

A Concise Introduction to Models and Methods for Automated Planning

Synthesis Lectures on Artificial Intelligence and Machine Learning

Editor

Ronald J. Brachman, *Yahoo! Labs*

William W. Cohen, *Carnegie Mellon University*

Peter Stone, *University of Texas at Austin*

[A Concise Introduction to Models and Methods for Automated Planning](#)

Hector Geffner and Blai Bonet

2013

[Essential Principles for Autonomous Robotics](#)

Henry Hexmoor

2013

[Case-Based Reasoning: A Concise Introduction](#)

Beatriz López

2013

[Answer Set Solving in Practice](#)

Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub

2012

[Planning with Markov Decision Processes: An AI Perspective](#)

Mausam and Andrey Kolobov

2012

[Active Learning](#)

Burr Settles

2012

[Computational Aspects of Cooperative Game Theory](#)

Georgios Chalkiadakis, Edith Elkind, and Michael Wooldridge

2011

Representations and Techniques for 3D Object Recognition and Scene Interpretation

Derek Hoiem and Silvio Savarese

2011

A Short Introduction to Preferences: Between Artificial Intelligence and Social Choice

Francesca Rossi, Kristen Brent Venable, and Toby Walsh

2011

Human Computation

Edith Law and Luis von Ahn

2011

Trading Agents

Michael P. Wellman

2011

Visual Object Recognition

Kristen Grauman and Bastian Leibe

2011

Learning with Support Vector Machines

Colin Campbell and Yiming Ying

2011

Algorithms for Reinforcement Learning

Csaba Szepesvári

2010

Data Integration: The Relational Logic Approach

Michael Genesereth

2010

Markov Logic: An Interface Layer for Artificial Intelligence

Pedro Domingos and Daniel Lowd

2009

Introduction to Semi-Supervised Learning

Xiaojin Zhu and Andrew B. Goldberg

2009

Action Programming Languages

Michael Thielscher

2008

Representation Discovery using Harmonic Analysis

Sridhar Mahadevan

2008

Essentials of Game Theory: A Concise Multidisciplinary Introduction

Kevin Leyton-Brown and Yoav Shoham

2008

A Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence

Nikos Vlassis

2007

Intelligent Autonomous Robotics: A Robot Soccer Case Study

Peter Stone

2007

Copyright © 2013 by Morgan & Claypool

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means—electronic, mechanical, photocopy, recording, or any other except for brief quotations in printed reviews, without the prior permission of the publisher.

A Concise Introduction to Models and Methods for Automated Planning

Hector Geffner and Blai Bonet

www.morganclaypool.com

ISBN: 9781608459698 paperback

ISBN: 9781608459704 ebook

DOI 10.2200/S00513ED1V01Y201306AIM022

A Publication in the Morgan & Claypool Publishers series

SYNTHESIS LECTURES ON ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Lecture #22

Series Editors: Ronald J. Brachman, *Yahoo! Labs*

William W. Cohen, *Carnegie Mellon University*

Peter Stone, *University of Texas at Austin*

Series ISSN

Synthesis Lectures on Artificial Intelligence and Machine Learning

Print 1939-4608 Electronic 1939-4616

A Concise Introduction to Models and Methods for Automated Planning

Hector Geffner

ICREA and Universitat Pompeu Fabra, Barcelona, Spain

Blai Bonet

Universidad Simón Bolívar, Caracas, Venezuela

*SYNTHESIS LECTURES ON ARTIFICIAL INTELLIGENCE AND MACHINE
LEARNING #22*



MORGAN & CLAYPOOL PUBLISHERS

ABSTRACT

Planning is the model-based approach to autonomous behavior where the agent behavior is derived automatically from a model of the actions, sensors, and goals. The main challenges in planning are computational as all models, whether featuring uncertainty and feedback or not, are intractable in the worst case when represented in compact form. In this book, we look at a variety of models used in AI planning, and at the methods that have been developed for solving them. The goal is to provide a modern and coherent view of planning that is precise, concise, and mostly self-contained, without being shallow. For this, we make no attempt at covering the whole variety of planning approaches, ideas, and applications, and focus on the essentials. The target audience of the book are students and researchers interested in autonomous behavior and planning from an AI, engineering, or cognitive science perspective.

