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
email: {petej,cba}@ida.liu.se
phone: +46 13 282415, +46 13 282429
fax: +46 13 282606

Abstract

We present a class 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 solution-polynomial time, that is, 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. This opens up some interesting possibilities for interleaving plan generation with plan execution.

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 (Chapman 1987) and PSPACE-complete for all common propositional variants (Bäckström 1995; Bylander 1994). 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. We show in this paper this need not at all be the case.

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 have presented elsewhere (Jonsson & Bäckström 1995) 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. If each plan step takes reasonably long time to execute, we can assume that the planner generates plan steps rapidly compared to the time it takes to execute them. 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 perhaps would start to execute) and then suddenly told us that no solution exists for the instance.

Due to space limitations, we have omitted much detail and completely removed all proofs in this paper. A longer version of the paper is available by anonymous ftp at ftp.ida.liu.se@pub/labs/rkllab /people/petej/ewsp95.

Basic Formalism

We will only consider propositional planning, using the *Propositional* STRIPS *with Negative Goals (PSN)* formalism (Bylander 1994), which is equivalent to most other propositional formalisms (Bäckström 1995). For full definitions, see (Bäckström 1995; Bylander 1994).

Definition 0.1 *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* $\langle pre^+, pre^-, add, del\rangle$, *where* $pre^+, pre^- \subseteq \mathcal{P}$ *denote the* **positive** *and* **negative precondition** *respectively,* $add, del \subseteq \mathcal{P}$ *denote the* **positive** *and* **negative postcondition** *(add and delete list) respectively.*
- $s_0 \subseteq \mathcal{P}$ *denotes the* **initial state** *and* $s_*{}^+, s_*{}^- \subseteq \mathcal{P}$ *denote the* **positive** *and* **negative goal** *respectively.*

Restrictions

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 1994) in his ALPINE system.

Definition 0.2 *Let* $p \in \mathcal{P}$ *and let* $Q \subseteq \mathcal{P}$. *Then,* $Affects(p) = \{o \in \mathcal{O} | p \in add(o) \text{ or } p \in del(o)\}$ *and* $Affects(Q) = \bigcup_{q \in Q} Affects(q)$.

Definition 0.3 *For a given PSN instance* $\Pi = \langle \mathcal{P}, \mathcal{O}, s_0, \langle s_*^+, s_*^- \rangle\rangle$, *we define the corresponding* **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* $o \in Affects(q)$ *such that* $p \in pre^+(o)$
- $\langle p, -, q\rangle \in \mathcal{D}$ *iff there exists an operator* $o \in Affects(q)$ *such that* $p \in pre^-(o)$.
- $\langle p, \sim, q\rangle \in \mathcal{D}$ *iff there exists an operator* $o \in \mathcal{O}$ *such that* $p, q \in add(o) \cup del(o)$ *and* $p \neq q$.

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 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.

Definition 0.4
Let $\Pi = \langle \mathcal{P}, \mathcal{O}, s_0, \langle s_*^+, s_*^- \rangle\rangle$ *be a PSN instance and 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* $p \in add(o)$ *or (2)* $p \in s_0$ *and there does not exist an* $o \in \mathcal{O}$ *such that* $p \in del(o)$ *or (3)* $p \notin s_0$ *and* $p \in s_*^-$ *and there does not exist an* $o \in \mathcal{O}$ *such that* $p \in del(o)$ *or (4)* $p \in s_0$ *and* $p \in s_*^+$ *and there does not exist an* $o \in \mathcal{O}$ *such that* $p \in add(o)$.

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* $pre^-(o) = pre^-(o')$.

Finally, an atom $p \in \mathcal{P}$ *is* **irreversible in** Π *iff it is not static in* Π *and not reversible in* Π.

Definition 0.5 *Let* $G = \langle V, E\rangle$ *be a directed labelled graph and let* $E' = \{(v, x, w), (w, x, v) | (v, x, w) \in E\}$. *Then, for* $v, w \in V$, w **is weakly reachable from** v *iff there exists a path from* v *to* w *in* $G = \langle V, E'\rangle$.

Definition 0.6 *Let* $\Pi = \langle \mathcal{P}, \mathcal{O}, s_0, \langle s_*^+, s_*^- \rangle\rangle$ *be a PSN instance,* $DG(\Pi) = \langle V, E\rangle$ *and let* $p \in \mathcal{P}$. *Furthermore, let* $Q_+^p = \{q | (p, +, q) \in E\}$, $Q_-^p = \{q | (p, -, q) \in E\}$, $DG_+^p(\Pi) = \langle V, E - \{(p, +, x) \in E | x \in V\}\rangle$ *and* $DG_-^p(\Pi) = \langle V, E - \{(p, -, x) \in E | x \in V\}\rangle$. *Then, we can divide* $\mathcal{P}$ *into the following three sets:*

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

Definition 0.7
Let $\Pi = \langle \mathcal{P}, \mathcal{O}, s_0, \langle s_*^+, s_*^- \rangle\rangle$ *be a PSN instance. An atom* $p \in \mathcal{P}$ *is* **splitting in** Π *iff* $\mathcal{P}_+^p$ *and* $\mathcal{P}_-^p$ *are disjoint.*

We can now define the 3S class of planning problems.

