Tractable Plan Existence Does Not Imply Tractable Plan Generation

Peter Jonsson and Christer Bäckström
Department of Computer and Information Science
Linköping University
S-581 83 Linköping, Sweden
Phone: +46 13 28 24 15
Fax: +46 13 28 26 06
{petej,cba}@ida.liu.se

Abstract

We present a class, 3S, of planning instances such that the plan existence problem is tractable while plan generation is provably intractable for instances of this class. The class is defined by simple structural restrictions, all of them testable in polynomial-time. Furthermore, we show that plan generation can be carried out in time bounded by a polynomial in the size of the input and the size of the generated solution. For this class, we propose a provably sound and complete incremental planner, *i.e.*, a planner that can usually output an executable prefix of the final plan before it has generated the whole plan.

Keywords: Artificial intelligence; Planning; Computational complexity.

AMS Classification: Primary: 68T20; Secondary: 68Q25.

1 Introduction

It is well-known that planning is computationally difficult in the general case; plan existence for STRIPS-style formalisms is undecidable in the first-order case [6] and PSPACE-complete for all common propositional variants [5, 2]. Most of these analyses have focussed on the plan existence problem and it seems to have been tacitly assumed that plan existence is not substantially easier than plan generation. From a practitioner's point of view, however, the complexity of plan existence *per se* is of limited interest since the ultimate goal is to generate an executable plan, not just find out that one exists. It seems often to be assumed in the literature that the difficulty of generating a plan is directly related to the difficulty of finding out whether a plan exists. That is, plan generation is hard if plan existence is hard and it is easy if plan existence is easy. However, the suspicion that this is *not* always the case has been expressed by Bäckström and Nebel [4]. We show in this paper that the complexity of plan existence is not necessarily related to the complexity of plan generation.

We present a class of propositional STRIPS planning problems, 3S, having the property that plan existence is tractable while plan generation is provably intractable. This is shown by first giving an algorithm that decides the plan existence problem in polynomial time and then showing that there exist instances in 3S with exponentially sized minimal solution. Hence, there cannot exist any planner whatsoever generating a plan in polynomial time. The class is defined by simple structural restrictions, all of them testable in polynomial time.

Even though we cannot generate plans in polynomial time, we can probably do better than an ordinary planner by exploiting our knowledge about the structure of 3S. We present a planning algorithm that generates plans for 3S in *solution-polynomial* time, that is, in time bounded by a polynomial in the size of the instance and in the size of the produced solution. This planner is *incremental*, *i.e.*, it outputs executable prefixes of the final plan before it has generated the whole plan. For this type of planners, it is important that we can tell in advance whether there exists a plan or not. It would be disappointing if the planner generated a large prefix, which we started to execute, and then suddenly told us that no solution exists for the instance. For further details about incremental planning, see Jonsson and Bäckström [11].

The paper is organized as follows. First, the PSN planning formalism and the 3S class are defined. Then, we show that the plan existence problem for 3S is solvable in polynomial time while the plan generation problem is provably intractable. In addition we show that the bounded existence problem is NP-hard. We continue by presenting a prefix-generating planner for the 3S class that runs in solution-polynomial time. We conclude with a brief discussion of the results.

2 Basic Formalism

We base our work in this paper on the propositional STRIPS formalism with negative goals [5], which is equivalent to most other variants of propositional STRIPS [2].

Definition 2.1 An instance of the *PSN planning problem* is a quadruple $\Pi = \langle \mathcal{P}, \mathcal{O}, s_0, \langle {s_*}^+, {s_*}^- \rangle \rangle$ where

- $\mathcal{P}$ is a finite set of *atoms*;
- $\mathcal{O}$ is a finite set of *operators* of the form $\langle pre^+, pre^-, add, del, name \rangle$, where $pre^+, pre^- \subseteq \mathcal{P}$ denote the *positive* and *negative precondition* respectively, satisfying $pre^+ \cap pre^- = \emptyset$, $add, del \subseteq \mathcal{P}$ denote the *positive* and *negative postcondition* (add and delete list) respectively, satisfying $add \cap del = \emptyset$, and $name$ is a unique identifier;

- $s_0 \subseteq \mathcal{P}$ denotes the *initial state* and ${s_*}^+, {s_*}^- \subseteq \mathcal{P}$ denote the *positive* and *negative goal* respectively, satisfying ${s_*}^+ \cap {s_*}^- = \varnothing$;

The unique identifier for each operator is not technically necessary but it will simplify the forthcoming proofs. For $o = \langle pre^+, pre^-, add, del, name \rangle \subseteq \mathcal{O}$, we write $pre^+(o)$, $pre^-(o)$, $add(o)$, $del(o)$ and $name(o)$ to denote pre^+, pre^-, add, del and $name$ respectively.

Definition 2.2 Given a set of operators $\mathcal{O}$, define $Seqs(\mathcal{O})$ to be the set of all finite sequences of operators from $\mathcal{O}$.

A sequence $\langle o_1, \ldots, o_n \rangle \in Seqs(\mathcal{O})$ of operators is called a *PSN plan* (or simply plan). We continue by defining when a plan solves a planning instance.

Definition 2.3 The ternary relation $Valid \subseteq Seqs(\mathcal{O}) \times 2^{\mathcal{P}} \times (2^{\mathcal{P}} \times 2^{\mathcal{P}})$ is defined such that for arbitrary $\langle o_1, \ldots, o_n \rangle \in Seqs(\mathcal{O})$ and $S, T^+, T^- \subseteq \mathcal{P}$, $Valid(\langle o_1, \ldots, o_n \rangle, S, \langle T^+, T^- \rangle)$ iff either

1. $n = 0$, $T^+ \subseteq S$ and $T^- \cap S = \varnothing$ or
2. $n > 0$, $pre^+(o_1) \subseteq S$, $pre^-(o_1) \cap S = \varnothing$ and $Valid(\langle o_2, \ldots, o_n \rangle, (S - del(o_1)) \cup add(o_1), \langle T^+, T^- \rangle)$.

