ECAI 2016
G.A. Kaminka et al. (Eds.)

doi:10.3233/978-1-61499-672-9-716

Upper and Lower Time and Space Bounds for Planning

Christer Bäckström and **Peter Jonsson** [1]

Abstract. There is an extensive literature on the complexity of planning, but explicit bounds on time and space complexity are very rare. On the other hand, problems like the constraint satisfaction problem have been thoroughly analysed in this respect. We provide a number of upper and lower bound results for both plan satisfiability (PSAT) and length-optimal planning (LOP), with an emphasis on monotone planning (where actions have only positive effects) which is used in, for instance, h^+ and similar heuristics. Let v and a be the number of variables and actions, respectively. We consider both restrictions on the number and polarity of preconditions and effects of actions and the PUBS restrictions in SAS$^+$. For all such classes, we show that PSAT and LOP is either tractable or cannot be solved in subexponential time $2^{o(v)}$ or time $2^{o(a)}$, unless the so-called *Exponential Time Hypothesis (ETH)* is false. There is also a sharp transition: monotone LOP can be solved in time $2^{o(v)}$ if $a \in o(\frac{v}{\log v})$ but not if $a \in \Omega(v)$. We also study upper bounds and discuss the trade-off between time and space, providing a polynomial-space algorithm for monotone LOP that beats depth-first search in most cases. This raises the important question how lower bounds are affected by polynomial space restrictions.

1 INTRODUCTION

The computational complexity of the plan satisfiability problem (PSAT) and the plan-length optimisation problem (LOP) is well-studied in the literature. Bylander [8] analysed subclasses of both problems based on restricting the preconditions and effects of actions in the STRIPS language. Bäckström and Nebel [6] made a similar study for the PUBS restrictions in the SAS$^+$ langugage. The complexity of cost-optimal planning (COP) has also been studied (cf. Katz and Domshlak [28]). More recently, parameterised complexity analysis has been used. Bäckström et al. [5] analysed LOP for a number of subclasses using plan length as parameter, Kronegger et al. [30] used many parameters, including plan length, analysing the complexity for combinations of these parameters, and Aghighi and Bäckström [3] analysed COP using plan cost as parameter. Many more examples can be found in the literature. However, they all have in common that they classify problems into complexity classes, rather than providing any explicit time bounds.

While explicit upper bounds can be provided by demonstrating algorithms, it is more difficult to prove non-trivial lower bounds. An important step forward was the Exponential Time Hypothesis (ETH) [22], which conjectures that k-SAT cannot be solved in subexponential time $2^{o(n)}$, where n is the number of variables. This has proven a very useful hypothesis since there is a large number of **NP**-complete problems that are related in the sense that either all of them can be solved in subexponential time or none of them can. Hence, proving that a problem cannot be solved in subexponential time under the assumption that the ETH holds is a very strong indication of hardness. While all **NP**-complete problems are equivalent under the theory of **NP**-completeness, it is known that they differ widely in hardness in practice. The ETH has enabled to separate the **NP**-complete problems with respect to concrete time bounds, which is more fine grained and better related to practice than the usual classifications into complexity classes. The ETH is nowadays a standard assumption in complexity theory [31].

The ETH, and similar assumptions, was recently used to analyze the constraint satisfaction problem (CSP) [13, 27]. This showed that if the ETH is true, CSP cannot be solved in subexponential time even for a large number of common restrictions, although some special cases were identified where subexponential algorithms exist. CSP is a very important **NP**-complete problem. Apart from its widespread use in AI and elsewhere, it is also an archetypical **NP**-complete problem in the sense that many other **NP**-complete problems can easily be modelled as CSP classes. Planning is similarly a good and natural modelling language for many problems in **PSPACE**, and in **NP**, but no similar analysis of lower bounds exists for planning.

We address the issue of explicit upper and lower bounds for planning, with an emphasis on monotone planning (where actions have only positive effects). One reason for the latter is that monotone planning is **NP**-complete, while general planning is **PSPACE**-complete, so it is a stronger result to prove that not even monotone planning can be solved in subexponential time. Another reason is that monotone planning is important for many heuristics, like h^+ [21].

First, we derive some straightforward upper bounds in Sec. 3, both for monotone and non-monotone planning, in order to put the forthcoming lower-bound results into a perspective. We then turn to lower bounds, using restrictions on the number and polarity of preconditions and effects of actions (cf. Bylander [8]). Let v and a be the number of variables and actions, respectively. In Sec. 4, we give a complete classification of PSAT for these restrictions in the sense that each such class is either tractable or not solvable in subexponential time in the number of variables or actions, i.e. in time $2^{o(v)}$ or $2^{o(a)}$, (unless the ETH is false). We then do similar analyses of the LOP problem for the same type of restrictions. In Sec. 5, we focus on monotone planning and show that not even severly restricted classes can be solved in time $2^{o(v)}$ or $2^{o(a)}$ (unless the ETH is false). For the general non-monotone case, we show a sharper result in Sec. 6 based on graph colouring. If LOP can be solved faster than time $2^{\frac{v}{2}} \cdot poly(v)$, then there would be a faster algorithm for graph colouring than currently known. We then show in Sec. 7 that there exists a sharp transition: Monotone LOP cannot be solved in time $2^{o(v)}$ if $a \in \Omega(v)$, i.e. the number of actions is at least linear in the number of variables, but it can be solved in time $2^{o(v)}$ if the number of actions is sublinear in the number of variables. This resembles a similar transition result for CSP [13]. Then we consider lower bounds for other

