. (1995). A sound and fast recognizer. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Quebec, Canada.

Little, I. and Thiébaux, S. (2006). Concurrent probabilistic planning in the Graphplan framework. In *Proceedings of the ICAPS'06 Workshop on Planning Under Uncertainty and Execution Control for Autonomous Systems*, The English Lake District, Cumbria, UK.

Little, I. and Thiébaux, S. (2007). Probabilistic planning vs replanning. In *Proceedings of the ICAPS'07 Workshop on Planning Competitions*, Providence, RI, USA.

Long, D. and Fox, M. (2003). Exploiting a Graphplan Framework in Temporal Planning. In *Proceedings of the International Conference on Artificial Intelligence Planning and Scheduling*, Trento, Italy.

McDermott, D. V. (1996). A Heuristic Estimator for Means-Ends Analysis in Planning. In *Proceedings of the International Conference on AI Planning Systems*, Edinburgh, Scotland.

McDermott, D. V. (1998). PDDL the Planning Domain Definition Language. Technical report, Yale University, Center for Computational Vision and Control.

Meuleau, N., Plaunt, C., Smith, D. E., and Smith, T. (2009). An emergency landing planner for damaged aircraft. In *Proceedings of the Annual Conference on Innovative Applications of Artificial Intelligence*, Pasadena, CA, USA.

Nguyen, X., Kambhampati, S., and Nigenda, R. S. (2002). Planning graph as the basis for deriving heuristics for plan synthesis by state space and CSP search. *Artificial Intelligence*, 135(1-2):73–123.

Nilsson, N. J. (1980). *Principles of Artificial Intelligence*. Tioga Publishing Company, Palo Alto, CA, USA.

Palacios, H. and Geffner, H. (2009). Compiling uncertainty away in conformant planning problems with bounded width. *Journal of Artificial Intelligence Research*, 35:623–675.

Pattinson, D. (2010). Domain independent goal recognition. In *Proceedings of the International Conference on Artificial Intelligence Planning and Scheduling*, Toronto, Canada.

Pearl, J. (1984). *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Pearl, J. and Kim, J. H. (1982). Studies in semi-admissible heuristics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(4):392–399.

Penberthy, J. S. and Weld, D. (1992). UCPOP: a sound, complete, partial order planner for ADL. In *Proceedings of the International Conference on Knowledge Representation and Reasoning*, Cambridge, MA, USA.

Penberthy, J. S. and Weld, D. (1994). Temporal planning with continuous change. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Seattle, WA, USA.

Plaza, J., R-Moreno, M. D., Castaño, B., Carbajo, M., and Moreno, A. (2008). PIPSS: Parallel Integrated Planning and Scheduling System. In *Proceedings of the Workshop of the UK Planning and Scheduling Special Interest Group*, Edinburgh, Scotland.

Pollack, M. E., Brown, L., Colbry, D., McCarthy, C. E., Orosz, C., Peintner, B., Ramakrishnan, S., and Tsamardinos, I. (2003). Autominder: an intelligent cognitive orthotic system for people with memory impairment. *Robotics and Autonomous Systems*, 44(3-4):273–282.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA.

Ramírez, M. (2012). *Plan Recognition as Planning*. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain.

Ramírez, M. and Geffner, H. (2009). Plan recognition as planning. In *Proceedings of International Joint Conference on Artificial Intelligence*, Pasadena, CA, USA.

Ramírez, M. and Geffner, H. (2010). Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Atlanta, GA, USA.

Ramírez, M. and Geffner, H. (2011). Goal recognition over POMDPs. In *Proceedings of the International Conference on Artificial Intelligence Planning and Scheduling*, Freiburg, Germany.

Richter, S., Helmert, M., and Westphal, M. (2008). Landmarks revisited. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Chicago, IL, USA.

Richter, S. and Westphal, M. (2010). The LAMA planner: guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, 39:127–177.