A plan $\omega \in Seqs(\mathcal{O})$ is said to be a *solution* to Π iff $Valid(\omega, s_0, \langle s_*^+, s_*^- \rangle)$.

We can now formally define the planning problems that we will consider in this paper.

Definition 2.4 Let $\Pi = \langle \mathcal{P}, \mathcal{O}, s_0, \langle s_*^+, s_*^- \rangle \rangle$ be a PSN instance. The *plan existence problem (PE)* is to decide whether there exists or not exists some $\omega \in Seqs(\mathcal{O})$ such that ω is a solution to Π. The *bounded plan existence problem (BPE)* takes an integer $K \geq 0$ as an additional parameter and decides whether a solution for Π of length K or shorter exists or not. The *plan generation problem (PG)* is to find some $\omega \in Seqs(\mathcal{O})$ such that ω is a solution to Π or answer that no such ω exists. The *bounded plan existence problem (BPG)* takes an integer $K \geq 0$ and finds a solution a solution for Π of length K or shorter exists or answers that no such solution exists.

3 The 3S Class

We begin by defining *dependency graphs* on planning instances. Such a graph represents for each atom p, which other atoms we will possibly have to add or delete in order to add or delete p. The idea is not new; a more restricted variant is used by Knoblock [12] in his ALPINE system. From now on, let $\Pi = \langle \mathcal{P}, \mathcal{O}, s_0, \langle s_*^+, s_*^- \rangle \rangle$ be an arbitrarily chosen PSN instance.

Definition 3.1 Let $p \in \mathcal{P}$ and let $Q \subseteq \mathcal{P}$. Then,

$$Affects(p) = \{o \in \mathcal{O} \mid p \in add(o) \text{ or } p \in del(o)\}$$

and $Affects(Q) = \bigcup_{q \in Q} Affects(q)$.

Definition 3.2 Define the *dependency graph* $DG(\Pi)$ as a directed labelled graph $DG(\Pi) = \langle \mathcal{P}, \mathcal{D} \rangle$ with vertex set $\mathcal{P}$ and arc set $\mathcal{D}$ such that for all $p, q \in \mathcal{P}$,

- $\langle p, +, q \rangle \in \mathcal{D}$ iff there exists an operator $o \in \mathit{Affects}(q)$ such that $p \in \mathit{pre}^+(o)$
- $\langle p, -, q \rangle \in \mathcal{D}$ iff there exists an operator $o \in \mathit{Affects}(q)$ such that $p \in \mathit{pre}^-(o)$.
- $\langle p, \sim, q \rangle \in \mathcal{D}$ iff there exists an operator $o \in \mathcal{O}$ such that $p, q \in \mathit{add}(o) \cup \mathit{del}(o)$ and $p \neq q$.

An example of an dependency graph for some Π with $\mathcal{P} = \{A, \ldots, I\}$ can be found in Figure 1. For example, we can see that there exists some operator affecting both A and B and that I is not dependent of the other atoms in any way. We continue by defining three classes of atoms, namely *static*, *irreversible* and *reversible* atoms. The intuition behind these classes is that a static atom must not or cannot be added or deleted, an irreversible atom can be added or deleted but not both and a reversible atom can be both added and deleted.

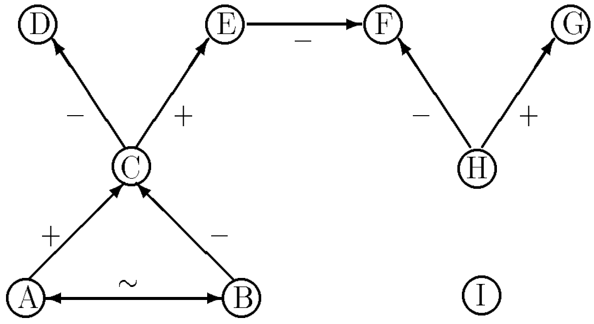

Figure 1: An example dependency graph.

Definition 3.3 Let $p \in \mathcal{P}$. Then, p is *static in* Π iff (1) $p \notin s_0$ and there does not exist an $o \in \mathcal{O}$ such that $p \in \mathit{add}(o)$ or (2) $p \in s_0$ and there does not exist an $o \in \mathcal{O}$ such that $p \in \mathit{del}(o)$ or (3) $p \notin s_0$ and $p \in s_*^-$ and there does not exist an $o \in \mathcal{O}$ such that $p \in \mathit{del}(o)$ or (4) $p \in s_0$ and $p \in s_*^+$ and there does not exist an $o \in \mathcal{O}$ such that $p \in \mathit{add}(o)$.

Case (1) says that there does not exist any operator that can add p and since $p \notin s_0$, p cannot occur in any state that any plan solving Π might achieve. Case (2) is analogous to case (1). Case (3) says that if we add p, then we cannot delete it again. But since $p \in s_*^-$ we must delete p if we have added it. Hence, p cannot be added by any plan solving Π. Case (4) is analogous to case (3). So, if an atom p is static in Π, then no plan solving Π can add or delete p.

Definition 3.4 An atom $p \in \mathcal{P}$ is *reversible in* Π iff for all $o \in \mathcal{O}$, whenever $p \in add(o)$ then there exists an $o' \in \mathcal{O}$ such that $p \in del(o)$ and vice versa. Moreover, p is *symmetrically reversible in* Π iff p is reversible and for all $o \in \mathcal{O}$, whenever $p \in add(o)$ then there exists an $o' \in \mathcal{O}$ such that $p \in del(o)$, $pre^+(o) = pre^+(o')$ and $pre^-(o) = pre^-(o')$, and vice versa.

