Paper III

Planning with Abstraction Hierarchies can be Exponentially Less Efficient

Christer Bäckström and Peter Jonsson

Abstract

It is well-known that state abstraction can *speed up* planning exponentially, under ideal conditions. We extend this finding—showing that state abstraction may likewise *slow down* planning exponentially, and even result in generating an exponentially longer solution than necessary. This phenomenon can occur for abstraction hierarchies which are generated automatically by the ALPINE and HIGHPOINT algorithms. We further show that there is little hope of any drastic improvement upon these algorithms—it is computationally difficult to generate abstraction hierarchies which allow finding good approximations of optimal plans.

1 Introduction

One common approach to improving the efficiency of planning is to use a hierarchical planner based on *state abstraction*—ignoring certain literals, either in the operator preconditions [Sacerdoti, 1974] or in the whole language [Knoblock, 1991, 1994]. First an abstracted version of the problem instance is solved, thus not taking all details into account and resulting in a plan which is correct at this abstraction level. This plan is then used as a skeleton plan to be filled in with more detail at the next lower level—a process referred to as *refinement.* Repeated refinement results in a solution to the original, non-abstract problem.

Although state abstraction cannot avoid exponential search spaces in the general case, it is usually considered a powerful method for reducing the search effort. It has been demonstrated that the method speeds up planning considerably for certain test examples [Knoblock, 1994; Bacchus and Yang, 1994]. This is augmented with theoretical results [Knoblock, 1991] showing that state abstraction can reduce the size of the search space from exponential to linear under certain ideal conditions. These conditions are very restrictive, however, and are not likely to be met in (m)any real applications. One of the conditions is that the hierarchy satisfies the *downward refinement property (DRP)* [Bacchus and Yang, 1994], which guarantees that no backtracking occurs between abstraction levels. Bacchus and Yang [1994] analysed the expected search complexity when this particular condition does not hold—more precisely, as a function of the probability that a plan at some abstraction level can be refined into a plan at the next lower level. They found that the search complexity is linear both when this probability is close to 1 and when it is close to 0. However, there is a phase-transition effect increasing the search complexity considerably when the probability is neither low nor high. Bacchus and Yang also reported that the expected search effort may be somewhat higher with abstraction than without in this middle region, namely if most search has to be redone at the ground level. However, the literature seems to tacitly assume that state abstraction will never cause any major harm. Contrary to this, we show that just as state abstraction can *speed up* planning exponentially, it can also *slow down* planning exponentially, and even force the hierarchical planner to produce an exponentially longer solution than a non-hierarchical planner!

Knoblock [1994] has further presented an algorithm, ALPINE, for generating abstraction hierarchies that are *ordered monotonic*—a property guaranteeing that no refinement of an abstract plan can undo any effects of the abstract plan. Bacchus and Yang [1994] have presented a modification of this algorithm, HIGHPOINT, whose hierarchies are ordered monotonic and expected to satisfy the DRP more closely. While these algorithms produce good hierarchies in many cases, they are not guaranteed to be harmless. In fact, we show that both algorithms may produce the type of abstraction hierarchy that leads to exponentially longer solutions. Furthermore, we show that using the same underlying principle as in ALPINE and HIGHPOINT, it is computationally difficult to generate an abstraction hierarchy that allows a hierarchical planner to generate a solution with length within a constant factor of the optimal plan length (we actually prove an even stronger approximation bound—a logarithmic factor in the size of the instance).

2 Basic Formalism

We first define some basic concepts.

Definition 2.1 Given a set S, we let $Seqs(S)$ denote the set of all sequences formed by members of S. We further use the symbol ";" to denote sequence concatenation. Given a set $\mathcal{P} = \{p_1, \ldots, p_n\}$ of propositional atoms, $\mathcal{L}_\mathcal{P}$ denotes the corresponding set of literals, ie. $\mathcal{L}_\mathcal{P} = \{p, \neg p \mid p \in \mathcal{P}\}$. A set $S \subseteq \mathcal{L}_\mathcal{P}$ of literals is *consistent* iff there is no atom p such that $\{p, \neg p\} \subseteq S$. For $S \subseteq \mathcal{L}_\mathcal{P}$ we further define $Gen(S) = \{p \mid p \in S \text{ or } \neg p \in S\}$, ie. the set of atoms generating the literals in S.

Since we will only prove hardness results, we need only consider a propositional formalism, and the results will carry over automatically to more expressive formalisms. More precisely, we will use the ground version of the TWEAK formalism [Chapman, 1987], which is known to be expressively equivalent, under polynomial reduction, to most other common variants of propositional STRIPS [Bäckström, 1995].

