



UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior
Ingeniería Informática

Planning and Learning under Uncertainty

Anteproyecto de Tesis para el programa de Doctorado de Ingeniería
Informática

AUTOR: Sergio Jiménez Celorrio

DIRECTORES: Daniel Borrajo Millán y Fernando Fernández Rebollo

Leganés, Mayo 2007

Contents

0.1	Resumen	1
0.2	Abstract	2
1	Introduction	3
2	State of the Art	7
2.1	Introduction	7
2.2	Deterministic Planning	8
2.2.1	The Conceptual Model	8
2.2.2	The Representation Languages	9
2.2.3	The Algorithms	13
2.2.4	The Planners	14
2.3	Planning Under Uncertainty	16
2.3.1	Probabilistic Planning	16
2.3.2	Contingent Planning	21
2.3.3	Conformant Planning	23
2.3.4	Probabilistic-Contingent Planning	26
2.3.5	Probabilistic-Conformant Planning	27
2.4	Learning to Improve Automated Planning	28
2.4.1	Learning Search Control Knowledge	30
2.4.2	Learning Domain-Specific Planners	34
2.4.3	Learning Domain Models	36
2.5	Discussion	37
3	Objectives	39
4	Methodology	41
4.1	The International Planning Competition	41
4.2	Evaluation	42
4.3	Work Plan	44

0.1 Resumen

La Planificación Automática es una importante rama de la Inteligencia Artificial que estudia la búsqueda computacional de conjuntos de acciones cuya ejecución permita alcanzar unos objetivos determinados. Históricamente, la investigación en esta rama ha estado centrada en resolver problemas teóricos en mundos controlados en los que se podía conocer exactamente el estado actual del entorno y se podía prever completamente los resultados de la ejecución de las acciones. El desarrollo de aplicaciones en la última década (planificación de las operaciones de extinción de incendios forestales [24], planificación de las actividades realizadas por la nave espacial *Deep Space 1* [8], planificación de las acciones de evacuación en emergencias [71]) ha evidenciado que estos modelos con los que tradicionalmente se trabajaba en planificación eran demasiado ingenuos para poder hacer frente a problemas reales.

Consciente de ello, la comunidad investigadora ha multiplicado sus esfuerzos en los últimos años para definir nuevos modelos de planificación capaces de ajustarse mejor a los problemas reales. Esto ha llevado al nacimiento de una nueva área dentro de la Planificación Automática, llamada planificación bajo incertidumbre, y al establecimiento de métricas y estándares con los que poder evaluar y comparar los nuevos planificadores. A pesar de estos esfuerzos, la nueva generación de planificadores no es capaz de resolver eficientemente problemas con incertidumbre si no disponen de una especificación detallada y completa del modelo de acciones. Sin embargo, conseguir un modelo de acciones preciso en entornos con incertidumbre es una tarea muy compleja y en algunos casos imposible (por ejemplo en la planificación de las acciones de los robots *Spirit* y *Opportunity* en Marte [1]).

Mi objetivo es definir un nuevo paradigma de planificación capaz de encontrar eficientemente planes cuya ejecución tenga una alta probabilidad de éxito en dominios con incertidumbre. Mi Tesis es que integrando técnicas de aprendizaje automático junto con los procesos de planificación y ejecución se puede desarrollar un planificador capaz de enriquecer automáticamente el conocimiento inicial del entorno con información adicional acerca de la ejecución de las acciones que ayude a encontrar planes más robustos.

Esta propuesta revisa y pone de relieve las deficiencias de los actuales paradigmas de Planificación Automática haciendo un énfasis especial en las técnicas de planificación bajo incertidumbre, repasa las diferentes técnicas de Aprendizaje Automático utilizadas para mejorar el rendimiento de los planificadores, y finalmente, describe los objetivos que se pretenden cubrir con esta Tesis y define una metodología para su consecución.

0.2 Abstract

Automated Planning is an important component of Artificial Intelligence that studies computationally the process of finding sets of actions whose execution achieves certain given objectives. Most of the research on Automated Planning has traditionally focused on solving theoretical problems in totally controlled environments where the state of the environment is totally observable and where the outcomes of action execution are completely predictable. The development of real planning applications during the last decade: planning fire extinction operations [24], planning spacecraft activities [8], planning emergency evacuation actions [71]. . . has evidenced that these classical planning models are too naive to address realistic problems.

The planning research community who is aware of this issue has multiply its efforts during the last years in defining new planning models able to tackle realistic problems. All these efforts have begun a new field in Automated Planning called planning under uncertainty, likewise the establishment of standards to compare and evaluate the new generation planners. In spite of all these efforts the new planners precise an accurate definition of the action model to guarantee good performance in problems with uncertainty. However, the definition by hand of accurate models is very complex and sometimes impossible. For example when planning the actions of the Mars Rovers [1].

My goal is to define a new planning paradigm that allows to efficiently build plans with a high probability of success in domains with uncertainty. My thesis is that, integrating machine learning techniques with the planning and execution processes, a planner can be build that automatically enriches its initial knowledge about the environment with information about the actions performance to build more robust plans.

This proposal revises the shortcomings of the existing planning paradigms highlighting the planning under uncertainty techniques, reviews the different Machine Learning techniques that historically have been used to improve planners performance and finally describes the objectives posed in the Thesis and defines a methodology to fulfil them.

Chapter 1

Introduction

Humans, unlike other living beings, can deliberate about the consequences of their actions. This human-intelligence ability that allows us to choose and organize actions by anticipating their outcomes is called *Planning*. *Planning* is really a hard intellectual job: the world where we live is stochastic, changes continuously and the vision we have of it is limited. So we resort to it just when it is strictly needed. Mainly when we have to address new tasks or complex problems or when we have to proceed constrained somehow. For example, when we organise a weekend trip to a city we have never been to we plan the itinerary, the places to visit, the accommodation, . . .

Automated Planning (AP) arose in the late '50s from converging studies into state-space search, theorem proving and control theory from the practical needs of robotics, scheduling and other domains. STRIPS (STanford Institute Problem Solver) [35], one of the first major planning systems, was designed to be the planning component of the controlling software for the autonomous robot Shakey [75] and illustrates perfectly the interaction of all these different influences.

Nowadays, AP is studied as an independent discipline within the framework of Artificial Intelligence(AI), and it is defined as the area of knowledge that studies computationally the generation of sequences of actions that solve given problems. In AP, a problem is defined by: a state-transition function of a dynamic system, an initial state of the system and a set of objectives or goals to be achieved. According to this definition, AP problems seem to be easily tackled by searching for a path in a graph, which is a well-known problem. However, in AP this graph can be so large that specifying it explicitly is not feasible. Since the first days different approaches have been tried, but it is just in the last ten years that the new approaches based on heuristic search and problem decompositions have allowed to tackle planning problems. In spite of the new advances in the field, solving complex planning problems with a general domain independent planner is still intractable [22]; as it is considered a PSpace-complete problem [21]. Because

of that, the planners implemented in applications frequently use expert domain knowledge that facilitates the search process.

When solving AP problems in the real world it is not always feasible to follow a detailed plan. At any step unexpected events can occur ruining the predictions. In the previous example, the bad weather, a closed restaurant or a too crowded museum can make us change our initial decisions. In spite of that, human beings plan in numerous situations of their daily life, we prioritise the different possible situations that might occur and plan covering the most likely ones while leaving others to be completed as more information is available.

Planning Under Uncertainty (PUU) is the subfield of AP that studies computationally how to address planning problems in non-deterministic environments. In the early nineties, several extensions to classical AP have been developed to reason about uncertainty in the perception of the environment and in the outcomes of actions' executions. These first systems use probabilistic representation for reasoning about uncertainty and consider the probability of reaching the goals as the plan goodness [62]. At the moment, there are more elaborated PUU planners based on algorithms for solving MDP's but still they precise of an accurate definition of the action model to guarantee good performance. However, the definition of these models by hand in problems with uncertainty is very complex and sometimes impossible. For example when planning the actions of the Mars Rovers [1].

How humans cope with the planning-inherent complexity? Using experience. Going back to the trip planning example, if we had previously visited the city we could use experience about how to get there, which places are worthy, how crowded they are, ... Human beings are endowed with skills that help them plan according to previous experiences. Psychology, biology or philosophy have widely studied the learning processes and tried to explain them. But learning, like intelligence, covers such a broad range of processes that it is difficult to define it precisely. One of the AI pioneers, Herbert Simon, defined learning as "*the changes in a system that allow to carry out a task or a task in the same domain in a more efficient way*" [93]. Regarding this definition one could try to develop AP systems able to modify its performance to carry out its objectives more efficiently.

Machine Learning (ML) is the AI's branch which studies computationally the process of changing the structure of programs and/or data based on its inputs or in response to input information in such a manner that its expected future performance improves. Since the beginning, research in AI has been concerned with ML (as early as 1959 Arthur Samuel developed a prominent program [89] that learned to play better the game of checkers) so, very often, ML has been referred to changes in systems that perform tasks associated with AI, such as perception, robot control or AP. More specifically, before the mid '90s planners were only able to synthesis no more than 10 action plans in many domains and ML was massively used to speed and scale up the existing planning algorithms. Nowadays, planners

achieve impressive performance in numerous domains, but a renewed interest in intersecting ML with AP has arisen with other aims like: to learn automatically the kinds of knowledge humans put in Knowledge-Based planners or to automate the processes of defining, validating and maintenance a domain model. In this Thesis we propose **the application of ML mechanisms to efficiently assist AP to build robust plans in environments with uncertainty even though the domain model is not complete.**

This document is organized as follows: chapter 2 summarizes the literature about deterministic planning, planning under uncertainty and ML to assist planning. Then, chapter 3 presents the objectives of the thesis work. And Finally, chapter 4 describes the methodology to achieve the presented goals.

Chapter 2

State of the Art

This chapter is a review of the main works done in the two research topics addressed in this Thesis: planning under uncertainty and planning assisted by ML. Section 2.1 of this chapter introduces the relevant concepts in AP following a general problem solving approach inspired by the IJCAI'05 tutorial [43]. Section 2.2 presents the main progress in deterministic planning taking as reference the AP textbook [44]. Section 2.3 describes the state-of-the-art techniques to plan in environments with uncertainty based on Jim Blythe's seminal Thesis [10]. Section 2.4 revises and classifies the ML techniques that have been used to improve AP based on the recent summary paper [106]. And finally section 2.5 discusses which are the current needs in planning under uncertainty and how ML can cover them.

2.1 Introduction

The aim of *general problem solving* is to find universal mechanisms that solve any problem that can be described in a high level representation using a general algorithm. Therefore, any *general problem solver* has these three components:

1. *The Conceptual Model*. The description of the problem to solve.
2. *The Representation Language*. The elements used to describe the problem to solve.
3. *The Algorithm*. The technique used to solve the problem.

AP is a form of *general problem solving* concerned with the selection and organization of actions to reach a desired state in a dynamic system. Because of that, AP requires a conceptual model able to describe dynamic systems. Most of the Automated Planning approaches take the state-transitions model as their conceptual model but making several assumptions that make it more operational:

1. *Finite World*. The dynamic system has a finite set of states.
2. *Static World*. The dynamic system stays in the same state until a new action is executed.
3. *Deterministic World*. When an action is applicable its execution brings the dynamic system to a single other state.
4. *Fully Observable World*. There is complete knowledge about the current state of the dynamic system.
5. *Implicit Time*. Actions have no duration so the state transitions are instantaneous.
6. *Restrictive Goals*. The only objective of the planner is to find a set of state transitions that ends at one of the states satisfying the goals.

Deterministic Planning is the AP subfield that studies how to tackle the planning problems that can be described according to this conceptual model. With the aim of addressing more realistic planning problems, some of these assumptions have been tried to be relaxed: **Temporal Planning** studies how to tackle planning problems where actions effects may not be instantaneous. **Scheduling** tackles planning problems where a limited number of resources has to be used in a limited amount of time. And **Planning Under Uncertainty** studies the relaxation of the *Full Observability* and *Deterministic World* assumptions. That is, planning problems where the knowledge of the current state of the problem is incomplete and where the outcome of plans is stochastic.

2.2 Deterministic Planning

Deterministic Planning is the task of finding a sequence of actions to reach a desired state in a deterministic and totally observable environment. Figure 2.1 shows an example of a deterministic planning problem consisting of a robot navigation in a grid. In this example the robot has to reach the cell D2 starting from the cell A5 and avoiding the obstacles (cells B2, D3, and E4). Most of the research done in AP has focused on solving Deterministic Planning tasks. Thanks to that, Deterministic Planning is now a well formalized and well characterized problem with algorithms and techniques that scale-up reasonably well.

