A Survey of Complexity Results for Planning*

*** DRAFT. COMMENTS VERY WELCOME ***

Marco Cadoli
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
via Salaria 113, I-00198 Roma, Italia
Tel. +39-6-8558418. Fax +39-6-85300849
E-mail: cadoli@assi.ing.uniroma1.it

October 21, 1993

Abstract

Recently several works about the computational complexity of planning appeared in the literature. In this paper we survey the main results in this area. We not only give results about the tractability/intractability of the individual problems but we also analyze sources of complexity and explain intuitively the nature of easy/hard cases.

1 Introduction

Planning is the reasoning task of finding a series of actions that achieve a goal from a given initial state. One of the most important features of a

*Work supported by the *Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo* and the *Progetto Speciale Pianificazione Automatica* of the CNR (Italian Research Council) and by the ESPRIT Basic Research Action 6810-COMPULOG II.

planning system is its *efficiency*, i.e. how many resources it needs in order to find a satisfactory plan. Many techniques, such as *hierarchical abstraction* [Sacerdoti, 1974], have been developed in order to make planning more efficient. In the same way, several properties of planning problems, such as *independent subgoals* [Korf, 1987], have been defined in order to characterize easy instances of planning.

Recently several works about the intrinsic computational complexity of planning appeared in the literature. Most of them focus on the analysis of the so-called "tractability threshold" [Brachman and Levesque, 1985; Levesque, 1988] between polynomial and intractable problems. The goal of such analyses is to find a meaningful balance between the expressiveness of the language used for representing a planning problem (i.e. initial and final states, available actions) and the complexity of the task of finding a solution.

Complexity analyses have been performed both for propositional and for first-order planning systems. Specific planning domains (e.g. blocks-world) as well as general-purpose planners have been considered. Several computational tasks have been taken into account: proving whether a plan exists, finding a plan if existing, finding an optimal plan (i.e. a plan with a minimum number of actions), finding a plan provided we already have a plan for a similar planning problem. Undecidable, decidable but polynomially intractable and polynomial cases have been shown. Almost all of the works perform a worst-case complexity analysis, although interesting considerations on average-case complexity recently appeared.

The goal of this paper is to survey the main results appearing in the literature on the computational complexity of reasoning about plans. We not only give results about the tractability/intractability of the individual problems but we also analyze sources of complexity and explain intuitively the nature of easy/hard cases. We use the jargon of computational complexity, as found in [Garey and Johnson, 1979; Johnson, 1990]. An appendix at the end of this paper summarizes the main concepts of computational complexity theory that are used in the survey.

In this paper we are interested in the intrinsic complexity of planning. Therefore works on heuristics are only marginally addressed. This paper is not intended as an overview of planning systems and their relative merits. A collection of papers on this issue has been edited by Allen, Hendler and Tate [1990].

The paper is organized as follows. In the next two sections we survey the complexity of several tasks in general-purpose planning systems (i.e. complexity analyses for the so-called "domain-independent planning"). In particular we address propositional planners in Section 2 and first-order planners in Section 3. In Section 4 we address the complexity of blocks-world planning, while in Section 5 we comment average-case analyses. In Section 6 we briefly survey complexity results for problems related to planning; some interesting relations with non monotonic reasoning are highlighted. In Section 7 we draw some conclusions and show a brief list of interesting computational analyses about planning that have not/have marginally been addressed in the literature.

2 Complexity of propositional planning

In this section we focus on the complexity of domain-independent propositional planning. We will consider several different propositional formalisms (STRIPS, SAS, SAS$^+$) and several computational problems, like existence of a (bounded size) plan, finding a (bounded size) plan, modifying a plan, ...

2.1 Propositional STRIPS planning

Terminology in planning is by no means uniform, and the sub-area of complexity analysis makes no exception. Throughout this section we adopt the notion of *propositional STRIPS planning*, as defined in [Bylander, 1991]. Other propositional planners are described in Subsection 2.3.

An instance of propositional STRIPS planning is specified by a tuple $\langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$, where:

- $\mathcal{P}$ is a finite set of propositional letters (the *conditions*);
- $\mathcal{O}$ is a finite set of *operators*, where each operator is a tuple $\langle \varphi, \eta, \alpha, \delta \rangle$:
  - $\varphi \subseteq \mathcal{P}$ is a set of *positive preconditions*;
  - $\eta \subseteq \mathcal{P}$ is a set of *negative preconditions*;
  - $\alpha \subseteq \mathcal{P}$ is a set of *positive postconditions*;
  - $\delta \subseteq \mathcal{P}$ is a set of *negative postconditions*; and
  - $\varphi \cap \eta = \emptyset$ and $\alpha \cap \delta = \emptyset$.

$\mathcal{I} \subseteq \mathcal{P}$ is the *initial state*; and

$\mathcal{G} = \langle \mathcal{M}, \mathcal{N} \rangle$ is the *goal*:

$\mathcal{M} \subseteq \mathcal{P}$ is a set of *positive goals*;

$\mathcal{N} \subseteq \mathcal{P}$ is a set of *negative goals*;

$\mathcal{M} \cap \mathcal{N} = \emptyset$.

$\mathcal{P}$ is the set of conditions that are relevant. A *state* is a set $S \subseteq \mathcal{P}$; a condition $p \in \mathcal{P}$ is true in a state S if $p \in S$, it is false otherwise. An operator $o \in \mathcal{O}$ can change one state into another. The initial state $\mathcal{I}$ specifies the conditions that are true and false in the initial state: a generic $p \in \mathcal{P}$ is initially true if $p \in \mathcal{I}$, and it is false otherwise (notice that the initial state is *total*). On the other hand the goal is a *partial* state: a generic state $S \subseteq \mathcal{P}$ satisfies a goal $\langle \mathcal{M}, \mathcal{N} \rangle$ iff $\mathcal{M} \subseteq S$ and $S \cap \mathcal{N} = \emptyset$.

An operator $o = \langle \varphi, \eta, \alpha, \delta \rangle \in \mathcal{O}$ has an effect on a state S iff its preconditions (both positive and negative) are satisfied in S, i.e. iff $\varphi \in S$ and $\eta \cap S = \emptyset$. The result of such an application amounts to add all of its positive postconditions and delete all of its negative postconditions, i.e. to obtain the state $S \cup \alpha \setminus \delta$ (cf. [Fikes and Nilsson, 1971]).

As an example, we consider the well-known *blocks world* scenario. A possible formulation of blocks world in propositional STRIPS is to have conditions such as $onAB, clearA, clearB, \ldots$ and operators such as $unstackAB$ (that moves block A from top of block B to the table), which is characterized by:

- positive preconditions: $\{onAB, clearA\}$;
- negative preconditions: $\{\}$;
- positive postconditions: $\{clearB\}$;
- negative postconditions: $\{onAB\}$,

A finite sequence of operators $\langle o_1, o_2, \ldots, o_n \rangle$ is a *solution* to an instance of a propositional STRIPS planning iff the result of the application of all the operators in the sequence is a goal state (cf. [Bylander, 1991] for more formal details and examples). The number n of operators occurring in a solution to an instance is its *length*.