If an atom p is reversible in Π, then plans that solve Π can contain both operators adding p and operators deleteing p. If an atom is symmetrically reversible, then we can always delete it under the same conditions as we can add it and vice versa.

Definition 3.5 An atom $p \in \mathcal{P}$ is *irreversible in* Π iff it is not static in Π and not reversible in Π.

If an atom p is irreversible in Π, then plans that solve Π must contain an operator that adds or deletes p, but not both.

Definition 3.6 Let $G = \langle V, E \rangle$ be a directed labelled graph and $G' = \langle V, E' \rangle$ be its undirected counterpart, that is, let

$$E' = \{(v, x, w), (w, x, v) \mid (v, x, w) \in E\}.$$

Then, for $v, w \in V$, w is *weakly reachable* from v in G iff there exists a path from v to w in $G' = \langle V, E' \rangle$.

Definition 3.7 Let $DG(\Pi) = \langle V, E \rangle$ and assume that $p \in \mathcal{P}$. Furthermore, let

$$Q_+^p = \{q \mid (p, +, q) \in E\},$$

$$Q_-^p = \{q \mid (p, -, q) \in E\},$$

$$DG_+^p(\Pi) = \langle V, E - \{(p, +, x) \in E \mid x \in V\} \rangle, \text{ and}$$

$$DG_-^p(\Pi) = \langle V, E - \{(p, -, x) \in E \mid x \in V\} \rangle.$$

Then, we can divide $\mathcal{P}$ into the following three sets:

1. $\mathcal{P}_+^p = Q_+^p \cup \{q \mid q \text{ is weakly reachable from some } r \in Q_+^p \text{ in } DG_+^p(\Pi)\}$.
2. $\mathcal{P}_-^p = Q_-^p \cup \{q \mid q \text{ is weakly reachable from some } r \in Q_-^p \text{ in } DG_-^p(\Pi)\}$.
3. $\mathcal{P}_0^p = \{q \in \Pi' \mid q \text{ is not weakly reachable from } p \text{ in } DG(\Pi)\}$.

Consider the dependency graph in Figure 1 and the vertex C. We can see that $Q_+^C = \{E\}$ and consequently, $\mathcal{P}_+^C = \{E\} \cup \{F, G, H\} = \{E, F, G, H\}$. Analogously, $Q_-^C = \{D\}$ and $\mathcal{P}_-^C = \{D\} \cup \varnothing = \{D\}$. Also note that $\mathcal{P}_0^C = \{I\}$. Obviously, $\mathcal{P}_+^p \cap \mathcal{P}_0^p = \varnothing$ and $\mathcal{P}_-^p \cap \mathcal{P}_0^p = \varnothing$ for all choices of p but, in the general case, $\mathcal{P}_+^p$ and $\mathcal{P}_-^p$ are not disjoint. This observation leads to the next definition:

Definition 3.8 An atom $p \in \mathcal{P}$ is *splitting in* Π iff $\mathcal{P}_+^p$ and $\mathcal{P}_-^p$ are disjoint.

The atom C in Figure 1 is a splitting atom because $\mathcal{P}_+^C \cap \mathcal{P}_-^C = \{E, F, G, H\} \cap \{D\} = \varnothing$. Another example is the atom F where both $\mathcal{P}_+^F$ and $\mathcal{P}_-^F$ equal $\varnothing$. Intuitively, if an atom p is splitting then the problem instance can be split into three subproblems which can be solved independently and the sets $\mathcal{P}_+^p, \mathcal{P}_-^p$ and $\mathcal{P}_0^p$ tells us which atom belongs to which subproblem. As a convention, we usually drop "in Π" when dealing with types of atoms if Π is clear from the context. We can now define the 3S class of planning problems.

Definition 3.9 3S is the set of PSN instances having acyclic dependency graphs and where every atom is static, symmetrically reversible or splitting.

Note that if $DG(\Pi)$ is acyclic, then Π is *unary*, that is, $|add(o) \cup del(o)| = 1$ for every $o \in \mathcal{O}$. Hence, every 3S instance is unary. It should also be noted that the restrictions on the atomic level are not orthogonal to each other. For example, an atom can be splitting and symmetrically reversible at the same time.

4 Polynomial Plan Existence

In this section, we show that the plan existence problem for instances in 3S is polynomial while the plan generation problem is provably intractable. However, we begin by defining some concepts that will facilitate the forthcoming proofs.

Definition 4.1 Let p be a member of $\mathcal{P}$ and $s, s^+, s^- \subseteq \mathcal{P}$. Then s is *compatible* with $\langle s^+, s^- \rangle$ wrt p iff (1) $p \in s$ and $p \notin s^-$ or (2) $p \notin s$ and $p \notin s^+$.

Loosely speaking, an initial state s_0 is compatible with a goal state $\langle s_*^+, s_*^- \rangle$ wrt. p if we do not have to add or delete p in order to satisfy the goal. We continue by defining a function $\Cap$ for restricting operators and planning instances to limited sets of atoms. We also define a function for recreating operators that have been restricted by $\Cap$.