2.2.1 The Conceptual Model

The conceptual model for deterministic planning is a deterministic, finite and fully observable state-transition system denoted by $\Sigma = (S, A, C)$, where:

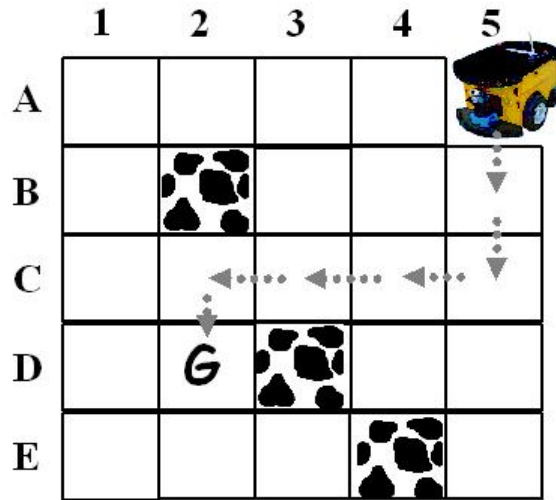


Figure 2.1: Example of a deterministic planning problem.

- S is a finite set of states.
- A is a finite set of actions.
- $C(s, a)$ represents the cost of applying the action $a \in A$ in the state $s \in S$.

According to this conceptual model, a deterministic planning problem is defined as a triple $P = (\Sigma, s_0, G)$ where $s_0 \in S$ is the initial state and $G \subseteq S$ is the set of goal states. Finding a solution to a deterministic planning problem P consists on generating a sequence of actions (a_1, a_2, \dots, a_n) corresponding to a sequence of state transitions (s_0, s_1, \dots, s_n) such that s_i results from executing the action a_i in the state s_{i-1} and $s_n \in G$ is a goal state. The optimal solution for a deterministic planning problem is the one that minimizes the expression $\sum_{i=0}^n c(s_i, a_i)$. Figure 2.2 shows the conceptual model corresponding to the problem of the robot navigation in the cells A4, A5, B4 and B5.

2.2.2 The Representation Languages

A planning representation language is a notation for representing syntactic and semantically planning problems. The planning representation languages are based on variants of first order logic and describe the types, the actions and the states of a planning problem.

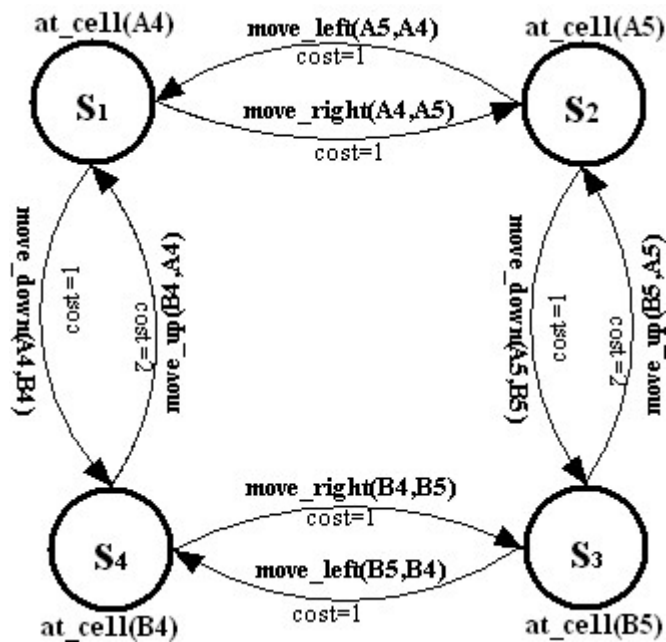


Figure 2.2: Conceptual model corresponding to the robot navigation in the cells A4, A5, B4 and B5.

STRIPS

The language used by STRIPS [35] to represent domains and problems, in spite of being one of the oldest planning representation languages is still very popular. The STRIPS representation for a planning problem is a tuple $P = \langle L, A, I, G \rangle$ where:

- L : represents the literals set.
- A : represents the actions set.
- $I \subseteq L$: represents the subset of literals holding in the initial state.
- $G \subseteq L$: represents the subset of literals that have to be true in a goal state.

The major contribution made by STRIPS is the *Closed World Assumption* to handle the *Frame Problem*¹. The *Closed World Assumption* is a proposal to solve this problem based on assuming that execution of actions only changes a small part of the world. That is, if a situation s_n is the result of applying action a_n ,

¹The *Frame Problem* [68] is the problem of expressing a dynamical domain in logic without explicitly specifying which literals are not affected by an action.

in situation s_{n-1} , then s_n and s_{n-1} , must be very much alike. Implicit in this assumption is that any formula not explicitly asserted in a state is taken to be false. This allows one to avoid explicitly specifying negated atomic formulae.

The STRIPS representation for an action $a \in A$ consists of the following three lists of literals from the set L :

1. The Preconditions List $Pre(a) \subseteq L$: subset of literals that need to be true for the action a to be applied.
2. The Delete List $Del(a) \subseteq L$: subset of literals no longer true after the application of the action a .
3. The Add List $Add(a) \subseteq L$: subset of literals made true by the application of the action a .

Figure 2.3 shows an example of a planning action represented in the STRIPS language. Specifically it is the action `pick-up` from the *Blocksworld*² domain described in the planning domain representation language STRIPS.

```
pick-up (TOP, BOTTOM)
Pre: [emptyhand, clear(TOP), on(TOP, BOTTOM) ]
Del: [emptyhand, clear(TOP), on(TOP, BOTTOM) ]
Add: [holding(TOP), clear(BOTTOM) ]
```

Figure 2.3: Action PICK-UP from the *Blocksworld* planning domain represented in STRIPS.

STRIPS represents a state space S with literals from the L set. A state $s_n \in S$ is the conjunction of literals from L representing the facts holding in the instant n . An initial state $I \in S$ is the subset of literals from L that are holding in the initial instant. And a goal state is any state s_n satisfying the subset of atoms G . According to this notation:

1. The actions applicable in a state s_n are those $a \in A$ such that $Pre(a) \subseteq s_n$
2. The state resulting from applying the action a in the state s_n is $result(s_n, a) = \{s_n - Del(a)\} \cup Add(a)$
3. A solution for a planning problem P is a sequence of applicable actions a_0, \dots, a_n that reaches some goal state s_n starting from the initial state I .

²The *Blocksworld* is a classic domain in AP which consists of a set of blocks, a table and a gripper. The Blocks can be on other blocks or on the table. A block that has nothing on it is clear. The gripper can hold one block or be empty. Because its simpleness and clarity the *Blocksworld* has been by far the most frequently used example in the AI planning literature since the 1960s.

ADL

One of the first extensions to the STRIPS language was the Action Description Language (ADL) [80]. The two main contributions of ADL are (1) the actions' preconditions and the problem goals can be expressed not just as conjunction of literals but also as disjunctions or quantified formulae and (2) actions can have conditional outcomes.

PDDL

The Planning Domain Definition Language (PDDL) [39] is a recent attempt to standardize the planning representation languages in order to allow comparatives among the diverse planners. A PDDL planning problem definition consists of two parts:

1. The Domain Definition: contains an enumeration of the predicates, the actions, the types, the constants and the static facts of the domain.
2. The Problem Definition: contains the objects present in the problem instance, the initial state and the goals. The initial state and the goals descriptions should be ground, meaning that all predicate arguments should be object or constant names rather than parameters. An exception is quantified goals where the quantified variables may be used within the scope of the quantifier.

Figure 2.4 shows the action PICK-UP from the *Blocksworld* domain represented in PDDL.

```
(:action pick-up
:parameters (?top - block ?bottom)
:precondition (and (not (= ?top ?bottom))
                  (forall (?b - block) (not (holding ?b)))
                  (on-top-of ?top ?bottom)
                  (forall (?b - block) (not (on-top-of ?b ?top))))
:effect (and (holding ?top)
             (not (on-top-of ?top ?bottom))))
```

Figure 2.4: Action PICK-UP from the *Blocksworld* domain represented in PDDL.

The development of PDDL is associated to the International Planning Competitions (IPC's). Along the different IPC's this language has evolved to cover the representation needs of the new planning challenges:

1. PDDL1.7 (IPC1 and IPC2) contains the STRIPS and ADL functionality.
2. PDDL2.1 (IPC3) supplements the original PDDL version with:
 - Numeric variables and the ability to test and update their values instantaneously.
 - Durative actions with both discrete and continuous effects.
3. PDDL2.2 (IPC4) extends the previous versions with:
 - Derived predicates which are backward-chaining axioms that allow a planner to achieve a goal by making the antecedent of one of them true.
 - Timed initial literals which are literals that will become true at a predictable time independent of what the planning agent does.
4. PDDL3.0 (IPC5) allows representing preferences and constraints using a restricted temporal logic.

In spite of all these functionalities most of the existing planners do not support full PDDL; in fact, the majority support only the STRIPS subset besides typing and the equality predicate.

2.2.3 The Algorithms

The Algorithms used to solve deterministic planning problems are mainly search algorithms. These algorithms explore a graph systematically trying to find a path to arrive to some goal node n_g starting from a given initial node n_0 . When solving search problems some elements need to be considered:

- *The search space.* In AP the states-space and the plans-space are the two most used search spaces. In the states-space search, each graph node corresponds to a state of the dynamic system and each arc corresponds to a state transition resulting from an action execution. In the plans-space search, the graph nodes are partially specified plans and arcs correspond to plan refinement operations.
- *The search direction.* From the initial node to the goals (forward), from the goals to the initial node (backward) or bidirectional.
- *The search algorithms.* The first planners such as STRIPS or PRODIGY simply implemented a depth first search. Heuristic planning introduced more elaborated search algorithms in the planners development: HSP was

endowed with a hill-climbing algorithm. To solve more problems or to solve them with a better quality HSP2 [11] implemented the complete algorithm (Best-first search). FF [49] was equipped with an improvement to the hill-climbing algorithm in the selection of the local search algorithm, called enforced hill-climbing, which automatically switches to a best-first search in case of failure. Recently divide-and-conquer approaches [105] have been used to reduced the memory requirements drawback of best-first search.

- The heuristics: rather than trying all possible search paths, these algorithms focus on paths that seem to be getting them nearer to a solution. They use an evaluation function $f(n)$ that scores a state s in the search tree according to how close to a goal node they are. The performance of search process is determined by the accuracy of the heuristic function guiding it. Heuristics could be derived as the cost of optimal solutions to relaxed problems. To derive planning domain-independent heuristics automatically the problem relaxations should be extracted from the problem encoding. In AP, the more common relaxations used are:
 1. Ignoring the actions' delete lists,
 2. Reducing the goal sets to subsets
 3. Ignoring certain atoms.

Usually the heuristics handled in AP are non-admissible because the existing admissible ones are poorly informed. Moreover, the application of optimal search algorithms to planning domains is too expensive in terms of computation time.

2.2.4 The Planners

This is a enumeration of the planners that have been specially relevant as they have introduced the major advances in the subfield of Deterministic Planning.

STRIPS

The STRIPS planning algorithm [35] arose in the early 70's. It was based on backward chaining in the state space without using any heuristic. STRIPS worked with a subgoals stack, trying to satisfy first always the subgoal in the top of the stack. The output of this planner was a total order sequence of actions.

UCPOP

UCPOP [81] appeared in the early nineties. It used ADL as a Representation Language. UCPop was based on searching on the plans space instead of in the state space so it could work on any subgoal non linearly and its output was not a totally ordered sequence of actions but a Partial-Order plan.

GRAPHPLAN

GRAPHPLAN [9] builds a state-space graph containing all possible parallel plans up to a certain length; then it extracts a plan by searching the graph backwards from the goals.

SATPLAN

SATPLAN [54] maps planning problems into Boolean satisfiability (SAT) problems and solve them using a state-of-the-art SAT solver. SATPLAN assumes that a given planning problem has a solution of length n , if no solution of this length was found it increases n and considers a new SAT problem. The output of this planner is a totally-ordered sequence of actions.