Definition 2.2 A *planning problem instance* is a quadruple $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$ where

- $\mathcal{P}$ is a finite set of *atoms*;
- $\mathcal{O}$ is a finite set of *operators* of the form $\langle pre, post \rangle$ where $pre, post \subseteq \mathcal{L}_\mathcal{P}$ are consistent and denote the *pre-* and *post-condition* respectively;
- $\mathcal{I}, \mathcal{G} \subseteq \mathcal{L}_\mathcal{P}$ are consistent and denote the *initial* and *goal state*, respectively.

For $o = \langle pre, post \rangle \subseteq \mathcal{O}$, $pre(o)$ and $post(o)$ denote *pre* and *post* respectively. A sequence $\langle o_1, \ldots, o_n \rangle \in Seqs(\mathcal{O})$ of operators is called a *plan* over Π. The function *Result* is defined for all consistent states $S \subseteq \mathcal{L}_\mathcal{P}$ and plans $\langle o_1, \ldots, o_n \rangle \in Seqs(\mathcal{O})$ as

$$\begin{aligned}
&Result(\langle\rangle, S) = S \\
&Result(\langle o_1, \ldots, o_n \rangle, S) = \\
&\quad Result(\langle o_2, \ldots, o_n \rangle, S \cup post(o) - \{p \mid \neg p \in post(o)\})
\end{aligned}$$

We say that a plan $\langle o_1, \ldots, o_n \rangle \in Seqs(\mathcal{O})$ is a *solution* to an instance $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$ iff

1. $pre(o_1) \subseteq \mathcal{I}$;
2. $\mathcal{G} \subseteq Result(\langle o_1, \ldots, o_n \rangle, \mathcal{I})$ and
3. $pre(o_i) \subseteq Result(\langle o_1, \ldots, o_i - 1 \rangle, \mathcal{I})$ for all $1 < i \leq n$.

3 State Abstraction

There are two common ways of performing state abstraction: the *relaxed* method and the *reduced* method. The relaxed method was pioneered for planning in the AbStrips planner [Sacerdoti, 1974]. *Criticality values* are assigned to the literals and at each abstraction level i, all literals with criticality value $< i$ are omitted from the operator preconditions. The reduced method [Knoblock, 1991, 1994] goes even further by restricting the whole language at level i to only those literals having criticality value $\geq i$. We will base our theorems on the reduced model, but they trivially hold also under the relaxed model.

Definition 3.1 Given a set of of atoms $\mathcal{P}$, an *abstraction* of $\mathcal{P}$ is a set of atoms $\mathcal{P}' \subseteq \mathcal{P}$. An *n-level abstraction hierarchy* on $\mathcal{P}$ is a chain $\mathcal{P}^n \subseteq \ldots \subseteq \mathcal{P}^1 \subseteq \mathcal{P}^0$ where $\mathcal{P}^n = \varnothing$ and $\mathcal{P}^0 = \mathcal{P}$. We will generally write the abstraction hierarchy as an ordered partitioning $\langle \mathcal{D}^{n-1}, \ldots, \mathcal{D}^0 \rangle$ of $\mathcal{P}$ where $\mathcal{D}^i = \mathcal{P}^i - \mathcal{P}^{i+1}$ for all i. The mapping of a state $S \subseteq \mathcal{L}_\mathcal{P}$ onto the abstract level i, for some $1 \leq i \leq n$, is denoted S^i and is defined as $S^i = S \cap \mathcal{L}_{\mathcal{P}^i}$. Similarly, the mapping of a ground operator $o = \langle pre, post \rangle$ onto the abstract level i is denoted o^i and defined as $o^i = \langle pre^i, post^i \rangle$. The mapping of an operator set to level i is consequently defined as $\mathcal{O}^i = \{o^i \mid o \in \mathcal{O}\}$ and the mapping of a planning instance $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$ to level i is defined as $\Pi^i = \langle \mathcal{P}^i, \mathcal{O}^i, \mathcal{I}^i, \mathcal{G}^i \rangle$. We refer to level 0 as the *ground level.*

procedure $\text{HPLAN}(\mathcal{O}, \mathcal{I}, \mathcal{G}, \langle \mathcal{D}^{n-1}, \ldots, \mathcal{D}^0 \rangle)$
$\omega \leftarrow \text{PLAN}\,(\mathcal{O}^{n-1}, \mathcal{I}^{n-1}, \mathcal{G}^{n-1})$
if no such plan **then fail**
for i from $n-1$ to 1 **do**
  $\omega \leftarrow Refine(\omega, i)$
return ω

procedure $Refine(\omega, i)$
Assume $\omega = \langle o_1^i, \ldots, o_k^i \rangle$
$S_0 \leftarrow \mathcal{I}^i$; $T_0 \leftarrow \mathcal{I}^{i-1}$
for j from 1 to k **do**
  $S_j \leftarrow Result(o_j^i, S_{j-1})$
for j from 1 to k **do**
  $\omega_j \leftarrow \text{PLAN}\,(\mathcal{O}^{i-1}, T_{j-1}, S_j)$
  if no such plan **then fail**
  $T_j \leftarrow Result(\omega_j, T_{j-1})$
$\omega_{k+1} \leftarrow \text{PLAN}\,(\mathcal{O}^{i-1}, T_k, \mathcal{G}^{i-1})$
return $\omega_1;\ldots;\omega_{k+1}$

Figure 1: The hierarchical planning algorithm (search control omitted).