KEYWORDS

planning, autonomous behavior, model-based control, plan generation and recognition, MDP and POMDP planning, planning with incomplete information and sensing, action selection, belief tracking, domain-independent problem solving

Contents

	Preface	xi
1	Planning and Autonomous Behavior	1
	1.1 Autonomous Behavior: Hardwired, Learned, and Model-based	1
	1.2 Planning Models and Languages	3
	1.3 Generality, Complexity, and Scalability	6
	1.4 Examples	8
	1.5 Generalized Planning: Plans vs. General Strategies	11
	1.6 History	12
2	Classical Planning: Full Information and Deterministic Actions	15
	2.1 Classical Planning Model	15
	2.2 Classical Planning as Path Finding	16
	2.3 Search Algorithms: Blind and Heuristic	16
	2.4 Online Search: Thinking and Acting Interleaved	20
	2.5 Where do Heuristics come from?	23
	2.6 Languages for Classical Planning	24
	2.7 Domain-Independent Heuristics and Relaxations	27
	2.8 Heuristic Search Planning	33
	2.9 Decomposition and Goal Serialization	34
	2.10 Structure, Width, and Complexity	34
3	Classical Planning: Variations and Extensions	37
	3.1 Relaxed Plans and Helpful Actions	37
	3.2 Multi-Queue Best-First Search	38
	3.3 Implicit Subgoals: Landmarks	38
	3.4 State-of-the-Art Classical Planners	39
	3.5 Optimal Planning and Admissible Heuristics	41
	3.6 Branching Schemes and Problem Spaces	42
	3.7 Regression Planning	43
	3.8 Planning as SAT and Constraint Satisfaction	45
	3.9 Partial-Order Causal Link Planning	46
	3.10 Cost, Metric, and Temporal Planning	47
	3.11 Hierarchical Task Networks	49

4	Beyond Classical Planning: Transformations	51
4.1	Soft Goals and Rewards	51
4.2	Incomplete Information	53
4.3	Plan and Goal Recognition	57
4.4	Finite-State Controllers	60
4.5	Temporally Extended Goals	62
5	Planning with Sensing: Logical Models	65
5.1	Model and Language	65
5.2	Solutions and Solution Forms	67
5.3	Offline Solution Methods	69
5.4	Online Solution Methods	72
5.5	Belief Tracking: Width and Complexity	73
5.6	Strong vs. Strong Cyclic Solutions	76
6	MDP Planning: Stochastic Actions and Full Feedback	79
6.1	Goal, Shortest-Path, and Discounted Models	79
6.2	Dynamic Programming Algorithms	84
6.3	Heuristic Search Algorithms	86
6.4	Online MDP Planning	92
6.5	Reinforcement Learning, Model-based RL, and Planning	95
7	POMDP Planning: Stochastic Actions and Partial Feedback	97
7.1	Goal, Shortest-Path, and Discounted POMDPs	97
7.2	Exact Offline Algorithms	99
7.3	Approximate and Online Algorithms	102
7.4	Belief Tracking in POMDPs	105
7.5	Other MDP and POMDP Solution Methods	107
8	Discussion	109
8.1	Challenges and Open Problems	109
8.2	Planning, Scalability, and Cognition	111
	Bibliography	113
	Author's Biography	129

Preface

Planning is a central area in Artificial Intelligence concerned with the automated generation of behavior for achieving goals. Planning is also one of the oldest areas in AI with the General Problem Solver being the first automated planner and one of the first AI programs [Newell et al., 1959]. As other areas in AI, planning has changed a great deal in recent years, becoming more rigorous, more empirical, and more diverse. Planners are currently seen as automated solvers for precise classes of mathematical models represented in compact form, that range from those where the state of the environment is fully known and actions have deterministic effects, to those where the state of the environment is partially observable and actions have stochastic effects. In all cases, the derivation of the agent behavior from the model is computational intractable, and hence a central challenge in planning is scalability. Planning methods must exploit the structure of the given problems, and their performance is assessed empirically, often in the context of planning competitions that in recent years have played an important role in the area.

In this book, we look at a variety of models used in AI planning and at the methods that have been developed for solving them. The goal is to provide a modern and coherent view of planning that is precise, concise, and mostly self-contained, without being shallow. For this, we focus on the essentials and make no attempt at covering the whole variety of planning approaches, ideas, and applications. Moreover, our view of the essentials is not neutral, having chosen to emphasize the ideas that we find most basic in a model-based setting. A more comprehensive treatment of planning, circa 2004, can be found in the planning textbook by Ghallab et al. [2004]. Planning is also covered at length in the AI textbook by Russell and Norvig [2009].

