Lecture Notes in Artificial Intelligence 4929

Edited by J. G. Carbonell and J. Siekmann

Subseries of Lecture Notes in Computer Science

Malte Helmert

Understanding Planning Tasks

Domain Complexity
and Heuristic Decomposition

Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Author

Malte Helmert
Albert-Ludwigs-Universität Freiburg, Institut für Informatik
Georges-Köhler-Allee, Geb. 052, 79110 Freiburg, Germany
E-mail: helmert@informatik.uni-freiburg.de

Library of Congress Control Number: 2007943037

CR Subject Classification (1998): I.2.8, F.1.3, F.2, E.1, G.1.6

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN	0302-9743
ISBN-10	3-540-77722-9 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-77722-9 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Markus Richter, Heidelberg
Printed on acid-free paper SPIN: 12215133 06/3180 5 4 3 2 1 0

Foreword

Action planning is an area that has played a central role in Artificial Intelligence since its beginning. Given a description of the current situation, a description of possible actions and a description of the goals, the task is to identify a sequence of actions, a plan, that transforms the current situation into one that satisfies the goal description.

Even if we restrict this problem to a setting where the environment is completely observable, all actions are deterministic, and there are no exogenous events, the planning problem is computationally very difficult. For planning formalisms that are considered these days, such as PDDL, the problem is EXPSPACE-complete, and even for a purely propositional setting the planning problem is still PSPACE-complete. For this reason, it is very unlikely that we will ever come up with planning algorithms that can solve arbitrary planning tasks in reasonable time. Instead, the planning community has concentrated on developing methods that work well with "typical" tasks.

In order to push the advance of the state of the art, in 1998 planning researchers initiated the biennial International Planning Competitions (IPC). These competitions had two very important effects on research in planning. First of all, they led to the process of defining a quasi standard planning formalism, called the *planning domain definition language* (PDDL). Secondly, for each competition, new sets of planning problems are introduced, giving the research community a rich set of benchmark problems.

And in this context, Malte Helmert makes two important contributions. First of all, the book contains an exhaustive analysis of the computational complexity of the benchmark problems that have been used in the first four competitions. Secondly, drawing on structural similarities between a number of the benchmark problems, a new planning technique is derived. The effectiveness of this technique was demonstrated at the fourth International Planning Competition in 2004, where the planning system *Fast Downward* won the first prize in the non-optimal propositional planning track.

When using benchmark problems to evaluate algorithms, one should have a good idea of the properties of these problems. In particular, in the planning case one should have an idea of what the inherent computational complexity of the problem is and what the source of the difficulty is. So far, only one planning domain, the so-called *blocks world* had been analyzed from a computational complexity point of view. However, nobody had a look at all the other planning domains. In fact, Malte was the first one to analyze all the benchmark problems from an analytical point of view. Furthermore, he also provided an in-depth analysis of what he called routing and transportation problems, which is a recurring topic in a large number of planning problems in the literature and of the benchmark problems designed for the IPC. Additionally, he analyzed the approximability of all the benchmarks, giving an indication of how easy it may be to come up with near-optimal solutions instead of optimal solutions. All in all, this book gives the most comprehensive analysis of planning benchmarks so far and it provides a good idea of how hard the different planning domains are and where the sources for (NP- or PSPACE-)hardness lie.

One outcome of the analysis was the observation that a large number of domains have a "hierarchical structure". However, in order to exploit this structure, it is necessary to move from binary to multi-valued state variables. So, one part of this contribution is a development of a method for automatic reformulation of planning domains. On the basis of such a reformulation of the domain description, it is then possible to derive heuristics that are based on the hierarchical structure of the domain. Interestingly, this resembles planning based on abstraction with the important difference that one does not need to rely on strong refinement properties. Thus, abstractions appear to be much more useful in the context of generating heuristics.

In summary, both contributions of this book advance the state of the art in automatic planning significantly, and in particular the new method for deriving heuristics appears to be quite powerful and seems to have potential beyond what has been explored in this book.