FF

FF [49] is a forward chaining-heuristic state space-planner. FF used a domain independent heuristic function derived as the cost of relaxed problems. The planning problem relaxation consists of ignoring the delete lists of all actions and extracting an explicit solution using a GRAPHPLAN-style algorithm. The number of actions in the relaxed solutions is used as a goal distance estimate. These estimations control a novel local search strategy, enforced hill-climbing which is a hill-climbing procedure that, in each intermediate state, uses breadth first search to find a strictly better, possibly indirect, successor. As a second heuristic, the relaxed plans are used to prune the search space: usually, the actions that are really useful in a state are contained in the relaxed plan, so one can restrict the successors of any state to those produced by members of the respective relaxed solution (the helpful actions).

SHOP

SHOP [72] is based on ordered task decomposition, which is a type of Hierarchical Task Network (HTN) planning. By problem reduction, the planner recursively decomposes tasks into subtasks, stopping when it reaches primitive tasks that can

be performed directly by planning actions. It needs to have a set of methods that indicate how to decompose non primitive tasks into subtasks.

2.3 Planning Under Uncertainty

When planners are used to provide course of actions to solve real-world problems, they must take into account the fact that they cannot have an exact knowledge of the environment and that actions may have different outcomes. In this cases, planners must balance the potential of some plan achieving a goal state against the risk of producing an undesirable states and against the cost of executing the plan. PUU studies how to extend the deterministic planning models in two different dimensions with the aim of developing more sophisticated planners able to tackle realistic planning problems:

1. *Actions Effects*. In many planning problems it is not reasonable to assume deterministic world dynamics. For instance, when the actions to plan involve the toggling of coins, the rolling of dices, . . . In these cases planners should manage the different possible outcomes of the actions.
2. *World Observability*. In many planning problems handling a complete and perfect description of the state of the environment is inconceivable. For example when planning the activities of a Mars rover. In these cases planners should be able to generate plans even if only limited information of the state is available.

As it is shown in figure 2.5 there are different planning paradigms according to how these two dimensions are extended. The following sections describe them in detail.

OBSERVABILITY	EFFECTS		
	STRIPS	Disjunctive	Probabilistic
Complete	<i>Deterministic</i>	<i>Deterministic</i>	<i>Probabilistic</i>
Partial	<i>Deterministic</i>	<i>Contingent</i>	<i>Probabilistic-Contingent</i>
None	<i>Deterministic</i>	<i>Conformant</i>	<i>Probabilistic-Conformant</i>

Figure 2.5: Planning under uncertainty paradigms.

2.3.1 Probabilistic Planning

Probabilistic Planning is the task of finding a plan that may satisfy a set of goals in a stochastic and completely observable environment. In many planning problems

there is no plan that completely guarantees reaching the goals. In these domains probabilistic planners reason about the likelihood of the actions' outcomes to obtain plans that maximize the probability of reaching the goals.

The Conceptual Model

Probabilistic Planning problems are modelled as stochastic state-transition systems with probability distributions associated to each state transition. This model is denoted by $\Sigma = (S, A, P, C)$, where:

- S is the finite set of states.
- A is the finite set of actions.
- $P_a(s_i|s)$, is the probability that action $a \in A$ executed in state $s \in S$, leads to state an $s_i \in S$. So, for each $s \in S$, if there exists $a \in A$ and $s_i \in S$ such that $P_a(s_i|s) \neq 0$, it is true that $\sum_i P_a(s_i|s) = 1$.
- $C(s, a)$ represents the cost of applying the action $a \in A$ in the state $s \in S$.

According to this conceptual model, a deterministic planning problem is defined as a tuple $P = (\Sigma, s_0, G)$ where $s_0 \in S$ is the initial state and $G \subseteq S$ is the set of goal states. Finding a solution to a deterministic planning problem P consists of generating a sequence of actions (a_1, a_2, \dots, a_n) corresponding to a sequence of state transitions (s_0, s_1, \dots, s_n) such that s_i results from executing the action a_i in the state s_{i-1} and $s_n \in G$ is a goal state. The quality of a solution to a probabilistic planning problem depends on two factors: the cost of the actions $\sum_{i=0}^n c(s_i, a_i)$ and the robustness of the solutions $\prod_{i=0}^n P_a(s_i|s_{i-1})$. Figure 2.6 shows the conceptual model corresponding to the robot navigation problem for the cells A4, A5, B4 and B5 in an environment where the robot can get confused when moving down or left. Specifically in this environment, the robot only succeeds when moving left with probability 0.5 and does nothing with probability 0.5. And it succeeds with probability 0.75 when moving down, doing nothing the rest of the times.

The Representation Languages

The Probabilistic Planning Domain Definition Language (PPDDL) [102] is the standard representation language to describe planning problems in stochastic environments. PPDDL1.0 is essentially a extension of PDDL2.1 [39] to provide support for actions with probabilistic effects. Probabilistic effects are an explicit declaration of the possible outcomes of an action execution. Their syntax is `(probabilistic p_1 o_1 p_2 o_2 p_3 o_3 ...)` meaning that the outcome o_i of the action occurs with probability p_i . It is required that $p_i \geq 0$ and

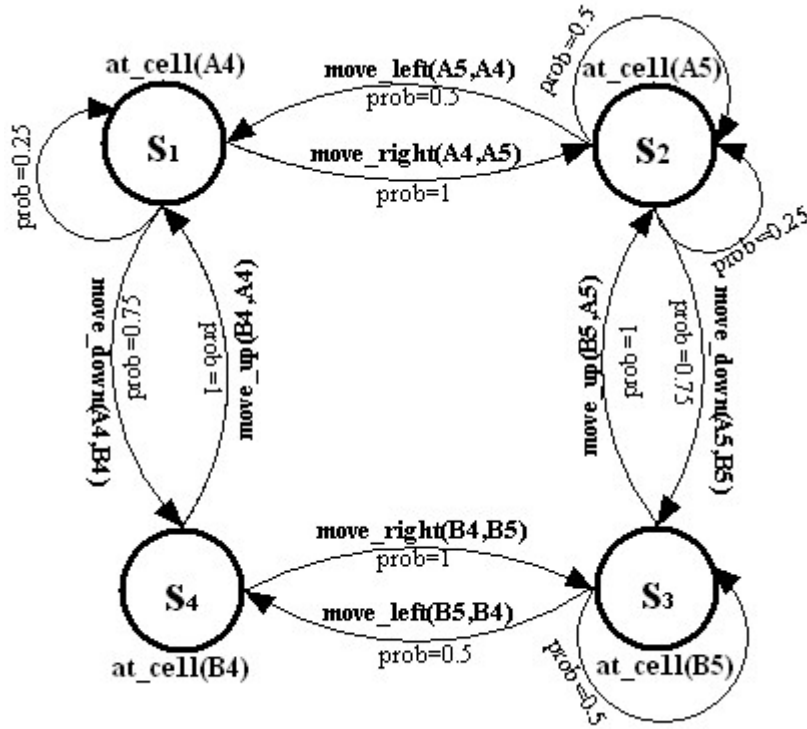


Figure 2.6: Conceptual model of the robot navigation problem for the cells A4, A5, B4 and B5 in a stochastic environment.

$\sum_{i=1}^k p_i = 1$. However, PPDDL1.0 allows a probability-outcome pair to be left out if the effect is empty. In other words, $(\text{probabilistic } p_1 \ o_1 \dots p_2 \ o_2 \ q)$ with $q = 1 - \sum_{i=1}^k p_i$.

PPDDL1.0 allows arbitrary nesting of conditional and probabilistic effects, contrary to popular propositional encodings, such as probabilistic STRIPS operators (PSO's) [63] and factored PSO's [16], which do not allow conditional effects nested inside probabilistic effects. While arbitrary nesting does not increase the expressiveness of the language, it can allow for exponentially more compact representations of certain effects given the same set of state variables and actions [86]. However, any PPDDL action can be translated into a set of PSOs with at most a polynomial increase in the size of the representation. Consequently, it follows from the results of Littman [66] that PPDDL is representationally equivalent to dynamic Bayesian networks, which is also another popular representation of probabilistic planning problems. Markovian rewards, associated with state transitions, can be encoded using fluents. PPDDL reserves the fluent `(reward)` to represent

the total accumulated reward since the start of execution. Action preconditions and effect conditions are not allowed to refer to the reward fluent, which means that the accumulated reward does not have to be considered part of the state space. The initial value of reward is zero. These restrictions on the use of the reward fluent allow a planner to handle domains with rewards, without having to implement full support for fluents.

As an example, Figure 2.7 shows the Probabilistic Action PICK-UP from the *Blocksworld* domain of the IPC4 represented in PPDDL1.0. This action indicates that with probability 0.75 a robot arm will pick-up successfully the block ?top from the block ?bot and that with probability 0.25 the block ?top will fall down.

```
(:action pick-up
:parameters (?top - block ?bot)
:precondition (and (not (= ?top ?bot))
                  (forall (?b - block)
                    (not (holding ?b)))
                  (on-top-of ?top ?bot)
                  (forall (?b - block)
                    (not (on-top-of ?b ?top))))
:effect (and (decrease (reward) 1)
             (probabilistic 0.75 (and (holding ?top)
                                     (not (on-top-of ?top ?bot)))
                                0.25 (when (not (= ?bot table))
                                     (and (not (on-top-of ?top ?bot))
                                           (on-top-of ?top table))))))
```

Figure 2.7: Action PICK-UP from the probabilistic *Blocksworld* domain represented in PPDDL1.0.

The algorithms

Regarding the IPC probabilistic planners there are two main approaches to tackle probabilistic planning problem:

- Complementing an extension of a classical planner with plan repairing techniques [36]. The first attempts to face probabilistic planning problems followed this approach; most were extensions to partial order planning [76]. But in recent years this work has continued over the new deterministic planning paradigms.

- Finding policies (mappings between world states and the preferred action to be executed to achieve the goals) through the optimisation of a given function [83]. These planning systems model the dynamics of the environment as a Markov Decision Process (MDP) and find policies optimising a numeric function, called utility function, which gives preference to the different states and transitions of the MDP. Classic dynamic programming algorithms solve MDP's in time polynomial in the size of the state space. However, the size of the state space grows exponentially with the number of features describing the problem. This state explosion problem limits the use of the MDP framework, and overcoming it has become an important topic of research. Over the last years, two approaches that do not rely on complete state enumeration have been developed:
 1. On one hand there is exploiting relational representation of the MDP's to create state abstractions that allow the problem to be represented and solved.
 2. On the other hand there is reachability analysis that limit computation to the states that are reachable from the initial state of the problem.

The planners

The probabilistic planners that participated in the probabilistic competition of IPC4 and IPC5 are.

- SYMBOLIC LAO* [12]. The IPC4 winner based on symbolic model checking techniques that combines the symbolic LAO* and symbolic RTDP algorithm.
- MGPT [33] based on heuristic search for solving MDP models. It uses the algorithm *Labelled Real-Time Dynamic Programming*(LRTDP) which is a heuristic-search algorithm that implements a labelling scheme on top of the RTDP algorithm to get a faster convergence time together with heuristic automatically extracted from the problem representation.
- NMRDPP [46] translated the planning problem into a problem of solving a decision process with no Markovian Rewards.
- FCP [53] based on the First-Order Value Iteration Algorithm(FOVIA) a symbolic counterpart of the classical value iteration algorithm for solving MDP's.
- LEARNING POLICIES FROM RANDOM WALKS [96]. Based on a variant of approximate policy iteration that combines inductive machine learning and simulation to perform policy improvement.

- FPG [20]. The winner of the IPC5 uses gradient ascent for direct policy search and factors the parameterised policy by using a function approximation for each action.
- PARAGRAPH [65] extends the Graphplan framework to probabilistic planning by introducing a node for each of an action's possible outcomes, so that there are three different types of nodes in the graph: proposition, action, and outcome.
- FOALP [90] translates the probabilistic planning problem into a First-Order Markov Decision Process (FOMDP) and uses approximate solution techniques for FOMDP's to derive a utility function using first-order extensions of approximate linear programming.
- SFDP uses a compact factored representation of MDP's based on Algebraic Decision Diagrams and a symbolic dynamic programming algorithm that focuses the policy on goal states.

In spite of all these different approaches the best overall performance in IPC4 and IPC5 was achieved by a planner simply based on complementing the deterministic planner FF with replanning to cope with actions failures. FF-REPLAN achieved such a good results in both IPC4 and IPC5 because the optimal policies for the test benches of the two competitions were very close to the ones obtained with the deterministic planner FF.