The book is organized into eight chapters. Chapter 1 is about planning as the model-based approach to autonomous behavior in contrast to approaches where behaviors are learned, evolved, or specified by hand. Chapters 2 and 3 are about the most basic model in planning, classical planning, where a goal must be reached from a fully known initial state by applying actions with deterministic effects. Classical planners can currently find solutions to problems over huge state spaces, yet many problems do not comply with these restrictions. The rest of the book addresses such problems in two ways: one is by automatically translating non-classical problems into classical ones; the other is by defining native planners for richer models. Chapter 4 focuses thus on reductions for dealing with soft goals, temporally extended goals, incomplete information, and a slightly different task: goal recognition. Chapter 5 is about planning with incomplete information and partial observability in a logical setting where uncertainty is represented by sets of states. Chapters 6 and 7 cover probabilistic planning where actions have stochastic effects, and the state is either fully or partially observable. In all cases, we distinguish between offline solution methods that derive the complete control offline, and online solution methods that derive the control as needed, by interleaving planning and execution, thinking and doing. Chapter 8 is about open problems.

We are grateful to many colleagues, co-authors, teachers, and students. Among our teachers, we would like to mention Judea Pearl, who was the Ph.D. advisor of both of us at different times, and always a role model as a person and as a scientist. Among our students, we thank in particular Hector

xii PREFACE

Palacios, Emil Keyder, Alex Albore, Miquel Ramírez, and Nir Lipovetzky, on whose work we have drawn for this book. The book is based on tutorials and courses on planning that one of us (Hector) has been giving over the last few years, more recently at the ICAPS Summer School (Thessaloniki, 2009; São Paulo, 2012; Perugia, 2013), the International Joint Conference on AI (IJCAI, Barcelona, 2011), La Sapienza, Università di Roma (2010), and the Universitat Pompeu Fabra (2012). We thank the students for the feedback and our colleagues for the invitations and their hospitality. Thanks also to Alan Fern who provided useful and encouraging feedback on a first draft of the book.

A book, even if it is a short one, is always a good excuse for remembering the loved ones.

A los chicos, caminante no hay camino, a Lito, la llama eterna, a Marina, mucho más que dos, a la familia toda; a la memoria del viejo, la vieja, la bobo, y los compañeros tan queridos – Hector

A Iker y Natalia, por toda su ayuda y amor, a la familia toda, por su apoyo. A la memoria de Josefina Gorgal Caamaño y la iaia Francisca Prat – Blai

Hector Geffner, Barcelona
Blai Bonet, Caracas
June 2013

CHAPTER 1

Planning and Autonomous Behavior

Planning is the model-based approach to autonomous behavior where the agent selects the action to do next using a model of how actions and sensors work, what is the current situation, and what is the goal to be achieved. In this chapter, we contrast programming, learning, and model-based approaches to autonomous behavior, and present some of the models in planning that will be considered in more detail in the following chapters. These models are all general in the sense that they are not bound to specific problems or domains. This generality is intimately tied to the notion of intelligence which requires the ability to deal with new problems. The price for generality is computational: planning over these models when represented in compact form is intractable in the worst case. A main challenge in planning is thus the automated exploitation of problem structure for scaling up to large and meaningful instances that cannot be handled by brute force methods.

1.1 AUTONOMOUS BEHAVIOR: HARDWIRED, LEARNED, AND MODEL-BASED

At the center of the problem of intelligent behavior is the *problem of selecting the action to do next*. In Artificial Intelligence (AI), three different approaches have been used to address this problem. In the *programming-based approach*, the controller that prescribes the action to do next is given by the programmer, usually in a suitable high-level language. In this approach, the problem is solved by the programmer in his head, and the solution is expressed as a program or as a collection of rules or behaviors. In the *learning-based approach*, the controller is not given by a programmer but is induced from experience as in reinforcement learning. Finally, in the *model-based approach*, the controller is not learned from experience but is derived automatically from a model of the actions, sensors, and goals. In all these approaches, the controller is the solution to the model.

The three approaches to the action selection problem are not orthogonal, and exhibit different virtues and limitations. Programming agents by hand, puts all the burden on the programmer that cannot anticipate all possible contingencies, and often results in systems that are brittle. Learning methods have the greatest promise and potential, but their flexibility is often the result of learning a model. Last, model-based methods require a model of the actions, sensors, and goals, and face the computational problem of solving the model—a problem that is computationally intractable even for the simplest models where information is complete and actions are deterministic.