An instance of a propositional STRIPS planning problem is *satisfiable* iff it has a solution.

2.2 Computational tasks for planning

The following computational tasks have been studied for propositional STRIPS planning as well as for other planning systems:[1]

PLANSAT: given an instance of the planning problem, decide whether there exists a solution;

k-PLANSAT: given an instance of the planning problem and an integer k, decide whether there exists a solution of length k or less;

PLANFIND: given an instance of the planning problem, find a solution or answer that there is no solution;

k-PLANFIND: given an instance of the planning problem and an integer k, find a solution of length k or less or answer that there is no such solution.

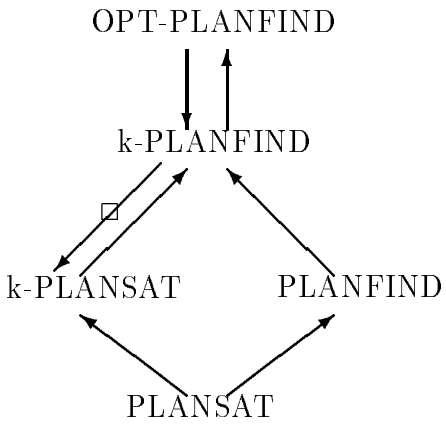
Obviously a search problem can never be easier than its corresponding existence problem (e.g. PLANFIND cannot be easier than PLANSAT). Moreover, the bounded version of a problem, can never be easier than its corresponding unbounded version. As an example, k-PLANSAT can never be easier than PLANSAT, since the longest solution in propositional STRIPS has length less than or equal to $2^{|\mathcal{P}|}$. Therefore if $k = 2^{|\mathcal{P}|}$ then k-PLANSAT and PLANSAT are the same problem.

We notice that k-PLANSAT is polynomial iff it is possible to compute in polynomial time the minimum length of a solution. We have to be careful with the representation of the integer k, though. If k is represented in unary notation, then the size of its representation is linear in its value. On the other hand, if k is represented in binary, then the size of its representation is only logarithmic in its value. In other words unary representations are not *parsimonious*. An algorithm which is polynomial in the *value* of k is called a *pseudo-polynomial* algorithm [Garey and Johnson, 1979]. Notice that a pseudo-polynomial algorithm could be exponential in the size of the input (if the representation of k is parsimonious). In the following, if not specified, we will assume binary representation for all numbers.

[1] The following computational tasks are sometimes named differently in the literature.

In the same way, k-PLANFIND is polynomial iff it possible to find in polynomial time a minimum-length (optimal) plan. Finding optimal plans (the OPT-PLANFIND) problem is the major computational problem in planning. If we are able to solve k-PLANSAT in polynomial time, then it is possible to solve k-PLANFIND in pseudo-polynomial time by using *prefix search* [Garey and Johnson, 1979].

Reducibility relations among all mentioned problems are reported in Figure 1.



A ⟶ B = A is polynomially reducible to B

A ⟶◇ B = A is peudo-polynomially reducible to B

Figure 1: Reducibility among planning problems.

It is important to notice that STRIPS and other planning systems admit exponentially-sized (with respect to the number of available operators) optimal solutions. This is why pseudo-polynomial algorithms are not polynomial in practice. Nevertheless it is possible that –even if the minimum length of solutions is exponential– the PLANSAT or the k-PLANSAT problem can be solved in polynomial time. As a matter of fact, this happens for the Towers of Hanoi and similar planning problems (cf. [Aho *et al.*, 1974]). We will return on the issue of exponentially-sized optimal plans later on.

Another computational task that has been studied is (plan modification):

- given an instance of the planning problem $\Pi' = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}', \mathcal{G}' \rangle$ and a solution Δ to an instance $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$ find a solution Δ' to Π' by minimally modifying Δ.

Complexity results for all of the above problems will be surveied in this section.

2.3 Other formalisms for planning

STRIPS is not the only propositional planning system that has been studied from the point of view of computational complexity. In particular, computational properties of SAS and SAS$^+$ formalisms have been thoroughly analyzed [Bäckström and Klein, 1991; Bäckström, 1992b; Bäckström, 1992a; Bäckström and Nebel, 1993].

The main difference between propositional STRIPS and SAS is in that the latter admits multivalued conditions, as opposed to binary conditions of STRIPS. On top of that, SAS$^+$ admits partial initial states (on the other hand in STRIPS and SAS one has to give a complete specification of the initial state). We refer to [Bäckström, 1992b; Bäckström, 1992a] for a thorough discussion of the differences between STRIPS, SAS and SAS$^+$.

As proven in [Bäckström, 1992b; Bäckström, 1992a], SAS, SAS$^+$ and STRIPS are equivalent, in the sense that there is a polynomial reduction that maps the problem of finding a (bounded-length) solution to a generic instance of STRIPS into the problem of finding a (bounded-length) solution to an instance of SAS$^+$; analogous polynomial reductions are shown for the other pairs of formalisms and for the (bounded-length) existence problem. Therefore throughout this section we refer to computational tasks for SAS and SAS$^+$ with the terminology introduced in Subsection 2.2.

We recall that STRIPS admits exponentially-sized optimal solutions. This holds for SAS and SAS$^+$ as well. As a consequence, a "reasonable" polynomial reduction from the problem of finding a solution in formalism A into the analogous problem for formalism B should map polynomially-sized solutions into polynomially-sized solutions. This topic is thoroughly analyzed in [Bäckström, 1992a], where such a reduction is shown.

2.4 Intractability results

Bylander [1991] shows that the PLANSAT problem in STRIPS is PSPACE-complete. We want to spend few words about this result.

A PLANSAT problem can be characterized as a graph-reachability problem. More precisely, given a PLANSAT problem $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$, a graph Γ is implicitly identified such that:

- Γ has $2^{|\mathcal{P}|}$ nodes, one for each state $S \subseteq \mathcal{P}$ of Π;
- for each pair of nodes N_1, N_2, there is an edge from N_1 to N_2 iff there is an operator $o \in \mathcal{O}$ that transforms the state corresponding to N_1 into the state corresponding to N_2;
- there is a single initial node I, corresponding to $\mathcal{I}$;
- there is a set of goal nodes G, corresponding to $\mathcal{G}$.

The graph Γ has the following properties:

- it has exponentially many nodes;
- each node has a polynomial number of adjacent nodes;
- it is possible to find in polynomial time all the nodes adjacent to a given node.

The PLANSAT problem amounts to test the existence of a path from node I to a node in G. PLANSAT has a structure similar to other PSPACE-complete problems, such as the problem of testing satisfiability of a formula in the modal system $\mathcal{K}$ [Ladner, 1977]. Comparison to other PSPACE-complete problems with a similar structure will be given in Section 6.1.

Since the STRIPS, the SAS and the SAS$^+$ formalisms are equivalent, PSPACE-completeness of PLANSAT holds for SAS and SAS$^+$ as well [Bäckström, 1992b; Bäckström, 1992a]. The k-PLANSAT problem is also PSPACE-complete for STRIPS [Erol *et al.*, 1992b] and for SAS and SAS$^+$ [Bäckström, 1992b; Bäckström, 1992a].

