Studies in Action Planning
Algorithms and Complexity

Peter Jonsson

Abstract

The action planning problem is known to be computationally hard in the general case. Propositional planning is PSPACE-complete and first-order planning is undecidable. Consequently, several methods to reduce the computational complexity of planning have been suggested in the literature. This thesis contributes to the advance and understanding of some of these methods.

One proposed method is to identify restrictions on the planning problem which ensure tractability. We propose a method using a state-variable model for planning and define structural restrictions on the state-transition graph. We also present a planning algorithm that is correct and tractable under these restrictions as well as a map of the complexity results for planning under our new restrictions and certain previously studied restrictions. The algorithm is further extended to apply to a miniature assembly line.

Another method that has been studied is state abstraction. The idea is to first plan for the most important goals and then successively refine the plan to also achieve the less important goals. It is known that this method can speed up planning exponentially under ideal conditions. We show that state abstraction may likewise slow down planning exponentially and even result in generating an exponentially longer solution than necessary.

Reactive planning has been proposed as an alternative to classical planning. While a classical planner first generates the whole plan and then executes it, a reactive planner generates and executes one action at a time, based on the current state. One of the approaches to reactive planning is to use universal plans. We show that polynomial-time universal plans satisfying even a very weak notion of completeness must be of exponential size in the worst case.

A trade-off between classical and reactive planning is incremental planning, *i.e.* a planner that can output valid prefixes of the final plan before it has finished planning. We present a correct incremental planner for a restricted class of planning problems. The plan existence problem is tractable for this class despite the fact that the plan generation problem is provably exponential. Hence, by first testing whether an instance is solvable or not, we can avoid starting to generate prefixes of invalid plans.

Acknowledgements

This thesis is based on work carried out at the Department of Computer and Information Science (IDA) in Linköping since the autumn of 1993. Many people, both inside and outside IDA, have enabled and supported this work and I am very grateful to them all. However, some of them deserve being mentioned explicitly.

My advisor Christer Bäckström for three years of excellent guidance. It has been a very interesting and productive collaboration which, hopefully, will continue in the future.

Erik Sandewall and Jan Małuszyński, my two co-advisors, for help and encouragement.

My colleagues at IDA for providing a stimulating research environment; in particular, the members of RKLLAB and its three daughter labs TOSCA, KPLAB and TASLAB. Special thanks go to Magnus Andersson, Marcus Bjäreland, Thomas Drakengren, Fredrik Eklund, Lars Karlsson and Simin Nadjm-Tehrani.

Inger Klein in the Automatic Control Group for collaboration and interesting discussions.

Lise-Lott Svensson and Lillemor Wallgren for administrative help. Leif Finmo for technical assistance.

Ivan Rankin for improving my English. Naturally, remaining faults are entirely my own.

Jonas Kvarnström who implemented some of the algorithms presented in this thesis.

Karl-Johan Bäckström for guidance and inspiration while I was writing my master's thesis.

Family and friends for constant support and encouragement.

The research of the author was sponsored by the Swedish Research Council for the Engineering Sciences (TFR) under grant Dnr. 93-00291.

List of Papers

This thesis includes the following five papers.

I. State-Variable Planning under Structural Restrictions: Algorithms and Complexity. Peter Jonsson and Christer Bäckström. Submitted to *Artificial Intelligence.*

II. Tractable Planning for an Assembly Line. Inger Klein, Peter Jonsson and Christer Bäckström. In Malik Ghallab and Alfredo Milani, editors, *New Directions in AI Planning: EWSP'95—3rd European Workshop on Planning, Frontiers in AI and Applications*, 1995, IOS Press.

III. Planning with Abstraction Hierarchies Can Be Exponetially Less Efficient. Christer Bäckström and Peter Jonsson. In Chris S. Mellish, editor, *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, 1995, Morgan Kaufmann.

IV. On the Size of Reactive Plans. Peter Jonsson and Christer Bäckström. Accepted at the 13th (US) National Conference on Artificial Intelligence (AAAI-96), 1996.

V. Incremental Planning. Peter Jonsson and Christer Bäckström. In Malik Ghallab and Alfredo Milani, editors, *New Directions in AI Planning: EWSP'95—3rd European Workshop on Planning, Frontiers in AI and Applications*, 1995, IOS Press. A modified version of this paper has been submitted to *Annals of Mathematics and Artificial Intelligence.*

Paper I is a compilation of the following published papers and research reports.

1. Tractable Planning with State Variables by Exploiting Structural Restrictions. Peter Jonsson and Christer Bäckström. In *Proceedings of the 12th (US) National Conference on Artificial Intelligence (AAAI-94),* 1994, AAAI Press/The MIT Press.

2. Complexity Results for State-Variable Planning under Mixed Syntactical and Structural Restrictions. Peter Jonsson and Christer Bäckström. In Phillipe Jorrand, editor, *Proceedings of the 6th International Conference on Artificial Intelligence: Methodology, Systems, Applications (AIMSA-94),* 1994, World Scientific Publishing.