The Wumpus game, shown in Figure 1.1 from the standard AI textbook [Russell and Norvig, 2009], is an example of a simple scenario where an agent must process information arriving from the sensors to decide what to do at each step. The agent, initially at the lower left corner, must obtain the gold while avoiding deadly pits and a killer wumpus. The locations of the gold, pits, and wumpus are

2 1. PLANNING AND AUTONOMOUS BEHAVIOR







Stench		Breeze	
	Breeze Stench 		Breeze
Stench		Breeze	
	Breeze		Breeze

Figure 1.1: Autonomous Behavior in the Wumpus World: What to do next?

not known to the agent, but each emits a signal that can be perceived by the agent when in the same cell (gold) or in a contiguous cell (pits and wumpus). The agent control must specify the action to be done by the agent as a function of the observations gathered. The three basic approaches for obtaining such a controller are to write it by hand, to learn it from interactions with a Wumpus simulator, or to derive it from a model representing the initial situation, the actions, the sensors, and the goals.

While planning is often defined as the branch of AI concerned with the “synthesis of plans of action to achieve goals,” planning is best conceived as *the model-based approach to action selection*—a view that defines more clearly the role of planning in intelligent autonomous systems. The distinction that the philosopher Daniel Dennett makes between “Darwinian,” “Skinnerian,” and “Popperian” creatures [Dennett, 1996], mirrors quite closely the distinction between hardwired (programmed) agents, agents that learn, and agents that use models respectively. The contrast between the first and the latter corresponds also to the distinction made in AI between reactive and deliberative systems, as long as deliberation is not reduced to logical reasoning. Indeed, as we will see, the inferences captured by model-based methods that scale up are not logical but heuristic, and follow from relaxations and approximations of the problem being solved.

PLANNING IS MODEL-BASED AUTONOMOUS BEHAVIOR

Model-based approaches to the action selection problem are made up of three parts: the *models* that express the dynamics, feedback, and goals of the agent; the *languages* that express these models in compact form; and the *algorithms* that use the representation of the models for generating the behavior.

A representation of the model for the Wumpus problem, for example, will feature *variables* for the locations of the agent, the gold, the wumpus, the pits, and a boolean variable for whether the agent is alive. The location variables can take 16 different values, corresponding with the cells in the 4×4 grid, except for the gold that can also be held by the agent and hence has 17 possible values.¹ A *state* for the problem is a valuation over these seven variables. The number of possible states is thus $16^5 \times 17 \times 2$, which is slightly more than 35 million. Initially, the agent is alive and knows its location

¹If the number of pits and wumpus is not known a priori, an alternative representation would be needed where each cell in the grid would contain a wumpus, a pit, or neither.

but not the value of the pit and wumpus variables. The state of the system is thus *not fully observable*. The agent gets partial knowledge about the hidden variables through each of its three *sensors* that relate the true but hidden state of the world with observable tokens. The agent receives the observation token “stench” in the states where the wumpus is in one of the (at most) four cells adjacent to the agent, the token “breeze” in the states where a pit is adjacent to the agent, and the token “bright” when the gold and the agent are in the same cell. The actions available to the agent are to move to an adjacent cell, and to pick up the gold if known to be in the same cell. The actions change the state of the system in the expected way, affecting the location of the agent or the location of the gold. Yet the agent dies if it enters a cell with a wumpus or a pit, and a dead agent cannot execute any of the actions, and hence cannot achieve the goal of getting the gold.

In this problem, an intelligent agent should notice first that there is no wumpus or pit in cells (1, 2) or (2, 1) as there is no stench or breeze at the initial agent location (1, 1). It is then safe to move either up or right. If it moves up, it’ll sense a stench at (1, 2) and conclude that the wumpus is at either (1, 3) or (2, 2). Likewise, since it senses no breeze, it can conclude that neither of these cells contains a pit. The only safe move is then to get back to (1, 1) where it can move safely to (2, 1). From the sensed breeze, it can conclude that there is a pit at (3, 1) or (2, 2), or one pit at each, and from sensing no stench, that there is no wumpus at either (3, 1) or (2, 2). At this point, it should conclude that cell (2, 2) is safe as it cannot contain either a wumpus or a pit. It should then move up to (2, 2), from which the process of visiting new cells that are safe is repeated until the gold is found.