Freiburg, November 2007 Bernhard Nebel

Preface

This volume is concerned with the *classical planning problem*, which can be informally defined as follows:

> Given a description of the current situation (an *initial state*) of an agent, the means by which the agent can alter this situation (a set of *actions*) and a description of desirable situations (a *goal*), find a sequence of actions (a *plan*) that leads from the current situation to one which is desirable.

Instances of the classical planning problem, called *planning tasks*, can model all kinds of abstract reasoning problems in areas as diverse as elevator control, the transportation of petroleum products through pipeline networks, or the solution of solitaire card games.

These applications have little in common apart from the fact that, at a suitable level of abstraction, they can be precisely modelled using the notions of initial states, actions, and goals. In the elevator example, the initial state is given by the current location of the elevator and the locations of passengers waiting at different floors to board the elevator. The set of actions comprises movements of the elevator between different floors along with the ensuing activities of passengers boarding and leaving the elevator. The goal specifies a destination floor for each passenger. In the pipeline example, the initial state describes the initial contents of the pipelines and of the areas they connect. Actions model the changes in the contents of a pipeline as products are pumped through it, and a typical goal requires a certain amount of petroleum product to be available in a certain area. In the card game example, the initial state is given by a randomly dealt card tableau. The set of actions models the different ways of moving cards between piles that are allowed by the rules of the game. The goal consists of achieving a certain arrangement of the cards.

Because planning is not limited to a particular application area, or indeed any finite set of application areas, it is an example of *general problem solving*, and in fact the planning problem was first introduced to the Artificial Intelligence community under that name. Classical planning has been an active research area for about half a century, with Newell and Simon's work on the General Problem Solver [94] usually seen as a starting point. A historical perspective on planning research is provided by a collection of classical papers edited by Allen et al. [1] and a more recent survey by Weld [112].

One of the well-established facts about the planning problem is that it is *hard.* In the very general variants commonly studied in the early days of planning research, such as the original STRIPS formalism [40,84], it is known to be undecidable [39]. In the more restricted formalisms typically considered today, the problem is still at least PSPACE-hard [19].

How do we solve such a hard problem? In his 1945 classic *How to Solve It*, mathematician George Pólya describes a four-step strategy to problem solving. Here is the first and most important step:

"First, you have to *understand* the problem." [99, p. 5]

This is solid general advice. To solve the planning problem, that is to design efficient planning algorithms, it is important to understand it. Is planning difficult? Can we *prove* that it is difficult? Are there relevant special cases which are easier to solve than others?

This volume contributes to the understanding of the planning problem by formally analyzing those special cases which have attracted most attention in the past decade, namely, the standard *benchmark domains* of the International Planning Competitions [8,66,86,87,91]. This is the topic of Part I, *Planning Benchmarks.*

Because planning is general *problem* solving, planning tasks are commonly called planning *problems* in the literature. This implies that Pólya's recommendation is equally applicable to *their* solution: To solve a planning task, one has to understand it. Without any kind of intuition of which actions are useful for achieving the goals in a certain situation, the problem solver is more or less limited to blindly exploring the space of possible solutions, which is usually a fruitless endeavour. Given that we are interested in *algorithmic* approaches to planning, this "understanding" must be arrived at algorithmically. One well-established approach to *informed* planning algorithms is the use of heuristic search techniques [16,68,90]. (Not entirely coincidentally, Pólya's work is also responsible for introducing the word *heuristic* into modern scientific discourse, although not quite with the meaning in which it is generally used in Artificial Intelligence these days.)

This volume contributes to the practice of solving planning tasks by presenting a new approach to heuristic planning based on two central ideas: reformulating planning tasks into a form in which its logical structure is more apparent than in the original specification, and exploiting the information encoded in the *causal graphs* and *domain transition graphs* of these reformulated tasks. This is the topic of Part II, *Fast Downward.*

This volume is a revised version of my doctoral thesis, *Solving Planning Tasks in Theory and Practice* [61], submitted to Albert-Ludwigs-Universität Freiburg in June 2006. It has been a long time in the making, with the first ideas conceived in 1998. Over the years, many people have contributed to the work in some way or other, and I would like to use this opportunity to thank them.