[1] Department of Computer and Information Science, Linköping University, Sweden. Email: christer.backstrom@liu.se, peter.jonsson@liu.se

types of restrictions. In Sec. 8 we consider the PUBS restrictions (cf. Bäckström and Nebel [6]) and provide complete classifications for both PSAT and LOP, in the sense that all combinations of restrictions are either tractable or cannot be solved in time $2^{o(v)}$ (unless the ETH is false). We also settle an open question and prove **NP**-hardness for those combinations that have remained unclassified in the literature. For the last type of lower bound results, we consider planning classes defined by the structure of the causal graph (cf. Katz and Domshlak [29] and Giménez and Jonsson [19]). We show that even if restricted to a number of simple types of causal graphs, including several types studied in the literature, planning cannot be solved in time $2^{o(v)}$ (unless the ETH is false). After that, we once again consider upper bounds in Sec. 10, and, in particular, we consider the trade-off between time and space. While the previously derived upper bounds are based on algorithms requiring exponential space, we ask how fast we can plan using only polynomial space. For monotone planning, depth-first search satisfies this criterion, since the depth is limited, but the time bound increases with the branching factor. We provide an alternative polynomial-space algorithm which outperforms depth-first search except for instances with small branching factor. This also raises an important open question: How would lower-bound results be affected by a restriction to polynomial space?

2 PRELIMINARIES

For a set or sequence X of objects, we write $|X|$ to denote the cardinality (the number of objects) of X and we write $||X||$ to denote the size (the number of bits of the representation) of X.

2.1 Planning

In the general case, we will use the SAS$^+$ planning framework [6], which uses variables with arbitrary finite domain. Let $V = \{v_1, \ldots, v_n\}$ be a finite set of *variables*, with an implicit order $v_1, \ldots, v_n$, each with a finite *domain* $D(v_i)$. This defines the *state space* $S(V) = D(v_1) \times \ldots \times D(v_n)$. A member $s \in S(V)$ is called a *(total) state* and can be viewed as a total function that specifies a value in D for each $v_i \in V$. A *partial state* may leave the value undefined for some (or all) variables, and is thus a partial function. The value of a defined variable v_i in a (total or partial) state s is called the *projection* of s onto v_i and is denoted $s[v_i]$. If s is a partial state, then $\mathrm{vars}(s)$ is the set of variables with a defined value in s.

A *planning instance* $\mathbb{P} = \langle V, A, I, G\rangle$ has a set of variables V over the domain D, a set of *actions* A, a total *initial state* I and a partial *goal state* G. Each action $a \in A$ has a *precondition* $\mathrm{pre}(a)$ and an *effect* $\mathrm{eff}(a)$, both partial states. Let $a \in A$ and $s \in S(V)$. Then a is *valid in* s if $\mathrm{pre}(a)[v] = s[v]$ for all $v \in \mathrm{vars}(\mathrm{pre}(a))$, and the *result of* a *in* s is a state $t \in S(V)$ such that for all $v \in V$, $t[v] = \mathrm{eff}(a)[v]$ if $v \in \mathrm{vars}(\mathrm{eff}(a))$ and $t[v] = s[v]$ otherwise. Let $s_0, s_\ell \in S(V)$ and let $\omega = a_1, \ldots, a_\ell$ be a sequence of actions. Then ω is a *plan from* s_0 *to* s_ℓ if either (1) $\omega = \langle\rangle$ and $\ell = 0$ or (2) there are states $s_1, \ldots, s_{\ell-1} \in S(V)$ such that for all i ($1 \le i \le \ell$), a_i is valid in s_{i-1} and s_i is the result of a_i in s_{i-1}. Furthermore, ω is a *plan (i.e. a solution) for* $\mathbb{P}$ if it is a plan from I to G.

For every class C of SAS$^+$ instances, we define the following two problems.

PLAN SATISFIABILITY $\big(\mathrm{PSAT}(C)\big)$
Instance: An instance $\mathbb{P} = \langle V, A, I, G\rangle$ in C.
Question: Does $\mathbb{P}$ have a plan?

LENGTH-OPTIMAL PLANNING $\big(\mathrm{LOP}(C)\big)$
Instance: An instance $\mathbb{P} = \langle V, A, I, G\rangle$ in C and a non-negative integer k.
Question: Does $\mathbb{P}$ have a plan ω of length $|\omega| \le k$?

Most of our results, in particular the lower-bound results, only make use of binary variables. In these cases, it is often clearer and more convenient to use Propositional STRIPS with Negative goals (PSN) [8], which can be viewed as a different way to define SAS$^+$ with binary variables. Let V be a set of binary *variables*, i.e. propositional atoms. For any set $V' \subseteq V$, the set of *literals* over V' is $L(V) = \{v, \overline{v} \mid v \in V\}$. A *total state* s over V is a subset $s \subseteq V$, where a variable v is true in s if and only if $v \in s$. The *space of total states* over V is $S(V) = 2^V$. A *partial state* p over V is a consistent subset $p \subseteq L(V)$, i.e. it does not contain both v and $\overline{v}$ for any $v \in V$. A variable v is true in p if $v \in p$, false if $\overline{v} \in p$ and undefined if $p \cap \{v, \overline{v}\} = \varnothing$. We also define $p^+ = \{v \in V \mid v \in p\}$ and $p^- = \{v \in V \mid \overline{v} \in p\}$. Let p be a partial state and s a total state. Then p is *satisfied* in s, denoted $p \sqsubseteq s$ if both $p^+ \subseteq s$ and $p^- \cap s = \varnothing$. The $\ltimes$ *operator* is defined as $s \ltimes p = (s \setminus p^-) \cup p^+$. Finally, $\mathrm{vars}(p) = \{v \mid v \in p \text{ or } \overline{v} \in p\}$.