The general method for planning with abstraction hierarchies can be cast as an algorithm, HPLAN (see Figure 1). This planner relies on a non-hierarchical planner PLAN for solving subproblems within abstraction levels. PLAN can be any planner for the language at hand, but it must be sound and complete to guarantee soundness and completeness of HPLAN. We will further assume that PLAN generates shortest plans.

When solving an instance $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$ under an abstraction hierarchy $\langle \mathcal{D}^{n-1}, \ldots, \mathcal{D}^0 \rangle$, HPLAN first uses PLAN to solve the most abstract version, Π^{n-1}, of this instance. This results in a plan $\langle o_1^{n-1}, \ldots, o_k^{n-1} \rangle$ over the abstract operator set $\mathcal{O}^{n-1}$. This plan is used as a skeleton for solving the instance Π^{n-2}, with initial and goal states $\mathcal{I}^{n-2}$ and $\mathcal{G}^{n-2}$ respectively. In addition, the intermediate states $S_1 = Result(o_1^{n-1}, \mathcal{I}^{n-1}), \ldots, S_k = Result(o_k^{n-1}, S_{k-1})$ on level $n-1$ are used as new subgoals on level $n-2$. In this way we get $k+1$ subproblems to solve on level $n-2$, each one hopefully easier than solving Π^{n-2} from scratch. Each of these subproblems is solved using PLAN, and these solutions are concatenated into a solution for Π^{n-2}. This process is then repeated until we reach the ground level, which results in a solution for $\Pi^0 = \Pi$.

The process of using a plan on one abstraction level as a skeleton for producing a plan at the next lower level is called *refining* the plan. In the general case, for abstraction hierarchies not satisfying the DRP, HPLAN must also use backtracking and try refining another skeleton plan on some level whenever a subproblem cannot be solved. However, to simplify matters we omit backtracking in this paper since we will only use HPLAN for hierarchies satisfying the DRP.

4 Exponential Slow-down

Knoblock [1991] has shown that, under certain ideal conditions, the size of the search space can be reduced from exponential to linear by using HPLAN and an abstraction hierarchy instead of an ordinary non-hierarchical planner. Most of these conditions are expressed in terms involving properties of the actual planning process and properties of the final solution, and are thus difficult to cast in terms involving only properties of the instance. One of the conditions is the DRP, *i.e.*, there is no backtracking between abstraction levels.

This section presents some complementary results: state abstraction can also cause an exponential blow-up of the search space, causing an exponential slow-down, under certain conditions—even for hierarchies satisfying the DRP. Furthermore, this exponential slow-down is accompanied by the even worse result that the generated solution is exponentially longer than the shortest one!

Consider the following generic planning instance, Σ_n, and the two possible abstraction hierarchies $\mathcal{H}_1$ and $\mathcal{H}_2$.

Definition 4.1 For all even $n > 0$ we define

$$\Sigma_n = \langle \{p_0, \ldots, p_{n-1}\}, \mathcal{O}_n, \varnothing, \{p_{n-2}, p_{n-1}\} \rangle,$$

where $\mathcal{O}_n$ contains the $2n$ operators $s_0, r_0, \ldots, s_{n-1}, r_{n-1}$ as defined in Table 1. We further define the following two abstraction hierarchies for Σ_n,

$$\begin{aligned} \mathcal{H}_1 &= \langle \{p_0\}, \{p_1\}, \{p_2\}, \{p_3\}, \ldots, \{p_{n-2}\}, \{p_{n-1}\} \rangle, \\ \mathcal{H}_2 &= \langle \{p_1\}, \{p_0\}, \{p_3\}, \{p_2\}, \ldots, \{p_{n-1}\}, \{p_{n-2}\} \rangle. \end{aligned}$$

Both $\mathcal{H}_1$ and $\mathcal{H}_2$ obviously satisfy the DRP and are ordered.

We can now prove that there is an exponential difference in the sizes of the solutions and search spaces depending on the choice of abstraction hierarchy.

Theorem 4.2 HPLAN will produce a solution of length n for Σ_n under $\mathcal{H}_1$.