Throughout my PhD studies, which began in 2001, Bernhard Nebel served as my advisor. In addition to providing a perfect work environment, his advice has always been very helpful. While giving me lots of freedom to pursue the scientific topics I was interested in, he pushed me at the right times and with the right amount of force to actually get the work done, which is a crucial contribution.

Sylvie Thiébaux served as the second reviewer of the thesis and gave me some exceptional feedback. I very strongly doubt that anyone else will ever read the thesis as closely as her, myself included.

This work has greatly benefited from feedback and discussions with many people in the AI planning community and some people outside. In addition to Bernhard and Sylvie, I particularly want to thank Carmel Domshlak, Stefan Edelkamp, Maria Fox, Inge Li Gørtz, Jörg Hoffmann, Derek Long, Silvia Richter, Jussi Rintanen and Menkes van den Briel for their scientific input.

Many of the results in Chaps. 4 and 5 build on the work of former students whose work I supervised. I want to thank Michael Drescher, Robert Mattmüller and Gabi Röger for their contributions.

I already mentioned the perfect work environment, and I want to use the opportunity to thank all my former and current colleagues at the Foundations of Artificial Intelligence group at Albert-Ludwigs-Universität Freiburg for useful discussions and for having a lot of fun together. Special thanks go to my officemates throughout the years, in chronological order: Rudi Triebel, Yacine Zemali, Christian Köhler, Yuliya Lierler, Marco Ragni, Yannis Dimopoulos and Sebastian Kupferschmid.

For their help – and patience – in the preparation of this volume, I want to thank Ursula Barth and Frank Holzwarth from Springer. Extra special thanks go to Gabi Röger, who undertook the very laborious task of preparing the keyword index.

Last but certainly not least, for moral support my heartfelt thanks goes to Stefan Franck, Sebastian Kupferschmid, Silvia Richter, Anna Seesjärvi and Libor Valevsky and to my family Gundula, Michael and Volker Helmert.

Freiburg, November 2007 Malte Helmert

Contents

Part I Planning Benchmarks

1 The Role of Benchmarks 3
1.1 Evaluating Planner Performance 3
1.1.1 Worst-Case Evaluation 4
1.1.2 Average-Case Evaluation 5
1.2 Planning Benchmarks Are Important 7
1.3 Theoretical Analyses of Planning Benchmarks 8
1.3.1 Why Theoretical Analyses Are Useful 8
1.3.2 Published Results on Benchmark Complexity 9
1.4 Standard Benchmarks 9
1.5 Summary and Overview 11

2 Defining Planning Domains 13
2.1 Optimization Problems 13
2.1.1 Minimization Problems 14
2.1.2 Approximation Algorithms 15
2.1.3 Approximation Classes 16
2.1.4 Reductions 18
2.2 Formalizing Planning Domains 21
2.3 General Results and Reductions 24
2.3.1 Upper Bounds 24
2.3.2 Shortest Plan Length 25
2.3.3 Approximation Classes of Limited Interest 26
2.3.4 Relating Planning and (Bounded) Plan Existence 28
2.3.5 Generalization and Specialization 29

3 The Benchmark Suite 31
3.1 Defining the Competition Domains 31
3.2 The Benchmark Suite 32
3.2.1 IPC1 Domains 32

3.2.2 IPC2 Domains 34
3.2.3 IPC3 Domains 34
3.2.4 IPC4 Domains 35
3.3 Domains and Domain Families 36

4 Transportation and Route Planning 39
4.1 Transport and Route 39
4.1.1 The Transport Domain 41
4.1.2 The Route Domain 43
4.1.3 Special Cases and Hierarchy 44
4.2 General Results 46
4.3 Plan Existence 52
4.4 Hardness of Optimization 54
4.5 Constant Factor Approximation 59
4.6 Hardness of Constant Factor Approximation 62
4.7 Summary 68
4.8 Beyond Transport and Route 71

5 IPC Domains: Transportation and Route Planning 75
5.1 Gripper 75
5.2 Mystery and MysteryPrime 76
5.3 Logistics 78
5.4 Zenotravel 83
5.5 Depots 85
5.6 Miconic-10 88
5.7 Rovers 93
5.8 Grid 98
5.9 Driverlog 103
5.10 Airport 108
5.11 Summary 111