A PSN *instance* is a tuple $\mathbb{P} = \langle V, A, I, G\rangle$ where V is a set of variables, A is a set of *actions*, the *initial state* I is a total state over V and the *goal* G is a partial state over V. Each action a in A has a precondition $\mathrm{pre}(a)$ and an effect $\mathrm{eff}(a)$, which are both partial states. For all total states s, t over V and all $a \in A$, a is *from* s *to* t if both (1) $\mathrm{pre}(a) \sqsubseteq s$ and (2) $t = s \ltimes \mathrm{eff}(a)$. A sequence $\omega = a_1, \ldots, a_\ell$ of actions in A is a *plan* from a state s_0 to a state s_ℓ if either (1) $s_0 = s_\ell$ and ω is the empty sequence or (2) there are total states $s_1, \ldots, s_{\ell-1}$ such that a_i is from s_{i-1} to s_i for all i ($1 \le i \le \ell$). The sequence $s_0, \ldots, s_\ell$ is the *state sequence* of ω. A solution for $\mathbb{P}$ is a plan from I to some total state s such that $G \sqsubseteq s$. A solution for $\mathbb{P}$ is called a *plan* for $\mathbb{P}$.

We write $a : P \Rightarrow E$ to define an action a with precondition P and effect E. We define PSN (and later SAS$^+$) subclasses based on the number and polarity of the preconditions and effects, e.g. PSN_1^{2+} denotes the class of PSN instances where the actions have at most two positive literals in the precondition and one literal in the effect. We use $*$ to denote an unrestricted number of literals, i.e. PSN_1^* allows any number of literals in the preconditions but only one literal in the effects. A PSN instance is *monotone* if no action has any negative effects, i.e. all monotone instances belong to PSN_{*+}^*. Note that $\mathrm{PSN}_*^* = \mathrm{PSN}$.

2.2 Satisfiability and the ETH

The k-SAT problem is defined as follows and it is known to be **NP**-complete for $k \ge 3$ [18].

k-SAT
Instance: A CNF formula $\mathbb{F}$ where each clause has at most k literals.
Question: Does $\mathbb{F}$ have a satisfying assignment?

We will use n for the number of variables and m for the number of clauses of k-SAT instances. A more precise complexity characterization is possible by using the *Exponential Time Hypothesis (ETH)* [22].

Definition 1 For all constant integers $k \ge 3$, let s_k be the infimum of all real numbers δ such that k-SAT can be solved in time $O(2^{\delta n})$. The ETH says that $s_k > 0$ for all $k \ge 3$.

Somewhat informally, the ETH says that k-SAT cannot be solved in subexponential time $2^{o(n)}$.

If the ETH holds, then for every $k \geq 3$, there is some constant c_k such that k-SAT cannot be solved in time $2^{c_k n}$. The ETH is a quite strong assumption that allows for defining a theory similar to the one of **NP**-completeness. There is a large number of **NP**-complete problems that form a completeness class in the sense that either all of them can be solved in subexponential time, or none of them can [24]. There is also strong variant of the hypothesis (SETH) that additionally conjectures that $\lim_{k\to\infty} s_k = 1$.

We will frequently make use of the following result, that it is sufficient to assume a linear number of clauses in the instances.

Lemma 2 (de Haan et al. [13, Lemma 1]) k-SAT ($k \geq 3$) is solvable in time $2^{o(n)}$ if and only if k-SAT with a linear number of clauses and in which the number of occurences of each variable is at most 3 is solvable in time $2^{o(n)}$.

3 UPPER BOUNDS

In order to set the forthcoming lower-bound results into a perspective we first derive some straightforward upper bounds for planning. Consider the general case, a SAS$^+$ instance with v variables, each with a domain of size d. Then the state space consists of d^v states. Each state can have an arc to every state, including itself, so the maximum number of arcs in the state-transition graph is $d^v \cdot d^v = d^{2v}$. A straightforward way to solve LOP is to use Dijkstra's algorithm. Fredman and Tarjan's [16] variant of Dijkstra's algorithm runs in time $O(|E| + |V| \log |V|)$. A planning instance $\mathbb{P}$ can be viewed as a compact representation (in the sense of Galperin and Wigderson [17]) of its state-transition graph, where we can check in time $poly(||\mathbb{P}||)$ if an arc exists. It follows that we can solve LOP(SAS$^+$) in time $O\big((d^{2v} + d^v \log d^v) \cdot poly(||\mathbb{P}||)\big) = O\big(d^{2v} \cdot poly(||\mathbb{P}||)\big)$. Heuristic search algorithms may be preferrable in many practical cases, but they give no advantage in the worst case, e.g. Dijkstra's algorithm is essentially equivalent to the A* algorithm [15].