2.3.2 Contingent Planning

Contingent planning is the task of solving a planning problem in environments where the current state is not completely defined, but it is possible to observe some aspects of the current state during the plan execution. Figure 2.8 shows an example of a contingent planning problem. In this example a mobile robot has to plan its actions to navigate from the cell A5 to the goal cell C2 avoiding obstacles. In this example the robot has no knowledge about the content of cells C2, C3, D3 and D4 when it gets close to them it can sense their contents and cross them if there is no obstacle in them.

The Representation Languages

Contingent Planning needs a representation language that is able to represent deterministic planning problems with two other extra functionalities: it has to be able to define the initial state of a problem as a disjunction of literals called belief state. And it has to be able to describe the acquisition of information about the

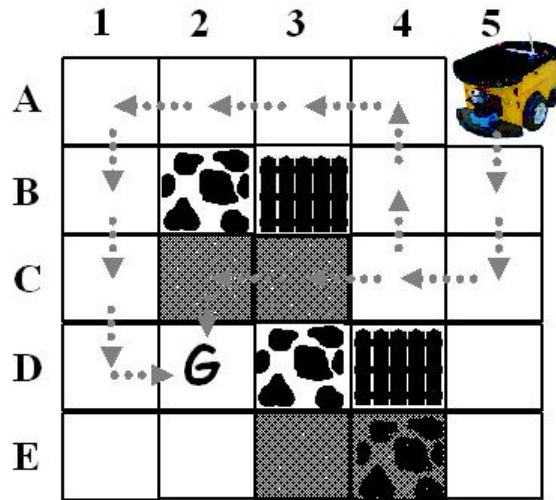


Figure 2.8: Example of a contingent planning problem.

current state in execution time. So, usually contingent planning problems are represented in a standard deterministic planning domain description language –like PDDL– extended with sensing actions.

Sensing actions are planning actions with effects representing some information observed from the current state. A planner should choose Sensing Actions when the lack of information can prevent it to achieve the problem goals. One of the first planners that included sensing actions was CNLP [82]. This system allowed to represent uncertain information in planning time with the predicate function `unknown`. And sensing actions were used to know if some literal is true or false during execution time. Figure 2.9 shows an example of a sensing action for the CNLP planner. In this action `effect1` and `effect2` are mutually exclusive and cover all the possible outcomes for the observation.

```
(:observation test-robot-handempty
  pre: (unknown (handempty))
  effects1: (handempty)
  effects2: (not (handempty)))
```

Figure 2.9: Sensing Action to check in the *Blocksworld* whether the robot hand is empty or not.

The Planners

- CONTINGENT-FF [48] extends the FF PLANNER with the ability to treat initial state uncertainty expressed in the form of a CNF formula. The output of Contingent-FF are tree-shaped plans with branches.
- SGP [99] is an extension of GRAPHPLAN to handle sensing actions.
- CASSANDRA [84] is a partial-order, contingency, domain-independent problem solver architecture based on the Deterministic Planner UCPOP.

2.3.3 Conformant Planning

Conformant planning is the task of finding a safe plan in a non observable environment. Conformant planners have to find a sequence of actions able to achieve a goal state for all possible contingencies without any sensing during the plan execution. Figure 2.10 shows an example of a conformant planning problem consisting of a robot navigation in a grid. In this example the robot has to reach the cell D2 starting from the cell A5 and avoiding the obstacles. In this case, as the robot is not able to perceive the contents of the cells C2,C3 and E3 it has to avoid them to generate a conformant solution.

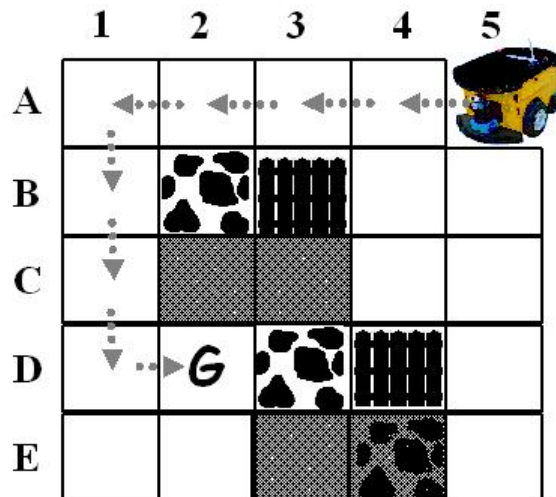


Figure 2.10: Example of a conformant planning problem.

The Conceptual Model

As actions are deterministic and no observation is possible conformant planning problems can be modelled with the same state-transition system used in deterministic planning problems (Figure 2.2). According to this conceptual model, a conformant planning problem is defined as a triple $P = (\Sigma, I, G)$ where $I \subseteq S$ is not a state –as in deterministic planning– but a set of possible initial states called belief state and $G \subseteq S$ is the set of goal states. Finding a solution to a conformant planning problem P consists of generating a sequence of actions (a_1, a_2, \dots, a_n) corresponding to a sequence of state transitions (s_0, s_1, \dots, s_n) such that s_i results from executing the action a_i in the state s_{i-1} and $s_n \in G$ is a goal state for all the possible $s_0 \in I$.

The Representation Language

Conformant planning problems are described using a standard deterministic planning representation language but specifying the set of possible initial states of the problems as a disjunction of literals. In IPC5 the conformant planning competition took place for the first time and PDDL was used as the common representation language. Figure 2.11 shows a conformant problem from the *Blocksworld* domain of the IPC5.

The Algorithms

There are three main approaches to tackle the conformant planning problem:

- Via Model Checking Planning. This approach relies on the symbolic representation Binary Decision Diagrams (BDD's).
- Extending heuristic planning techniques to search in the belief space.
- Compiling the conformant planning problems into other kinds of problems such as SAT or deterministic planning problems.

The Planners

This is a enumeration of the conformant planners that participated in the non-deterministic track of the IPC.

- BURIDAN [62] takes as input a probability distribution over states and produces a plans such that the probability to reach the goals after the plan execution is no less than a given threshold.

```

(define (problem p01)
  (:domain conformant-blocks)
  (:objects A B - block)
  (:init
    (and (oneof (handempty) (holding A) (holding B))
         (oneof (holding A) (clear A) (on B A))
         (oneof (holding A) (ontable A) (on A B))
         (oneof (holding B) (clear B) (on A B))
         (oneof (holding B) (ontable B) (on B A))

         (or (not (handempty)) (not (holding A)))
         (or (not (handempty)) (not (holding B)))
         (or (not (holding A)) (not (holding B)))
         (or (not (holding A)) (not (clear A)))
         (or (not (holding A)) (not (on B A)))
         (or (not (clear A)) (not (on B A)))
         (or (not (holding A)) (not (ontable A)))
         (or (not (holding A)) (not (on A B)))
         (or (not (ontable A)) (not (on A B)))
         (or (not (holding B)) (not (clear B)))
         (or (not (holding B)) (not (on A B)))
         (or (not (clear B)) (not (on A B)))
         (or (not (holding B)) (not (clear B)))
         (or (not (holding B)) (not (on A B)))
         (or (not (clear B)) (not (on A B)))
         (or (not (on A B)) (not (on B A))))))
  (:goal (and (ontable A) (on B A))))

```

Figure 2.11: Definition of a Conformant planning problem from the *Blocksworld* domain of the IPC5.

- CONFORMANT-GRAPHPLAN (CGP) [94]. It develops separate plan graphs for each different possible state of the environment.
- CMBP [25] is based on the planning via model checking paradigm, relies on symbolic techniques such as Binary Decision Diagrams to compactly represent and efficiently analyse a planning domain.
- CONFORMANT-FF [17] performs a forward heuristic search in belief space guided by an extension of the FF's heuristic to the conformant setting using SAT-based techniques to reason about uncertainty.
- POND [19]. Performs a forward search in the space of belief states, guided by a relaxed plan heuristic. It estimates the conformant plan distance between the belief state(s) at the end of its current plan prefix and a goal belief

state. The distance between belief states is taken as an aggregate measure of the underlying distances between states in the belief states.

- CF2CS [78]. The winner of the IPC5 conformant planning track. This planner maps conformant planning problems into deterministic problems which are then solved by the off-the-shelf classical planner FF.

2.3.4 Probabilistic-Contingent Planning

Probabilistic-contingent planning is the task of generating a contingent plan in environments where actions can take the system to different possible states determined by a probability distribution.

The Conceptual Model

The conceptual model for probabilistic-contingent planning problems is a partially observable stochastic system. It is modelled as a non deterministic state transition system that can be represented as a tuple $\Sigma = (S, A, P, O, C)$, where:

- S is a finite set of states.
- A is a finite set of actions.
- $P_a(s_i|s)$, is the probability that action $a \in A$ executed in state $s \in S$, leads to state an $s_i \in S$. So, for each $s \in S$, if there exists $a \in A$ and $s_i \in S$ such that $P_a(s_i|s) \neq 0$, it is true that $\sum_i P_a(s_i|s) = 1$.
- O is a finite set of observations with probabilities $P_a(o|s)$, for any $a \in A$, $s \in S$ and $o \in O$. $P_a(o|s)$ represents the probability of observing o in the state s after executing action a . It is required that the probabilities are defined for each state $s \in S$ and action $a \in A$ and that, given a state s and an action a , $\sum_{o \in O} P_a(o|s) = 1$. There is no other way to obtain any information about the state than through observation.
- $C(s, a)$ represents the cost of applying the action $a \in A$ in the state $s \in S$.

Instead of handling the exact state of the dynamic system, the probabilistic-contingent planners handle a probability distribution over states called *belief state*. Given a belief state b , $b(s)$ is the probability assigned to the state $s \in S$ by the belief state b . Considering b_a as the new belief state resulting from the execution of an action $a \in A$, the probability $b_a(s)$ can be computed as the sum of the probability distribution determined by b weighted by the probability that action a leads from $s' \in S$ to s :

$$b_a(s) = \sum_{s' \in S} P_a(s|s')b(s')$$

The Representation Language

Probabilistic-contingent planning needs a representation language that is able to describe probabilistic actions and the acquisition of information about the current state at execution time. This can be represented with a probabilistic planning domain description language –like PPDDL– extended with sensing actions.

The Algorithms

A probabilistic-contingent planning problem can be seen as solving Partially Observable MDP's (POMDP's). POMDP's theoretically could be solved extending the algorithms for MDP's over belief states. But computationally, this is intractable because the resulting space of belief states is infinite and continuous. Currently, there are methods developed that approximate solutions for POMDP's, but most POMDP's are computationally intractable to solve exactly for optimal behaviour.

2.3.5 Probabilistic-Conformant Planning

Probabilistic-conformant planning is the task of generating safe plans in non-observable environments where actions can bring the system to different possible states determined by a probability distribution and no sensing can be done during plan execution.

The Conceptual Model

The Conceptual Model for the Probabilistic Conformant Planning is a non-observable and non-deterministic state-transition system, i.e., a stochastic system where no information about the world state can be obtained.

The Representation Language

Probabilistic-Conformant planning problems can be described in PPDDL specifying the initial state through a disjunction of literals.

The Planners

- PROBAPOP [77] is based on generating base plans with the deterministic partial-order planner VHPOP and then refine the plans in the search queue until finding a solution plan that meets or exceeds the probability threshold.

- COMPLAN [50] performs depth-first branch-and-bound search in the plan space. For each potential search node, an upper bound is computed on the success probability of the best plans under the node, and the node is pruned if this upper bound is not greater than the success probability of the best plan already found. A major source of efficiency for this algorithm is the efficient computation of these upper bounds, which is possible by encoding the original planning problem as a propositional formula and compiling the formula into deterministic decomposable negation normal form.

2.4 Learning to Improve Automated Planning

Before the mid '90s domain independent planners could only synthesize no more than 10 actions plans in an acceptable time. Along those years there was a massive dependence on speed-up techniques to solve planning problems and the definition of search control through ML became a very popular solution to accelerate planning processes.

In the late 90's a significant scale-up in planning happened: the appearance of powerful reachability heuristics based on solving a relaxed task in each single search state using a Graphplan [9] made domain independent planners synthesize 100 actions plans just in seconds. As a direct consequence, the planning community decreased its interest in the ML techniques.