Proof: Let L_n be the length of the shortest plan HPLAN can produce under $\mathcal{H}_1$. We prove by induction over n that for even $n > 0$,

$$L_n = \begin{cases} 2, & \text{for } n = 2, \\ 2 + L_{n-2}, & \text{for } n > 2. \end{cases}$$

Base step: For $n = 2$ all operators have empty preconditions, so the behaviour of HPLAN will correspond to the two uppermost levels of Figure 2. The resulting plan is $\langle s_1, s_0 \rangle$, which clearly must be a shortest plan. Hence, $L_2 = 2$.

Induction step: Assume the claim holds for all even $k < n$ for some even $n > 2$. Planning on the four most abstract levels will proceed as shown in Figure 2. The initial state will be empty on all abstraction levels and orderedness of the abstraction hierarchy guarantees that the last three states on level $n - 4$ will be refined into states subsuming these states. Hence, the operators s^{n-4}_{n-1} and s^{n-4}_{n-2} will be refined into the single-operator plans $\langle s^i_{n-1} \rangle$ and $\langle s^i_{n-2} \rangle$ respectively at each level $i < n - 4$. It remains to analyse the subplan $\langle s^{n-4}_{n-3}, s^{n-4}_{n-4} \rangle$. Orderedness guarantees that the atom p_{n-1} cannot be affected and will not be required for any refinement of this subplan. Hence, this atom can be ignored for the expansion. Substituting indices $n - 1$ and $n-2$ for $n-3$ and $n-4$, respectively, then shows that the subplan $\langle s^{n-4}_{n-3}, s^{n-4}_{n-4} \rangle$ is isomorphic to the plan $\langle s^{n-2}_{n-1}, s^{n-2}_{n-2} \rangle$ on level $n - 2$, and similarly for the adjacent states. That is, by ignoring the atom p_{n-1} we see that the subplan

operator	*pre*	*post*
s_{2i}	$\{p_{2i-1}, p_{2i-2}\}$	$\{p_{2i}\}$
r_{2i}	$\{p_{2i-1}, p_{2i-2}\}$	$\{\neg p_{2i}\}$
s_{2i+1}	$\{\neg p_{2i-1}, \neg p_{2i-2}\}$	$\{p_{2i+1}\}$
r_{2i+1}	$\{\neg p_{2i-1}, \neg p_{2i-2}\}$	$\{\neg p_{2i+1}\}$

Table 1: Operators for the planning instance Σ_n, where $0 \leq 2i < n$ and with the exception that the operators s_0, r_0, s_1 and r_1 have empty preconditions.

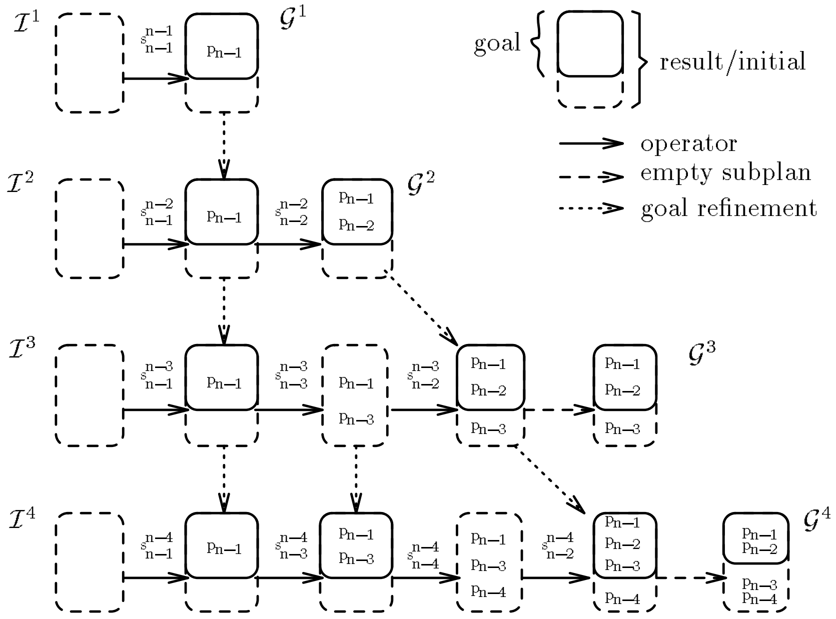

Figure 2: Applying HPLAN *to* Σ_n *under* $\mathcal{H}_1$.