3. Tractable Planning with State Variables by Exploiting Structural Restrictions. Peter Jonsson and Christer Bäckström. Research report LiTH-IDA-R-95-16, Department of Computer and Information Science, Linköping University, Sweden, 1995.

4. Complexity Results for State-Variable Planning under Mixed Syntactical and Structural Restrictions. Peter Jonsson and Christer Bäckström. Research report LiTH-IDA-R-95-17, Department of Computer and Information Science, Linköping University, Sweden, 1995.

5. Complexity of State-Variable Planning under Structural Restrictions. Peter Jonsson. Licentiate thesis no 478, Department of Computer and Information Science, Linköping University, Sweden, 1995.

In order to improve readability, some of the proofs in Paper I are only sketched or—in some cases—entirely omitted. The complete proofs can be found in the research reports III-V listed above. Papers II-V are also available as research reports:

II. Research report. LiTH-ISY-R-1717. Department of Electrical Engineering. Linköping University, Sweden, 1995.

III. Research report. LiTH-IDA-R-95-12. Department of Computer and Information Science, Linköping University, Sweden, 1995.

IV. Research report. LiTH-IDA-R-96-10. Department of Computer and Information Science, Linköping University, Sweden, 1996.

V. Research report. LiTH-IDA-R-95-31. Department of Computer and Information Science, Linköping University, Sweden, 1995.

1 Thesis Overview

1.1 About this Thesis

The main topic of this thesis is the study of computational complexity in action planning[1]. The action planning problem has been studied by the artificial intelligence community over the past thirty years. However, the intersection of complexity theory and planning has only recently become a topic of study. To our knowledge, the computational complexity of planning was first addressed by David Chapman in the paper "Planning for Conjunctive Goals" [1987] where it was shown that first-order planning is undecidable. Ever since, the complexity of planning has received a constantly increasing amount of interest.

This thesis consists of a collection of papers which fall into two categories. Papers I, II and V are attempts to extend the class of planning problems solvable in polynomial time. Papers III and IV are critical analyses of approaches to reduce the complexity of planning proposed in the literature. All of the papers were originally written to be self-contained. Consequently some material is repeated in the various papers, such as basic definitions. Hopefully, this will not disturb the reader too much. The papers can be read in arbitrary order but it is recommended that paper I is read before paper II and paper IV before paper V. The reader should note that the papers presented in this thesis are somewhat different from the original papers. Minor changes such as typographical adjustments and corrections of misspellings are left without notice whereas substantial changes are pointed out explicitly in the papers.

Some familiarity with complexity theory and action planning is required to read this thesis. It does not require special mathematical knowledge; however, it presupposes an acquaintance with mathematical and logical reasoning. The complexity theory needed can be found in any standard textbook on the subject such as Garey and Johnson [1979] or Papadimitriou [1994]. Introductions to planning can be found in, for example, Tate *et al.* [1990], Allen [1990] and Charniak and McDermott [1985].

[1]This name has been chosen in order to distinguish it from path planning [Schwartz and Sharir, 1990]. It should be noted that action planning is sometimes referred to as *AI planning* in the literature. Unfortunately, the term planning has been given many different meanings, which is why we specialize it to action planning.

1.2 Brief Summary of Papers

Below, we give very brief summaries of the five papers in this thesis. For more extensive summaries, we refer the reader to the abstracts of the papers and Section 2.4-2.7 in this introduction.

Paper I

Classes of tractable planning problems previously reported in the literature have almost exclusively been defined by simple syntactical restrictions on the set of operators [Bylander, 1994; Bäckström and Nebel, 1995]. In this paper, we propose an alternative approach. We use the SAS^+ [Bäckström and Nebel, 1995] state variable model for planning and define restrictions on the state-transition graph that allow for polynomial-time planning. In addition we present a map over the complexity results for planning under all combinations of four previously studied syntactical restrictions and our new structural restrictions.

Paper II

This paper extends the methods for tractable planning given in paper I to apply to a miniature assembly line. Although somewhat limited, this assembly line has many features in common with real industrial processes.

Paper III

This paper provides an analysis of *state abstraction* in planning. It is well-known that state abstraction can speed up planning exponentially if certain conditions are met [Knoblock, 1994]. We show that state abstraction may likewise slow down planning exponentially and generate a solution that is exponentially longer than necessary. We also show that efficient algorithms for constructing good abstraction hierarchies are not likely to exist.

Paper IV

The goal of this paper is to analyse the *universal planning* approach [Schoppers, 1987] to reactive planning. We show that polynomial-time universal plans satisfying even a very weak notion of completeness must be of exponential size unless the polynomial hierarchy collapses.

Paper V

Ambros-Ingerson and Steel [1988] have suggested interleaving planning and execution through incremental planning, *i.e.* using a planner that can output valid prefixes of the final plan before it has finished planning. We present a correct incremental planner for a restricted class of planning problems. The plan existence problem is tractable for this class despite the fact that the plan generation problem is provably exponential. Hence, by first testing whether an instance is solvable or not, we can avoid starting to generate prefixes of invalid plans.