For PSN we have $d = 2$, so the state space is of size 2^v and can have up to $2^{2v} = 4^v$ arcs, i.e. we can solve LOP(PSN) in time $O\big(4^v \cdot poly(||\mathbb{P}||)\big)$. We then proceed to the monotone case, when the actions have no negative effects, i.e. PSN$^*_{*+}$. Then there can only be an arc from s to t if $s \subseteq t$. For each i, there are $\binom{v}{i}$ states of size i. For each such state, there are 2^i subsets. Hence, the maximum number of arcs is

$$e(v) = \sum_{i=1}^{v} \binom{v}{i} 2^i = \sum_{i=0}^{v} \binom{v}{i} 2^i - 1.$$

Using the binomial formula

$$(a+b)^v = \sum_{i=0}^{v} \binom{v}{i} a^{v-i} b^i$$

and setting $a = 1$ and $b = 2$ we get

$$\sum_{i=0}^{v} \binom{v}{i} 2^i = \sum_{i=0}^{v} \binom{v}{i} 1^{v-i} 2^i = (1+2)^v = 3^v$$

i.e. $e(v) < 3^v = 2^{v \log 3}$.

Observation 3 LOP(PSN$^*_{*+}$) can be solved in time $O\big(3^v \cdot poly(||\mathbb{P}||)\big)$ using Dijkstra's algorithm.

4 SUBEXPONENTIAL LOWER BOUNDS FOR PSAT

Both problems PSAT(PSN$^1_{1+}$) and PSAT(PSN$^{1+}_2$) are known to be **NP**-hard [8]. We will strengthen these results below by also proving that neither can be solved in subexponential time (in either the number of variables or the number of actions), unless the ETH is false.

Construction 4 Let $\mathbb{F}$ be a 3-SAT instance with variables $x_1, \ldots, x_n$ and clauses $c_1, \ldots, c_m$, where each clause c_j is of the form $\{l^1_j, l^2_j, l^3_j\}$. Construct a corresponding PSAT(PSN$^1_{1+}$) instance $\mathbb{P} = \langle V, A, I, G\rangle$ as follows:

- $V = \{f_i, t_i \mid 1 \leq i \leq n\} \cup \{y_j \mid 1 \leq j \leq m\}$;
- A contains the actions
  - $setf_i : \{\overline{t_i}\} \Rightarrow \{f_i\}$, for all i ($1 \leq i \leq n$),
  - $sett_i : \{\overline{f_i}\} \Rightarrow \{t_i\}$, for all i ($1 \leq i \leq n$) and
  - $vfy^k_j : \{\hat{l}^k_j\} \Rightarrow \{y_j\}$, for all j, k ($1 \leq j \leq m$, $1 \leq k \leq 3$), where $\hat{l}^k_j = f_i$ if $l^k_j = \overline{x_i}$ and $\hat{l}^k_j = t_i$ if $l^k_j = x_i$;
- $I = \varnothing$ and $G = \{y_1, \ldots, y_m\}$.

Theorem 5 If PSAT(PSN$^1_{1+}$) can be solved in time $2^{o(v)}$ or time $2^{o(a)}$, then the ETH is false.

Proof. Proof by reduction from 3-SAT to PSAT(PSN$^1_{1+}$). Let $\mathbb{F}$ be a 3-SAT instance and let $\mathbb{P}$ be the corresponding PSN instance according to Construction 4. For each variable x_i in $\mathbb{F}$, a plan for $\mathbb{P}$ can set either f_i or t_i to true, but not both. It follows that $\mathbb{P}$ has a plan if and only if $\mathbb{F}$ is satisfiable, so Construction 4 is a polynomial reduction from 3-SAT to PSAT(PSN$^1_{1+}$).

Let S_d denote the class of 3-SAT instances where $m \leq dn$ and choose d such that S_d cannot be solved in time $2^{o(n)}$ if the ETH is true. Such a d exists according to Lemma 2. Let P_d be the class of PSN$^1_{1+}$ instances we can get by applying Construction 4 to S_d.

Suppose PSAT(PSN$^1_{1+}$) can be solved in time $2^{o(v)}$. Choose an arbitrary $c > 0$. Then PSAT(PSN$^1_{1+}$) can be solved in time in 2^{cv} for large v. Let $\mathbb{F}$ be a 3-SAT instance in S_d with n variables. Then $\mathbb{F}$ has $m \leq dn$ clauses. Let $\mathbb{P}$ be the corresponding PSN instance. We have $v = 2n + m \leq 2n + dn = (2+d)n$ and $||\mathbb{P}||$ is polynomial in n, so it follows from our assumption that we can solve satisfiability for S_d in time $poly(n) + 2^{c(2+d)n} \leq 2^{(c(2+d)+\epsilon)n}$, for all $\epsilon > 0$ and large n. However, c is arbitrary so we can choose arbitrary $c', \epsilon > 0$ such that $c(2+d) + \epsilon \leq c'$ and S_d can be solved in time $2^{c'n}$ for large n. Unless the ETH is false, this contradicts our assumptions and it follows that PSAT(PSN$^1_{1+}$) cannot be solved in time $2^{o(v)}$.