Bylander [1991] and Bäckström [1992b; 1992a] show some restrictions of the general PLANSAT problem such that the PSPACE-completeness result still holds. As an example, in [Bylander, 1991] it is shown that the complexity

is the same if either each operator has at most one postcondition or all preconditions are positive.

There are restrictions that make PLANSAT easier. As an example, Bylander [1991] notices that if all operators have positive postconditions, then the existence problem (called PLANSAT$^+$) is NP-complete. The reason why PLANSAT$^+$ is in NP is easily seen by looking at the path-existence problem in the corresponding graph Γ: Since operators have no negative postconditions, the state grows monotonically when operators are applied. This implies that only a polynomial number of operators can be applied to the initial state $\mathcal{I}$, i.e. the longest path in Γ has polynomial length.

Similarly, in [Bäckström and Nebel, 1993] a class (namely SAS$^+$-US) is identified having a similar property: All optimal solutions in SAS$^+$-US have polynomially bounded length, hence the PLANSAT problem is in NP. Other restrictions leading to NP-completeness for the PLANSAT and the k-PLANSAT problems can be found in [Bylander, 1991; Bäckström and Nebel, 1993; Erol *et al.*, 1992b].

In the following subsection we address restrictions of the PLANSAT and PLANFIND problems that lead to polynomial algorithms.

We close this subsection by addressing another source of intractability. As we already noticed, some STRIPS, SAS and SAS$^+$ instances admit exponentially-sized optimal plans. In particular in [Bäckström, 1992a; Bäckström and Nebel, 1993] several restrictions of SAS are shown having this property. The PLANFIND problem in these classes is therefore *provably* intractable[2]. Nevertheless the PLANSAT problem could still be polynomial. As suggested by several authors [Garey and Johnson, 1979; Charniak and McDermott, 1985; Bäckström, 1992a] we should hardly regard these solutions as realistic, since we are asking for more information that we could ever hope to use.

An interesting computational task, that does not seem to have been addressed so far, is how to "minimally change" the set of operators, so that optimal solutions have polynomial length.

[2]This is a result stronger than PSPACE-completeness, whose meaning is *probably* intractable

2.5 Tractability results

As we noticed in the previous subsection, there are two sources of complexity in propositional planning problems: firstly, we have an exponentially-sized search space; secondly, we may need an exponential number of actions in order to go from the initial state into the goal state. The interaction of the two sources leads to PSPACE-completeness of PLANSAT. Restrictions that ensure membership of the problem in NP typically cut off the second source of complexity. In order to ensure polynomiality, it is necessary to work on the first source as well. We now briefly comment a restriction of the expressiveness of STRIPS that leads to polynomiality.

In [Bylander, 1991, Theorem 7] it is proven that PLANSAT is polynomial if operators only have positive preconditions and one postcondition. The reason why this problem is easy is that if a solution to the planning problem exists, then it can be found by applying operators with a positive postcondition first, then operators with a negative postcondition. Moreover the right sequence of operators can be found by means of a simple local search.

In order to guarantee polynomial algorithms for planning problems, several restrictions have been proposed in the literature. Most of the restrictions are made on the operators. A restriction on the goal state has been proposed in [Bylander, 1991].

According to Bäckström [1992a], there are two possible kinds of restriction on operators:

local: i.e. restrictions on single operators. As an example, we may allow an operator to have only one postcondition (this restriction is called *unariness* by Bäckström);

global: i.e. restrictions affecting the whole set of operators. As an example, we may not allow two distinct operators to change the same condition (this restriction is called *post-uniqueness* by Bäckström).

Bylander [1991] and Erol *et al.* [1992b] consider only local restrictions. On the other hand in [Bäckström and Klein, 1991; Bäckström, 1992b; Bäckström, 1992a; Bäckström and Nebel, 1993] global conditions are taken into account.

It is clearly possible to check in polynomial time if an instance of a planning problem satisfies a local restriction. Bäckström and colleagues propose

global restrictions for SAS and SAS$^+$ planning problems that are testable in polynomial time. In their work they consider four different restrictions: two local and two global. All of the $2^4 = 16$ possible combinations of the restrictions have been studied from the computational point of view, for the four problems PLANSAT, k-PLANSAT, PLANFIND, k-PLANFIND. Polynomial cases have been found for all of the four problems. It is worth noticing that there are problems (namely k-PLANSAT for the so-called UBS restriction) that are polynomial for the SAS formalism, and NP-complete for the SAS$^+$ formalism (cf. [Bäckström and Nebel, 1993]). This shows that, although the formalisms are equivalent in the general case, they are not equivalent when restrictions are taken into account.

Nebel and Bäckstrom [1993] show a restriction (namely SAS$^+$-US) such that k-PLANFIND is NP-equivalent, while PLANFIND is polynomial. In particular they show how to find a polynomial-length plan in polynomial time, although the ratio of its length wrt the minimum is not known. In other words it is not known how good is the plan which is obtained. Similar results appear in [Erol *et al.*, 1992b; Erol *et al.*, 1992a], where restrictions such that k-PLANSAT is NP-complete, while PLANSAT is polynomial are shown.

Other polynomial cases for PLANSAT are shown in [Bylander, 1991; Erol *et al.*, 1992b]. In particular, Erol *et al.* show a general technique (called *composition*) for widening classes of tractable operators. The composition of two operators a and b is a sort of pipelining, so that a and b are executed in sequence. Adding a set of operators that are the composition of a set of tractable operators does not spoil polynomiality.

2.6 Plan modification

In [Nebel and Koehler, 1993] the following issue is addressed: Suppose that we already solved a PLANFIND problem and we have a solution Δ in hand; now we have a slightly different PLANFIND problem, for example the initial state is changed and/or the goal is changed (but the operators are still the same). Is it better –from a purely computational point of view– to modify Δ, trying to use as much of the old solution as possible, or to solve the new PLANFIND problem from scratch?

Such question is motivated by experimental results [Kambhampati and Hendler, 1992], that seem to suggest that the first approach (named *plan*

modification) is more efficient. Nebel and Koehler [1993] propose the following definitions for the plan modification problem:

MODFIND: given an instance of the planning problem $\Pi' = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}', \mathcal{G}' \rangle$ and a solution Δ to an instance $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$ find a solution Δ' to Π' that *minimally modifies* Δ;

MODSAT: given an instance of the planning problem $\Pi' = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}', \mathcal{G}' \rangle$ and a solution Δ to an instance $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$ decide whether there exists a solution Δ' to Π' that *minimally modifies* Δ.

Obviously it is necessary to specify what "minimally modifies" means. A solution Δ is a sequence of operator symbols; it is reasonable to define minimal modification so that two sequences Δ and Δ' sharing long subsequences are similar. The authors specify three methods for obtaining Δ by modifying Δ'. We briefly describe the first of such methods: It allows to obtain Δ' by means of deletions in Δ and additions before and after Δ. As an example, if

$$\Delta =< a, b, c, d, e, f, g, h, i, j, k >,$$

then the following sequence contains a subsequence of Δ (in bold face) of length 5:

$$\Delta' =< i, c, z, \mathbf{c}, \mathbf{d}, \mathbf{e}, \mathbf{h}, \mathbf{i}, s, a > .$$

A solution Δ' minimally modifies a solution Δ if Δ' contains a subsequence of Δ of length at least k, where k is a number given in the input.