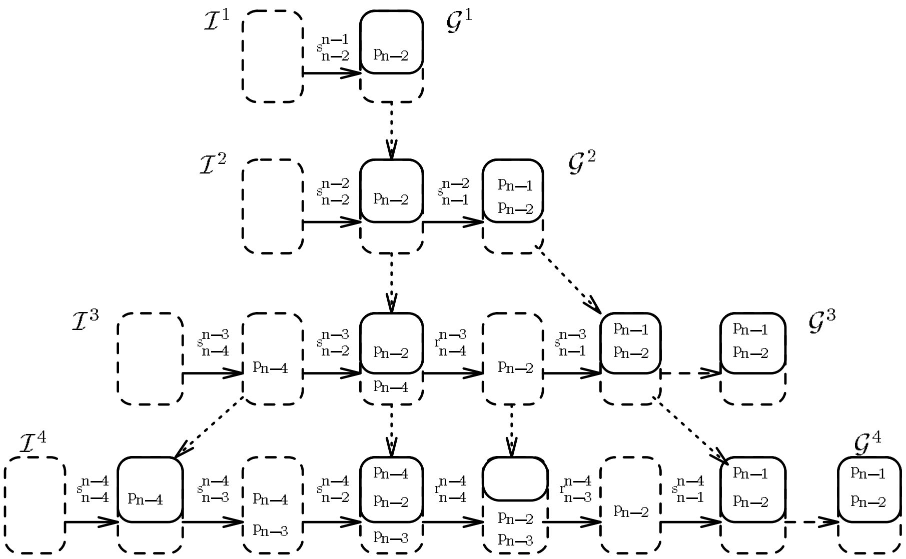

Figure 3: Applying HPLAN *to* Σ_n *under* $\mathcal{H}_2$*.*

$\langle s^{n-4}_{n-3}, s^{n-4}_{n-4}\rangle$ is the solution at level $n-4$ for the instance Σ_{n-2}, so it follows from the induction hypothesis that it will be refined into a ground solution of length L_{n-2}. It follows that $L_n = 2 + L_{n-2}$ for even $n > 4$, which proves the claim and ends the induction.

The solution to the recursive equation is $L_n = n$, which proves the theorem. □

Theorem 4.3 The shortest solution HPLAN can generate for Σ_n under $\mathcal{H}_2$ is of size $\Omega(2^{\frac{n}{2}})$.

Proof: Let L_n be the length of the shortest plan HPLAN can produce under $\mathcal{H}_2$. We prove by induction over n that for even $n > 0$,

$$L_n = \begin{cases} 2, & \text{for } n = 2, \\ 2 + 2L_{n-2} & \text{for } n > 2. \end{cases}$$

Base step: For $n = 2$ all operators have empty preconditions, so the behaviour of HPLAN will correspond to the two uppermost levels of Figure 3. The resulting plan is $\langle s_0, s_1\rangle$, which clearly must be a shortest plan. Hence, $L_2 = 2$.

Induction step: Assume the claim holds for all even $k < n$ for some even $n > 2$. For $n \geq 4$, planning on the four most abstract levels will proceed as shown in Figure 3. Analogous to the previous proof we see that the operators s_{n-1}^{n-4} and s_{n-2}^{n-4} will be refined into the single-operator plans $\langle s_{n-1}^{i} \rangle$ and $\langle s_{n-2}^{i} \rangle$, respectively, at each level $i < n-4$. Also by analogous reasoning, the atoms p_{n-1} and p_{n-2} can be ignored wrt. the expansions of the subplans $\langle s_{n-3}^{n-4}, s_{n-4}^{n-4} \rangle$ and $\langle r_{n-3}^{n-4}, r_{n-4}^{n-4} \rangle$. The first of these is clearly the solution at level $n-4$ for Σ_{n-2} and it, thus, follows from the induction hypothesis that it expands into a ground subplan of length L_{n-2}. Since the operators r_{n-3} and r_{n-4} have the same preconditions as s_{n-3} and s_{n-4} respectively, it is clear that the two subplans will have isomorphic refinements. Hence, the second subplan expands into a ground subplan of length L_{n-2}, too. It follows that $L_n = 2 + 2L_{n-2}$ for even $n > 4$, which proves the claim and completes the induction.

The solution to the recursive equation is $L_n = 2^{\frac{n}{2}+1} - 2$ so $L_n \in \Omega(2^{\frac{n}{2}})$, which proves the theorem. $\Box$

These results mean that if we happen to make a fortuitous choice of abstraction hierarchy, then HPLAN will generate a linear-size solution, using only a linear-size search space. On the other hand, if we are less fortunate, then HPLAN is forced to explore an exponential number of nodes generating an exponentially longer solution.

Clearly an unfortunate choice of abstraction hierarchy can force HPLAN to take exponential time, producing an exponentially suboptimal solution. It is thus interesting to compare this to the performance of a non-hierarchical planner. Such a planner may also have to explore an exponential-size search space. However, allowing the planner to search the whole, exponential-size search space would at least guarantee generating a shortest, *i.e.*linear-size, solution. Furthermore, a standard planner using a domain-independent standard heuristic can guarantee finding a solution exploring only a linear number of nodes in this case.

