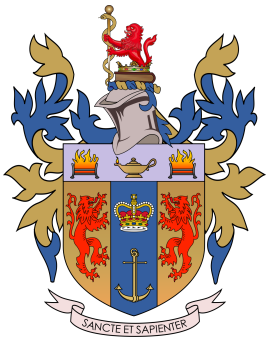

From Optimal Planning to Robotics

A work in progress

Ionut Moraru

Department of Informatics
King's College London

A thesis submitted in partial fulfilment for the degree of
Doctor of Philosophy

August 2021

I would like to dedicate this thesis to my loving parents ...

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Ionut Moraru
August 2021

Acknowledgements

And I would like to acknowledge ...

Abstract

This is where you write your abstract ...

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Motivation

I like AI...

1.2 Thesis Structure and Contribution

1.3 Publications

Chapter 2

Related Works and Background

2.1 Introduction

2.2 Planning

2.3 Optimal Planning

2.4 Satisfacing

2.5 Heuristic Search

2.6 Related Works

2.7 Plan Library in ROSPlan

Chapter 3

Optimal Planning with Pattern Databases

3.1 Introduction

The automated generation of search heuristics is one of the holy grails in AI, and goes back to early work of Gaschnik [11], Pearl [23], and Prieditis [24]. In most cases, lower bound heuristics are problem relaxations: each plan in the original state space maps to a shorter one in some corresponding abstract one. In the worst case, searching the abstract state spaces at every given search nodes exceeds the time of blindly searching the concrete search space [26]. With pattern databases (PDBs), all efforts in searching the abstract state space are spent prior to the plan search, so that these computations amortize through multiple lookups.

Initial results of Culberson and Schaeffer [5] in sliding-tile puzzles, where the concept of a pattern is a selection of tiles, quickly carried over to a number of combinatorial search domains, and helped to optimally solve random instances of the Rubik's cube, with non-pattern labels being removed [19]. When shifting from breadth-first to shortest-path search, the exploration of the abstract state-space can be extended to include action costs.

The combination of several databases into one, however, is tricky [13]. While the maximum of two PDBs always yields a lower bound, the sum usually does not. Korf and Felner [20] showed that with a certain selection of disjoint (or additive) patterns, the values in different PDBs can be added while preserving admissibility. Holte et al. [15] indicated that several smaller PDBs may outperform one large PDB. The notion of a pattern has been generalized to production systems in vector notation [16], while the

automated pattern selection process for the construction of PDBs goes back to the work of Edelkamp [7].

Many planning problems can be translated into state spaces of finite domain variables [14], where a selection of variables (pattern) influences both states and operators. For disjoint patterns, an operator must distribute its original cost, if present in several abstractions [18, 27].

During the PDB construction process, the memory demands of the abstract state space sizes may exceed the available resources. To handle large memory requirements, symbolic PDBs succinctly represent state sets as binary decision diagrams [6]. However, there are an exponential number of patterns, not counting alternative abstraction and cost partitioning methods. Hence, the automated construction of informative PDB heuristics remains a combinatorial challenge. Hill-climbing strategies have been proposed [13], as well as more general optimization schemes such as genetic algorithms [7, 10]. The biggest area of research in this area remains the quality evaluation of a PDB (in terms of the heuristic values for the concrete state space) which can only be estimated. Usually, this involves generating the PDBs and evaluating them [8, 19].

3.2 Automatically Created Pattern Databases

Planning is a PSPACE-complete problem [4], heuristic search has proven to be one of the best ways to find solutions in a timely manner.

3.2.1 Pattern Selection

3.3 Results

Chapter 4

Plan Library

4.1 Introduction

In order for robots like this to become helpful in dynamic and stochastic environments, their reasoning about what to do must combine two qualities: autonomy and speed. They need autonomy in order to be able to decide for themselves how to achieve their goals, regardless of the situation they are placed in [17]. They need speedy reasoning so that they can perform in dynamic environments where plans can become unusable if a robot takes too long to synthesise them. Any environment containing people is dynamic, because people act in it, and include the additional constraint that people who interact with robots will not tolerate waiting long for them to respond.

For a long time, robots have been designed using the three layered architecture [12], [1], [21], known as the Sense-Plan-Act (SPA) paradigm. The *sensing* component makes sense of real time observations from the environment, and of monitors if observations are consistent with the plan being executed. The *planning* component is tasked with reaching the robot's goals while taking into consideration what the sensing component provides as an initial state. Finally, the *acting* or executing part will put the plan into action by using the robot's actuators. For the remainder of this paper, we shall focus on the *planning* component of the SPA.

One way to ensure that the robot has a high degree of autonomy is by reasoning directly about the state of the environment. Systems that do this are mostly based on STRIPS [9], and help the robot to come up with a sequence of actions (i.e. plans), from a set of available operators that would satisfy a set of explicit goals given in a planning task. Such an approach trades speed for autonomy, because planning with a suitably expressive language — able, for example, to reason about the duration and cost of actions — is computationally expensive. The success of a robot is also dependant in the speed

of which planning is done (i.e.: the time from which the planning component receives a planning task until it submits the plan for execution). If the robot has a valid plan for a task, but the task is no longer consistent with the current state of the environment (due to the long time taken to create the plan), then it needs to re-plan, restarting the reasoning process.