With this notion of plan modification the MODSAT problem can never be easier than the PLANSAT problem: PLANSAT reduces to MODSAT by choosing an old planning problem such that the initial state already satisfies the goal and $k = 0$; in this case the answer to MODSAT is yes iff the new planning problem admits a solution of arbitrary length. Analogously, MODFIND can never be easier than the PLANFIND problem.

The major results shown in [Nebel and Koehler, 1993] are:

- MODSAT is PSPACE-complete. In other words MODSAT has exactly the same complexity as PLANSAT and k-PLANSAT in the general case;
- there are cases in which PLANSAT is polynomial and MODSAT is NP-complete. This holds even if the old situation and the new situation

(in MODSAT) have identical initial states and goal states that differ only on one condition. In other words MODSAT can be harder than PLANSAT for restricted cases;

- whatever restriction is considered, MODFIND is still at least as hard as PLANSAT. This holds even if the old situation and the new situation (in MODFIND) have identical initial states and goal states that differ only on one condition.

2.7 Extensions to STRIPS

An extension of the propositional STRIPS planning has been proposed by Bylander [1992]. In particular the input of a problem also includes a propositional formula Σ (the *domain theory*). All states must be consistent with the domain theory, which represents therefore some sort of integrity constraint of a planning problem. As an example, in formalizing the blocks-world it is possible to express the constraint that a generic block has to be either on top of another block or on the table. It may happen that the application of an operator results in a state that violates the domain theory. In such a case, a *default preference ordering*, i.e. a total ordering among the conditions, is used to re-establish consistency (cf. [Bylander, 1992] for formal definitions and examples).

Bylander carries out a complexity analysis of the EPLANSAT problem (Extended PLANning SATisfiability) for some syntactic restrictions on the domain theory. In particular he focuses only on domain theories such that the propositional satisfiability problem is polynomial[3]. With this restriction it is still possible to obtain in polynomial time all the states "adjacent" to a given state (i.e. states reachable by applying just one operator). As a consequence the EPLANSAT problem is still in PSPACE (cf. Subsection 2.4). The results of the complexity analysis can be summarized as follows:

- all the polynomial cases for PLANSAT shown in [Bylander, 1991] become PSPACE-complete for EPLANSAT if Definite Horn domain theories are allowed. In other words it is possible to *simulate* "intractable"

[3]Two classes of propositional formulae in Conjunctive Normal Form are considered: Definite Horn (each clause has exactly one positive literal) and Krom (each clause has at most two literals). The satisfiability problem is well-known to be solvable in linear time for formulae of this kind (cf. [Dowling and Gallier, 1984; Even *et al.*, 1976].)

STRIPS operators by means of "tractable" operators and simple domain theories;

- some of the polynomial cases for PLANSAT become PSPACE-complete for EPLANSAT if Krom domain theories are allowed. In particular Krom theories permit to simulate multiple postconditions, hence EPLANSAT is PSPACE-complete if there is just one postcondition and the preconditions are positive (the corresponding case is polynomial for PLANSAT [Bylander, 1991, Theorem 7]);

- some of the polynomial cases for PLANSAT stay polynomial for EPLANSAT if Krom domain theories are allowed. In particular it is possible to have Krom domain theories "for free" whenever the PLANSAT problem admits an arbitrary number of postconditions.

3 Complexity of first-order planning

3.1 First-order STRIPS planning

Erol *et al.* [1992b; 1992c] define first-order STRIPS planning. In order to be consistent with the previous section, we recall their definition using a slightly different terminology.

Instead of a finite set of propositional letters, there is a first-order language $\mathcal{L}$ generated by finitely many constant, function and predicate symbols. If function symbols are not allowed, then we have *datalog* STRIPS planning[4]. A state is a set of ground atoms of $\mathcal{L}$, i.e. a subset of the Herbrand Base. In datalog STRIPS the state is always finite.

An operator is denoted by a *name* and a list $(X_1, \ldots, X_n)$ of variable symbols. The name is not a symbol of the language $\mathcal{L}$. Preconditions and postconditions (both positive and negative) are finite sets of atoms whose variables are all from the set $\{X_1, \ldots, X_n\}$. As an example the operator $unstack(X_1, X_2)$ (that moves block X_1 from top of block X_2 to the table) is characterized by:

- positive preconditions: $\{on(X_1, X_2), clear(X_1)\}$;

[4]If all predicates have arity 0 then we have propositional planning as in Section 2.1.

- negative preconditions: $\{\}$;
- positive postconditions: $\{clear(X_2)\}$;
- negative postconditions: $\{on(X_1, X_2)\}$,

where $clear()$ and $on(,)$ are predicate symbols.

Given an operator $o(X_1, \ldots, X_n)$ and a substitution θ that maps all of the X_i into ground terms, o can be *executed* in a state S iff its preconditions (both positive and negative) are satisfied in S, provided the substitution θ has been performed. The result of such an application amounts to add to S all of its positive postconditions and delete all of its negative postconditions, provided the substitution θ has been performed. If the resulting state is S', then the execution of the operator is denoted by $S \stackrel{o,\theta}{\Longrightarrow} S'$.

The initial state is a state (conjunction of *ground* atoms), while the goal is an existentially closed conjunction of atoms. A solution to a STRIPS problem is a sequence $\langle o_1, o_2, \ldots, o_n \rangle$ of operators, along with a sequence $\langle \theta_1, \theta_2, \ldots, \theta_n \rangle$ of substitutions and a sequence $\langle S_0, S_1, \ldots, S_{n-1}, S_n \rangle$ of states such that:

$$S_0 \stackrel{o_1,\theta_1}{\Longrightarrow} S_1 \stackrel{o_2,\theta_2}{\Longrightarrow} S_2 \cdots S_{n-1} \stackrel{o_n,\theta_n}{\Longrightarrow} S_n$$

and the goal G is satisfied by S_n, i.e. there exists a ground instance of G that is true in S_n.

In [Erol *et al.*, 1992a; Erol *et al.*, 1992b; Erol *et al.*, 1992c] the PLANSAT problem is studied for the full first-order case as well as for the datalog case. The k-PLANSAT problem is studied for the datalog case.

It is important to notice that the input of a first-order (datalog) planning instance is a first-order language $\mathcal{L}$, a set of operator *schemata*, the initial state and the goal. A huge number of propositional operators can be compactly represented by means of operator schemata: an exponential number in the datalog case and infinitely many in the first-order case. It is therefore reasonable to expect the complexity of planning to be higher than in the propositional case.

3.2 Complexity of datalog planning

As noticed by Bylander [1991, p. 276], a datalog planning instance can be polynomially reduced to a propositional planning instance if each operator schema is limited to a constant number of variables[5]. A polynomial number of variables would lead to an exponential number of propositional planning operators. Nevertheless the PLANSAT problem remains decidable.