Definition 4.2 Let $\Pi = \langle \mathcal{P}, \mathcal{O}, s_0, \langle s_*^+, s_*^- \rangle \rangle$ be a PSN instance, $o \in \mathcal{O}$ and $\mathcal{P}' \subseteq \mathcal{P}$. Then, $o \Cap \mathcal{P}'$ (the *restriction of* o *to* $\mathcal{P}'$) is the operator

$$\langle add(o) \cap \mathcal{P}', del(o) \cap \mathcal{P}', pre^+(o) \cap \mathcal{P}', pre^-(o) \cap \mathcal{P}', name(o) \rangle.$$

We define $\Cap$ for a set $\mathcal{O}' \subseteq \mathcal{O}$ of operators in the following way: $\mathcal{O}' \Cap \mathcal{P}' = \{o \Cap \mathcal{P}' \mid o \in \mathcal{O}'\}$. Finally, we define $\Cap$ for a PSN problem instance Π such that $\Pi \Cap \mathcal{P}' = \langle \mathcal{P}', \mathcal{O} \Cap \mathcal{P}', s_0 \cap \mathcal{P}', \langle s_*^+ \cap \mathcal{P}', s_*^- \cap \mathcal{P}' \rangle \rangle$

Definition 4.3 Let o be an operator and $\mathcal{O}$ a set of operators. Then, $\downarrow^{\mathcal{O}}(o)$ is defined as the unique operator $o' \in \mathcal{O}$ such that $name(o) = name(o')$. We generalize $\downarrow^{\mathcal{O}}$ to operate on plans in the obvious way, namely $\downarrow^{\mathcal{O}}(\langle o_1, \ldots, o_n \rangle) = \langle \downarrow^{\mathcal{O}}(o_1), \ldots, \downarrow^{\mathcal{O}}(o_n) \rangle$.

Observe that the previous definition is sound since we have assumed that every operator has a unique name in every operator set. In the next definition, we provide a method for removing certain operators from a planning instance.

Definition 4.4 Let $\mathcal{O}' \subseteq \mathcal{O}$ and $p \in \mathcal{P}$. Then, $R^+(p, \mathcal{O}') = \{o \in \mathcal{O}' \mid p \notin pre^+(o)\}$ and $R^-(p, \mathcal{O}') = \{o \in \mathcal{O}' \mid p \notin pre^-(o)\}$. We also define R^+ and R^- for PSN problem instances the obvious way; namely $R^+(p, \Pi) = \langle \mathcal{P}, R^+(p, \mathcal{O}), s_0, \langle s_*^+, s_*^- \rangle \rangle$ and $R^-(p, \Pi) = \langle \mathcal{P}, R^-(p, \mathcal{O}), s_0, \langle s_*^+, s_*^- \rangle \rangle$.

We can view $R^+(p, \Pi)$ as the problem instance Π with all operators o such that $p \in pre^+(o)$ is removed and $R^-(p, \Pi)$ as Π with all operators o such that $p \in pre^-(o)$ is removed. Finally, we define a well-known graph-theoretic concept.

Definition 4.5 Let $G = \langle V, E \rangle$ be a directed (labelled) graph. A vertex $v \in V$ is *minimal* iff there does not exist any $e \in E$ ending in v.

We claim that the PE-3S algorithm which is presented in Figure 2 solves the plan existence problem in polynomial time for problem instances in 3S. To prove the claim, we need the following three lemmata.

Lemma 4.6 Let $\Pi = \langle \mathcal{P}, \mathcal{O}, s_0, \langle s_*^+, s_*^- \rangle \rangle \in$ 3S and let $\mathcal{P}' = \mathcal{P} - \{p\}$ for some arbitrary $p \in \mathcal{P}$. Then, $\Pi \Cap \mathcal{P}', R^+(p, \Pi) \Cap \mathcal{P}', R^-(p, \Pi) \Cap \mathcal{P}' \in$ 3S.

Proof: If $\Pi' = \Pi \Cap \mathcal{P}'$, then $DG(\Pi')$ is acyclic and every atom in Π' is either static, symmetrically reversible or splitting since $\Pi \in$ 3S. Assume $\Pi' = R^+(p, \Pi) \Cap \mathcal{P}'$. $DG(\Pi')$ is acyclic since $DG(\Pi)$ is acyclic. Choose an arbitrary $q \in \mathcal{P}'$. If q is static, then q is static in Π' because $|\mathcal{O}'| \leq |\mathcal{O}|$. If q is symmetrically reversible, then q is either reversible or static in Π'. This follows from the fact that if some add operator that affects p is removed by R^+, then the corresponding delete operator is removed as well. Finally, if q is splitting, then q is still splitting in Π' because $|\mathcal{O}'| \leq |\mathcal{O}|$. The case when $\Pi' = R^-(p, \Pi \Cap \mathcal{P}')$ is analogous. $\Box$

function PE-3S(Π) : boolean (* $\Pi = \langle \mathcal{P}, \mathcal{O}, s_0, \langle s_*^+, s_*^- \rangle \rangle$ *)
if $\mathcal{P} = \varnothing$ **then return true**
else
    choose an atom p that is minimal in $DG(\Pi)$
    if p is static **then**
        if s_0 is not compatible with $\langle s_*^+, s_*^- \rangle$ wrt. p
        then return false
        elsif $p \notin s_0$ **then return** PE-3S($R^+(p, \Pi) \Cap (\mathcal{P} - \{p\})$)
        else return PE-3S($R^-(p, \Pi) \Cap (\mathcal{P} - \{p\})$)
    else return PE-3S($\Pi \Cap (\mathcal{P} - \{p\})$)

Figure 2: The PE-3S algorithm.

Lemma 4.7 Let $\Pi = \langle \mathcal{P}, \mathcal{O}, s_0, \langle s_*^+, s_*^- \rangle \rangle \in 3S$. Then, PE-3S($\Pi$) returns true if Π has a solution.