2 Background

This section serves the purpose of placing the papers in this thesis in the context of other related work. Subsections 2.1-2.3 present a brief introduction to planning in general and the subsequent subsections provide background material to papers I-V respectively.

2.1 The Planning Problem

Assume that we are given some *world*, such as a room together with the objects in it, an industrial process or the abstract activities going on inside a computer. Furthermore, suppose that this world can be modelled by some kind of *state*, describing what the world looks like at a certain point in time, and that there is a set of *actions* that can be performed in this world, triggering transitions between states in the world. An action has some *effects*, modelling how the current state of the world would change if this action were to be executed and, typically, some *preconditions* which express what conditions must hold in a state for the action to be executable there. Then, the planning problem is: Given an *initial* state, describing the state of the world when we will start executing the plan and a *goal* state, describing what we want the world to look like, find a sequence of actions that will change the state of the world from the initial state to the goal state. Observe that there are certain restrictions that this sequence of actions must satisfy. First of all, the first action must be executable in the initial state, *i.e.* its preconditions are satisfied there. The successful application of this action in the initial state results in a new state. Then the second action in the sequence must be executable in this new state, resulting in yet a new state and so forth. Finally, the state resulting after executing the last action in the sequence must satisfy the goal. The actions are usually described using *operators*, which are action templates that can be instantiated into specific action occurrences. An action is then a specific instance of an operator.

The world states are typically modelled by formulae in some logic. This leads to a basic distinction made in planning between *first-order* and *propositional* planning formalisms. By a first-order formalism, we mean a formalism where states are described as formulae in some first-order logic. Often, we are not allowed to use any first-order formula in our world descriptions. A typical restriction is that the formulae are required to be atomic and ground, and this restricted formalism is referred to as propositional.

Instead of propositional atoms, multi-valued *state variables* are sometimes used [Sandewall and Rönnquist, 1986; Bäckström and Klein, 1991b]. In such formalisms, each variable can take its values from some pre-defined, finite domain. Clearly, this is a more flexible approach than the propositional one where each atom can take only one of two values, namely true or false. It has been shown by Bäckström [1995a], that multi-valued, discrete state variables can be simulated by propositional atoms in traditional planning formalisms. In some cases, the methods for simulating state-variables are rather intricate and can considerably decrease the performance of the planner [Bäckström, 1994]. Going the other way round, simulating propositional planning with state-variables, is of course trivial (simply by using two-valued state variables).

Robotics has historically been the archetypical application of action planning. It was early realized that planning is vital for robotics; in fact, some authors seem to equate these two issues [Charniak and McDermott, 1985]. However, many non-robotics applications exist and other authors have a broader view. Tate *et al.* [1990, p. 26] write

> Ideally, the set of actions so produced is then passed on to a robot, a manufacturing system or some other form of effector.

For instance, non-robotic applications naturally arise in sequential control[2]. Most industrial processes are continuous systems equipped with a superior discrete level. On this superior level the actions can often be viewed as actions in the AI planning sense. Consequently, many problems on this level can be viewed as planning problems. Typical actions can be to open or close a valve or to start or stop a motor. It should also be noted that the system

[2]Sequential control is a sub-area of automatic control. Sequential control is concerned with the control of discrete systems.

does not necessarily have to be a physical system; the abstract activities going on inside a computer are a typical example of a non-physical system.

2.2 Classical Approaches to Planning

There have traditionally been two major approaches to solving planning problems. We will refer to these as *deductive* planning and *operator-search* planning, respectively.

In deductive planning, both the world and the operators are axiomatized in some suitable logic. Planning is performed by attempting to prove that a state satisfying the goal can be reached from the initial state, using the available operators. If the resulting proof is constructive, the actual solution is also generated as a side-effect. Many logic formalizations of planning have been proposed in the literature and the most widely known is the *situation calculus* by McCarthy and Hayes [1969]. However, few of these formalisms have led to implemented planners or even a planning algorithm. An early exception is the QA3 system [Green, 1969] which is a resolution-based planner for the situation calculus. A more recent example is the LLP system by Biundo *et al.* [1992] which builds on an interval-based temporal logic.

In operator-search planning, world states and operators are represented separately and an operator models how a world state is changed if the operator is executed there. As indicated by the name, operator-search planning uses some search-based method to find a solution, *i.e.* a sequence of actions transforming the initial state into the goal state.

An early example of an operator-search planner is the STRIPS system [Fikes and Nilsson, 1971]. STRIPS is a so-called *total-order* planner since it always works on a sequence of operators. STRIPS introduced (at least) two features that have had great impact on later planners.

- The STRIPS *assumption* was introduced as a solution to the frame problem. (The frame problem is the problem of specifying what remains unchanged when an operator is executed in the world. A thorough investigation of this problem can be found in [Sandewall, 1994].) The assumption states that everything that is not explicitly changed by an operator remains unaffected when executing the operator. Most planners that have been developed after STRIPS have advocated the STRIPS assumption.