Erol *et al.* [1992a; 1992b] prove that datalog PLANSAT is EXPSPACE-complete and that k-PLANSAT is NEXPTIME-complete. We recall that PSPACE $\subseteq$ NEXPTIME $\subseteq$ EXPSPACE. In other words:

- both PLANSAT and k-PLANSAT are harder in the datalog case than in the propositional case;
- k-PLANSAT is easier than PLANSAT in the datalog case (they have the same complexity in the propositional case).

Let us comment the last point. As we said in Section 2.2, in the propositional case k-PLANSAT can never be easier than PLANSAT, since the longest solution has size at most exponential in the input of the problem. On the other hand Erol *et al.* prove that in datalog STRIPS there are solutions having length doubly exponential in the size n of the input, i.e. length up to $2^{2^{p(n)}}$, where $p(n)$ is a polynomial. If we want to represent k with size polynomial with respect to the rest of the input of a k-PLANSAT problem, then k cannot be larger than $2^{p(n)}$. In other words k-PLANSAT confines us to a strict subset of the possible solutions, those of length at most exponential in the size of the input. Hence in the worst case k-PLANSAT is easier than PLANSAT.

In [Erol *et al.*, 1992a; Erol *et al.*, 1992b] a detailed analysis of datalog STRIPS subject to several restrictions is carried out. The following restrictions are considered:

- all predicate symbols are 0-ary (propositional STRIPS);
- the operators are fixed (not given in the input);

[5]This assumption is similar to the *data complexity* assumption in database theory cf. [Vardi, 1982].

- negative preconditions are not allowed;
- negative postconditions are not allowed.

All of the $2^4 = 16$ possible combinations of the restrictions have been studied from the computational point of view, for the two problems PLANSAT and k-PLANSAT.

Datalog STRIPS appears to be significantly harder than propositional STRIPS. In particular polynomial problems become EXPTIME-complete, NP-complete problems become NEXPTIME-complete, PSPACE-complete problems become EXPSPACE-complete. This is not surprising, since in datalog STRIPS there is a doubly exponential number of states (as opposed to a single exponential number in propositional STRIPS) and each state has an exponential number of "adjacent" states (as opposed to a polynomial number).

As we already noticed in the propositional case, if negative postconditions are not allowed then the state grows monotonically after each action. This results in a decrease of complexity, in the propositional as well as in the datalog case.

Polynomial cases in datalog STRIPS are shown only if the operators are fixed in advance. It is worth noticing that fixing the set of operators in advance allows to reduce in polynomial time datalog STRIPS to propositional STRIPS by considering all the ground instances of the operators.

3.3 Decidability of first-order planning

The first result in complexity of first-order planning has been proved in [Chapman, 1987]. In that work existence of a plan for the first-order TWEAK planner was shown to be undecidable.

Erol *et al.* [1992a; 1992c] make a detailed analysis of the cause of such undecidability. Their basic result is that logic programming is essentially the same as first-order STRIPS planning with no negative postconditions. In particular, they show a transformation from a generic planning instance Π with no negative postconditions into a logic program Ψ such that Π has a solution iff a query γ corresponding to the goal of Π is provable in Ψ. Moreover they show that every logic program can be transformed into an equivalent planning instance with no negative preconditions and no negative postconditions.

The similarity between the two problems allows to use "classic" results for decidability and undecidability of logic programming in order to obtain analogous results for planning.

In particular Erol *et al.* show that PLANSAT is decidable iff the language $\mathcal{L}$ contains only finitely many terms. In other words PLANSAT is undecidable (more precisely: semi-decidable) if function symbols are allowed or infinitely many constant symbols are allowed; the problem is decidable if there are no function symbols and only a finite number of constant symbols. In order to achieve decidability of PLANSAT while still retaining infinitely many terms, severe syntactic restrictions must be adopted. Some restrictions of this kind are shown.

Significant insights wrt the analysis of [Chapman, 1987] have been obtained. In particular it is shown that semi-decidability arises even if the initial state contains only a finite number of conditions. This is an important result, since it is not realistic to consider planning instances with infinite initial states.

4 Complexity of blocks-world planning

Blocks-world is one of the most widely studied planning domains and one of the most popular scenarios in AI. Several different formulations of the problem have been proposed. In its simplest description, blocks-world is characterized by:

- n cubical blocks, one table large enough to hold all of them;
- each block is either on another block or on the table;
- each block is either clear or there is a single block on top of it;
- a single kind of action is possible: move a single clear block onto the table, or onto another clear block;

  the preconditions and postconditions (both positive and negative) are obvious;
- the initial state is totally specified; the goal is not totally specified.

This scenario is called the Elementary Blocks World (EBW) in [Gupta and Nau, 1992]. If the goal is a completely specified state, then the scenario is called Primitive Blocks World (PBW) in [Bäckström, 1992a, Chapter 4].

EBW can be formalized in propositional STRIPS with:

- n conditions of the kind $clear(X)$, one for each block X;
- $n^2 + n$ conditions of the kind $on(X, Y)$, where X is a block and Y is either a block or a table;
- $n^2 + n$ operators of the kind $move(X, Y)$, where X is a block and Y is the destination (i.e. either a block or a table).

Different formalizations in STRIPS and in SAS$^+$ are possible. In particular, in [Bylander, 1991; Erol *et al.*, 1992b], [Bäckström, 1992a, Chapter 4] formalizations in which the only possible action is to move a block onto the table or from the table are shown. This scenario is called EBW$^-$ in [Bäckström, 1992a, Chapter 4].

Several researchers analyzed the complexity of the blocks-world. Here we summarize the major results:

- PLANSAT for EBW can be solved in $O(n\ log\ n)$ time [Gupta and Nau, 1992];
- PLANFIND for EBW can be solved in $O(n^3)$ time. In particular, a plan of length no more than twice the minimum length can be found [Gupta and Nau, 1992];
- k-PLANFIND for EBW$^-$ can be solved in $O(n^4)$ time [Bäckström, 1992a, Chapter 4];
- it is possible to transform in $O(n^3)$ an EBW problem into a PBW problem such that optimal solutions are preserved [Gupta and Nau, 1992];
- k-PLANSAT for EBW and PBW is NP-complete [Gupta and Nau, 1992].

  In [Chenoweth, 1991] a sligthly different version of the problem is proved to be NP-complete;

- k-PLANFIND for EBW and PBW is NP-hard and in FNP [Gupta and Nau, 1992].

In [Gupta and Nau, 1992] some extensions of EBW are studied. In particular EBW is extended by allowing varying block size (VBW), limited capacity of the table (LBW) and the combination of the two (VLBW). Clearly these three scenarios are at least as hard as EBW.

VBW and LBW have computational properties very similar to those of EBW. In particular PLANFIND is polynomial and k-PLANFIND is NP-hard and in FNP. On the other hand the minimum size of a solution for VLBW might be exponential.