Definition 0.8 3S *is the set of PSN instances having acyclic dependency graphs and where every atom is static, symmetrically reversible or splitting.*

Polynomial-time 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. We begin by defining some concepts used in the definition of the algorithm.

Definition 0.9
Let $\Pi = \langle \mathcal{P}, \mathcal{O}, s_0, \langle s_*^+, s_*^- \rangle \rangle$ *be a PSN instance,* $p \in \mathcal{P}$ *and let* $s, s^+, s^- \subseteq \mathcal{P}$*. 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* $p \notin s^+$*.*

Definition 0.10
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* $\langle add(o) \cap \mathcal{P}', del(o) \cap \mathcal{P}', pre^+(o) \cap \mathcal{P}', pre^-(o) \cap \mathcal{P}' \rangle$*. We define* $\Cap$ *for a set* $\mathcal{O}' \subseteq \mathcal{O}$ *of operators in the following way:* $\mathcal{O}' \Cap \mathcal{P}' = \{o \Cap \mathcal{P}' | o \in \mathcal{O}'\}$*. 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 0.11
Let $\Pi = \langle \mathcal{P}, \mathcal{O}, s_0, \langle s_*^+, s_*^- \rangle \rangle$ *be a PSN instance,* $\mathcal{O}' \subseteq \mathcal{O}$ *and* $p \in \mathcal{P}$*. Then,* $R^+(p, \mathcal{O}') = \{o \in \mathcal{O}' | p \notin pre^+(o)\}$ *and* $R^-(p, \mathcal{O}') = \{o \in \mathcal{O}' | p \notin pre^-(o)\}$*. 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* $R^-(p, \Pi) = \langle \mathcal{P}, R^-(p, \mathcal{O}), s_0, \langle s_*^+, s_*^- \rangle \rangle$*.*

Definition 0.12 *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 1 solves the plan existence problem in polynomial time for problem instances in 3S.

Theorem 0.13
Let $\Pi = \langle \mathcal{P}, \mathcal{O}, s_0, \langle s_*^+, s_*^- \rangle \rangle \in$ 3S*. Then, If* Π *has a solution or not can be decided in polynomial time by* PE-3S*.*

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 0.14 *For all* $n > 0$*, there is some instance* $\Pi = \langle \mathcal{P}, \mathcal{O}, s_0, \langle s_*^+, s_*^- \rangle \rangle \in$ 3S *such that* $|\mathcal{P}| = n$ *and all minimal plans solving* Π *are of length* $2^n - 1$*.*

So, plan existence is polynomial for instances in 3S while plan generation takes exponential time in the worst case.

Incremental Planning

We have seen that it is not possible to generate plans in polynomial time for 3S. However, we can generate plans in *solution-polynomial* time.

Definition 0.15 *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.*

Theorem 0.16 *For* 3S *there exists a planning algorithm* PPG-3S *with the following properties:*

- *It is sound, complete and runs in solution-polynomial time.*
- *It produces executable prefixes of the final plan before it has generated the complete plan.*

Having a solution-polynomial algorithm is probably as good as we can hope for when dealing with instances having exponentially sized plans. It is important to notice that the planner is polynomial in the length of the *generated* plan, not in the shortest possible plan. Hence, it is possible that the algorithm can take exponential time when solving an instance Π though it is possible to solve Π in polynomial time with some other algorithm.

We claim that this planner is efficient in practice. If each plan step takes a reasonable amount time to execute, we can assume that the planner generates plan steps rapidly compared to the time it takes to execute them. Hence, the time spent on planning is negligible. Note that, 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 perhaps would start to execute) and then suddenly told us that no solution exists for the instance. Executing a prefix of a non-solution is not advisable, since we may wish to try planning for some alternative goal if there is no solution for the first one. However, executing the invalid prefix may prevent us from reaching the alternative goal.

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 1: The PE-3S algorithm.

Discussion

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.

The idea of incremental planning is not new (see, for example, (Ambros-Ingerson & Steel 1988)) but the problem seems to have been little studied in the literature. This is somewhat surprising, since we believe incremental planning to be an interesting alternative to replanning in domains with uncertainty. If failures are frequent, actions may often fail already when executing the first prefix. We can then start replanning immediately, but need not wait for a whole new plan to be generated—only the first prefix—until we can start executing again. However, much work remains before incremental planners are applicable in practice. Typical problems to tackle ranges from the worst- and average-case time of producing the first prefix to the quality of the produced plan, for example, in terms of its length. An important question is if it is possible to determine upper and/or lower bounds on the length of a plan before generating it.

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 incremental planner for the class that is provably sound and complete. This opens up some interesting possibilities for interleaving plan generation with plan execution.

References

Ambros-Ingerson, J. A., and Steel, S. 1988. Integrating planning, execution and monitoring. In *Proceedings of the 7th (US) National Conference on Artificial Intelligence (AAAI-88)*, 83–88. St. Paul, MN, USA: American Association for Artificial Intelligence.

Bäckström, C. 1995. Expressive equivalence of planning formalisms. *Artificial Intelligence* 76(1–2):17–34. ARTINT 1234.

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

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

Jonsson, P., and Bäckström, C. 1995. Incremental planning. In Ghallab, M., and Milani, A., eds., *New Directions in AI Planning: EWSP'95—3rd European Workshop on Planning*, Frontiers in AI and Applications, 79–90. Assisi, Italy: IOS Press.

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