In the last decade we are living a renewed interest in applying ML to assist the planning processes. This renewed interest is mainly because of three factors: first, the results of the IPC3 showed that knowledge-based planners outperform significantly domain-independent planners. The development of techniques that automatically define the kind of knowledge that humans put in these planners would bring great advances in the field. Second, there is a current need for automatic tools that assist in the definition, validation and maintenance of planning domain models. Along the last years planning applications have been implemented successfully but the process of specifying their domain models is still done by hand. And third, domain-independent planners are still not able to cope with complex problems. Currently they are solved by defining ad hoc planning strategies by hand. ML can be used to automatically learn these strategies.

Mitchell's ML textbook [70] points out the questions that have to be raised when applying ML to improve the performance of any automatic processes:

1. *The choice of the target concept.* The target concept will exactly determine the type of knowledge that the ML process will learn. When talking in terms of planning this knowledge can be: **search control knowledge, domain-specific planners, planning domain models, ...**

2. *The representation for the acquired knowledge.* The representation for the acquired knowledge –**heuristics, policies, actions’ schema**, ... – have to harmonise with the used planning technique: forward or backwards chaining, state-space or plan-space, heuristic or non-informed search, ...
3. *The extraction of the experience.* When planning there are three different opportunities where learning processes can take place: **before** the planning starts, **during** the planning process or **after** the planning process, i.e., during the plans’ executions.
4. *The choice of the learning approach:*
 - **Inductive Learning.** These learning techniques induce general theories based on observations. In this approach, the input to the learning process is a set of observed instances and the output is a classification method consistent with them used to classify the subsequent instances. Inductive Learning can be broken into Supervised, Unsupervised and Reinforcement Learning. In **Supervised Learning** inputs and correct outputs are ‘a priori’ known. In **Unsupervised Learning** just the inputs are specified. And in **Reinforcement Learning** the inputs and a long-term feedback signal from a external source are given.
 - **Analytical Learning.** These techniques use prior knowledge and deductive reasoning to explain the information provided by the training examples. Because of that, analytical techniques are not so constrained to the requirement of training examples to achieve a good level of generalization accuracy.
 - **Inductive-Analytical Learning.** Purely inductive learning methods formulate general hypotheses by finding empirical regularities over the training examples. Purely analytical methods use prior knowledge to derive general hypothesis deductively. This learning approach combines the two methods to obtain the benefits of both: a better generalization accuracy when prior knowledge is available and reliance on observed training data to overcome shortcomings of prior knowledge.
5. *The Learning Environment.* Different learning methods are used according to the features of the environment. For example, learning to assist planning in **continuous** environments involves using regression methods; learning in **partially observable** environments means using learning methods able to cope with incomplete examples, learning in **stochastic** environments implies coping with noise, ...

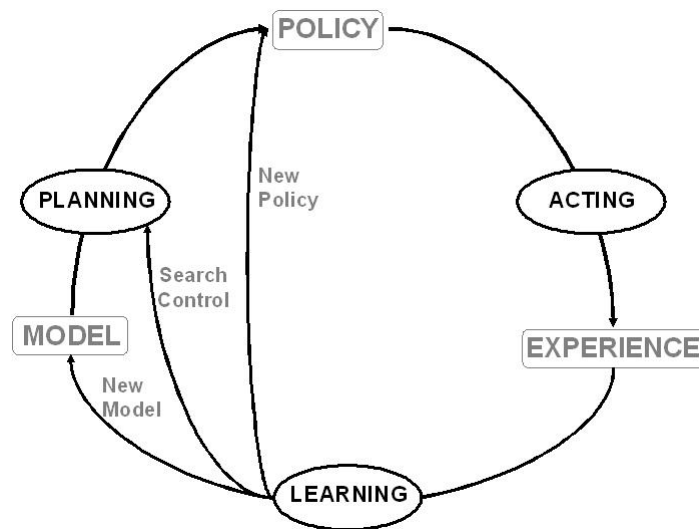


Figure 2.12: Integration of learning search control, learning planning models and policy learning with the planning process.

The following sections are a description of the main ML techniques used to assist AP processes classified according to their learning target concept: (1) **search control learning**, (2) **planning model learning** and (3) **learning domain-specific planners**. Figure 2.12 illustrates how these learning processes are integrated with AP.

2.4.1 Learning Search Control Knowledge

This section revises the different systems that automatically learn search control knowledge to improve the speed and/or the quality of the planning processes. The speed-up techniques are based on making the planner avoid unpromising portions of the search space without exploring it and the quality improvement methods are based on biasing the search process towards the kind of solutions preferred by the user.

Learning Macro-Actions

Macro-actions are the first attempt to speed-up the planning process. They are extra actions added to a given domain theory resulting from combining the actions that are frequently used together.

The use of macro-actions reduces the depth of the search tree. More precisely, learning macro-actions is quite useful in domains with problems that can be de-

composable in non serializable goals. But the benefits of using macro-actions decreases with the number of new macro-actions as they enlarge the branching factor of the search tree causing the *utility problem*. To decrease this negative effect filters deciding the applicability of the macro-actions should be defined.

Since the beginning of AP the use of macro-actions has been widely explored. Traditionally most of the techniques use an off-line approach to generate and filter macro-actions before using them in search. Early works on macro-actions began with a version of the STRIPS planner [34]. It used previous solution plans and segments of the plans as macro-actions to solve subsequent problems. Later, MORRIS [59] extended this approach by adding some filtering heuristics to prune the generated set of macro-actions. This approach distinguished between two types of macro-actions: S-macros that occur frequently during search and T-macros, those that occur less often, but model some weakness in the heuristic. Minton observed that the T-macros, although used less frequently, offered a greater improvement in search performance. The REFLECT system [29] took the alternative approach of forming macro-actions based on pre-processing of the domain. All sound pairwise combinations of actions were considered as macro-actions and filtered through some basic pruning rules. Due to the small size of the domains with which the planner was reasoning, the number of macro-actions remaining following this process was sufficiently small to use in planning.

More recent works on macro-actions include Macro-FF [15]. In this work Macro-actions are extracted in two ways: from solution plans; and by the identification of statically connected abstract components. Besides, an off-line filtering technique is used to prune the list of macro-actions. [74] uses a genetic algorithm to generate a collection of macro-actions, and then filters this collection through an off-line filtering technique similar to that used by Macro-FF. Marvin [28] is a heuristic planner that competed in IPC4 using action-sequence-memoisation techniques to generate macro-actions on-line that allow the planner escape from plateaus without any exploration.

Learning Control Rules

A control rule is an IF-THEN rule to guide the exploration of the planning search tree. Control rules guide the exploration in two different ways: pruning portions of the search space or ordering the nodes exploration.

On one hand there are systems that inductively learn control rules. Among these systems Inductive Learning Programming (ILP) is the most popular learning technique. The GRASSHOPPER system [64] used the FOIL [85] ILP system to learn control rules that guide the PRODIGY planner when selecting goals, operators and binding operators. GRASSHOPPER used as learning examples previous decisions in search traces from solving similar planning problems. Purely induc-

tive systems need a big number of learning examples to acquire efficient control knowledge. Besides, the quality of the acquired knowledge completely depends on the learning examples. When the learning examples are not significant enough the induced control rules will mislead the search process of the planner.

On the other hand there are analytical systems. STATIC [32] that obtains control rules without any training example just using Explanation Based Learning (EBL) to analyse the relations between actions' preconditions and effects. The main disadvantage of these methods is that, as there are no training examples, there is no measure of the learned knowledge utility.

To solve the problems of the purely inductive and purely analytical approaches some researchers have tried to combine them: the pioneering systems based on this principle are LEX-2 [69] and META-LEX [55]. AXA-EBL [27] combined EBL + induction. It first learns control rules with EBL and then refines them with training examples. DOLPHIN was an extension of AXA-EBL which used FOIL [103] as the inductive learning module. The HAMLET [14] system combines deduction and induction in an incremental way. First it learns control rules with EBL, that usually are too specific and then uses induction to generalize and correct the rules. EVOCK [2] applies the genetic programming paradigm to solve the learning problems caused by the hypothesis representation language.

Learning Cases

Cases are past experiences which are memorized in order to assist in the solution of future problems. In general terms, a case can be described as the tuple $C=(Problem, Solution, Effects)$ where *problem* is the problem solved in a past episode, *solution* is a description of the way the *problem* was solved and *effects* is a description of the new situation resulting from applying the *solution* to the *problem*.

By developing the first case-based planner (CHEF), Hammond helped to define the case-based approach to problem solving and to explanation [47]. Given a set of goals and a current situation, the first task for CHEF was to find an old plan that solved a past problem that is similar to the current problem. The next tasks were to adapt the old solution to fit the new circumstances and to store the new solution so that it can be reused in the future. However in addition to old plans, Hammond illustrated the use of memory for plan adaptation, plan repair, and failure anticipation. PRODIGY/ANALOGY [97] introduced the application of derivational analogy and abstraction to planning. This system stored planning traces to avoid failure paths in future exploration of the search tree. To retrieve similar cases, PRODIGY/ANALOGY indexed them using the minimum preconditions to reach a set of goals. The case-based planning system PARIS [7] proposed the introduction of abstraction techniques to store the cases organised in a

hierarchical memory. This technique improves the flexibility of the cases adaptation, thus increasing the coverage of a single case. Nowadays case based systems try to apply CBR techniques to assist the search process of the state-of-the-art planners. According to this idea, the SAYPHI planner [30] uses CBR to reduce the number of explored nodes in the FF planner.

Learning Heuristic Functions

Most of the state-of-the-art planners are based on heuristic search over the state-space (12 over the 20 IPC5 participants). The performance of these planners depend strongly on defining a domain-independent heuristic function that provides good guidance across the wide range of different domains. But as Jorg Hoffman shows in [49] there are several domains where these heuristic functions underestimate the distance to the goal leading to poor guidance.

In [101] SungWook Yoon et al. proposed using an inductive approach to correct the domain-independent heuristic in those domains based on learning a supplement to the heuristic from observations of solved problems in these domains.

Learning Hierarchical Control Knowledge

The input to a hierarchical planner includes a set of operators similar to the ones used in classical planning and a set of methods describing how tasks should be decomposed into subtasks in this particular domain. The specification of the hierarchical methods by hand is a hard task as expert knowledge is needed. Learning how to automatically define them is still an open issue.

The first system to automate the construction of abstraction hierarchies for planning was ABSTRIPS [88]. In ABSTRIPS, the abstraction spaces are constructed by assigning number indicating the relative difficulty of achievement of the preconditions of each operators (criticalities). First ABSTRIPS finds an abstract plan that satisfies only the preconditions with the highest criticalities. Then the abstract plan is refined by considering the preconditions of the next levels of criticality and inserting steps to the plan to satisfy these preconditions until the plan considers the lowest criticality level.

The ALPINE system [58] completely automates the generation of abstraction hierarchies from the definition of a problem space. Each abstraction space in a hierarchy is formed by dropping literals from the original problem space, thus it abstracts the preconditions and effects of operators as well as the states and goals of a problem.

The HICAP [71] architecture integrates a CBR system NACODAE [18] with the hierarchical planner SHOP. NACODAE interacts with users to extract a stored similar case that allows to solve the current problem. The system CASEADVISOR

[23] stores plans to get information about how to solve a task instead of choosing the refinement method. The CAMEL system [52] learns the expansion methods for an HTN planner in military domains.

2.4.2 Learning Domain-Specific Planners

An alternative approach to learning planning search control consists of learning domain-specific planners, i.e., programs that generate output plans to solve problems in a particular domain. This section reviews the diverse techniques to automatically learn to solve planning problems of a particular domain given a number of solved instances.

Learning General Policies with Decision Lists

A policy is a mapping between the world states and the preferred action to be executed in order to achieve a given set of goals. The problem of learning general policies was first studied by Roni Khardon. Khardon's approach, L2ACT [57], induced policies using a variation from the decision list learning algorithm [87]. This first approach had two main weaknesses: (1) it relied on background knowledge that expressed key features of the domain, and (2) the resulting policies do not generalize well. Hector Geffner in [67] proposes to solve the weaknesses of the Khardon's approach representing the general policies in a different way, specifically in a concept language [42]. Concept languages have the expressive power of fragments of standard first-order logic but with a syntax that is suited for representing and reasoning with classes of objects.