We further have $a = 2n + 3m$, i.e. $v \in O(a)$, so PSAT(PSN$^1_{1+}$) cannot be solved in time $2^{o(a)}$ unless the ETH is false. □

Construction 6 Let $\mathbb{F}$ be a 3-SAT instance with variables $x_1, \ldots, x_n$ and clauses $c_1, \ldots, c_m$. Construct a corresponding PSAT(PSN$^{1+}_2$) instance $\mathbb{P} = \langle V, A, I, G\rangle$ as follows:

- $V = \{e_i, f_i, t_i \mid 1 \leq i \leq n\} \cup \{y_j \mid 1 \leq j \leq m\}$;
- A contains the actions
  - $setf_i : \{e_i\} \Rightarrow \{\overline{e_i}, f_i\}$, for all i ($1 \leq i \leq n$),
  - $sett_i : \{e_i\} \Rightarrow \{\overline{e_i}, t_i\}$, for all i ($1 \leq i \leq n$) and
  - $vfy^k_j : \{\hat{l}^k_j\} \Rightarrow \{y_j\}$, for all j, k ($1 \leq j \leq m$, $1 \leq k \leq 3$), where $\hat{l}^k_j = f_i$ if $l^k_j = \overline{x_i}$ and $\hat{l}^k_j = t_i$ if $l^k_j = x_i$;
- $I = \{e_1, \ldots, e_n\}$ and $G = \{y_1, \ldots, y_m\}$.

Theorem 7 If $\text{PSAT}(\text{PSN}_2^{1+})$ can be solved in time $2^{o(v)}$ or time $2^{o(a)}$, then the ETH is false.

Proof. Analogous to the proof of Theorem 5, but using Construction 6 instead. For each variable x_i in $\mathbb{F}$, a plan for $\mathbb{P}$ can set either f_i or t_i to true, but not both since we can never set e_i to true again once it is reset. Hence, this is a polynomial reduction from 3-SAT to $\text{PSAT}(\text{PSN}_2^{1+})$ such that $v = 3n + m$ and $a = 2n + 3m$. □

It is further known that the problems $\text{PSAT}(\text{PSN}_{*+}^{*+})$, $\text{PSAT}(\text{PSN}_*^0)$ and $\text{PSAT}(\text{PSN}_1^{*+})$ can be solved in polynomial time [8]. Hence, we have a complete classification of all PSN classes defined by the number and polarity of preconditions and effects of actions, each being classified as either tractable or not solvable in subexponential time.

5 LOWER BOUNDS FOR MONOTONE LOP

We start with LOP for monotone instances. Finding a length-optimal plan is hard even when the actions are restricted to only one positive precondition and one positive effect.

Construction 8 Let $\mathbb{F}$ be a 3-SAT instance with variables $x_1, \ldots, x_n$ and clauses $c_1, \ldots, c_m$, where each clause c_j is on the form $\{l_j^1, l_j^2, l_j^3\}$. Construct a corresponding PSN_{1+}^{1+} instance $\mathbb{P} = \langle V, A, I, G\rangle$ as follows:

- $V = \{f_i, t_i, s_i \mid 1 \leq i \leq n\} \cup \{y_j \mid 1 \leq j \leq m\}$;
- A contains the actions
  $set_i^f : \varnothing \Rightarrow \{f_i\}$, $set_i^t : \varnothing \Rightarrow \{t_i\}$, $sets_i^f : \{f_i\} \Rightarrow \{s_i\}$ and
  $sets_i^t : \{t_i\} \Rightarrow \{s_i\}$, for all i $(1 \leq i \leq n)$, and
  $vfy_j^k : \{\hat{l}_j^k\} \Rightarrow \{c_j\}$, for all j, k $(1 \leq j \leq m, 1 \leq k \leq 3)$, where $\hat{l}_j^k = x_i^f$ if $l_j^k = \overline{x_i}$ and $\hat{l}_j^k = x_i^t$ if $l_j^k = x_i$;
- $I = \varnothing$ and $G = \{s_1, \ldots, s_n\} \cup \{y_1, \ldots, y_m\}$.

Theorem 9 If $\text{LOP}(\text{PSN}_{1+}^{1+})$ can be solved in time $2^{o(v)}$ or time $2^{o(a)}$, then the ETH is false.

Proof. Proof by reduction from 3-SAT to $\text{LOP}(\text{PSN}_{1+}^{1+})$. Let $\mathbb{F}$ be a 3-SAT instance with variables $x_1, \ldots, x_n$ and clauses $c_1, \ldots, c_m$, where each clause c_j is of the form $\{l_j^1, l_j^2, l_j^3\}$. Let $\mathbb{P}$ be the corresponding PSN instance according to Construction 8. We claim that $\mathbb{F}$ is satisfiable if and only if $\mathbb{P}$ has a plan of length $2n + m$.

$\Rightarrow$: Suppose α is a satisfying assignment for $\mathbb{F}$. Construct a plan ω as follows. For each x_i, let ω contain set_i^f and $sets_i^f$ if $\alpha(x_i) = 0$ and otherwise let ω contain set_i^t and $sets_i^t$. Then, for each clause $c_j = \{l_j^1, l_j^2, l_j^3\}$, there is at least one k such that α makes l_j^k true. Choose such a k and add action vfy_j^k at the end of ω. Clearly, ω is a plan for $\mathbb{P}$ of length $2n + m$.