Theorem 4.4 SNLP [McAllester and Rosenblitt, 1991] solves Σ_n in polynomial time if equipped with a heuristic which prefers existing actions to new ones for goal establishment.

5 Building Abstraction Hierarchies

Knoblock [1994] defines a preorder $\sqsubseteq$ in the set of atoms and then uses this order to define an abstraction hierarchy satisfying the following restriction.

Restriction 5.1 Define $\sqsubseteq$ on $\mathcal{P}$ s.t. for all $p, p' \in \mathcal{P}$ and every $o \in \mathcal{O}$,

1. if $p, p' \in Gen(post(o))$ and $p \neq p'$, then $p \sqsubseteq p'$ and $p' \sqsubseteq p$;
2. if $p \in Gen(pre(o))$ and $p' \in Gen(post(o))$, then $p \sqsubseteq p'$.

For all atoms $p \in \mathcal{P}^i, p' \in \mathcal{P}^j$, if $p \sqsubseteq p'$, then $i \leq j$.

The intention is that if $p \sqsubseteq p'$, then p must not occur higher up in the abstraction hierarchy than p'. Restriction 5.1 is known to be a sufficient, though not necessary, condition for an abstraction hierarchy to be ordered monotonic [Knoblock, 1994].

Knoblock [1994] has further presented an algorithm, ALPINE, for generating maximally deep abstraction hierarchies satisfying Restriction 5.1, thus generating ordered abstraction hierarchies. The basic ALPINE algorithm appears in Figure 4.[1] The actual ALPINE algorithm [Knoblock, 1994] is somewhat more advanced and also comes equipped with certain heuristics. Further, it handles a first-order language, while our version is intended only for a propositional language. These differences do not affect the results to be proven in the following section, however—a topic which will be further discussed later in this paper.

ALPINE builds a directed graph, G, corresponding to the preorder $\sqsubseteq$ and then collapses all strong components in G, resulting in a set C of equivalence classes over $\mathcal{P}$. The final line of the algorithm sorts the partially ordered set C topologically, but does not specify any preference for a particular topological sort. Hence, ALPINE cannot always distinguish between good and bad abstraction hierarchies, such as $\mathcal{H}_1$ and $\mathcal{H}_2$.

Theorem 5.1 Given the planning instance Σ_n, ALPINE arbitrarily generates any of a number of possible abstraction hierarchies including $\mathcal{H}_1$ and $\mathcal{H}_2$.

Proof: The first step of ALPINE will produce the graph G in Figure 5 (corresponding to the preorder $\sqsubseteq$). Since there are no strong components of size > 1, step 2 will produce an isomorphic graph, with each element being a singleton component, inducing a partial order on the atoms. Finally, step 3 may produce any topological sorting of this partial order, which clearly includes the total orders reflected by the graphs T_1 and T_2 in Figure 5. Obviously, T_1 correspond to $\mathcal{H}_1$ and T_2 to $\mathcal{H}_2$, which proves the theorem. □

[1] Note that, contrary to Knoblock, we direct the arcs in the standard way.

procedure ALPINE($\mathcal{P}, \mathcal{O}$)
$G \leftarrow \langle \mathcal{P}, \varnothing \rangle$
for all $p, p' \in \mathcal{P}$ **do**
  if $p \sqsubseteq p'$ **then** insert arc $\langle p, p' \rangle$ in G
Collapse the strong components in G and let
$G' = \langle C, A \rangle$ be the reduced graph
return any topological sorting of A

Figure 4: The ALPINE *algorithm.*

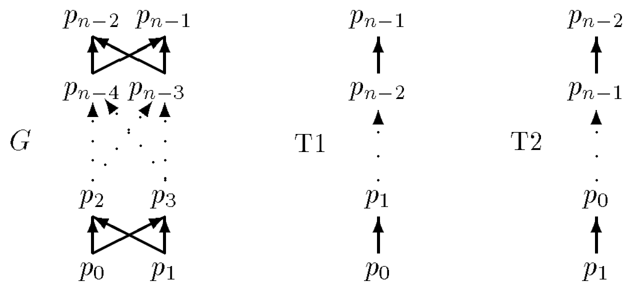

Figure 5: The preorder on $\mathcal{P}$ induced by the first step of the ALPINE *algorithm and two of the possible topological sortings of the reduced graph.*

What, then, are the chances of improving ALPINE by making a more informed choice in line 6? The implemented version comes equipped with certain heuristics [Knoblock, 1994, pp. 272–273], of which only one (number 3) applies to the propositional case. This heuristic specifies that adjacent levels not containing any goal literals should be merged into one single level. Applying this heuristic would cause the atoms $p_0, \ldots, p_{n-3}$ to end up on the same level.[2]

A modified version of the algorithm, HIGHPOINT [Bacchus and Yang, 1994], uses a sampling method to determine for each pair of components $c_i, c_j \in C$ that could be ordered the expected probability that a plan at level i can be refined at level j if ordering i above j. These probabilities are then used to further collapse some components and to guide the topological sorting of the remaining components. However, for Σ_n, HIGHPOINT will always find that the probability of refinement is 1, so it is provided with no extra information to guide the topological sorting. Hence, HIGHPOINT is bound to suffer from the same problem as ALPINE, *i.e.*, not being able to prefer $\mathcal{H}_1$ to $\mathcal{H}_2$.