Learning General Policies with Relational Decision Trees

Inducing decision trees is a very popular ML technique for classification. In a decision tree the leaves nodes contain the classification values and the internal nodes contain tests to divide the examples according to values of their attributes. To make a classification with a decision tree one starts in the root of the tree and applies the root's test to the example. Then one takes the branch that corresponds to the outcome of the test in the example and propagates the example to the corresponding subtree. If the resulting subtree is a leaf node, one gets the classification value; otherwise one applies the procedure recursively on the example and the subtree. Unlike traditional decision trees, relational decision trees work with examples described in a relational language such as logic predicates. This means that each example is not described by a single feature vector but by a set of facts. This way, in relational decision trees test nodes do not contain tests about values of examples' attributes but queries about the facts holding in the examples.

Planning the navigation of a mobile robot is a complex problem as environments are usually not totally observable and the information obtained through sensor is frequently noisy. [91] and [26] propose the use of relational decision trees to learn which action should be executed according to the observed current state.

Learning General Policies with Genetic Programming

The conventional Genetic Algorithms [13] used evolution-inspired techniques to manipulate and produce fixed-length chromosome strings that encoded solutions to problems. Genetic programming (GP) is an automatic programming technique developed by Koza [60] that extended the GA techniques to automatically manipulate and produce computer programs. Lee Spector applied in [95] GP to learn policies to solve planning problems in the blocksworld domain.

Learning General Policies with Reinforcement Learning

Reinforcement Learning (RL) include a set of learning techniques to compute the optimal policy to reach a given goal by exploring the space state through trial and error. In RL each learning experience consists of the action executed and the reward received. The optimization methods used to obtain the best policy are dynamic programming by Bellman and the control theory developed in the mid 50's [4, 5, 61]. The major important benefit of these techniques is that can be used to solve problems whether the actions model is known or not. In these cases RL can follow two approaches: (1) *Model-Based methods* that try first to learn the model of the environment and then apply dynamic programming methods to obtain the optimal policy or (2) *Model-Free Methods* apply directly RL to obtain the optimal policies.

Most work on RL has focused on propositional representations for the state and the actions, so RL was not applicable to domains with a relational structure -as the planning domains- without extensive feature engineering. Recently there have been a few successful attempts to generalize RL to the relational setting [31, 56]. These attempts are based on combining RL with the new relational learning techniques such as relational decision trees. Due to the use of a more expressive representation language to describe states, actions and utility functions, RRL can be potentially applied to planning tasks. But the problem of both RL and RRL is that acquired policies, as they do not explicitly include knowledge about the problem goals, do not generalise well the acquired knowledge, so every time a new goal has to be achieved, new policies have to be learnt again, even if the dynamics of the environment did not change.

2.4.3 Learning Domain Models

This section reviews the planning systems that use ML techniques to automatically acquire knowledge about domain models.

Domain Analysis

Domain Analysis include the ML techniques that acquire knowledge to improve the speed and the quality of the planning process by analysing the description of the problem before any planning has been done.

Huffman, Pearson and Laird in [51] proposed a system to detect errors in planning domain descriptions searching for too general or too specific preconditions, incomplete effects or missing operators. The RIFO [73] system automatically detected irrelevant information to state based planners. Literals were considered irrelevant if they are not necessary to be true or false to obtain the solution plan. TIM [37] extracted automatically the objects types according to their functionality. This information is useful to find symmetries in typed objects. SYMMETRICSTAN [38] took advantage of this to save planning time for the GRAPHPLAN planner. This way if GRAPHPLAN chooses an action that results in a failure path SYMMETRICSTAN prevents GRAPHPLAN from choosing its symmetric actions.

Learning Actions' Schema

These techniques learn the parameters, preconditions and/or the effects of planning actions from past episodes of actions executions. Learning action's schema is a classic problem in AI so there are numerous approaches that have addressed it. We revise the major ones according to two different dimensions: determinism and observability.

- In *deterministic and totally observable environments* this problem has been well studied. The LIVE system [92] learns operators with quantified variables while exploring the world. The EXPO system [45] refines incomplete planning operators when the monitoring of a plan execution detects a divergence between internal expectations and external observations. [98] learns PRODIGY operators from the execution traces of the resulting solutions or execution failures. The TRAIL system [6] limits its operators to an extended version of Horn clauses so that ILP can be applied. The LOPE system [41] proposed an integrated framework for learning, planning and executing actions.
- In *stochastic and totally observable environments*. Probabilistic world dynamics are commonly represented using graphical models, such as Bayesian

networks (BNs) [40]. However, these representations do not make any assumptions tailored towards representing planning actions dynamics. In [79] Leslie Kaelbling et al. presented the first specific algorithm to learn probabilistic planning operators.

- In *partially observable environments* there has been less work. The ARMS system [100] learns planning actions through encoding example plan traces as a weighted maximum satisfiability problem, from which a candidate STRIPS action model is extracted. A data-mining style algorithm is used in order to examine only a subset of the data given to the learner, so the approach is approximate by nature. Hidden Markov Models can be used to estimate a stochastic transition model from observations. However, the more complex nature of the problem prevents scaling up to large domains. These approaches represent the state transition matrix explicitly, and can only handle relatively small state spaces. In [3] Eyal Amir introduced an algorithm to exactly learn actions in partially observable STRIPS domains. But there is no general efficient approach yet to cope with the problem of partial observability in stochastic environments.

2.5 Discussion

The review of the existing deterministic planning techniques reveals there are still two important open issues to be addressed:

- **Domain independent planners are not currently able to address realistic problems.** Most planning systems require expert knowledge that help them face realistic planning problems, i.e. control rules to handle user preferences, restriction, quality metrics, However, in many of these problems, such expert knowledge may not be completely available; this is partly because it is very hard to compile such knowledge, and it is partly because there is no expert to provide it, e.g., in the space missions.
- **Planning domain models are hard to design, validate and maintain.** The process of encoding a planning domain model is laborious. Diverse bugs can be in models for a long time. At the present, most of the domain model designers use only planners to develop and debug domain models, but planners are not designed for this purpose. The International Competition on Knowledge Engineering for Planning and Scheduling (ICKEPS) takes place for the second time in September of 2007 to motivate developments of specific automated tools to assist in designing, validating and maintaining planning domains.

Besides that, in the subfield of PUU there are some more challenges that have to be faced:

- **Probabilistic planners only obtain robust plans when they have an accurate description of the environment.** But designing, validating and maintaining non-deterministic planning domain models is very hard. Even in simple planning domains like the blocksworld, when actions have stochastic behaviour they may result in innumerable different outcomes. Specifying by hand all of them and their probabilities is intractable, and if at a given instant it is viable, they could vary over time.
- **Current probabilistic planners do not find robust plans on a wide set of benchmarks.** The existing optimal approaches based on solving MDP's are only able to solve toy problems and the suboptimal ones based on reachability analysis or relational representation of the MDP's do not scale well.

Providing planners with learning abilities promises to be a successful approach to face some of these problems, but at the moment deeper studies have to be done:

- **The current learning-based planners are not competitive with state-of-the-art non-learning planners.** Most of the learning-based planners, except recent approaches based on learning macro-actions and heuristics [15, 101, 28] are not competitive with the state-of-the-art domain independent planners with respect to performance on a wide set of benchmarks. One of the main reasons is that frequently, the learning-based planners are not evaluated very well in most studies, typically the evaluation is only focusing on a very small number of domains. So this planners are usually quite brittle when encountering new domains.
- **The model learning task has not been solved in stochastic environments.** There is a lot of research work done in learning actions models from observations in deterministic environments. But when the environment is stochastic, there are just few works that only can learn simple probabilistic actions model with no conditional effects [79].

Chapter 3

Objectives

This chapter is a description of the objectives established for this Thesis.

The main aim of the thesis is providing an answer to the main open issues in PUU based on complementing the planning process with learning abilities. According to this, **the overall objective of the Thesis is the development of ML mechanisms to efficiently assist AP to build robust plans in environments with uncertainty even though the domain model is not complete.**

Probabilistic planners only obtain robust plans when they have an accurate and complete description of the environment. But recent planning applications like controlling underwater autonomous vehicles' (UAV's), planetary rovers or spacecrafts have to address non-deterministic planning problems in domains whose dynamics are not accessible. At the moment, the model learning in environments with uncertainty is very poorly-studied and there are just few works that only learn simple probabilistic planning actions model with no conditional effects [79, 104]. **The first objective is to propose machine learning mechanisms to capture the performance of actions in environments with uncertainty.** This ML mechanisms have to induce features to model the performance of the actions in the environment from observations of the actions executions.

Most of the existing learning-based planners are not competitive with the state-of-the-art planners across a wide range of different domains. **The second objective is to propose a new PUU paradigm based on the integration of planning and execution with learning mechanisms competitive with the state-of-the-art probabilistic planners.** The new PUU have to be based on efficient planning and learning techniques that guarantee good results in terms of quality and time in the standard test benches.

The existing test benches for PUU systems assume the worlds dynamics are static, i.e., the actions' performance does not change. This is a too restrictive assumption for many real problems where the world dynamics change over time. **The third objective is the definition of test benches to evaluate the perfor-**

mance of PUU systems in dynamic environments.

The planning techniques proposed to tackle realistic planning problems are frequently only used in simulators. **The fourth objective is the development of a software to address real planning task in environments with uncertainty.** Our aim is to have a software able to solve planning problems for at least these two real applications:

- Synthesizing tailor-made work plans for university students in order to ensure their incremental and consistent learning of a given subject. The Spanish MEC project ADAPTAPLAN ¹ is developing an intelligent architecture to assist teachers and student in the new tasks of the European Higher Education Space. The developed PUU software has to serve plans for this architecture considering the workload of the different tasks and the students profile. Besides, the PUU software have to monitor the execution of the plans and propose new ones in case the students are not making the predicted progresses or in case they are exceeding the expectations.
- Controlling the navigation of the mobile robot *Pioneer P3-DX* (Figure 3.1). Navigation is one of the fundamental tasks for a mobile robot. The majority of the existing robot navigation techniques already yield highly efficient solutions but as they do not manage relational knowledge they typically do not transfer to other similar tasks. A relational abstraction of the navigation plans has several interesting and important properties: It would allow the robot to generalize its navigation plans, and moreover, relational plans are understandable by humans that can easily validate them by hand. The developed PUU software have to propose relational navigation plans, and monitor its executions proposing new plans in case the robot reaches unexpected situations.



Figure 3.1: The Pioneer P3-DX mobile robot in a corridor of the University Carlos III de Madrid.

¹<http://adenu.ia.uned.es/adaptaplan/>

Chapter 4

Methodology

The International Planning Competition provides researchers with a framework to evaluate the new developed planning systems. This framework consist of a standard representation language (PDDL for deterministic planners and PPDDL for non-deterministic planners), test benches, problem generators and metrics to compare the planners' performance. In this chapter I propose an evaluation of the achievement of the this Thesis objectives based on the resources provided by this competition:

4.1 The International Planning Competition

The IPC is a biennial event, hosted at the International Conference on Automated Planning and Scheduling (ICAPS)¹ series. It was in 1998 when Drew McDermott leaded a committee to create a common specification language for automated planners called PDDL and a collection of benchmarks to evaluate the different existing planning paradigms. In this first competition, **IPC1**, participated deterministic planners from seven different universities. In 2000, Fahiem Bacchus continued the work; this time the **IPC2** attracted 15 different planners. In 2002, the **IPC3** competition was organised by Derek Long and Maria Fox and it focused on planning in temporal and metric domains posing a number of important new challenges for planning. In 2004 **IPC4** was organised by Stefan Edelkamp and Jörg Hoffmann, and the event was extended and revised in several respects. In particular, for the first time, the overall competition was split into a classical part –a continuation of the previous events– and a non-deterministic part. The non-deterministic part, co-organized by Michael Littman and Hakan Younes pretended to evaluate the performance of the existing probabilistic planning systems.

¹<http://icaps-conference.org/>

The main contribution of this event was the introduction of a common representation language for probabilistic planners (PPDDL), and the establishment of the first standard benchmarks. In 2006 the **IPC5** competition went on with this two tracks format, in this case the organizers of the non-deterministic track, Blai Bonet and Bob Givan, created a specific competition for conformant planners.

4.2 Evaluation

The achievement of the Thesis objectives will be tested in a four steps evaluation:

1. **Test the performance of the proposed ML mechanisms** (*Objective 1*). To evaluate the efficiency of the developed learning techniques I will compute the Mean Quadratic Error (MQE) between the domain model estimated through ML and the true domain model, the one of the simulator. Starting from a simple deterministic domain model, which is only consisting of STRIPS actions, a probabilistic action model is estimated using the observations of actions' executions after trying to solve 5 random probabilistic. The execution of actions is simulated with the software provided by the probabilistic track of the IPC. This simulator allows to emulate stochastic worlds where actions may have diverse outcomes. The current state of the simulator can be totally observed at any time, and it is updated only when a new action is executed. Figure 4.1 shows an overview of the architecture proposed to test the performance of the learning mechanisms.