$\Leftarrow$: Suppose ω is a plan for $\mathbb{P}$ of length $2n + m$. It must contain n actions setting the s_i variables and m actions setting the y_j variables. In order to set the s_i variables, it must also set either of f_i and t_i for each i, but it cannot set both since there can only be n such actions in total. Hence, ω corresponds to a satisfying assignment.

It follows that the construction is a polynomial reduction.

Let S_d denote the class of 3-SAT instances where $m \leq dn$ and choose d such that S_d cannot be solved in time $2^{o(n)}$ if the ETH is true. Such a d must exist according to Lemma 2. Let P_d be the class of PSN_{1+}^{1+} instances we get by applying the reduction above to S_d.

Suppose $\text{LOP}(\text{PSN}_{1+}^{1+})$ can be solved in time in $2^{o(v)}$. Choose an arbitrary $c > 0$. Then $\text{LOP}(\text{PSN}_{1+}^{1+})$ can be solved in time 2^{cv} for large v. Let $\mathbb{F}$ be a 3-SAT instance in S_d with n variables. Then $\mathbb{F}$ has $m \leq dn$ clauses. Let $\mathbb{P}$ be the corresponding PSN instance. We have $v = 3n + m \leq 3n + dn = (3 + d)n$ and $||\mathbb{P}||$ is polynomial in n, so it follows from our assumption that we can solve the class S_d in time $poly(n) + 2^{c(3+d)n} \leq 2^{(c(3+d)+\epsilon)n}$, for all $\epsilon > 0$ and large n. However, c is arbitrary so we can also choose arbitrary $c', \epsilon > 0$ such that $c(3 + d) + \epsilon \leq c'$ and S_d can be solved in time $2^{c'n}$ for large n. Unless the ETH is false, this contradicts our assumptions and it follows that $\text{LOP}(\text{PSN}_{1+}^{1+})$ cannot be solved in time $2^{o(v)}$.

We further have $a = 4n + 3m$, i.e. $v \in O(a)$, so $\text{LOP}(\text{PSN}_{1+}^{1+})$ cannot be solved in time $2^{o(a)}$ unless the ETH is false. □

This lower bound for $\text{LOP}(\text{PSN}_{1+}^{1+})$ holds even if the number of actions is linear in the number of variables.

Theorem 10 If $a \in \Omega(v)$, then $\text{LOP}(\text{PSN}_{1+}^{1+})$ cannot be solved in time $2^{o(v)}$ unless the ETH is false.

Proof. In the proof of Theorem 9 we have $|V| = 3n + m \leq 3n + dn = (3 + d)n$ and $|A| = 4n + 3m \leq 4n + 3dn = (4 + 3d)n$. Hence, the class P_d satisfies that $a \in \Omega(v)$ and the proof works also in this case. □

We will see in Sec. 7 that this no longer holds if the number of actions is sublinear in the number of variables.

The previous theorem covers all cases of monotone planning, except when actions have no preconditions at all. In the case of no preconditions, we have to rely on a conjecture about the SET COVER problem, which is defined as follows:

k-SET COVER
Instance: A set S and a set C of subsets of S.
Question: Does S have a cover of size k, i.e. is there a subset $C' \subseteq C$ such that $\bigcup_{X \in C'} X = S$ and $|C'| \leq k$?

Cygan et al. [11] conjectured that k-SET COVER cannot be solved in time $2^{o(n)}$ unless the SETH is false, where $n = |S|$.

Theorem 11 $\text{LOP}(\text{PSN}_{k+}^0)$ cannot be solved in time $2^{o(v)}$ unless k-SET COVER can be solved in time $2^{o(n)}$.

Proof. Polynomial reduction from k-SET COVER. Given an instance $\mathbb{I} = \langle S, C\rangle$ of k-SET COVER, construct a $\text{LOP}(\text{PSN}_{k+}^0)$ instance $\mathbb{P} = \langle V, A, I, G\rangle$, where $V = S \cup \{y_c \mid c \in C\}$, A contains the action $a_c : \varnothing \Rightarrow \{x \mid x \in c\}$ for all $c \in C$ and all $x \in c$, $I = \varnothing$ and $G = S$. Clearly, $\mathbb{P}$ has a plan of length k if and only if $\mathbb{I}$ has a cover of size k. □

6 LOWER BOUNDS FOR GENERAL LOP

For the general case, we prove a sharper bound than for monotone LOP based on results about graph colouring.

GRAPH COLOURABILITY
Instance: A graph $G = \langle V, E\rangle$ and a positive integer $k \leq |V|$.
Question: Is G k-colourable, i.e. is there a function $f : V \to \{1, \ldots, k\}$ such that $f(u) \neq f(v)$ whenever $\{u, v\} \in E$?

The best known upper bound for GRAPH COLOURABILITY is time $2^n \cdot poly(n)$, where $n = |V|$, [7, Prop. 1], and it is considered an important open question whether a faster algorithm can exist [23]. Hence, we can use GRAPH COLOURABILITY instead of the ETH to prove sharper bounds, but under somewhat different assumptions. In particular, we show a sharper limit for planning with negative effects.

Theorem 12 If LOP(PSN) can be solved in time $2^{\frac{cv}{2}} \cdot poly(v)$ for some $c > 0$, then GRAPH COLOURABILITY can be solved in time $2^{cn} \cdot poly(n)$.