Proof: Suppose there exists a plan ω that solves Π, but PE-3S(Π) returns false. We show that this is impossible by induction over $|\mathcal{P}|$:
Basis step: $|\mathcal{P}| = 0$. PE-3S(Π) returns true by definition.
Induction hypothesis: Suppose the lemma holds for $|\mathcal{P}| \leq k$, $k \geq 0$.
Induction step: We want to show that the lemma holds for $|\mathcal{P}| = k + 1$. We have four cases:

1. PE-3S returns false in line 7. Obviously, Π does not have any solution. Contradiction.

2. PE-3S returns false in line 8. Since we cannot add p, we must check if $R^+(p, \Pi) \Cap (\mathcal{P} - \{p\})$ has any solution. By Lemma 4.6, $R^+(p, \Pi) \Cap (\mathcal{P} - \{p\}) \in 3S$ so, by the induction hypothesis, we can do this recursively. Hence, if PE-3S($R^+(p, \Pi) \Cap (\mathcal{P} - \{p\})$) returns false, then Π does not have any solution. Contradiction.

3. PE-3S returns false in line 9. Analogous to the previous case.

4. PE-3S returns false in line 10. By Lemma 4.6, $\Pi \Cap (\mathcal{P} - \{p\}) \in 3S$. Hence, by the induction hypothesis, we can check whether $\Pi \Cap (\mathcal{P} - \{p\})$ has a solution or not with PE-3S. If $\Pi \Cap (\mathcal{P} - \{p\})$ has no solution, then Π has no solution. Contradiction. $\Box$

Lemma 4.8 Let $\Pi = \langle \mathcal{P}, \mathcal{O}, s_0, \langle s_*^+, s_*^- \rangle \rangle \in 3S$ Then, Π has a solution if PE-3S(Π) returns true.

Proof: Assume PE-3S(Π) returns true. We show the lemma by induction over $|\mathcal{P}|$:
Basis step: If $|\mathcal{P}| = 0$, then PE-3S(Π) returns true in line 2. Obviously, $\langle\rangle$ is a valid plan for Π so the lemma holds in this case.
Induction hypothesis: Suppose the lemma holds for $|\mathcal{P}| \leq k$, $k \geq 0$.
Induction step: We want to show that the lemma holds for $|\mathcal{P}| = k + 1$. Let p be the minimal atom in $DG(\Pi)$ that PE-3S chooses in line 7 and let $\mathcal{P}' = \mathcal{P} - \{p\}$. (Note that the algorithm always can choose such a p since $DG(\Pi)$ is acyclic). We have three cases:

1. p is static. Since PE-3S(Π) returns true, s_0 is compatible with $\langle s_*^+, s_*^- \rangle$ wrt. p. Hence, PE-3S must return true in line 8 or 9. Both $R^+(p, \Pi) \Cap \mathcal{P}'$ and $R^-(p, \Pi) \Cap \mathcal{P}'$ are members of 3S by Lemma 4.6. So, by the induction hypothesis, there exists a valid plan ω for $R^+(p, \Pi) \Cap \mathcal{P}'$ or $R^-(p, \Pi) \Cap \mathcal{P}'$. Since p is static, $\downarrow^{\mathcal{O}}(\omega)$ is a valid plan for Π.

2. p is reversible. We know that p is not static so PE-3S must return true in line 10. Since p is minimal in $DG(\Pi)$, there exists operators o^+, o^- that adds p and deletes p having no preconditions at all. Hence, we can add and delete p freely. By Lemma 4.6, $\Pi \Cap \mathcal{P}' \in 3S$ so by the induction

hypothesis, there exists a valid plan ω for $\Pi \Cap \mathcal{P}'$. Consequently, there exists a plan ω' for Π. (Simply by inserting o^+ before every operator in $\downarrow^{\mathcal{O}}(\omega)$ that needs p to be true and inserting o^- before every operator in $\downarrow^{\mathcal{O}}(\omega)$ that needs p to be false. Possibly, we also have to insert some operator last in the plan to ensure that the goal state is satisfied.)

3. p is irreversible. We begin by showing that p is splitting. Remember that p is either static, symmetrically reversible or splitting. The case when p is static was taken care of earlier. Since p is irreversible, p cannot be symmetrically reversible. The only remaining possibility is that p is splitting. Consequently, PE-3S must return true in line 10. By Lemma 4.6, $\Pi' = \Pi \Cap \mathcal{P}' \in$ 3S and by the induction hypothesis, there exists a plan ω that solves Π'. Since p is splitting, $\mathcal{P}^p_+, \mathcal{P}^p_-$ and $\mathcal{P}^p_0$ are disjoint. Form the following three subinstances: $\Pi'_+ = \Pi \Cap \mathcal{P}^p_+, \Pi'_- = \Pi \Cap \mathcal{P}^p_-, \Pi'_0 = \Pi \Cap \mathcal{P}^p_0$. Assume that $p \in s_0$. As we know that p is not static, there exists an operator o^- that deletes p. Furthermore, we know that $\mathcal{P}^p_+, \mathcal{P}^p_-$ and $\mathcal{P}^p_0$ are disjoint, so we can reorder ω to the plan $\omega' = (\omega_+; \omega_-; \omega_0)$ where ω_+ solves Π'_+, ω_- solves Π'_- and ω_0 solves Π'_0. As a consequence, $\omega'' = (\downarrow^{\mathcal{O}}(\omega_+); o^-; \downarrow^{\mathcal{O}}(\omega_-); \downarrow^{\mathcal{O}}(\omega_0))$ is a valid plan solving Π. The case where $p \notin s_0$ is analogous. □

We are now able to prove that the plan existence problem for instances in 3S is polynomial.

Theorem 4.9 Let $\Pi = \langle \mathcal{P}, \mathcal{O}, s_0, \langle s^+_*, s^-_* \rangle \rangle \in$ 3S. Then, whether Π has a solution or not can be decided in polynomial time.