Writing a program for solving any instance of the Wumpus domain, for any (solvable) initial situation and grid size, is interesting enough. Yet, the task in planning is quite different. We want a program that can take a representation of *any problem* exhibiting a certain mathematical structure, not limited to the Wumpus domain, and find a solution to it. A number of planning models will make these mathematical structures explicit. Other problems that have a number of features in common with the Wumpus domain include the familiar Battleship game or the popular PC game Minesweeper. These are all problems where a goal is to be achieved by acting and sensing in a world where the state of the system, that may change or not, is partially observable.

While a program that has been designed to play the Wumpus can be deemed as intelligent, a program that can play the Wumpus without having been designed specifically for it will be intelligent in a much broader sense. The first contains the recipes for playing the Wumpus; the latter contains “recipes” for playing an infinite collection of domains, known or unknown to the programmer, that share a general mathematical structure. The formulation of these mathematical structures and the general “recipes” for solving them is what planning is about.

1.2 PLANNING MODELS AND LANGUAGES

A wide range of models used in planning can be understood as variations of a *basic state model* featuring:

- a finite and discrete state space S ,
- a *known initial state* $s_0 \in S$,
- a non-empty set $S_G \subseteq S$ of goal states,
- actions $A(s) \subseteq A$ applicable in each state $s \in S$,

4 1. PLANNING AND AUTONOMOUS BEHAVIOR

- a *deterministic* state transition function $f(a, s)$ such that $s' = f(a, s)$ stands for the state resulting of applying action a in s , $a \in A(s)$, and
- *positive action costs* $c(a, s)$.

This is the model underlying *classical planning* where it is normally assumed that action costs $c(a, s)$ do not depend on the state, and hence $c(a, s) = c(a)$. A solution or *plan* in this model is a sequence of applicable actions that map the initial state into a goal state. More precisely, a plan $\pi = a_0, \dots, a_{n-1}$ must generate a state sequence s_0, \dots, s_n such that $a_i \in A(s_i)$, $s_{i+1} = f(a_i, s_i)$, and $s_n \in S_G$, for $i = 0, \dots, n - 1$. The cost of the plan is the sum of the action costs $c(a_i, s_i)$, and a plan is optimal if it has minimum cost over all plans.

Classical planners accept a compact description of models of this form in *languages* featuring *variables*, where the states are the possible valuations of the variables. A classical plan $\pi = a_0, \dots, a_n$ represents an *open-loop controller* where the action to be done at time step i depends just on the step index i . The solution of models that accommodate uncertainty and feedback, produce *closed-loop controllers* where the action to be done at step i depends on the actions and observations collected up to that point. These models can be obtained by relaxing the assumptions in the model above displayed in italics.

The model for *partially observable planning*, also called *planning with sensing* or *contingent planning*, is a variation of the classical model that features both *uncertainty* and *feedback*—namely, uncertainty about the initial and next possible state, and partial information about the current state of the system. Mathematically such a model can be expressed in terms of the following ingredients:

- a finite and discrete state space S ,
- a non-empty set S_0 of *possible* initial states, $S_0 \subseteq S$,
- a non-empty set $S_G \subseteq S$ of goal states,
- a set of actions $A(s) \subseteq A$ applicable in each state $s \in S$,
- a *non-deterministic* state transition function $F(a, s)$ for $s \in S$ and $a \in A(s)$, where $F(a, s)$ is non-empty and $s'' \in F(a, s)$ stands for the possible successor states of state s after action a is done, $a \in A(s)$,
- a set of observation tokens O ,
- a sensor model $O(s, a) \subseteq O$, where $o \in O(s, a)$ means that token o may be observed in the (possibly hidden) state s if a was the last action done, and
- positive *action costs* $c(a, s)$.