This is hardly surprising, however, since it is possible to show that no modification or heuristic can improve the topological sorting to always allow HPLAN to produce shortest plans.

Definition 5.2 The search problem ALPGENMIN is defined as follows:
Instance: A planning instance $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$.
Problem: When executing the ALPINE algorithm on Π, find a topological sorting in the final step that results in an abstraction hierarchy which *allows* HPLAN to find a shortest solution.

Theorem 5.3 ALPGENMIN is NP-hard.

Proof: Proof by reduction from MINIMUM COVER [Garey and Johnson, 1979, p. 222], which is NP-complete. Let $X = \{x_1, \ldots, x_m\}$ be a set, let $C = \{C_1, \ldots, C_n\}$ be a set of subsets of X and let K be an integer. Without loss of generality we restrict the problem to having covers of even size only, by requiring that m is even and that the atoms x_{2i} and x_{2i+1} always appear together in members of C. Define a planning instance $\Pi = \langle \mathcal{P}, \mathcal{O}, \varnothing, \{p\} \rangle$

[2] Knoblock [personal comm., 1995] argues that this is the right behaviour in this case. However, our Theorems 4.2 and 4.3 would also hold under heuristic 3 if setting $\mathcal{G} = \mathcal{P}$ in Σ_n.

where $\mathcal{P}$ is partitioned into the three sets $\mathcal{P}_{top} = \{p\}$, $\mathcal{P}_{MC} = X \cup \{r\}$, and $\mathcal{P}_{fix} = \{q_0, \ldots, q_{K+1}\}$. The set of operators is similarly partitioned into three sets $\mathcal{O}_{top}$, $\mathcal{O}_{MC}$ and $\mathcal{O}_{fix}$, such that the operators in $\mathcal{O}_{top}$ change only atoms in $\mathcal{P}_{top}$ *etc.* The set $\mathcal{O}_{MC}$ contains one operator o_i for each member C_i of C, having no precondition and $C_i \cup \{r\}$ as its effect. The set $\mathcal{O}_{fix}$ contains one operator o_i for each atom $q_i \in \mathcal{P}_{fix}$, having no precondition and $\{q_0, q_i\}$ as its effect. Finally, $\mathcal{O}_{top}$ consists of the two operators o_{MC} and o_{fix}, both having the effect p and having the preconditions $\mathcal{P}_{MC}$ and $\mathcal{P}_{fix}$, respectively.

When applying ALPINE to Π, it will find the three maximal strong components $\mathcal{P}_{top}$, $\mathcal{P}_{MC}$ and $\mathcal{P}_{fix}$, the first being ordered above the two latter, which are mutually unordered. Hence, there are two possible abstraction hierarchies: $\mathcal{H}_{MC} = \langle \mathcal{P}_{top}, \mathcal{P}_{fix}, \mathcal{P}_{MC} \rangle$ and $\mathcal{H}_{fix} = \langle \mathcal{P}_{top}, \mathcal{P}_{MC}, \mathcal{P}_{fix} \rangle$. Obviously, under hierarchy $\mathcal{H}_{MC}$ it is possible to find a plan of lenght $K^* + 1$, where K^* is the size of the minimum cover for X. $\mathcal{H}_{fix}$, on the other hand, will force the planner to generate a plan of length $K + 2$, which is optimal iff $K^* > K$ (remember that K^* must be even). Now, if we could choose in polynomial time the hierarchy allowing us to find an optimal plan, then we could also solve MINIMUM COVER in polynomial time. Hence, ALPGENMIN is NP-hard. □

Note that the theorem is not about whether HPLAN *will* generate a shortest plan, but only about whether the abstraction hierarchy prevents it from doing so or not. This is a disappointing result since one of the conditions guaranteeing a linear-size search space for HPLAN is that HPLAN generates a shortest plan [Knoblock, 1991]. Knoblock mentions, however, that this condition can be relaxed; it is sufficient that HPLAN finds a plan of length within a constant factor longer than the shortest one. Unfortunately, ALPGENMIN cannot be approximated within any constant factor, unless P=NP. In fact, an even stronger approximation limit can be proven.

Theorem 5.4 ALPGENMIN cannot be asymptotically approximated within a factor $c \log_2 \frac{|\mathcal{P}|}{2}$ for any $c < \frac{1}{8}$ unless $NP \subseteq \mathrm{DTIME}(n^{\log \log n})$.