Rintanen, J. (2004). Evaluation strategies for planning as satisfiability. In *Proceedings of European Conference on Artificial Intelligence*, Valencia, Spain.

Rintanen, J. (2006). Planning as satisfiability: parallel plans and algorithms for plan search. *Artificial Intelligence*, 170(12):1031–1080.

Rintanen, J. (2011). Planning with SAT, admissible heuristics and A*. In *Proceedings of International Joint Conference on Artificial Intelligence*, Barcelona, Spain.

Rintanen, J. (2012). Planning as satisfiability: heuristics. *Artificial Intelligence*, 193:45–86.

Robinson, N., Gretton, C., Pham, D.-N., and Sattar, A. (2009). SAT-basd parallel planning using a split representation of actions. In *Proceedings of the International Conference on Artificial Intelligence Planning and Scheduling*, Thessaloniki, Greece.

Sanchez Nigenda, R. and Kambhampati, S. (2005). Planning graph heuristic for selecting objectives in over-subscription planning problems. In *Proceedings of the International Conference on Automated Planning and Scheduling*, Monterey, CA, USA.

Shapiro, S. and Wilk, M. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, 53(3-4):591–611.

Smith, D. E. and Weld, D. (1999). Temporal Graphplan with mutual exclusion reasoning. In *Proceedings of International Joint Conference on Artificial Intelligence*, Stockholm, Sweden.

Tate, A. (1974). INTERPLAN: a plan generation system which can deal with interactions between goals. Technical report, Memo MIP-R-109, Machine Intelligence Research Unit, University of Edinburgh.

Tate, A., Dalton, J., and Levine, J. (2000). P-Plan: a web-based AI planning agent. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Austin, TX, USA.

Teichteil-Königsbuch, F., Kuter, U., and Infantes, G. (2010). Incremental plan aggregation for generating policies in MDPs. In *Proceedings of the Autonomous Agents and Multiagent Systems*, Toronto, Canada.

Veloso, M., Carbonell, J., Pérez, A., Borrajo, D., Fink, E., and Blythe, J. (1995). The PRODIGY architecture. *Journal of Experimental and Theoretical Artificial Intelligence*, 7(1):81–120.

Vidal, V. and Pierre, R. (1999). Total order planning is more efficient than we though. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Orlando, FL, USA.

Warren, D. H. (1974). WARPLAN: a system for generating plans. Technical report, Memo 76, Computational Logic Department, School of AI, University of Edinburgh.

Weber, J. S. and Pollack, M. E. (2008). Evaluating user preferences for adaptive reminding. In *Proceedings of Human Factors in Computing Extended Abstracts*, Florence, Italy.

Wehrle, M. and Rintanen, J. (2007). Planning as satisfiability with relaxed ∃-step plans. In *Proceedings of the Australian Joint Conference on Advances in Artificial Intelligence*, Gold Coast, Australia.

Wu, J., Osuntogun, A., Choudhury, T., Philipose, M., and Rehg, J. M. (2007). A scalable approach to activity recognition based on object use. In *Proceedings the IEEE International Conference on Computer Vision*, Rio de Janeiro, Brazil.

Wu, J.-H., Kalyaman, R., and Givan, R. (2011). Stochastic enforced hill-climbing. *Journal of Artificial Intelligence Research*, 42:815–850.

Yoon, S., Fern, A., and Givan, R. (2007). FF-Replan: a baseline for probabilistic planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, Providence, RI, USA.

Yoon, S., Fern, A., Givan, R., and Kambhampati, S. (2008). Probabilistic planning via determinization in hindsight. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Chicago, IL, USA.

Yoon, S., Ruml, W., Benton, J., and Do, M. (2010). Improving determinization in hindsight for on-line probabilistic planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, Toronto, Ontario, Canada.

Younes, H. L. S., Littman, M. L., Weissman, D., and Asmuth, J. (2005). The first probabilistic track of the International Planning Competition. *Journal of Artificial Intelligence Research*, 24:841–887.