In the model for the Wumpus problem, the state space S is given by the set of possible valuations over the problem variables, S_0 is the set of states where the agent is initially alive and at location (1, 1), S_G is the set of states where the agent is holding the gold, and A stands for the actions of moving and picking up the gold, provided that the agent can't leave the grid and can't pick the gold if not in the same cell. Likewise, the state transitions $F(a, s)$ associated with these actions is deterministic, meaning that $F(a, s)$ contains a single state s' so that $|F(a, s)| = 1$. The same is true for the sensor

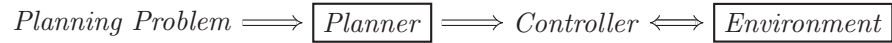


Figure 1.2: A planner takes a compact representation of a planning problem over a certain class of models (classical, conformant, contingent, MDP, POMDP) and automatically produces a controller. For fully and partially observable models, the controller is closed-loop, meaning that the action selected depends on the observations gathered. For non-observable models like classical and conformant planning, the controller is open-loop, meaning that it is a fixed action sequence.

model $O(s, a)$, which does not depend on a but just on the hidden state s . Namely, O contains nine observation tokens o , corresponding to the possible combinations of the three booleans stench, breeze, and bright, so that if s is a state where the agent is next to a pit and a wumpus but not in the same cell as the gold, then $o \in O(s, a)$ iff o represents the combination where stench and breeze are true, and bright is false. The action costs for the problem, $c(a, s)$, can be all assumed to be 1, and in addition, no action can be done by the agent when he is not alive.

A partially observable planner is a program that accepts compact descriptions of instances of the model above, like the one for the Wumpus, and automatically outputs the control (Figure 1.2). As we will see, planners come in two forms: *offline* and *online*. In the first case, the behavior specifies the agent response to each possible situation that may result; in the second case, the behavior just specifies the action to be done in the current situation. These types of control, unlike the control that results in classical planning, are closed-loop: the actions selected usually depend on the observation tokens received.

Offline solutions of partially observable problems are not fixed action sequences as in classical planning, as observations need to be taken into account for selecting actions. Mathematically, thus, these solutions are functions mapping the stream of past actions and observations into actions, or more conveniently, functions mapping *belief states* into actions. The belief state that results after a given stream of actions and observations represents the set of states that are deemed possible at that point, and due to the Markovian state-transition dynamics, it summarizes all the information about the past that is relevant for selecting the action to do next. Moreover, since the initial belief state b_0 is given, corresponding to the set of possible initial states S_0 , a solution function π , called usually the *control policy*, does not need to be defined over all possible beliefs, but just over the beliefs that can be produced from the actions determined by the policy π from the initial belief state b_0 and the observations that may result. Such *partial* policies π can be represented by a directed graph rooted at b_0 , where nodes stand for belief states, edges stand for actions a_i or observations o_i , and the branches in the graph from b_0 , stand for the stream of actions and observations $a_0, o_0, a_1, o_1, \dots$, called *executions*, that are possible. The policy solves the problem when all these possible executions end up in belief states where the goal is true.²

The models above are said to be *logical* as they only encode and keep track of what is possible or not. In *probabilistic* models, on the other hand, each possibility is weighted by a probability measure. A probabilistic version of the partially observable model above can be obtained by replacing the set of possible initial states S_0 , the set of possible successor states $F(a, s)$, and the set of possible observation tokens $O(s, a)$, by probability distributions: a prior $P(s)$ on the states $s \in S_0$ that are initially possible,

²We will make this all formal and precise in Chapter 5.

6 1. PLANNING AND AUTONOMOUS BEHAVIOR

transition probabilities $P_a(s'|s)$ for encoding the likelihood that s' is the state that follows s after a , and observation probabilities $P_a(o|s)$ for encoding the likelihood that o is the token that results in the state s when a is the last action done.

The model that results from changing the sets S_0 , $F(a, s)$, and $O(s, a)$ in the partially observable model, by the probability distributions $P(s)$, $P_a(s'|s)$, and $P_a(o|s)$, is known as a *Partially Observable Markov Decision Process* or POMDP [Kaelbling et al., 1998]. The advantages of representing uncertainty by probabilities rather than sets is that one can then talk about the *expected cost* of a solution as opposed to the cost of the solution in the *worst case*. Indeed, there are many meaningful problems that have infinite cost in the worst case but perfectly well-defined expected costs. These include, for example, the problem of preparing an omelette with an infinite collection of eggs that may be good or bad with non-zero probabilities, but that can be picked up and sensed one at a time. Indeed, while the scope of probabilistic models is larger than the scope of logical models, we will consider both, as the latter are simpler, and the computational ideas are not all that different.

A *fully observable* model is a partially observable model where the state of the system is fully observable, i.e., where $O = S$ and $O(s, a) = \{s\}$. In the logical setting such models are known as *Fully Observable Non-Deterministic* models, abbreviated FOND. In the probabilistic setting, they are known as *Fully Observable Markov Decision Processes* or MDPs [Bertsekas, 1995].