Proof: The recursion depth of PE-3S is bounded above by $|\mathcal{P}|$ since the number of atoms decreases strictly for each recursive level. Hence, PE-3S must eventually return true or false. By Lemmata 4.7 and 4.8, Π has a solution iff PE-3S(Π) returns true. Conversely, Π lacks a solution iff PE-3S(Π) is false. It remains to show that PE-3S runs in polynomial time for every Π. Constructing $DG(\Pi)$ and performing the different tests takes only polynomial time. We have already shown that we PE-3S will make less than $|\mathcal{P}|$ recursive calls. Consequently, PE-3S runs in polynomial time. □

5 Bounded Plan Existence and Plan Generation

We proceed by showing that the bounded plan existence problem for 3S is NP-hard. The reduction is based on the MINIMUM COVER problem, which is known to be NP-complete.

Definition 5.1 An instance of the MINIMUM COVER problem is given by a finite set $X = \{x_1, \ldots, x_m\}$, a set $C = \{C_1, \ldots, C_n\}$ of subsets of X and a positive integer $K \leq |C|$. The question is whether there exists a cover for X, *i.e.*, a subset $C' \subseteq C$ with $|C'| \leq K$ such that every element of X belongs to at least one member of C'.

Theorem 5.2 (Garey and Johnson [8]) The problem MINIMUM COVER is NP-complete.

Theorem 5.3 Bounded plan existence is NP-hard for PSN problem instances such that every atom is symmetrically reversible and their corresponding dependency graphs are acyclic.

Proof: NP-hardness is shown by reduction from MINIMUM COVER. Let $C = \{c_1, \ldots, c_m\}$ be a collection of subsets of a finite set $X = \{x_1, \ldots, x_n\}$ and let K be an integer such that $K \leq |C|$. Define a PSN problem instance $\Pi = \langle \mathcal{P}, \mathcal{O}, s_0, \langle s_*^+, s_*^- \rangle \rangle$ such that

- $\mathcal{P} = \{c_k \mid 1 \leq k \leq m\} \cup \{x_l \mid 1 \leq l \leq n\}$.
- $\mathcal{O} = \{c_k^+, c_k^- \mid 1 \leq k \leq m\} \cup \{x_{k,l}^+, x_{k,l}^- \mid 1 \leq k \leq n \text{ and } x_k \in C_l\}$ where

$$\begin{array}{lclcl}
pre^+(c_k^+) & = & pre^+(c_k^-) & = & \varnothing \\
pre^-(c_k^+) & = & pre^-(c_k^-) & = & \varnothing \\
add(c_k^+) & = & del(c_k^-) & = & \{c_k\} \\
add(c_k^-) & = & del(c_k^+) & = & \varnothing
\end{array}$$

$$\begin{array}{lclcl}
pre^+(x_{k,l}^+) & = & pre^+(x_{k,l}^-) & = & \{c_l\} \\
pre^-(x_{k,l}^+) & = & pre^-(x_{k,l}^-) & = & \varnothing \\
add(x_{k,l}^+) & = & del(x_{k,l}^-) & = & \{x_k\} \\
add(x_{k,l}^-) & = & del(x_{k,l}^+) & = & \varnothing
\end{array}$$

- $s_0 = \varnothing$
- $s_*^+ = \{x_l \mid 1 \leq l \leq n\}$, $s_*^- = \varnothing$

It is obvious that X has cover C' such that $|C'| \leq K$ iff there is a plan of size $|X| + K$ or less solving Π. Since Π satisfies the given restrictions, the theorem follows. □

Now, we turn our attention to the complexity of plan generation for instances in 3S. In the next theorem, we show that there exists instances having exponentially sized minimal solution in 3S.

Theorem 5.4 Let n be an arbitrary integer strictly greater than 0. Then there exists an instance $\Pi_n = \langle \mathcal{P}, \mathcal{O}, s_0, \langle s_*^+, s_*^- \rangle \rangle \in$ 3S such that $|\mathcal{P}| = n$ and all minimal plans solving Π are of length $2^n - 1$.

Proof: Define Π_n as follows:

- $\mathcal{P} = \{p_1, \ldots, p_n\}$;

- $\mathcal{O} = \{o_1^+, o_1^-, \ldots, o_n^+, o_n^-\}$ where for $1 \leq k \leq m$,

$$\begin{array}{lclcl} pre^+(o_k^+) & = & pre^+(o_k^-) & = & \begin{cases} \{p_{k-1}\} & \text{if } k > 1 \\ \varnothing & \text{otherwise} \end{cases} \\ pre^-(o_k^+) & = & pre^-(o_k^-) & = & \begin{cases} \{p_1, \ldots, p_{k-2}\} & \text{if } k > 2 \\ \varnothing & \text{otherwise} \end{cases} \\ add(o_k^+) & = & del(o_k^-) & = & \{p_k\} \\ add(o_k^-) & = & del(o_k^+) & = & \varnothing \end{array}$$

- $s_0 = \varnothing$;
- $s_*^+ = \{p_n\}$ and $s_*^- = \{p_1, \ldots, p_{n-1}\}$.

It is easy to see that Π_n is a 3S instance. Furthermore, the length of the shortest plan solving Π_n is $2^n - 1$ as shown by Bäckström and Nebel [4]. □

So, plan generation takes exponential time in the worst case. Consequently, the bounded plan generation problem also takes exponential time.

6 Incremental Planning

Since Lemma 4.8 is constructive, it allows us to devise a planner that generates a solution plan whenever one exists. By exploiting our knowledge of the structure of the plan, we can even construct a *incremental* planner, *i.e.*, a planner that attempts outputting an executable prefix of the final solution plan before the whole of this solution is completed. This algorithm, IP-3S, appears in Figure 3. To be able to output prefixes before the whole solution is computed, we use *streams*. The procedure *output* puts one or more elements on the stream and the function *read* returns the first element of the stream and removes it.