Proof sketch: Suppose the theorem were not true. Then it would follow from the construction in the proof of Theorem 5.3 that we could approximate MINIMUM COVER within $c \log_2 |X|$ for some $c < \frac{1}{8}$. However, this is impossible unless $NP \subseteq \mathrm{DTIME}(n^{\log \log n})$ [Bellare *et al.*, 1993], contradicting the assumption. □

We have previously required that the algorithm Plan underlying Hplan always generates optimal plans. Unfortunately, generating an optimal plan at an abstract level does not guarantee that we find an optimal plan at the ground level. This does not affect Theorems 5.3 and 5.4, however, since it is obvious from their proofs that neither theorem depends on the assumption that Plan generates an optimal plan.

6 Discussion

It is well-known that state abstraction can speed up planning exponentially [Knoblock, 1991]. Under certain ideal conditions, plans can be generated in linear time in the length of the solution for some planning problems, *e.g.* the Towers-of-Hanoi problem. However, the value of this demonstration is questionable since the problem is unrealistic in the sense that it has exponentially-sized minimal solutions.[3] One of these ideal conditions is the downward refinement property (DRP), which guarantees that no backtracking occurs between abstraction levels. We have added to previous analyses of state abstraction by showing that not only can state abstraction achieve exponential speed-up in some cases, it can also cause exponential slow-down in other cases—even for hierarchies satisfying the DRP. More precisely, there exist problem instances such that the ideal choice of abstraction hierarchy leads to the generation of a linear-size plan, while a more unfortunate choice forces the generation of an exponential-size plan, taking exponentially longer time to generate. This may even happen in cases where a standard non-hierarchical planner equipped with a simple, domain-independent heuristic produces a shortest, *i.e.* linear-size, solution in polynomial time. Instances of this kind seem no less realistic than, for instance, Towers-of-Hanoi.

We have further shown that the Alpine [Knoblock, 1994] and Highpoint [Bacchus and Yang, 1994] algorithms for generating abstraction hierarchies are not able to distinguish between such good and bad hierarchies as mentioned above. Furthermore, we have also shown that it is even impossible to design an algorithm based on the same underlying principle as Alpine and Highpoint that always produces hierarchies allowing a hierarchical planner to generate plans of length within a constant factor of the shortest length (actually, not even within a logarithmic factor in the size of the instance). We have chosen in this paper to concentrate on state ab-

[3] See Bäckström and Nebel [1993] or [Garey and Johnson, 1979, pp. 11–12] for a discussion of this topic.

straction as defined and used by Knoblock [1994], *i.e.* using a total-order hierarchical planner. We are currently investigating the consequences of using a partial-order hierarchical planner like AbTweak [Yang and Tenenberg, 1990] instead. Although AbTweak seems to handle correctly the particular example we have used to demonstrate the exponential slow-down effect, we do not believe there is any fundamental difference in general. In fact, the approximation result mentioned above should also be valid for partial-order planners like AbTweak.

The message of this paper is not that state abstraction and the use of algorithms such as Alpine and Highpoint should be abandoned; in many cases, these can still be powerful tools for tackling the search complexity in planning. However, the results tell us that we must be very careful; state abstraction is a powerful tool, but a tool that may occasionally turn its power against us, making things exponentially worse. Even if good abstraction hierarchies exist in many domains, the task of finding these is non-trivial and seems to remain a highly domain-dependent heuristic endeavour. We believe that more research is needed in order to understand when state abstraction works and how to exploit the inherent structure of problems for building good abstraction hierarchies.

Acknowledgements

We would like to thank Craig Knoblock, Jalal Maleki, Qiang Yang and the anonymous referees for comments which helped improve this paper.

References

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

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

[Bäckström and Nebel, 1993] Christer Bäckström and Bernhard Nebel. Complexity results for SAS$^+$ planning. In Ruzena Bajcsy, editor, *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 1430–1435, Chambéry, France, August–September 1993. Morgan Kaufmann.

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

[Bellare *et al.*, 1993] M. Bellare, S. Goldwasser, C. Lund, and A. Russel. Efficient probabilistically checkable proofs and applications to approximation. In *25th ACM Symposium on Theory of Computing (STOC-93)*, pages 294–304, San Diego, CA, May 1993. ACM.

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

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

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

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

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

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

[Yang and Tenenberg, 1990] Qiang Yang and Josh D Tenenberg. ABTWEAK: Abstracting a nonlinear, least commitment planner. In *Proceedings of the 8th (US) National Conference on Artificial Intelligence (AAAI-90)*, pages 204–209, Boston, MA, USA, August 1990. American Association for Artificial Intelligence, MIT Press.