5 Average-case analysis

The only study of average-case complexity of planning appeared so far has been done by Bylander [1993]. The PLANSAT problem is analyzed making several assumptions on the distribution of planning instances. In particular it is assumed that operators are randomly generated and that each possible precondition and postcondition is equally likely to appear within an operator, independently of other pre- and postconditions and other operators. Initial and goal states are also chosen at random independently of other parameters. No condition in the goal is satisfied in the initial state.

Even if the above assumptions do not approximate planning domains very well (see [Bylander, 1993] for a discussion on this point) the results obtained are definitely interesting.

The major result is that the space of problem instances can be separated into three regions, if appropriate parameters are considered. In particular, the three regions are delimited by different values of the number of operators in a planning instance:

1. in the first region there are few operators (overconstrained problem). In this case it is unlikely that a solution exists, as an example because there are conditions in the goal that are not made true by the postconditions of any operator. In the latter case it would be computationally easy to prove that a solution to a planning instance does not exist;

2. in the second region there are many operators (underconstrained problem). The density of solutions is high, thus making it easy to find a solution. As an example we could use a simple hill-climbing strategy that decreases the number of conditions to be achieved. Such a method has low complexity, and it is likely to find a solution[6];

3. finally, there is an intermediate number of operators, and nothing is said in general.

We now give only a rough description of the boundaries of the regions. In particular the number of pre- and postconditions must remain fixed (or small enough) when the size of the problem increases. Let n be the number of possible conditions, g the number of conditions in the goal and δ a real number such that $0 < \delta < 1$. δ is called the *accuracy parameter*:

1. if there are at most $O(n\ \sqrt[g]{\delta})$ operators, then the problem is overconstrained, and non-existence of plans can be proven in linear time with probability $(1 - \delta)$.

   Notice that –as expected– the smaller δ –i.e. the bigger the accuracy we want– the smaller the number of possible operators;

2. if there are at least $\Omega(n\ (log\ g)(log\ g/\delta)$ operators, then the problem is underconstrained, and a polynomial-time search (e.g. the greedy heuristic) finds a solution with probability $(1 - \delta)$.

   Notice that the smaller δ, the bigger the number of necessary operators.

These results suggest that the hard planning instances are confined to a narrow range of the number of operators. Such a narrow range should also contain all instances in which the minimum length of the solution is exponential.

Similar results have already been obtained for other problems such as graph coloring and propositional satisfiability [Cheeseman *et al.*, 1991; Mitchell *et al.*, 1992]. These analyses showed that the computational difficulty of many NP-complete problems is characterized by a parameter and that a critical

[6]Clearly this implies that in this situation the minimum length of a solution is only polynomial, and not exponential like in the worst case.

value for such a parameter exists. The critical value splits the space of problems into two regions. Problem instances are easily solvable in both regions, although for different reasons: too many constraints or too few constraints, respectively. The *really* hard problems occur at the boundary of these two regions.

The results of Bylander are particularly interesting because the underlying problem (PLANSAT) is PSPACE-complete, hence harder than the problems previously analyzed.

The average case complexity of the plan modification problem is also studied. In particular it is proven that, if the number of operators is big enough, plan modification can be easier than planning from scratch.

6 Related analyses

In this section we briefly survey results and give pointers to the literature for computational analyses of some problems related to planning.

6.1 Model preference default theories

In this subsection we analyze the relation between propositional STRIPS and a form of non monotonic reasoning. This analysis is –to our knowledge– novel.

A non monotonic formalism similar to propositional STRIPS has been defined by Selman and Kautz [1990]. The authors introduce the notion of *model-preference default theory*, which is a pair $< \Gamma; \Delta >$, where Γ is a propositional formula and Δ is a set of *defaults*, i.e. a syntactic object of the form $\beta \xrightarrow{D} \lambda$, where β is a set of literals and λ is a singleton. An example of default is the following:

$$\mathsf{student},\ \mathsf{young} \xrightarrow{D} \overline{\mathsf{worker}}$$

whose intuitive meaning is that a young student is by default (i.e. normally) not a worker.

The *preference (directed) graph* $G(\Gamma, \Delta)$ obtained from Γ and Δ is defined in the following way: Its nodes are truth assignments to the letters of Γ, and

there is an edge from node M to node M' if there is a default $\beta \xrightarrow{D} \lambda \in \Delta$ such that

(a) both M and M' satisfy β;

(b) M' satisfies λ and M does not;

(c) M and M' differ only in λ.

Papadimitriou [1991] gives a similar definition, the only difference being that the nodes of the graph $G(\Gamma, \Delta)$ are necessarily models of Γ. The difference is immaterial if Γ is a tautology.

If there is a path in $G(\Gamma, \Delta)$ from M to M', then we write $M \geq M'$. Truth assignment M' *dominates* M if $M \geq M'$ and it is not the case that $M' \geq M$. Finally, a truth assignment M is *undominated* if it is a model of Γ and there is no other truth assignment that dominates it, i. e. if it is in a sink strongly connected component of $G(\Gamma, \Delta)$. Undominated models exist iff Γ is satisfiable.

The above definition clearly relates to propositional STRIPS planning: STRIPS states correspond to truth assignments, and STRIPS operators correspond to defaults. More specifically, given a model preference default theory $< \texttt{true} ; \Delta >$ – i.e. the purely propositional part is a tautology – an instance of propositional STRIPS planning $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$ can be built as follows:

- $\mathcal{P}$ is the set of letters occurring in Δ;
- for each default $\beta^+\beta^- \xrightarrow{D} \lambda$ in Δ, where the literals in β^+ are positive and those in β^- are negative, there is an operator $\langle \varphi, \eta, \alpha, \delta \rangle \in \mathcal{O}$ such that
  - $\varphi = \beta^+$ is the set of positive preconditions;
  - $\eta = \beta^-$ is the set of negative preconditions;
  - if λ is a positive literal then $\alpha = \{\lambda\}$ is the set of positive postconditions and the set δ of negative postconditions is empty;
  - if λ is a negative literal then $\delta = \{\lambda\}$ is the set of negative postconditions and the set α of positive postconditions is empty;

- the initial state $\mathcal{I}$ and the goal $\mathcal{G}$ are not specified.

We call *default* STRIPS the restriction of propositional STRIPS in which all operators have one postcondition. The graph $G(\mathtt{true}, \Delta)$ is isomorphic to the graph corresponding to Π, as defined in Section 2.1. A sink strongly connected component of $G(\mathtt{true}, \Delta)$ corresponds to a set of states in $\mathcal{P}$ that is closed under the application of any number of operators. Therefore an undominated model corresponds to a state such that – regardless of the initial state – if reached once is reached over and over applying any sequence of operators. We call such a state a *sink* state.

If Γ is not a tautology, then $< \Gamma; \Delta >$ is related to propositional STRIPS with domain theories, as defined in [Bylander, 1992] and surveied in Section 2.7.

Model preference default theories have been thoroughly investigated from the computational point of view. The main computational problem that has been addressed is the following:

UNDOMINATEDFIND: given a model preference default theory $< \Gamma; \Delta >$, find an undominated model.