- The STRIPS *operator*. In STRIPS, operators are modelled by three sets named the pre-condition, the add list and the delete list. A state is described by a set of logical formulae and the pre-conditions of an operator must hold in the state to execute the operator. The new state is then acheived by deleting the formulae in the delete list from the state and adding the formulae in the add list. STRIPS operators in some form are used in almost every existing planning system.

Several problems with STRIPS have been identified and most of them stem from the strict sequential ordering of operators. To overcome this problem, *partial-order* planners were introduced. The basic idea is to defer decisions on the ordering between operators until such decisions are forced to resolve some conflict. The first example of a partial-order planner was NOAH by Sacerdoti [1975]. Most modern planners belong to this class and typical examples are TWEAK [Chapman, 1987] and SNLP [McAllester and Rosenblitt, 1991].

2.3 The Importance of Tractability

Throughout this thesis, the importance of tractability will be stressed over and over again. By tractability, we usually mean that the algorithms under consideration are required to run in polynomial time. It should be clear that if we intend to apply results from computer science to real-world problems, then tractability is a fundamental issue. If a problem is not tractable, we cannot hope for solving an arbitrary problem instance in reasonable time with any type of computer. Hence, if we want to use computers for solving planning problems, we should demand that the problems we try to solve are tractable. This does not seem to be the common point of view in the artificial intelligence community, however, although there has been an increasing interest in complexity issues over the past few years. Common arguments as to why tractability is not of any significant importance are:

- The intractable cases are rare, pathological cases which do not occur in practice.
- Since planning is computationally hard (even undecidable) in the general case, planning will never succeed anyway.

We will try to meet these arguments in turn. If the hard cases do not occur in real problems, then we have not charachterized the problem properly.

In other words, the real problem is a tractable subproblem of the problem we have characterized. Levesque Levesque [1988, p. 386] has stated this argument as follows:

> If the only problematic cases are the ones that do not seem to occur in practice...we can simply decide to eliminate them from consideration...But this does not eliminate our concern with extreme cases; it merely changes what cases we consider to be extreme.

One objection to this view is that eliminating the problematic cases can be an extremely non-trivial task. In fact, this is what finding tractable subcases of an intractable problem is all about.

Another answer to the first argument is that, in certain cases, we must be absolutely sure that these pathological cases do not occur in practice. Predicting all the infrequent anomalies a real-world system might encounter is obviously an insurmountable task. As a consequence, we will never be able to construct reliable complex systems with "pathological cases never occur" as an underlying assumption.

The second argument leads to a rather interesting observation. Chapman [1987, p. 350] writes about his TWEAK planner:

> The restrictions on action representation make TWEAK almost useless as a real-world planner.

However, on page 344 of the same publication he writes:

> Any Turing machine with its input can be encoded as a planning problem in the TWEAK representation.

Consequently, TWEAK can solve any problem that a computer can solve (assuming Church's hypothesis). Combining these two quotations leads to the conclusion that real-world planning problems cannot be solved by any computer. Most likely, this is not the conclusion Chapman intended. A more reasonable conclusion is that TWEAK is too restricted in some aspects (the world-modelling power), but at the same time being unrestricted in other aspects (the computational power).

2.4 Papers I and II: Tractable Planning

In some branches of automated reasoning and artificial intelligence, *e.g.* terminological logics, abduction and non-monotonic reasoning, research into identifying tractable subcases and sources of intractability has been highly active. Fortunately, we have seen a growing interest in the computational aspects of planning during recent years.

It is now well-known that planning is computationally hard in the general case. The first result concerning the complexity of first-order planning was presented by Chapman [1987] who showed that first-order planning is undecidable. Erol *et. al* [1992] strengthened this result by proving (with a less contrived proof) that first-order planning is undecidable if there are function symbols or infinitely many constants in the language. If no function symbols and only finite domains are allowed, first-order planning is decidable with a complexity ranging from EXPSPACE-complete to polynomial, depending on further restrictions.

Somewhat later, it was shown that propositional planning is PSPACE-complete [Bylander, 1994]. Even worse, there exist propositional planning problems with exponentially-sized minimal solutions [Bäckström and Klein, 1991b], thus making the generation problem for propositional planning provably intractable. A number of restricted cases have been analysed by Bylander [1994] and Erol *et al* [1992] and their complexity ranges from PSPACE-complete to, in rare cases, tractable. These restricted cases are based on simple syntactic restrictions on operators, such as bounding the number of pre- and post-conditions.

Using multi-valued state-variables instead of propositional atoms cannot, of course, simplify the problem. However, it seems that state-variable formalisms can facilitate the search for restrictions leading to tractability. A line of research concerning the complexity of state-variable planning was initiated by Bäckström and Klein [1990]. This paper was the first in a row of papers [Bäckström and Klein, 1991a; Bäckström and Klein, 1991b; Bäckström, 1992; Bäckström and Nebel, 1993] on the complexity of state-variable planning in the SAS$^+$ formalism. By starting with a simple, severely restricted problem and successively removing and replacing restrictions, several tractable subproblems could be identified. This research is summarized in Bäckström and Nebel [1995] and Bäckström [1995b].