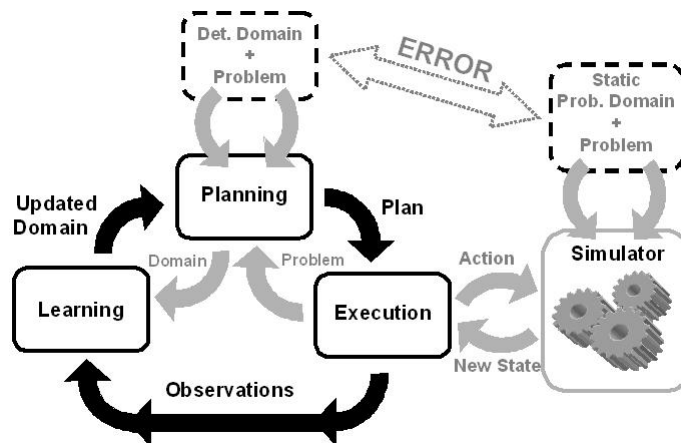


Figure 4.1: Architecture proposed to test a PUU planner in a static environment.

In order to compute the MQE of the estimate domain model I will maintain a set of pairs $(state, action)$ obtained after solving problems randomly-

generated with the IPC problem-generators. I will take each pair and compare the estimation of the probabilities of the actions' outcomes using the estimated domain model and the true domain model. I will do this for all the domains of the probabilistic track of IPC4 and IPC5.

2. **Comparison between the proposed PUU paradigm and the state-of-the-art PUU planners** (*Objective 2*). I will evaluate the performance the new PUU paradigm solving the IPC4 and IPC5 test benches under similar conditions. In the probabilistic tracks of the IPC, each planner attempts to solve 30 times every problem and its performance is measured according to the average values of three dimensions:

- The number of times the planner has solved the given problem.
- The time the planner has spent solving the given problem.
- The number of actions the planner has needed to solve the given problem.

Unlike the IPC participant probabilistic planners, our PUU paradigm will only have available a deterministic domain description consisting of STRIPS actions because it does not need to a priori have a probabilistic action model to solve stochastic planning problems.

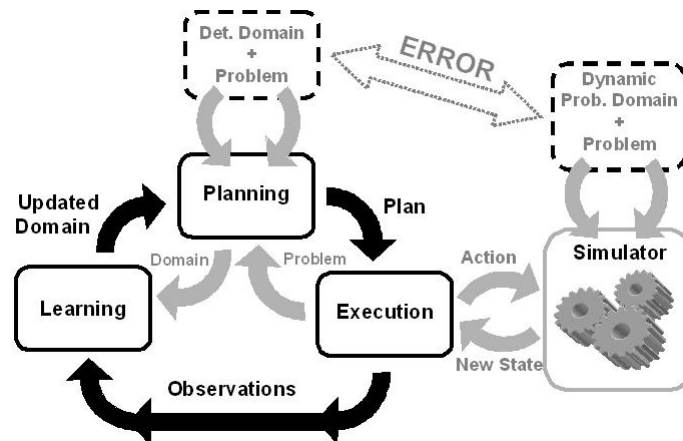


Figure 4.2: Architecture proposed to test the PUU planner in a dynamic environment.

3. **Test the performance of the proposed PUU paradigm in dynamic environments** (*Objective 3*). I will evaluate the performance of the new PUU paradigm in problems where the probabilities of the actions outcomes vary

over time. I will define dynamics environments by randomly changing the probabilities of the actions outcomes of the domain theory in the simulator after a fixed number of problem solving attempts. Figure 4.2 shows an overview of the architecture proposed to test a PUU planner in a dynamic environment.

4. **Test the performance of the proposed PUU paradigm when addressing real planning tasks (Objective 4).** Figure 4.3 shows an overview of the architecture proposed to test the PUU planner in real applications. I will evaluate this architecture when addressing the planning tasks of two real applications:

- An intelligent architecture to assist teachers and students in the new tasks posed in the European Higher Education Space.
- Controlling the navigation of the mobile robot *Pioneer P3-DX*.

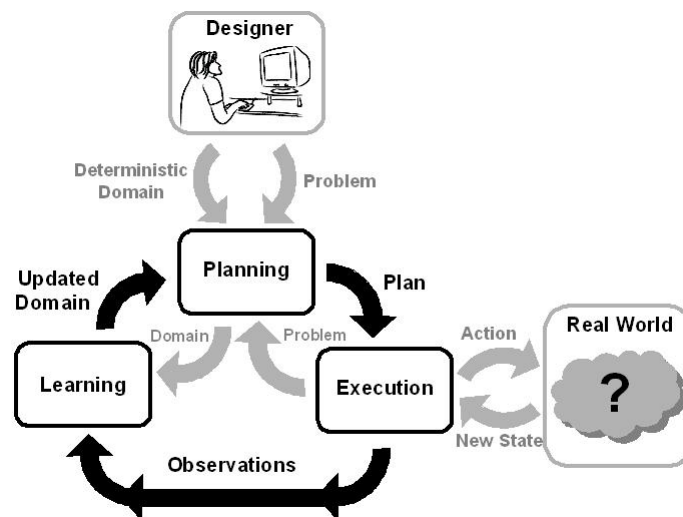


Figure 4.3: Architecture proposed to test the PUU planner in a real application.

4.3 Work Plan

The following programme is set up with the aim of guiding the work towards the achievement of the Thesis objectives. The steps 1-6 of the program are already finished. Figure 4.4 shows the schedule for the fulfilling of steps 7, 8 and 9.

	May	June	July	August	Sep	Oct	Nov	Dec
Task	7	7	8	-	8	-	-	defence

Figure 4.4: Schedule for fulfilling steps 7,8 and 9 of the Thesis program.

1. Exhaustive review of the existing literature in the PUU subfield. A deep study about the advantages and limitations of the existing paradigms and how they can be improved.
2. Exhaustive review of the current literature in ML assisted planning. An analysis of how these techniques are applied in deterministic planning and how they can be applied to the PUU paradigms.
3. Exhaustive review of the current literature about learning actions' model. This include learning action models in deterministic and stochastic domains with full and partial observability.
4. Development and evaluation of diverse ML techniques to learn knowledge about the performance of stochastic actions.
5. Implementation of an architecture that integrates planning, execution and the new developed learning mechanisms.
6. Comparison of the performance of the architecture with the state-of-the-art non-deterministic planners test benches from the IPC4 and IPC5.
7. Definition of a test bench that allows the evaluation of the performance of non-deterministic planners in dynamic and stochastic environments and test of the performance of the architecture in this new test bench for non-deterministic planners.
8. Test of the performance of the architecture in two real planning applications: in the frame of the ADAPTAPLAN project and in the frame of a robotic navigation controlling.

Bibliography

- [1] M. Ai-Chang, J. Bresina, L. Charest, J. Hsu, A. K. Jhonsson, B. Kanefsky, P. Maldague, P. Morris, K. Rajan, and J. Yglesias. Mapgen planner: Mixed-initiative activity planning for the mars exploration rover mission. In *Planning and Sequencing Information Systems Workshop*, 2002.
- [2] R. Aler, D. Borrajo, and P. Isasi. Using genetic programming to learn and improve control knowledge. *Artificial Intelligence*, 141(1-2):29–56, October 2002.
- [3] E. Amir. Learning partially observable action schemas. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI'06)*, 2006, 2006.
- [4] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ., 1957.
- [5] R. Bellman and R. Kalaba. *Dynamic Programming and Modern Control Theory*. Academic Press, 1965.
- [6] S. S. Benson. *Learning Action Models for Reactive Autonomous Agents*. PhD thesis, Stanford University, 1997.
- [7] R. Bergmann and W. Wilke. Paris: Flexible plan adaptation by abstraction and refinement. In A. Voss, editor, *ECAI (1996) Workshop on Adaptation in Case-Based Reasoning*. John Wiley & Sons, 1996.
- [8] D. Bernard, E. Gamble, N.Rouquette, B.Smith, Y.Tung, N.Muscettola, G. Dorias, B.Kanefsky, J. Kurien, W. Millar, P. Nayak, and K. Rajan. Remote agent experiment. ds1 technology validation report. Technical report, NASA Ames and JPL repot, 1998.
- [9] A. Blum and M. Furst. Fast planning through planning graph analysis. In C. S. Mellish, editor, *Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI-95*, volume 2, pages 1636–1642, Montreal, Canada, August 1995. Morgan Kaufmann.

- [10] J. Blythe. *Planning under Uncertainty in Dynamic Domains*. PhD thesis, School of Computer Science, Carnegie Mellon University, 1998.
- [11] B. Bonet and H. Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1–2):5–33, 2001.
- [12] B. Bonet and H. Geffner. mgpt: A probabilistic planner based on heuristic search. In *Proceedings of the fourth International Planning Competition. Fourteenth International Conference on Automated Planning and Scheduling. ICAPS-04*, 2004.
- [13] L. Booker, D. Goldberg, and J. Holland. Classifier systems and genetic algorithms. *Artificial Intelligence*, pages 235–282, 1989.
- [14] D. Borrajo and M. Veloso. Lazy incremental learning of control knowledge for efficiently obtaining quality plans. *AI Review Journal. Special Issue on Lazy Learning*, 11(1-5):371–405, February 1997.
- [15] A. Botea, M. Enzenberger, M. Müller, and J. Schaeffer. Macro-ff: Improving ai planning with automatically learned macro-operators. *Journal of Artificial Intelligence Research*, 24:581–621, 2005.
- [16] C. Boutilier, R. Dearden, and M. Goldszmidt. Exploiting structure in policy construction. In C. Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1104–1111, San Francisco, 1995. Morgan Kaufmann.
- [17] R. Brafman and J. Hoffmann. Conformant planning via heuristic forward search: A new approach. In S. Koenig, S. Zilberstein, and J. Koehler, editors, *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS-04)*, 2004.
- [18] L. Breslow and D. W. Aha. NaCoDAE: Navy conversational decision aids environment. Technical Report AIC-97-018, Washington, DC: Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence, 1997.
- [19] D. Bryce. The partially-observable and non-deterministic planner. In *Proceedings of the fifth International Planning Competition. Fourteenth International Conference on Automated Planning and Scheduling. ICAPS-06*, 2005.
- [20] O. Buffet and D. Aberdeen. The factored policy gradient planner. In *Proceedings of the Fifth International Planning Competition*, June 2006.

- [21] T. Bylander. Complexity results for planning. In R. Myopoulos, John; Reiter, editor, *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 274–279, Sydney, Australia, Aug. 1991. Morgan Kaufmann.
- [22] T. Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1-2):165–204, 1994.
- [23] C. Carrick, Q. Yang, I. Abi-Zeid, and L. Lamontagne. Activating CBR systems through autonomous information gathering. *Lecture Notes in Computer Science*, 1650, 1999.
- [24] L. Castillo, J. Fdez.-Olivares, O. García-Pérez, and F. Palao. Bringing users and planning technology together. experiences in SIADEX. In *16th International Conference on Automated Planning and Scheduling (ICAPS 2006)*, 2006.
- [25] A. Cimatti and M. Roveri. Conformant planning via symbolic model checking. In *JAIR*, 13, 2000.
- [26] A. Cocora, K. Kersting, C. Plagemann, W. Burgard, and L. D. Raedt. Learning relational navigation policies. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-06)*, Beijing, China, 2006.
- [27] W. W. Cohen. Learning approximate control rules of high utility. In *ML90*, pages 268–276, Austin, TX, 1990. Morgan Kaufmann.
- [28] A. Coles and A. Smith. Marvin: A heuristic search planner with online macro-action learning. *Journal of Artificial Intelligence Research*, 28:119–156, 2007.
- [29] C. Dawson and L. Silklosly. The role of preprocessing in problem solving system. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence, IJCAI-77*, pages 465–471, 1977.
- [30] T. DelaRosa, A. García-Olaya, and D. Borrajo. Using cases utility for heuristic planning improvement. In M. Richter and R. Weber, editors, *Proceedings of the 7th International Conference on Case-Based Reasoning*, Belfast, Northern Ireland, August 2007. Springer Verlag.
- [31] S. Dzeroski, L. D. Raedt, and H. Blockeel. Relational reinforcement learning. In *International Workshop on Inductive Logic Programming*, pages 11–22, 1998.