Proof. Proof by reduction from GRAPH COLOURABILITY to LOP(PSN). Let $\mathbb{I} = \langle G, k\rangle$ be an instance of GRAPH COLOURABILITY, where $G = \langle V, E\rangle$ is a graph and $k > 0$ is an integer. Assume $V = \{v_1, \ldots, v_n\}$. Construct a corresponding LOP(PSN) instance $\mathbb{I}' = \langle \mathbb{P}', k'\rangle$ as follows. Let $\mathbb{P}' = \langle V', A', I', G'\rangle$, where

- $V' = \{v_1, \ldots, v_n, b_1, \ldots, b_n\}$;
- A' contains the actions $a_s : \varnothing \Rightarrow \{\overline{b_1}, \ldots, \overline{b_n}\}$ and $a_i : \{\overline{b_i}\} \Rightarrow \{v_i\} \cup \{b_j \mid \{v_i, v_j\} \in E\}$ for all $v_i \in V$;
- $I' = \{\overline{b_1}, \ldots, \overline{b_n}, \overline{v_1}, \ldots, \overline{v_n}\}$ and $G' = \{v_1, \ldots, v_n\}$.

Let $k' = n + k - 1$.

A plan colours the vertices in phases, one colour in each phase. The phases are separated by occurences of action a_s, which switches to the next colour. Variable v_i is true if vertex v_i has been coloured and variable b_i is a blocking variable, preventing v_i from being coloured for the moment. The actual colour of a vertex is only implicit in the plan, and not explicitly represented. At the start of each phase, any node can be (re)coloured and colouring a node immediately blocks its neighbours from being coloured in the same phase.

We now claim that $\mathbb{I}$ is k-colourable if and only if $\mathbb{I}'$ has a plan of length k'.

$\Rightarrow$: Suppose G has a k-colouring. Then there is a partition $C_1, \ldots, C_k$ of V such that C_i is an independent set for all i. Create the action sequence $\omega = \omega_1, a_s, \omega_2, a_s, \ldots, a_s, \omega_k$, where ω_i contains action a_j for each $v_j \in C_i$ in arbitrary order. The initial state guarantees that all b_j variables are false at the start of sequence ω_1 and the a_s actions guarantee that all b_j variables are false at the start of sequence ω_i for each $i > 1$. Since all vertices in C_i have the same colour, there are no two $v_j, v_h \in C_i$ such that $\{v_j, v_h\} \in E$. Hence, no action in ω_i will set b_j for any $v_j \in C_i$. It follows that all actions in ω_i are valid. Furthermore, the a_s actions are always valid. Since each $v_j \in V$ occurs in some C_i, it follows that action a_j occurs somewhere in ω for each $v_j \in V$. Hence, the resulting state state satisfies G.

$\Leftarrow$: Suppose ω is a plan for $\mathbb{P}'$ of length k' or less. Without losing generality, assume ω is a shortest such plan. Then ω does not contain any successive occurences of action a_s, so it is of the form $\omega = \omega_1, a_s, \omega_2, a_s, \ldots, a_s, \omega_m$, for some m, where the subplans ω_i do not contain any occurences of action a_s. Since ω must contain at least one occurence of action a_j for each $v_j \in V$, it follows that $|\omega| \geq n+m-1$, i.e. $m \leq k$. Suppose there is some i and two actions a_j, a_h in ω_i such that $\{v_j, v_h\} \in E$. Without losing generality, assume v_j occurs before v_h. Then a_j sets b_h, but this blocks the execution of a_h. Hence, the assumption must be false and $\{v_j, v_h\} \notin E$ for all $a_j, a_h \in \omega_i$. It follows that G must have an m-colouring, and, thus, a k-colouring.

It follows that the construction is a polynomial reduction from GRAPH COLOURABILITY to LOP(PSN).

Now, suppose there is some $c > 0$ such that LOP(PSN) can be solved in time $2^{\frac{cv}{2}} \cdot poly(v)$. Since $|V'| = 2|V|$, we can solve GRAPH COLOURABILITY in time $2^{cn} \cdot poly(n)$. □

This is a sharper result than the previous ones for monotone planning in the following sense. If LOP(PSN) can be solved faster than time $2^{\frac{v}{2}} \cdot poly(v)$, then there is a faster algorithm for GRAPH COLOURABILITY than previously known, i.e. this result is based on an assumption about a specific fixed value for the constant in the exponent. Note that Theorem 9 still applies, i.e. LOP(PSN) cannot be solved in time $2^{o(v)}$ unless the ETH is false.

7 SUBEXPONENTIAL SOLVABILITY

We will now demonstrate three PSN classes that can be solved in subexponential time in the number of variables, if the number of actions is subexponential in the number of variables.

For the first class, we need the following lemma.

Lemma 13 LOP(PSN^{*+}_{*+}) can be solved in time $O(2^{|A|})$.

Proof. Enumerate all subsets of A. For each such subset A', we can apply the actions greedily until we either have a plan, or no more action is applicable. Since we try to find plan for each subset of A, we must find an optimal plan, so it is sufficient to keep track of the shortest plan found. Since there are $2^{|A|}$ subsets of A and each subset can be checked in polynomial time by the greedy strategy, it follows that we can solve LOP(PSN^{*+}_{*+}) in time $O(2^{|A|})$. □

Theorem 14 LOP(PSN^{*+}_{*+}) can be solved in time $2^{o(v)}$ if $a \in o(v)$.