It is appealing to study syntactic restrictions since they are typically easy to define and not very costly to test. However, to gain any deeper insight into what makes planning problems hard and easy, respectively, probably requires that we study the structure of the problem, in particular the state-transition graph[3]. Putting explicit restrictions on the state-transition graph must be done with great care since this graph is typically of exponential size in the size of the planning problem instance, making it extremely costly to test arbitrary properties. To exemplify previous usage of structural restrictions in planning, we provide two examples.

Korf [1987] has analysed how different properties of the subgoals affect the complexity of planning. He identified a number of properties, such as subgoal independence and subgoal serializability, and showed that the search space can be drastically reduced when such properties hold. Even though Korf managed to show impressing decreases in the size of the search space in some cases, there are drawbacks with his methods.

- The properties reduce the search needed, but they do not guarantee tractability.
- It is hard to test some of the proposed properties. Bylander [1992] has shown that deciding whether an instance has serializable subgoals or not is PSPACE-complete. Probably we are not willing to spend that much effort, especially since we are not guaranteed tractability in the end anyway.

Another method was suggested by Smith and Peot [1993]. They use an *operator graph* for preprocessing planning problem instances, identifying potential threats that can be safely postponed during planning, thus pruning the search tree. The operator graph can be viewed as an abstraction of the full state-transition graph, containing all the information relevant to analysing threats. As in Korf's approach, Smith and Peot's method does not guarantee tractability. The advantage with operator graphs is that they are not very costly to construct from a given planning problem.

2.5 Paper III: State Abstraction

A common approach to improving the efficiency of planning is to use some abstraction technique, *i.e.* ignoring certain details and iteratively refining the solution. Proposed methods include *state abstraction* [Sacerdoti, 1974; Knoblock, 1994], *hierarchical task networks* [Sacerdoti, 1975; Tate, 1977]

[3]The state-transition graph is a directed graph $G = \langle V, A \rangle$ with the states of the system represented as nodes in V and $(v, w) \in A$ iff there exists an operator that transforms state v to state w.

and *predicate relaxation* [Christensen, 1990]. Of these paper III discusses state abstraction.

In state abstraction, certain literals are ignored, either in the operator preconditions [Sacerdoti, 1974] or in the whole language [Knoblock, 1991, 1994]. First an abstracted version of the problem instance is solved, thus not taking all details into account, which results in a plan that is correct at this abstraction level. This plan is then used as a skeleton plan to be filled in with more detail at the next lower level—a process referred to as refinement. Repeated refinement results in a solution to the original, non-abstract problem. This technique was first employed in the AbStrips system [Sacerdoti, 1974]. It was shown that AbStrips was more efficient than Strips on a number of test examples, but no formal analysis was made of how and if the efficiency was improved in general. Since then, several methods and systems using state abstraction have been proposed and some of the more well-known are Abtweak [Yang and Tenenberg, 1990] and Prodigy [Knoblock, 1994].

Knoblock [1994] has tried to characterize under which restrictions an exponential reduction of the search space is possible. Using b for the branching factor and l for the length of the shortest solution, he found that the worst case size of the search space can be reduced from $O(b^l)$ to $O(l)$ under ideal conditions, thus leading to an exponential speed-up of the planning process. Unfortunately, these restrictions are not restrictions on the problem instance *per se*, but rather on the planning process. This makes them extremely costly to test and, probably, not very useful in practice.

To automatically generate abstraction hierarchies that are "good" in some sense, Knoblock [1994] has suggested an algorithm, Alpine, for generating abstraction hierarchies that are *ordered monotonic*—a property guaranteeing that no refinement of an abstract plan can undo any effects of the abstract plan. The main vehicle of this algorithm is a graph of dependencies between literals occurring in the pre- and post-conditions of the operators. Knoblock shows that a simple analysis of this graph is sufficient for the generation of ordered monotonic hierarchies. This idea has been elaborated upon by Bacchus and Yang [1994] in their HighPoint algorithm.

Paper III complements these mainly positive results by providing equally negative results for state abstraction and the automatic generation of abstraction hierarchies.

2.6 Paper IV: Reactive Planning

The term *reactive planning* has been used for a number of more or less related approaches. It seems that these approaches have at least one thing in common: There is usually very little or no planning ahead. Rather the idea is centered around the stimulus-response principle—prompt reaction to the input.

The proponents of the reactive-planning approach argue that classical planning has some disadvantages in comparison with reactive planning. Two common arguments (*c.f.* Schoppers [1987], Georgeff and Lansky [1987]) are the following.

- Classical planning cannot handle uncertain and dynamic worlds. This may be true to some extent, although effort in planning research is spent on extending classical planning to also handle such worlds.

- Classical planning is not tractable and, hence, cannot meet the demands of real-time behaviour. Naturally, reactive planning is no solution to this problem. First-order planning is undecidable and propositional planning is PSPACE-complete. This is an inherent difficulty in the problem itself and we cannot escape from it by trying alternative methods.