6 IPC Domains: Others 113
6.1 Assembly 113
6.2 Blocksworld 117
6.3 FreeCell 117
6.4 Movie 126
6.5 Pipesworld 127
6.6 Promela 132
6.7 PSR 138
6.8 Satellite 142
6.9 Schedule 145
6.10 Summary 149

7 Conclusions . . . 151
7.1 Ten Conclusions . . . 151
7.2 Going Further . . . 154

Part II Fast Downward

8 Solving Planning Tasks Hierarchically . . . 157
8.1 Introduction . . . 157
8.2 Related Work . . . 163
8.2.1 Causal Graphs and Abstraction . . . 164
8.2.2 Causal Graphs and Unary STRIPS Operators . . . 165
8.2.3 Multi-Valued Planning Tasks . . . 167
8.3 Architecture and Overview . . . 168

9 Translation . . . 171
9.1 PDDL and Multi-valued Planning Tasks . . . 171
9.2 Translation Overview . . . 175
9.3 Normalization . . . 176
9.3.1 Compiling Away Types . . . 177
9.3.2 Simplifying Conditions . . . 177
9.3.3 Simplifying Effects . . . 179
9.3.4 Normalization Result . . . 179
9.4 Invariant Synthesis . . . 180
9.4.1 Initial Candidates . . . 182
9.4.2 Proving Invariance . . . 183
9.4.3 Refining Failed Candidates . . . 186
9.4.4 Examples . . . 188
9.4.5 Related Work . . . 188
9.5 Grounding . . . 190
9.5.1 Overview of Horn Exploration . . . 191
9.5.2 Generating the Logic Program . . . 191
9.5.3 Translating the Logic Program to Normal Form . . . 193
9.5.4 Computing the Canonical Model . . . 195
9.5.5 Axiom and Operator Instantiation . . . 197
9.6 Multi-valued Planning Task Generation . . . 197
9.6.1 Variable Selection . . . 198
9.6.2 Converting the Initial State . . . 199
9.6.3 Converting Operator Effects . . . 200
9.6.4 Converting Conditions . . . 201
9.6.5 Computing Axiom Layers . . . 202
9.6.6 Generating the Output . . . 202
9.7 Performance Notes . . . 203
9.7.1 Relative Performance Compared to MIPS Translator . . . 203
9.7.2 Absolute Performance . . . 205

10 Knowledge Compilation 207
10.1 Overview 207
10.2 Domain Transition Graphs 208
10.3 Causal Graphs 213
10.3.1 Acyclic Causal Graphs 214
10.3.2 Generating and Pruning Causal Graphs 215
10.3.3 Causal Graph Examples 217
10.4 Successor Generators and Axiom Evaluators 220
10.4.1 Successor Generators 220
10.4.2 Axiom Evaluators 221

11 Search 223
11.1 Overview 223
11.2 The Causal Graph Heuristic 224
11.2.1 Conceptual View of the Causal Graph Heuristic 225
11.2.2 Computation of the Causal Graph Heuristic 226
11.2.3 States with Infinite Heuristic Value 228
11.2.4 Helpful Transitions 229
11.3 The FF Heuristic 230
11.4 Greedy Best-First Search in Fast Downward 231
11.4.1 Preferred Operators 231
11.4.2 Deferred Heuristic Evaluation 232
11.5 Multi-heuristic Best-First Search 233
11.6 Focused Iterative-Broadening Search 234

12 Experiments 239
12.1 Experiment Design 239
12.1.1 Benchmark Set 240
12.1.2 Experiment Setup 242
12.1.3 Translation and Knowledge Compilation vs. Search 243
12.2 STRIPS Domains from IPC1–3 243
12.3 ADL Domains from IPC1–3 246
12.4 Domains from IPC4 248
12.5 Conclusions from the Experiment 251

13 Discussion 253
13.1 Summary 253
13.2 Major Contributions 254
13.2.1 Multi-valued Representations 254
13.2.2 Task Decomposition Heuristics 256
13.3 Minor Contributions 257
13.4 Going Further 258

References 259

Index 267