- [32] O. Etzioni. Acquiring search-control knowledge via static analysis. *Artificial Intelligence*, 62(2):255–301, 1993.
- [33] Z. Feng and E. A. Hansen. Symbolic heuristic search for probabilistic planning. In *Proceedings of the fourth International Planning Competition. Fourteenth International Conference on Automated Planning and Scheduling. ICAPS-04*, 2004.
- [34] R. Fikes, P. Hart, and N. J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3:251–288, 1972.
- [35] R. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.
- [36] M. Fox, A. Gerevini, D. Long, and I. Serina. Plan stability: Replanning versus plan repair. *Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS'06)*, pages 193–202, 2006.
- [37] M. Fox and D. Long. The automatic inference of state invariants in tim. *Journal of AI Research*, 9:367–421, 1998.
- [38] M. Fox and D. Long. Extending the exploitation of symmetries in planning. In *In Proc. of AIPS-02*, 2002.
- [39] M. Fox and D. Long. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, pages 61–124, 2003.
- [40] N. Friedman and M. Goldszmidt. Learning Bayesian networks with local structure. In *Learning and Inference in Graphical Models*, pages 252–262, 1996.
- [41] R. Garcia-Martinez and D. Borrajo. An integrated approach of learning, planning, and execution. *Journal of Intelligent and Robotics Systems*, 29:47–78, 2000.
- [42] H. Geffner. Functional strips: a more general language for planning and problem solving. In *Presented at the Logic-based AI Workshop*, 1999.
- [43] H. Geffner, A. Darwiche, and R. Dechter. Tutorial: Principles of ai problem solving, 2005.
- [44] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning Theory and Practice*. Morgan Kaufmann, 2004.

- [45] Y. Gil. *Acquiring Domain Knowledge for Planning by Experimentation*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh., 1992.
- [46] C. Gretton, D. Price, and S. Thiébaux. Nmrhpp: Decision-theoretic planning with control knowledge. In *Proceedings of the fourth International Planning Competition. Fourteenth International Conference on Automated Planning and Scheduling. ICAPS-04*, 2004.
- [47] K. J. Hammond. Explaining and repairing plans that fail. *Artificial Intelligence*, 45:173–228, 1990.
- [48] J. Hoffmann and R. Brafman. Contingent planning via heuristic forward search with implicit belief states. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS-05)*, pages 71–80, 2005.
- [49] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.
- [50] J. Huang. Complanner: A conformant probabilistic planner. In *Proceedings of the fifth International Planning Competition. Fourteenth International Conference on Automated Planning and Scheduling. ICAPS-06*, 2005.
- [51] S. Huffman, D. Pearson, and J. Laird. Correcting imperfect domain theories: A knowledge level analysis. In S. Chipman and A. Meyrowitz, editors, Kluwer Academic Press, 1992.
- [52] O. Ilghami, D. Nau, H. Muñoz-Avila, and D. W. Aha. Camel: Learning method preconditions for HTN planning. In M. Ghallab, J. Hertzberg, and P. Traverso, editors, *Proceedings of the Sixth International Conference on Artificial Intelligence Planning Systems (AIPS-02)*, pages 131–141, Toulouse (France), 23-17 April 2002. AAAI Press.
- [53] E. Karabaev and O. Skvortsova. Fcplanner: A planning strategy for first-order mdps. In *Proceedings of the fourth International Planning Competition. Fourteenth International Conference on Automated Planning and Scheduling. ICAPS-04*, 2004.
- [54] H. A. Kautz and B. Selman. Planning as satisfiability. In *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI'92)*, pages 359–363, 1992.

- [55] R. Keller. *The Role of Explicit Contextual Knowledge in Learning Concepts to Improve Performance*. PhD thesis, Rutgers University, 1987.
- [56] K. Kersting, M. V. Otterlo, and L. D. Raedt. Bellman goes relational. In *Proceedings of the Twenty-First International Conference on Machine Learning (ICML-04)*, 2004.
- [57] R. Khardon. Learning action strategies for planning domains. *Artificial Intelligence*, 113:125–148, 1999.
- [58] C. Knoblock. Learning abstraction hierarchies for problem solving. In *Knoblock, C.A. Learning Abstraction Hierarchies for Problem Solving, in Seventh International Workshop on Machine Learning, pp. 923-928 (1990).*, 1990.
- [59] R. E. Korf. Macro-operators: A weak method for learning. *Artificial Intelligence*, 1985, 26:35–77, 1985.
- [60] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, 1992.
- [61] P. R. Kumar. *Stochastic Systems: Estimation, Identification, and Adaptive Control*. Prentice Hall, 1986.
- [62] N. Kushmerick, S. Hanks, and D. S. Weld. An algorithm for probabilistic planning. *Artificial Intelligence*, 76(1-2):239–286, 1995.
- [63] N. Kushmerick, S. Hanks, and D. S. Weld. An algorithm for probabilistic planning. *Artificial Intelligence*, 76(1-2):239–286, 1995.
- [64] C. Leckie and I. Zukerman. Learning search control rules for planning: An inductive approach. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 422–426, Evanston, IL, 1991. Morgan Kaufmann.
- [65] I. Little and S. Thiébaux. Concurrent probabilistic planner in the Graph-Plan framework. In *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling (ICAPS '06), The English Lake District, Cumbria, UK*, 2006.
- [66] M. L. Littman. Probabilistic propositional planning: Representations and complexity. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97)*, pages 748–761, Providence, Rhode Island, 1997. AAAI Press / MIT Press.

- [67] M. Martin and H. Geffner. Learning generalized policies in planning using concept languages. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems (AIPS00)*, June 2000.
- [68] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. L. Webber and N. J. Nilsson, editors, *Readings in Artificial Intelligence*, pages 431–450. Kaufmann, Los Altos, CA, 1981.
- [69] T. Mitchell, T. Utgoff, and R. Banerji. *Machine Learning: An Artificial Intelligence Approach*, chapter Learning problem solving heuristics by experimentation. Morgan Kaufmann, 1982.
- [70] T. M. Mitchell. *Machine Learning*. mcgraw-hill, 1997.
- [71] Muñoz-Avila, H. Aha, D. Breslow, and L. Nau. Hicap: An interactive case-based planning architecture and its application to noncombatant evacuation operations, 1999.
- [72] D. Nau, T. Au, O. Ilghami, U. Kuter, W. Murdock, D. Wu, and F. Yaman. Shop: An htn planning system. In *JAIR*, 20, 2003.
- [73] B. Nebel, Y. Dimopoulos, and J. Koehler. Ignoring irrelevant facts and operators in plan generation. In *ECP-97*, pages 338–350, 1997.
- [74] M. Newton, J. Levine, and M. Fox. Genetically evolved macro-actions in a.i. planning problems. In *Proceedings of the 24th UK Planning and Scheduling SIG*, pages 163–172, 2005.
- [75] N. J. Nilsson. Shakey the robot. Technical Report 323, AI Center, SRI International, Menlo Park, CA, 1984.
- [76] N. Onder and M. E. Pollack. Conditional, probabilistic planning: A unifying algorithm and effective search control mechanisms. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI'99)*, 1999.
- [77] N. Onder, G. C. Whelan, and L. Li. Probapop: Probabilistic partial-order planning. In *Proceedings of the fourth International Planning Competition. Fourteenth International Conference on Automated Planning and Scheduling. ICAPS-04*, 2004.
- [78] H. Palacios and H. Geffner. Compiling uncertainty away: Solving conformant planning problems using a classical planner (sometimes). In *Proc. 21st Nat. Conf. on Artificial Intelligence (AAAI-06)*, 2006.

- [79] H. Pasula, L. Zettlemoyer, and L. Kaelbling. Learning probabilistic relational planning rules. *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling*, 2004.
- [80] E. P. D. Pednault. ADL and the state-transition model of action. *Journal of Logic and Computation*, 4(5):467–512, Oct. 1994.
- [81] J. Penberthy and D. Weld. UCPOP: A sound, complete, partial order planner for ADL. In *Proceedings of KR-92*, pages 103–114, 1992.
- [82] M. Peot and D. Smith. Conditional nonlinear planning. In J. Hendler, editor, *Proceedings of the First International Conference on AI Planning Systems*, pages 189–197, College Park, Maryland, June 15–17 1992. Morgan Kaufmann.
- [83] P. Poupart and C. Boutilier. Value-directed belief state approximation for POMDPs. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 497–506, 2000.
- [84] L. Pryor and G. Collins. Planning for contingencies: A decision-based approach. *Journal of Artificial Intelligence Research*, 4:287–339, 1996.
- [85] J. Quinlan and R. Cameron-Jones. Introduction of logic programs: FOIL and related systems. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4):287–312, 1995.
- [86] J. Rintanen. Expressive equivalence of formalisms for planning with sensing, 2003.
- [87] R. L. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, Nov. 1987.
- [88] E. D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artif. Intell.*, 5(2):115–135, 1974.
- [89] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):211–229, 1959.
- [90] S. Sanner and C. Boutilier. Probabilistic planning via linear value-approximation of first-order mdps. In *Proceedings of the fifth International Planning Competition. Fourteenth International Conference on Automated Planning and Scheduling. ICAPS-06*, 2005.

- [91] M. Schmill, T. Oates, and P. Cohen. Learning planning operators in real-world, partially observable environments. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems (AIPS00)*, June 2000.
- [92] W. Shen and Simon. Rule creation and rule learning through environmental exploration. In *Proceedings IJCAI-89*, pages 675–680, 1989.
- [93] H. A. Simon. *Why should machine learn?. Machine Learning: an Artificial Intelligence Approach*. Morgan Kaufmann, Palo Alto, CA, 1983.
- [94] D. E. Smith and D. S. Weld. Conformant graphplan. In *AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pages 889–896, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence.
- [95] L. Spector. Genetic programming and AI planning systems. In *Proceedings of Twelfth National Conference on Artificial Intelligence*, Seattle, Washington, USA, 1994. AAAI Press/MIT Press.
- [96] A. F. SungWook Yoon and R. Givan. Learning reactive policies for probabilistic planning domains. In *Proceedings of the fourth International Planning Competition. Fourteenth International Conference on Automated Planning and Scheduling. ICAPS-04*, 2004.
- [97] M. M. Veloso and J. G. Carbonell. Derivational analogy in prodigy: Automating case acquisition, storage, and utilization. *Machine Learning*, 10:249–278, 1993.
- [98] X. Wang. Learning planning operators by observation and practice. In *Proceedings of the Second International Conference on AI Planning Systems, AIPS-94*, pages 335–340, Chicago, IL, June 1994. AAAI Press, CA.
- [99] D. S. Weld, C. R. Anderson, and D. E. Smith. Extending graphplan to handle uncertainty and sensing actions. In *AAAI/IAAI*, pages 897–904, 1998.
- [100] Q. Yang, K. Wu, and Y. Jiang. Learning action models from plan examples with incomplete knowledge. In *In Proceedings of the 2005 International Conference on Automated Planning and Scheduling, (ICAPS 2005) Monterey, CA USA*, pages 241–250, 2005.
- [101] S. Yoon, A. Fern, and R. Givan. Learning heuristic functions from relaxed plans. In *International Conference on Automated Planning and Scheduling (ICAPS-2006)*, 2006.

- [102] H. Younes, M. L. Littman, D. Weissman, and J. Asmuth. The first probabilistic track of the international planning competition. *Journal of Artificial Intelligence Research*, 24:851–887, 2005.
- [103] J. Zelle and R. Mooney. Combining FOIL and EBG to speed-up logic programs. In *IJCAI93*, pages 1106–1113, Chambéry, France, 1993. Morgan Kaufmann.
- [104] L. Zettlemoyer, H. Pasula, and L. Kaelbling. Learning planning rules in noisy stochastic worlds. *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, 2005.
- [105] R. Zhou and E. A. Hansen. Breadth-first heuristic search. In J. K. Shlomo Zilberstein and S. Koenig, editors, *Fourteenth International Conference on Automated Planning and Scheduling (ICAPS-04)*, pages 92–100, Whistler, British Columbia (Canada), June 2004.
- [106] T. Zimmerman and S. Kambhampati. Learning-assisted automated planning: looking back, taking stock, going forward. *AI Magazine*, 24:73 – 96, 2003.