Several approaches to reactive planning have been reported in the literature, *e.g.* *universal plans* [Schoppers, 1987], *situated control rules* [Drummond, 1989], and the *subsumption-architecture* approach by Brooks [1991]. Paper IV concentrates on universal plans. A universal plan is a function from the set of states into the set of operators. Hence, a universal plan does not generate a sequence of operators leading from the current state to the goal state as a classical planner; it decides after each step what to do next based on the current state.

Universal plans have been much discussed in the literature. In a famous debate (Ginsberg [1989b], Schoppers [1989], Ginsberg [1989a], Schoppers [1994]), Ginsberg criticised the approach while Schoppers defended it. (It should be noted that other authors too, such as Chapman [1989], have joined the discussion.) Based on a counting argument, Ginsberg claimed that almost all (interesting) universal plans take an infeasibly large amount of space. Schopper's defence has, to a large extent, built on the observation that planning problems are structured. According to Schoppers, this structure can be exploited in order to create small, effective universal plans.

From a complexity-theoretic standpoint, these papers do not settle the question. A more formal view on reactive planning is adopted by Selman [1994]. He shows that the existence of small (polynomially-sized) universal plans with the ability to generate *minimal* plans implies a collapse of the polynomial hierarchy. Since a collapse of the polynomial hierarchy is widely conjectured to be false, the existence of such universal plans seems highly unlikely. Paper IV extends Selman's work by continuing the study of space requirements in universal planning.

2.7 Paper V: Incremental Plan Generation

As was pointed out in the previous subsection, classical planning is sometimes considered inadequate for coping with dynamic worlds. The complexity of planning is a problem in rapidly changing and time-critical applications since we have to wait for the planner to generate the whole plan before we can start executing it. Furthermore, if the execution of an action fails, or other changes in the world force us into some unexpected world state, we have to reinvoke the planner to find a new plan from the current world state to the goal. This is known as *replanning* and is often considered an infeasible method since it can be as costly as planning. Reactive planning is sometimes considered a better alternative, but it must be noted that, from a computational perspective, a reactive planner can not perform any better than a classical planning/replanning system as was pointed out in the previous subsection.

Ambros-Ingerson and Steel [1988] and Drummond *et al.* [1993] have addressed the problems above by suggesting interleaving planning with execution. Their idea is to use a planner which will as soon as possible output a *prefix* of the final solution—that is, a set of actions which are the first actions of the final solution. We can, thus, immediately start executing this plan prefix and the planner will concurrently continue to generate the rest of the solution and output successive prefixes for execution whenever possible. We will refer to this method as *incremental planning,* a name that was introduced by Drummond *et al.* [1993]. The difference between reactive, incremental and classical planning lies in how eager the planner is to output operators. A classical planner computes the complete plan and then executes it. An incremental planner generates chunks of the plan and executes them in an interleaved fashion. A reactive planner generates one operator at a time and immediately executes it.

The incremental approach has two advantages when compared to the classical approach. First, even if it takes a long time to generate the whole solution, we can hope to start executing a prefix within a reasonably short time. In most applications, we can expect action execution to take place on a relatively much slower time scale than planning, so outputting prefixes now and then will keep the plan executor busy and not much time will be lost in planning. Second, if we have to replan, we only have to wait for the first prefix of the new plan before we can, once again, start execution. That is, for each failure, we only lose the time it takes to generate a prefix of the new plan. However, there is one obvious disadvantage with incremental planning. In general, we cannot know if there is a solution to a given problem before the planner has terminated. If the planner outputs a prefix, we cannot know whether it is a prefix of a solution or not. Executing the prefix of a non-solution is, for several reasons, not advisable. Hence, it is important to be able to tell in advance whether a solution exists or not. The main contribution of paper V is to provide a correct incremental planner for a restricted class of problems. Furthermore, this planner can decide whether a solution exists or not in polynomial time.

References

[AAAI-90, 1990] American Association for Artificial Intelligence. *Proceedings of the 8th (US) National Conference on Artificial Intelligence (AAAI-90)*, Boston, MA, USA, August 1990. MIT Press.

[AAAI-91, 1991] American Association for Artificial Intelligence. *Proceedings of the 9th (US) National Conference on Artificial Intelligence (AAAI-91)*, Anaheim, CA, USA, July 1991. AAAI Press/MIT Press.

[AAAI-92, 1992] American Association for Artificial Intelligence. *Proceedings of the 10th (US) National Conference on Artificial Intelligence (AAAI-92)*, San José, CA, USA, July 1992.

[AAAI-94, 1994] American Association for Artificial Intelligence. *Proceedings of the 12th (US) National Conference on Artificial Intelligence (AAAI-94)*, Seattle, WA, USA, July–August 1994.

[Allen *et al.*, 1990] James Allen, James Hendler, and Austin Tate, editors. *Readings in Planning.* Morgan Kaufmann, San Mateo, CA, 1990.

[Allen, 1990] James Allen. Formal models of planning. In Allen et al. [1990], pages 50–55.