The corresponding problem in the planning framework has not been addressed from the computational point of view. We briefly survey the main results about UNDOMINATEDFIND, focusing on the case $\Gamma = \mathtt{true}$:

- it is in FPSPACE and PSPACE-hard [Papadimitriou, 1991];
- it is polynomial if for each default $\beta \xrightarrow{D} \lambda \in \Delta$ the set β is a singleton [Papadimitriou, 1991];
- it is polynomial if for each default $\beta \xrightarrow{D} \lambda \in \Delta$ all literals in β are positive [Selman and Kautz, 1990].

As we saw in Section 2.4, PLANSAT for default STRIPS (i.e. each operator has at most one postcondition) is PSPACE-complete. On the other hand PLANSAT for default STRIPS with single preconditions is NP-complete, and it is polynomial when only positive preconditions are allowed. Summing up, as far as default STRIPS is concerned, PLANSAT and finding a sink state have the same complexity in the worst case, and there are cases (namely single preconditions) in which the latter is harder.

As for the case when $\Gamma \neq \mathtt{true}$, interesting analyses have been shown in [Selman and Kautz, 1990; Papadimitriou, 1991].

6.2 Temporal projection

Temporal projection is the problem of computing the consequences of a set of actions. In particular the set of actions could be partially ordered. In this case, a specific property could be true after each possible execution of the actions compatible with the partial order, or true after some of them.

In [Chapman, 1987; Dean and Boddy, 1988b] the complexity of temporal reasoning is analyzed. Actions are represented in a formalism similar to propositional STRIPS. This analysis has been further developed in [Bäckström and Nebel, 1992; Nebel and Bäckström, 1992], where relationships of temporal projection to STRIPS planning are also shown.

6.3 Quantitative analyses of heuristics

A number of heuristics for speeding-up planning appeared in the literature. Here we cite few works in which quantitative analyses of the benefits of those heuristics are performed.

In [Korf, 1987] a study on the influence of properties of the goal on the complexity of planning has been carried out. In particular it is shown that if the subgoals are completely independent, then it is possible to reduce exponentially the time required to solve a problem instance. This analysis is further developed in [Yang *et al.*, 1992].

A planning algorithm is *non-systematic* if the same partial plan can be generated several times. Non-systematicity is a source of inefficiency in planning systems. In [McAllester and Rosenblitt, 1991; Minton and Bresina, 1991] systematic planners are proposed, and it is shown how the search of a plan can be exponentially faster than with traditional algorithms.

In the STRIPS approach finding a plan is basically the problem of finding the proof of a theorem (the goal) starting from some premises (the initial state). On the other hand in [Kautz and Selman, 1992] the problem of finding a plan is defined as the problem of finding the model of a propositional formula – a problem which is in FNP and NP-hard. The authors make a detailed comparison of different algorithms for finding a model of a propositional formula. The algorithms are tested on several benchmark planning problems such as blocks world or towers of Hanoi.

7 Conclusions

In this paper we surveied the major results appearing in the literature on the computational complexity of planning. We showed results for domain-independent planning and planning in specific domains (blocks-world), for first-order as well as propositional planners. We considered several propositional planners with different formal specifications. Moreover decision as well as search computational problems were addressed. We surveied tractable, intractable and undecidable cases, showing causes of intractability and the restrictions that have been studied for reducing the complexity. Preliminary results on the average case complexity of propositional planning were also surveied.

Computational studies about problems related to planning (e.g. temporal projection) have been briefly referenced. Relations between STRIPS planning and a form of non monotonic reasoning have been highlighted.

We close this work with a list of interesting computational analyses about planning that have not/have marginally been addressed in the literature.

- In several planning scenarios the minimal length of a solution is exponential in the input (e.g. Tower of Hanoi [Aho *et al.*, 1974], Extension of Blocks World [Gupta and Nau, 1992], SAS$^+$-PUB, SAS$^+$-PBS [Bäckström and Nebel, 1993; Bäckström, 1992b]). In some sense solutions of this kind are not reasonable, because we are asking for more information that we could ever hope to use.

  An interesting computational problem is the following: find a "small" and "reasonable" set of operators so that a polynomial-length solution exists. This would be a form of plan modification different from that surveied in Section 2.6. Loosely speaking, this problem is related to belief-revision.

- Generalizing the previous point, the problem of synthesizing a set of operators given a suitable set of restrictions on the initial and goal states has not been addressed. Obviously some restrictions on the available operators should be made, in order to avoid trivial solutions, such as "magical" operators that achieve the goal in a single step.

- The ultimate goal of a planner is to find an optimal plan. Anyway, it might be interesting to find a plan whose length is "not much higher" than the minimum. As a matter of fact, the approximability of NP-complete planning search problems has only marginally been addressed. The only result of this kind is about the approximability of finding a solution in blocks-world (cf. Section 4, [Bäckström, 1992a; Bäckström and Nebel, 1993; Gupta and Nau, 1992]).

  A different notion of approximate temporal reasoning is discussed in [Dean and Boddy, 1988a], where the authors propose to trade completeness for efficiency.

- Throughout this paper the underlying assumption has been that plan A is preferred wrt plan B if A is shorter than B, i.e. it contains less steps. Other metrics can be developed (e.g. associate a weight to each operator, and minimize the whole weight of the solution).

  Obviously this cannot lead to simpler computational problems. It would be interesting to analyze whether fancy metrics deliver planning problems that are computationally harder. An analysis of this kind has been done by Eiter and Gottlob [1993] for the abduction problem.

Acknowledgements

The author is grateful to Diego Calvanese for proofreading an earlier version and suggesting improvements.

References

[Aho *et al.*, 1974] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The design and analysis of computer algorithms.* Addison Wesley, Reading, MA, 1974.

[Allen *et al.*, 1990] J. Allen, J. Hendler, and A. Tate, editors. *Readings in planning.* Morgan Kaufmann, 1990.

[Bäckström and Klein, 1991] C. Bäckström and I. Klein. Parallel non-binary planning in polynomial time. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 268–273, 1991.

[Bäckström and Nebel, 1992] C. Bäckström and B. Nebel. On the computational complexity of planning and story understanding. In *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI-92)*, pages 349–353, 1992.

[Bäckström and Nebel, 1993] C. Bäckström and B. Nebel. Complexity results for SAS^+ planning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 1430–1453, 1993.

[Bäckström, 1992a] C. Bäckström. *Computational Complexity of Reasoning about Plans*. PhD thesis, Linköping University, 1992.

[Bäckström, 1992b] C. Bäckström. Equivalence and tractability results for SAS^+ planning. In *Proceedings of the Third International Conference on the Principles of Knowledge Representation and Reasoning (KR-92)*, pages 126–137, 1992.

[Brachman and Levesque, 1985] R. J. Brachman and H. J. Levesque. A fundamental tradeoff in knowledge representation and reasoning. In R. J. Brachman and H. J. Levesque, editors, *Readings in knowledge representation*, pages 41–70. Morgan Kaufmann, 1985.

[Bylander, 1991] T. Bylander. Complexity results for planning. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 274–279, 1991.

[Bylander, 1992] T. Bylander. Complexity results for extended planning. In *Proceedings of the First International Conference on Artificial Intelligence Planning Systems (AIPS-92)*, pages 20–27, 1992.