Finally, an *unobservable* model is a partially observable model where no relevant information about the state of the system is available. This can be expressed through a sensor model O containing a single dummy token o that is “observed” in all states, i.e., $O(s, a) = O(s', a) = \{o\}$ for all s , s' , and a . In planning, such models are known as *conformant*, and they are defined exactly like partially observable problems but with *no sensor* model. Since there are no (true) observations, the solution form of conformant planning problems is like the solution form of classical planning problems: a fixed action sequence. The difference between classical and conformant plans, however, is that the former must achieve the goal for the given initial state and unique state-transitions, while the latter must achieve the goal in spite of the uncertainty in the initial situation and dynamics, for *any possible initial state and any state transition that is possible*. As we will see, conformant problems make up an interesting stepping stone in the way from classical to partially observable planning.

In the book, we will consider each of these models in turn, some useful special cases, and some variations. This variety of models is the result of several orthogonal dimensions: uncertainty in the initial system state (fully known or not), uncertainty in the system dynamics (deterministic or not), the type of feedback (full, partial or no state feedback), and whether uncertainty is represented by sets of states or probability distributions.

1.3 GENERALITY, COMPLEXITY, AND SCALABILITY

Classical planning, the simplest form of planning where actions have deterministic effects and the initial state is fully known, can be easily cast as a path-finding problem over a directed graph where the nodes are the states, the initial node and target nodes are the initial and goal states, and a directed edge between two nodes denotes the existence of an action that maps one state into the other. Classical planning problems can thus be solved in theory by standard path-finding algorithms such as Dijkstra's that run in time that is polynomial in the number of nodes in the graph [Cormen et al., 2009, Dijkstra, 1959]. Yet in planning, this is not good enough as the nodes in the graph stand for the problem states, whose number is exponential in the number of problem variables. If these variables have at least two

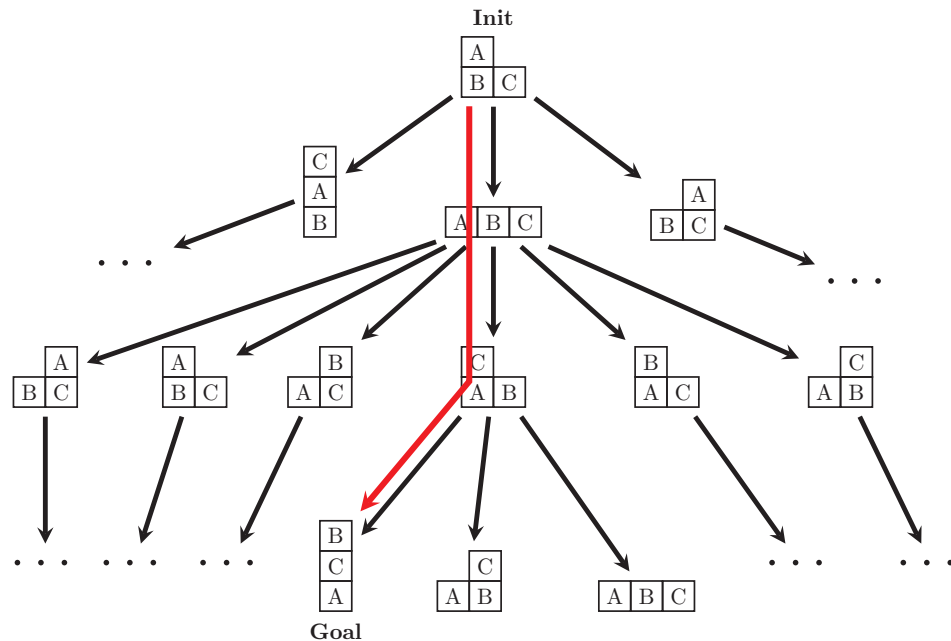


Figure 1.3: The graph corresponding to a simple planning problem involving three blocks with initial and goal situations as shown. The actions allow to move a clear block on top of another clear block or to the table. The size of the complete graph for this domain is exponential in the number of blocks. A plan for the problem is shown by the path in red.