[Ambros-Ingerson and Steel, 1988] José A. Ambros-Ingerson and Sam Steel. Integrating planning, execution and monitoring. In *Proceedings of the 7th (US) National Conference on Artificial Intelligence (AAAI-88)*, pages 83–88, St. Paul, MN, USA, August 1988. American Association for Artificial Intelligence, Morgan Kaufmann.

[Bacchus and Yang, 1994] Fahiem Bacchus and Qiang Yang. Downward refinement and the efficiency of hierarchical problem solving. *Artificial Intelligence*, 71:43–100, 1994.

[Bäckström and Klein, 1990] Christer Bäckström and Inger Klein. Planning in polynomial time. In G Gottlob and W Nejdl, editors, *Expert Systems in Engineering: Principles and Applications. International Workshop*, volume 462 of *Lecture Notes in Artificial Intelligence*, pages 103–118, Vienna, Austria, September 1990. Springer.

[Bäckström and Klein, 1991a] Christer Bäckström and Inger Klein. Parallel non-binary planning in polynomial time. In Ray Reiter and John Mylopoulos, editors, *Proceedings of the 12th International Joint Conference* *on Artificial Intelligence (IJCAI-91)*, pages 268–273, Sydney, Australia, August 1991. Morgan Kaufmann.

[Bäckström and Klein, 1991b] Christer Bäckström and Inger Klein. Planning in polynomial time: The SAS-PUBS class. *Computational Intelligence*, 7(3):181–197, August 1991.

[Bäckström and Nebel, 1993] Christer Bäckström and Bernhard Nebel. Complexity results for SAS$^+$ planning. In Bajcsy [1993], pages 1430–1435.

[Bäckström and Nebel, 1995] Christer Bäckström and Bernhard Nebel. Complexity results for SAS$^+$ planning. *Computational Intelligence*, 11(4):625–655, 1995.

[Bäckström, 1992] Christer Bäckström. Equivalence and tractability results for SAS$^+$ planning. In Bill Swartout and Bernhard Nebel, editors, *Proceedings of the 3rd International Conference on Principles on Knowledge Representation and Reasoning (KR-92)*, pages 126–137, Cambridge, MA, USA, October 1992. Morgan Kaufmann.

[Bäckström, 1994] Christer Bäckström. Planning using transformation between equivalent formalisms: A case study of efficiency. In David Wilkins, editor, *Comparative Analysis of AI Planning Systems*, 1994. Held in conjunction with AAAI-94 [1994].

[Bäckström, 1995a] Christer Bäckström. Expressive equivalence of planning formalisms. *Artificial Intelligence*, 76(1–2):17–34, 1995.

[Bäckström, 1995b] Christer Bäckström. Five years of tractable planning. In Malik Ghallab and Alfredo Milani, editors, *New Directions in AI Planning: EWSP'95—3rd European Workshop on Planning, Frontiers in AI and Applications*, Assisi, Italy, September 1995. IOS Press. Invited paper.

[Bajcsy, 1993] Ruzena Bajcsy, editor. *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI-93)*, Chambéry, France, August–September 1993. Morgan Kaufmann.

[Biundo *et al.*, 1992] Susanne Biundo, Dietmar Dengler, and Jana Koehler. Deductive planning and plan reuse in a command language environment. Research Report RR-92-11, German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany, March 1992.

[Brooks, 1991] Rodney A Brooks. Intelligence without representation. *Artificial Intelligence*, 47(1–3):139–159, 1991.

[Bylander, 1992] Tom Bylander. Complexity results for serial decomposability. In AAAI-92 [1992], pages 729–734.

[Bylander, 1994] Tom Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69:165–204, 1994.

[Chapman, 1987] David Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333–377, 1987.

[Chapman, 1989] David Chapman. Penguins can make cake. *AI Magazine*, pages 45–50, Winter 1989.

[Charniak and McDermott, 1985] Eugene Charniak and Drew McDermott. *Introduction to Artificial Intelligence*. Addison Wesley, Reading, MA, 1985.

[Christensen, 1990] Jens Christensen. A hierarchical planner that generates its own hierarchies. In AAAI-90 [1990], pages 1004–1009.

[Drummond *et al.*, 1993] Mark Drummond, Keith Swanson, John Bresina, and Richard Levinson. Reaction-first search. In Bajcsy [1993], pages 1408–1413.

[Drummond, 1989] Mark Drummond. Situated control rules. In Ronald J Brachman, Hector J Levesque, and Raymond Reiter, editors, *Proceedings of the 1st International Conference on Principles on Knowledge Representation and Reasoning (KR-89)*, Toronto, ON, Canada, May 1989. Morgan Kaufmann.

[Erol *et al.*, 1992] Kutluhan Erol, Dana S Nau, and V S Subrahmanian. On the complexity of domain-independent planning. In AAAI-92 [1992], pages 381–386.

[Fikes and Nilsson, 1971] Richard E Fikes and Nils J Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[Garey and Johnson, 1979] Michael Garey and David Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.