The ability of IP-3S to produce executable prefixes stems from two facts: (1) if an atom p is splitting, then the problem can be divided into three subproblems which can be solved independently and (2) *Interweave* does not have to wait for its incoming stream to be closed before it can begin putting operators on its output stream. To take full advantage of the prefix-generation, a certain amount of parallelism is needed. We do not want to wait until a recursive call in line 15 of IP-3S is completed before we begin to process the output on the stream from this call. Instead, we want the recursive call to be a new process that executes concurrently with the process that called it.

It should be noted that $\downarrow$ has to be redefined in a straight-forward way to operate on streams in order to make IP-3S work correctly. The IP-3S algorithm clearly follows the cases in the proof of Lemma 4.8 so the following theorem follows immediately.

Theorem 6.1 If Π is a soluble instance of 3S, then IP-3S will generate a plan ω solving Π

Since we can check if $\Pi \in 3S$ has a solution or not in polynomial time, it is not very restrictive that IP-3S requires Π to have a solution in order to work; It would, in fact, be very disappointing if IP-3S generated a large prefix (which we perhaps would start to execute) and then suddenly told us that no solution exists for the instance. We continue with defining a complexity concept for capturing the time complexity of IP-3S.

Definition 6.2 An algorithm runs in *solution-polynomial* time iff its running time is bounded above by some polynomial in the size of the input and in the size of the generated solution.

IP-3S is an example of an *output-sensitive* algorithm. Other examples of output-sensitive algorithms exist in the literature, *e.g.*, the convex hull algorithm by Jarvis [9]. By Theorem 5.4, IP-3S cannot run in polynomial time. However, it runs in solution-polynomial time, which is as good as we can hope for when dealing with problems having exponentially sized solutions.

Theorem 6.3 IP-3S runs in solution-polynomial time.

Proof: Suppose we want to compute IP-3S(Π) for some arbitrary $\Pi = \langle \mathcal{P}, \mathcal{O}, s_0, \langle s_*^+, s_*^- \rangle \rangle \in 3S$ and assume that the resulting plan ω has length L. Then, IP-3S will perform less than $|\mathcal{P}|$ recursive calls on non-empty subinstances of the original instance at most. This is trivial if the chosen p is static or reversible and follows from the fact that $\mathcal{P}_+^p$, $\mathcal{P}_-^p$ and $\mathcal{P}_0^p$ are disjoint otherwise. We can over-estimate the consumed time by assuming that every recursive calls works on a plan of length L. The construction of $DG(\Pi)$, finding a minimal p, the different tests and the $\downarrow$ and *Interweave* functions are all bounded by some polynomial p in the size of the instance $|\Pi|$ and L. Hence, the running time is $|\mathcal{P}| \cdot p(|\Pi|, L)$ at most and the theorem follows. $\Box$

It is important to notice that IP-3S is polynomial in the length of the *generated* plan, not in the shortest possible plan. Hence, it is possible that IP-3S can take exponential time when solving an instance Π though it is possible to solve Π in polynomial time with some other algorithm.

7 Discussion

We have presented a class of planning problems where we can tell in advance, in polynomial time, whether a solution exists or not. We have also presented a provably correct planner for this class that runs in polynomial time in the length of the solution it produces. Furthermore, this planner will, whenever possible, output successive valid prefixes of the final solution for immediate execution, concurrently with the continuing planning process. As we have noted before, it is very important that we can guarantee that the

```
function IP-3S(Π) : stream (* Π = ⟨𝒫, 𝒪, s_0, ⟨s_*^+, s_*^-⟩⟩ *)
if 𝒫 ≠ ∅ then
    choose an atom p that is minimal in DG(Π)
    if p is static then
      if p ∉ s_0 then output(↓^𝒪 (IP-3S(R^+(p, Π) ⋒ (𝒫 − {p}))))
      else output(↓^𝒪 (IP-3S(R^-(p, Π) ⋒ (𝒫 − {p}))))
    elsif p is irreversible then
      if p ∉ s_0 then
          output(↓^𝒪 (IP-3S(Π ⋒ 𝒫_-^p)))
          output(o) where o ∈ 𝒪 adds p
          output(↓^𝒪 (IP-3S(Π ⋒ 𝒫_+^p)))
          output(↓^𝒪 (IP-3S(Π ⋒ 𝒫_0^p)))
      else the case when p ∈ s_0 is analogous
    else (* p is reversible *)
      output(Interweave(↓^𝒪 (IP-3S(Π ⋒ (𝒫 − {p}))), p, 𝒪, s_0, ⟨s_*^+, s_*^-⟩))
close output stream
```

```
function Interweave(ω : stream, p, 𝒪, s_0, s_*^+, s_*^-) : stream
if p ∈ s_0 then added ← T
else added ← F
let o^+, o^- ∈ 𝒪 be such that p ∈ add(o^+) and p ∈ del(o^-)
while ω is not closed do
    o ← read(ω)
    if p ∈ pre^+(o) and not added then output(o^+); added ← T
    elsif p ∈ pre^-(o) and added then output(o^-); added ← F
    output(o)
if p ∈ s_*^+ and not added then output(o^+)
elsif p ∈ s_*^- and added then output(o^-)
close output stream
```

Figure 3: The IP-3S algorithm.

generated prefixes are prefixes of a valid plan. Other incremental planners, such as those presented by Ambros-Ingerson and Steel [1] and Drummond *et al.* [7], do not have this property. It is, however, important to note that they consider a much more general planning problem than the 3S class. It seems highly improbable that such general incremental planners can have the ability to tell whether a solution exists or not in polynomial time. In the first-order case, it would imply that the undecidable plan existence problem could be solved in polynomial time. In the propositional case, this would imply that the PSPACE-complete plan existence problem could be solved in polynomial time. Hence, we are probably forced to consider severely restricted problems if we want to use incremental planning techniques.