Because of these issues, approaches have been developed that make use of predefined plans. One influential family of approaches are those based on the Belief-Desire-Intention (BDI) paradigm [3]. Systems that are based on the BDI model include PRS [25] and Jason [2] which have a prescribed Plan Library, comprising a set of plan rules. Each plan rule consists of a *header*, which defines the situation where it is applicable, and a *sequence of actions* that will fulfil the robot's goals. The downside to this approach is that the robot is limited to the behaviours described in the Plan Library, and therefore has less autonomy than a robot that can compute its own plans.

Over the last decades, researchers have introduced task planning into BDI agents [22], where the main focus is planning when there is no available option in the Plan Library. In other words, planning is considered to be a last resort, only to be invoked if necessary.

4.2 Plan Libraries for ROSPlan

Planning is a PSPACE-complete problem [4], heuristic search has proven to be one of the best ways to find solutions in a timely manner.

4.2.1 Plan Selection

Study on Pattern Selection

4.3 Related Works

4.4 Summary of Results

Chapter 5

Other works

5.1 Introduction

Chapter 6

Conclusion and Future Work

6.1 Experimental Analysis

6.1.1 Summary Evaluation

6.1.2 The 2018 Internationa Planning Competition

6.1.3 Summary

6.2 Conclusions

6.2.1 Contributions

6.2.2 Future Work

References

[1] Ambros-Ingerson, J. A. and Steel, S. (1988). Integrating planning, execution and monitoring. In *AAAI*, volume 88, pages 21–26.

[2] Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007). *Programming multi-agent systems in AgentSpeak using Jason*. John Wiley & Sons.

[3] Bratman, M. et al. (1987). *Intention, plans, and practical reason*, volume 10. Harvard University Press Cambridge, MA.

[4] Bylander, T. (1994). The computational complexity of propositional strips planning. *Artificial Intelligence*, 69(1-2):165–204.

[5] Culberson, J. C. and Schaeffer, J. (1998). Pattern databases. *Computational Intelligence*, 14(4):318–334.

[6] Edelkamp, S. (2002). Symbolic pattern databases in heuristic search planning. In *AIPS*, pages 274–283.

[7] Edelkamp, S. (2006). Automated creation of pattern database search heuristics. In *International Workshop on Model Checking and Artificial Intelligence*, pages 35–50. Springer.

[8] Edelkamp, S. (2014). Planning with pattern databases. In *Sixth European Conference on Planning*.

[9] Fikes, R. E. and Nilsson, N. J. (1971). Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208.

[10] Franco, S., Torralba, A., Lelis, L. H., and Barley, M. (2017). On creating complementary pattern databases. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4302–4309. AAAI Press.

[11] Gaschnig, J. (1979). A problem similarity approach to devising heuristics: First results. pages 434–441.

[12] Gat, E., Bonnasso, R. P., Murphy, R., et al. (1998). On three-layer architectures. *Artificial intelligence and mobile robots*, 195:210.

[13] Haslum, P., Botea, A., Helmert, M., Bonet, B., and Koenig, S. (2007). Domain-independent construction of pattern database heuristics for cost-optimal planning. pages 1007–1012.

[14] Helmert, M. (2004). A planning heuristic based on causal graph analysis. pages 161–170.

[15] Holte, R., Newton, J., Felner, A., Meshulam, R., and Furcy, D. (2004). Multiple pattern databases. pages 122–131.

[16] Holte, R. C. and Hernádvölgyi, I. T. (1999). A space-time tradeoff for memory-based heuristics. In *AAAI/IAAI*, pages 704–709. Citeseer.

[17] Ingrand, F. and Ghallab, M. (2017). Deliberation for autonomous robots: A survey. *Artificial Intelligence*, 247:10–44.

[18] Katz, M. and Domshlak, C. (2008). Optimal additive composition of abstraction-based admissible heuristics. In *ICAPS*, pages 174–181.

[19] Korf, R. E. (1997). Finding optimal solutions to Rubik's Cube using pattern databases. pages 700–705.

[20] Korf, R. E. and Felner, A. (2002). *Chips Challenging Champions: Games, Computers and Artificial Intelligence*, chapter Disjoint Pattern Database Heuristics, pages 13–26. Elsevier.

[21] Kortenkamp, D., Schreckenghost, D., and Bonasso, R. (1998). Three nasa application domains for integrated planning, scheduling and execution.

[22] Meneguzzi, F. and De Silva, L. (2015). Planning in BDI agents: a survey of the integration of planning algorithms and agent reasoning. *The Knowledge Engineering Review*, 30(1):1–44.

[23] Pearl, J. (1984). *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley.

[24] Preditis, A. (1993). Machine discovery of admissible heuristics. *Machine Learning*, 12:117–142.

[25] Rao, A. S. and Georgeff, M. P. (1995). Bdi agents: from theory to practice. In *Proceedings of the First International Conference on Multiagent Systems.*, pages 312–319.

[26] Valtorta, M. (1984). A result on the computational complexity of heuristic estimates for the A* algorithm. *Information Sciences*, 34:48–59.

[27] Yang, F., Culberson, J., Holte, R., Zahavi, U., and Felner, A. (2008). A general theory of additive state space abstractions. *Journal of Artificial Intelligence Research*, 32:631–662.