[Georgeff and Lansky, 1987] M Georgeff and A Lansky. Reactive reasoning and planning. In *Proceedings of the 6th (US) National Conference on Artificial Intelligence (AAAI-87)*, pages 677–682, Seattle, WA, USA, July 1987. American Association for Artificial Intelligence.

[Ginsberg, 1989a] Matthew L Ginsberg. Ginsberg replies to Chapman and Schoppers. *AI Magazine*, pages 61–62, Winter 1989.

[Ginsberg, 1989b] Matthew L Ginsberg. Universal planning: An (almost) universally bad idea. *AI Magazine*, pages 40–44, Winter 1989.

[Green, 1969] Cordell Green. Application of theorem proving to planning. In Donald E Walker and Lewis M Norton, editors, *Proceedings of the 1st International Joint Conference on Artificial Intelligence (IJCAI-69)*, pages 219–239, Washington, DC, USA, May 1969. William Kaufmann. Reprinted in Allen *et al* [1990], pages 67–87.

[Knoblock, 1991] Craig A Knoblock. Search reduction in hierarchical problem solving. In AAAI-91 [1991], pages 686–691.

[Knoblock, 1994] Craig A. Knoblock. Automatically generating abstractions for planning. *Artificial Intelligence*, 68:243–302, 1994.

[Korf, 1987] Richard E Korf. Planning as search: A quantitative approach. *Artificial Intelligence*, 33:65–88, 1987.

[Levesque, 1988] Hector J Levesque. Logic and the complexity of reasoning. *Journal of Philosophical Logic*, 17:355–389, 1988.

[McAllester and Rosenblitt, 1991] David McAllester and David Rosenblitt. Systematic nonlinear planning. In AAAI-91 [1991], pages 634–639.

[McCarthy and Hayes, 1969] John McCarthy and Patrick J Hayes. Some philosophical problems from the standpoint of AI. *Machine Intelligence*, 4, 1969. Reprinted in [Webber and Nilsson, 1981], pages 431–450.

[Papadimitriou, 1994] Christos H. Papadimitriou. *Computational Complexity*. Addison Wesley, Reading, MA, 1994.

[Sacerdoti, 1974] Earl D Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5(2):115–135, 1974.

[Sacerdoti, 1975] Earl D Sacerdoti. The non-linear nature of plans. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence (IJCAI-75)*, Tbilisi, USSR, September 1975. IJCAI, William Kaufmann.

[Sandewall and Rönnquist, 1986] Erik Sandewall and Ralph Rönnquist. A representation of action structures. In *Proceedings of the 5th (US) National Conference on Artificial Intelligence (AAAI-86)*, pages 89–97, Philadelphia, PA, USA, August 1986. American Association for Artificial Intelligence, Morgan Kaufmann.

[Sandewall, 1994] Erik Sandewall. *Features and Fluents*. Oxford University Press, 1994.

[Schoppers, 1987] M J Schoppers. Universal plans for reactive robots in unpredictable environments. In John McDermott, editor, *Proceedings of the 10th International Joint Conference on Artificial Intelligence (IJCAI-87)*, pages 1039–1046, Milano, Italy, August 1987. Morgan Kaufmann.

[Schoppers, 1989] Marcel J Schoppers. In defense of reaction plans as caches. *AI Magazine*, pages 51–62, Winter 1989.

[Schoppers, 1994] Marcel Schoppers. Estimating reaction plan size. In AAAI-94 [1994], pages 1238–1244.

[Schwartz and Sharir, 1990] Jacob T. Schwartz and Micha Sharir. Algorithmic motion planning in robotics. In van Leeuwen [1990], chapter 8, pages 391–430.

[Selman, 1994] Bart Selman. Near-optimal plans, tractability, and reactivity. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proceedings of the 4th International Conference on Principles on Knowledge Representation and Reasoning (KR-94)*, pages 521–529, Bonn, Germany, May 1994. Morgan Kaufmann.

[Smith and Peot, 1993] David E. Smith and Mark A. Peot. Postponing threats in partial-order planning. In *Proceedings of the 11th (US) National Conference on Artificial Intelligence (AAAI-93)*, pages 500–506, Washington DC, USA, July 1993. American Association for Artificial Intelligence.

[Tate *et al.*, 1990] Austin Tate, James Hendler, and Mark Drummond. A review of AI planning techniques. In Allen et al. [1990].

[Tate, 1977] Austin Tate. Generating project networks. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence (IJCAI-77)*, pages 888–893, Cambridge, MA, USA, August 1977. Reprinted in Allen *et al* [1990], pages 291–296.

[van Leeuwen, 1990] Jan van Leeuwen, editor. *Handbook of Theoretical Computer Science: Algorithms and Complexity*, volume A. Elsevier, Amsterdam, 1990.

[Webber and Nilsson, 1981] Bonnie Lynn Webber and Nils J Nilsson, editors. *Readings in Artificial Intelligence*. Morgan Kaufmann, 1981.

[Yang and Tenenberg, 1990] Qiang Yang and Josh D Tenenberg. ABTWEAK: Abstracting a nonlinear, least commitment planner. In AAAI-90 [1990], pages 204–209.