This research continues as well as complements our ongoing research into tractable planning, using syntactic restrictions [4] as well as structural ones [10]. Incremental planning seems to provide one way of tackling nontractable classes of planning problems, while also making replanning feasible. The variable-graph approach is an obvious continuation of our research into structural restrictions [10]. Interestingly, these graphs can be viewed as a generalization of the dependency graphs Knoblock [12] uses for generating abstraction hierarchies, where our graphs contain more information.

In earlier publications [4], we have argued that planning problems allowing exponential-size optimal solutions should be considered unrealistic.[1] This does not imply that the 3S class is unrealistic, however. It is important to distinguish between the inherent complexity of an application problem and the complexity of the hardest problems allowed by a planning formalism *per se.* The only natural examples of problems with exponential-size optimal solutions seem to be artificial puzzles, like Towers of Hanoi, which are deliberatly designed to have this property. Application problems arising 'naturally' in industry *etc.* can be expected not to exhibit this property. In other words, we can expect real problems fitting within the 3S class to have reasonably sized solutions. Note, however, that this would not be of much help to us if the 3S class did not allow tractable plan existence, since we would then still face the intractability for the unsolvable instances—not being able to tell in advance that these are unsolvable.

Most of the research in planning complexity have only addressed the plan existence problem. However, the class 3S puts in doubt whether it is relevant to concentrate research into planning complexity on plan existence. As is demonstrated in this paper, there might be a considerable gap in the hardness between plan existence and plan generation. Since we are usually interested in actually generating a plan, it seems reasonable to concentrate on the complexity of plan generation instead.

One problem with the IP-3S algorithm is that although it runs in polynomial time in the length of the solution it generates, it is not guaranteed to generate an optimal plan. In fact, in the worst case it could happen to generate an exponential plan when there is a short, optimal one. Although we can hope for this not to happen in practice, it seems hard to rule out the possibility by any simple means and this problem arises also for 'standard' general-purpose planners, like Tweak. However, while such planners can avoid the problem through backtracking, although at a considerably higher cost, this may not be possible if we want to generate prefixes greedily. This problem is not unique for incremental planning, however. An analogous problem arises in state abstraction, where the wrong choice of abstraction hierarchy can force the hierarchical planner to spend exponentially longer time generating an exponentially longer solution than a non-hierarchical planner [3]. For incremental planners, there seems to be a tuning factor between outputting prefixes early and guaranteeing reasonably short plans respectively—an interesting challenge for future research.

[1] This is simply a specialization of the 'same' claim for problems in general [8, p. 11].

8 Conclusions

We have presented a class of planning instances such that the plan existence problem is polynomial while plan generation is provably intractable for instances of this class. The class is defined by simple structural restrictions, allowing for polynomial-time membership testing. Furthermore, we have shown that plan generation can be carried out in solution-polynomial time, that is, in time bounded by a polynomial in the size of the input and the size of the generated solution. We have further proposed a solution-polynomial and prefix-generating planner IP-3S for the class that is provably sound and complete.

Acknowledgements

We would like to thank Christos Papadimitriou and the anonymous referees for discussions and comments which helped improving the paper. The research was supported by the *Swedish Research Council for Engineering Sciences (TFR)* under grants Dnr. 92-143 and Dnr. 93-00291.

References

[1] J. A. Ambros-Ingerson and S. Steel, Integrating planning, execution and monitoring, in: *Proc. 7th (US) Nat'l Conf. on Artif. Intell. (AAAI-88)*, Morgan Kaufmann, 1988, pp. 83–88.

[2] C. Bäckström, Expressive equivalence of planning formalisms, *Artif. Intell.* 76(1–2) (1995) 17–34.

[3] C. Bäckström and P. Jonsson, Planning with abstraction hierarchies can be exponentially less efficient, in: *Proc. 14th Int'l Joint Conf. on Artif. Intell. (IJCAI-95)*, Morgan Kaufmann, 1995, pp. 1599–1604.

[4] C. Bäckström and B. Nebel, Complexity results for SAS$^+$ planning, *Comput. Intell.* 11(4) (1995) 625–655.

[5] T. Bylander, The computational complexity of propositional STRIPS planning, *Artif. Intell.* 69 (1994) 165–204.

[6] D. Chapman, Planning for conjunctive goals, *Artif. Intell.* 32 (1987) 333–377.

[7] M. Drummond, K. Swanson, J. Bresina, and R. Levinson, Reaction-first search, in: *Proc 13th Int'l Joint Conf. on Artif. Intell. (IJCAI-93)*, Morgan Kaufmann, 1993, pp. 1408–1413.

[8] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, 1979.

[9] R. A. Jarvis, On the identification of the convex hull of a finite set of points in the plane, *Inf. Process. Lett.* 2 (1973) 18–21.

[10] P. Jonsson and C. Bäckström, Tractable planning with state variables by exploiting structural restrictions, in: *Proc. 12th (US) Nat'l Conf. on Artif. Intell. (AAAI-94)*, Morgan Kaufmann, 1994, pp. 998–1003.

[11] P. Jonsson and C. Bäckström, Incremental planning, in: *New Directions in AI Planning: Proc. 3rd Eur. WS. Planning (EWSP'95)*, IOS Press, 1995, pp. 79–90.

[12] C. A. Knoblock, Automatically generating abstractions for planning, *Artif. Intell.*, 68 (1994) 243–302.