[Bylander, 1993] T. Bylander. An average case analysis of planning. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pages 480–485, 1993.

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

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

[Cheeseman *et al.*, 1991] P. Cheeseman, B. Kanefsky, and W. M. Taylor. Where the *really* hard problems are. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 331–337, 1991.

[Chenoweth, 1991] S. V. Chenoweth. On the NP-hardness of blocks world. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pages 623–628, 1991.

[Dean and Boddy, 1988a] T. Dean and M. Boddy. An analysis of time-dependent planning. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, pages 49–54, 1988.

[Dean and Boddy, 1988b] T. Dean and M. Boddy. Reasoning about partially ordered events. *Artificial Intelligence Journal*, 36:375–399, 1988.

[Dowling and Gallier, 1984] W. P. Dowling and J. H. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *Journal of Logic Programming*, 1:267–284, 1984.

[Eiter and Gottlob, 1993] T. Eiter and G. Gottlob. The complexity of logic-based abduction. In *Tenth Symposium on Theoretical Aspects of Computer Science (STACS-93)*, volume 665 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[Erol *et al.*, 1992a] K. Erol, D. S. Nau, and V. S. Subrahmanian. Complexity, decidability and undecidability results for domain-independent planning. Technical Report CS TR-2797, University of Maryland, Department of Computer Science, 1992.

[Erol *et al.*, 1992b] K. Erol, D. S. Nau, and V. S. Subrahmanian. On the complexity of domain-independent planning. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 381–386, 1992.

[Erol *et al.*, 1992c] K. Erol, D. S. Nau, and V. S. Subrahmanian. When is planning decidable? In *Proceedings of the First International Conference on Artificial Intelligence Planning Systems (AIPS-92)*, pages 222–227, 1992.

[Even *et al.*, 1976] S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM Journal of Computing*, 5:691–703, 1976.

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

[Garey and Johnson, 1979] M. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman, 1979.

[Gupta and Nau, 1992] N. Gupta and D. S. Nau. On the complexity of blocks-world planning. *Artificial Intelligence Journal*, 56:223–254, 1992.

[Johnson, 1990] D. S. Johnson. A catalog of complexity classes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A, chapter 2. Elsevier Science Publishers B. V. (North Holland), 1990.

[Kambhampati and Hendler, 1992] S. Kambhampati and J. A. Hendler. A validation-structure-based theory of plan modification and reuse. *Artificial Intelligence Journal*, 55:193–238, 1992.

[Kautz and Selman, 1992] H. A. Kautz and B. Selman. Planning as satisfiability. In *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI-92)*, pages 359–363, 1992.

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

[Ladner, 1977] R. E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal of Computing*, 6:467–480, 1977.

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

[McAllester and Rosenblitt, 1991] D. McAllester and R. Rosenblitt. Systematic nonlinear planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pages 634–639, 1991.

[Minton and Bresina, 1991] S. Minton and M. Bresina, J. ans Drummond. Commitment strategies in planning: A comparative analysis. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 259–265, 1991.

[Mitchell *et al.*, 1992] D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions for SAT problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 459–465, 1992.

[Nebel and Bäckström, 1992] B. Nebel and C. Bäckström. On the computational complexity of temporal projection and plan validation. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 748–753, 1992.

[Nebel and Koehler, 1993] B. Nebel and J. Koehler. Plan modification versus plan generation: A complexity-theoretic perspective. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 1436–1440, 1993.

[Papadimitriou, 1991] C. H. Papadimitriou. On selecting a satisfying truth assignment (extended abstract). In *Proceedings of the 32nd Annual Symposium on the Foundations of Computer Science (FOCS-91)*, pages 163–169, 1991.

[Sacerdoti, 1974] E. D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence Journal*, 5:115–135, 1974.

[Selman and Kautz, 1990] B. Selman and H. A. Kautz. Model-preference default theories. *Artificial Intelligence Journal*, 45:287–322, 1990.

[Vardi, 1982] M. Y. Vardi. The complexity of relational query languages. In *Proceedings of the 14th ACM Symposyum on Theory Of Computing (STOC-82)*, pages 137–146, 1982.

[Yang *et al.*, 1992] Q. Yang, D. S. Nau, and J. Hendler. Merging separately generated plans with restricted interactions. *Computational Intelligence*, 8:648–676, 1992.

Appendix: Terminology for computational complexity

Problems. Two kinds of problems are addressed in the survey: *decision* and *search* problems. In the former kind the answer is a simple boolean value, while the latter requires a more detailed answer. Deciding whether a plan exists is an example of decision problem, while finding a plan is an example of search problem.

Complexity classes. The following classes of decision problems are cited: P, NP, PSPACE, EXPTIME, NEXPTIME, EXPSPACE, semi-decidable. P (NP) is the class of decision problems that are solved in polynomial time by deterministic (non-deterministic) Turing machines. As for EXPTIME (NEXPTIME) the deterministic (non-deterministic) Turing machine works in exponential time, i.e. time bounded by $2^{p(n)}$, where $p(n)$ is a polynomial in the length n of the input. As for PSPACE and EXPSPACE the machine is deterministic, and works in polynomial, resp. exponential, space.

As far as the relations between complexity classes are concerned, it is known that P $\subseteq$ NP $\subseteq$ PSPACE $\subseteq$ EXPTIME $\subseteq$ NEXPTIME $\subseteq$ EXPSPACE. All containment relations are conjectured to be strict, although only the trivial relations P $\subset$ EXPTIME, NP $\subset$ NEXPTIME and PSPACE $\subset$ EXPSPACE have been proved.

Complexity classes for search problems are denoted by putting the letter F before the symbol of the corresponding class of decision problems. As an example the symbol FP denotes the class of problems solvable in polynomial time by a deterministic Turing machine.

Reducibility. The standard reduction between decision problems is the *many-one* polynomial reduction $\leq_p$. If X, Y are two problems, $X \leq_p Y$ holds iff there exists a polynomial time function f such that the answer to x is `yes` under X iff the answer to $f(x)$ is `yes` under Y. A problem D which is in a class C of decision problems and such that for all problems X in C it holds that $X \leq_p D$ is called C-complete. This delivers the well-known notions of NP-completeness, PSPACE-completeness, etc.

Although several notions of reducibility between search problems have been proposed, in this paper we refer to the *Turing* polynomial reduction $\leq_T$. If X, Y are two problems, $X \leq_T Y$ holds iff X can be solved in polynomial time using a subroutine without cost (usually called *oracle*) for the problem Y. A search or decision problem S such that for all problems Y in a class C of decision problems it holds that $Y \leq_T S$ is called C-hard. This delivers the well-known notions of NP-hardness, PSPACE-hardness, etc.

A search problem which is C-hard and in FC is sometimes called C-equivalent.

Pseudo-Polynomiality. A pseudo-polynomial algorithm is one whose running time would be polynomial if all input numbers were expressed in unary notation. Not all pseudo-polynomial algorithms are polynomial in the ordinary sense, i.e. when numbers are expressed in binary notation. Some of the problems presented in this paper admit pseudo-polynomial algorithms but do not seem to admit polynomial algorithms.