possible values, the number of nodes in the graph to search can be in the order of 2^n , where n is the number of variables. In particular, if the problem involves 30 variables, this means 1,073,741,824 nodes, and if the problem involves 100 variables, it means more than 10^{30} nodes. In order to get a concrete idea of what exponential growth means, if it takes one second to generate 10^7 nodes (a realistic estimate given current technology), it would take more than 10^{23} seconds to generate 10^{30} nodes. This is however almost one million times the estimated age of the universe.³

A more vivid illustration of the complexity inherent to the planning problem can be obtained by considering a well known domain in AI: the Blocks World. Figure 1.3 shows an instance of this domain where blocks A, B, and C, initially arranged so that A is on B, and B and C are on the table, must be rearranged so that B is on C, and C is on A. The actions allow to move a clear block (a block with no block on top) on top of another clear block or on the table. The problem can be easily expressed as a classical planning problem where the variables are the block locations: blocks can be on the table or on top of another block. The figure shows the graph associated to the problem whose solution is a path connecting the node representing the initial situation with a node representing a goal situation. The number of states in a Blocks World problem with n blocks is exponential in n , as the states include all the $n!$ possible towers of n blocks plus additional combinations of lower towers. Thus, a planner

³The age of the universe is estimated at 13.7×10^9 years approximately. Visiting 2^{100} nodes at 10^7 nodes a second would take in the order of 10^{15} years, as $2^{100}/(10^7 * 60 * 60 * 24 * 365) = 4.01969368 \times 10^{15}$.

8 1. PLANNING AND AUTONOMOUS BEHAVIOR

able to solve arbitrary Blocks World instances should be able to search for paths over huge graphs. This is a crisp computational challenge that is very different from writing a *domain-specific* Blocks World solver—namely, a program for solving any instance of this specific domain. Such a program could follow a domain-specific strategy, like placing all misplaced blocks on the table first, in order, from top to bottom, then moving these blocks to their destination in order again, this time, from the bottom up. This program will solve any instance of the Blocks World but will be completely useless in other domains. The challenge in planning is to achieve both *generality* and *scalability*. That is, a classical planner must accept a description of *any problem* in terms of a set of variables whose initial values are known, a set of actions that change the values of these variables deterministically, and a set of goals defined over these variables. The planner is domain-general or domain-independent in the sense that it does not know what the variables, actions, and domain stand for, and for any such description it must decide effectively which actions to do in order to achieve the goals.

For classical planning, as for the other planning models that we will consider, the general problem of coming up with a plan is NP-hard [Bylander, 1994, Littman et al., 1998]. In Computer Science, an NP-hard problem (non-deterministic polynomial-time hard) is a problem that is at least as hard as any NP-complete problem; these are problems that can be solved in polynomial time by a non-deterministic Turing Machine but which are widely believed not to admit polynomial-time solutions on deterministic machines [Sipser, 2006]. The complexity of planning and related models has been used as evidence for contesting the possibility of general planning and reasoning abilities in humans or machines [Tooby and Cosmides, 1992]. The complexity of planning, however, just implies that no planner can efficiently solve every problem from every domain, not that a planner cannot solve an infinite collection of problems from seen and unseen domains, and hence be useful to an acting agent. This is indeed the way modern AI planners are empirically evaluated and ranked in the AI planning competitions, where they are tried over domains that the planners’ authors have never seen. Thus, far from representing an insurmountable obstacle, the twin requirements of generality and scalability have been addressed head on in AI planning research, and have resulted in simple but powerful computational principles that make domain-general planning feasible. The computational challenge aimed at achieving both scalability and generality over a broad class of intractable models, has actually come to characterize a lot of the research work in AI, that has increasingly focused on the development of effective algorithms or *solvers* for a wide range of tasks and models (Figure 1.4); tasks and models that include SAT and SAT-variants like Weighted-Max SAT and Weighted Model Counting, Bayesian Networks, Constraint Satisfaction, Answer Set Programming, General Game Playing, and Classical, MDP, and POMDP Planning. This is all work driven by theory and experiments, with regularly held competitions used to provide focus, to assess progress, and to sort out the ideas that work best empirically [Geffner, 2013a].

1.4 EXAMPLES

We consider next a simple navigation scenario to illustrate how different types of planning problems call for different planning models and different solution forms. The general scenario is shown in Figure 1.5 where the agent marked as A has to reach the goal marked as G. The four actions available let the agent move one unit in each one of the four cardinal directions, as long as there is no wall. Actions that lead the agent to a wall have no effect. The question is how should the agent select the actions for achieving the goal with certainty under different knowledge and sensing conditions. In all