Proof. Immediate from Lemma 13. □

There is a similarly sharp bound for actions with arbitrary preconditions, if we limit their effects to a constant number of variables.

Theorem 15 LOP(PSN^{*}_{k+}) can be solved in time $2^{o(v)}$ if $a \in o(v)$.

Proof. With a actions, we can set at most ka variables, so $v - ka$ variables are redundant and can be removed from the instance before solving it. Since $a \in o(v)$, we get $o(v)$ remaining variables. □

If allowing also arbitrary preconditions, we get a somewhat less sharp bound.

Lemma 16 Generating all plans of length ℓ, or less, can be done in time $O(\ell^{|A|+2}|V|^2)$.

Proof. For $|A| \geq 2$ and $\ell \geq 2$, there are at most

$$0^{|A|} + 1^{|A|} + \cdots + \ell^{|A|} = 1 + \sum_{i=1}^{\ell} i^{|A|} \leq \ell \cdot \ell^{|A|} = \ell^{|A|+1}$$

action sequences of length ℓ, or less. Each plan of length ℓ, or less, can be verified in time $O(\ell|V|^2)$. Hence, we can generate all plans of length ℓ, or less, in time $O(\ell^{|A|+1} \cdot \ell|V|^2) = O(\ell^{|A|+2} \cdot |V|^2)$. □

Theorem 17 If $a \in o(\frac{v}{\log v})$, then LOP($\text{PSN}^{*}_{*+}$) can be solved in time $2^{o(v)}$.

Proof. No action need to occur more than once in a plan for a monotone instance, so the maximum plan length is a. Hence, we know from Lemma 16 that we can generate all plans of length a, or less, in time $O(a^{a+2}v^2)$. Hence, we can solve LOP(PSN^{*}_{*+}) by keeping track of the shortest plan found.

It remains to prove that $a^{a+2}v^2 \in 2^{o(v)}$, but $a^{a+2}v^2 = a^a \cdot a^2v^2$ so we can show separately that $a^a \in 2^{o(v)}$ and $a^2v^2 \in 2^{o(v)}$.

We first prove that $a^a \in 2^{o(v)}$. Since $a \in o(\frac{v}{\log v})$, it holds for all $c > 0$ that $a < c\frac{v}{\log v}$, for large v. We get

$$a^a = 2^{a \log a} < 2^{(\frac{cv}{\log v}) \log \frac{cv}{\log v}} = 2^{cv \frac{\log \frac{cv}{\log v}}{\log v}}.$$

Choose an arbitrary $c' > 0$. We want to prove that there is a $c > 0$ such that

$$2^{cv\frac{\log \frac{cv}{\log v}}{\log v}} \leq 2^{c'v},$$

that is,

$$cv\frac{\log \frac{cv}{\log v}}{\log v} \leq c'v.$$

We rewrite to

$$cv\frac{\log c + \log v - \log\log v}{\log v} \leq c'v,$$

but

$$\lim_{v\to\infty} \frac{\log c + \log v - \log\log v}{\log v} = 1$$

so it is sufficient to choose $c = c'$, which is allowed since we only require that $c > 0$. It follows that $a^a \in 2^{o(v)}$, since c' was chosen arbitrarily. We must next prove that also $a^2v^2 \in 2^{o(v)}$. We know that $a \in o(v)$ so it holds for all $c > 0$ that $a < cv$, for large v. Choose $c = 1$. We get $a^2v^2 \leq v^2v^2 = v^4$ and it is straightforward that $v^4 \in 2^{o(v)}$. We have now shown that $a^a \in 2^{o(v)}$ and that $a^2v^2 \in 2^{o(v)}$, so it follows that $a^a \cdot a^2v^2 = a^{a+2}v^2 \in 2^{o(v)}$. □

8 THE PUBS RESTRICTIONS

For the SAS$^+$ language, it is common to consider classes defined by combinations of the following four restrictions on instances [6].

P (post-unique): For all $v \in V$ and $x \in D(v)$, $\text{eff}(a)[v] = x$ for at most one $a \in A$.

U (unary): For each $a \in A$, $|\text{vars}(\text{eff}(a))| = 1$.

B (binary): $|D(v)| = 2$ for all $v \in V$.

S (single-valued): For all $a, b \in A$ and $v \in V$, if $v \in \text{vars}(\text{pre}(a)) \cap \text{vars}(\text{pre}(b))$ and $v \notin \text{vars}(\text{eff}(a)) \cup \text{vars}(\text{eff}(b))$ then $\text{pre}(a)[v] = \text{pre}(b)[v]$.

Combinations of these restrictions are written by juxtaposing the corresponding letters, eg. SAS$^+$-PUB is the class of all SAS$^+$ instances that are post-unique, unary and binary. We will now prove subexponential lower-bound results for all non-tractable combinations of such restrictions. The following construction can be used to implement disjunctions in SAS$^+$-PUB.

Construction 18 (Bäckström et al. [5], proof of Lemma 2) An OR gate g with two inputs x_1, x_2 and output y can be encoded as a SAS$^+$-PUB instance $\mathbb{P} = \langle V, A, I, G\rangle$ as follows:

- $V = \{x_1, x_2, y, y_1, y_2, i_1, i_2\}$, all with domain $\{0, 1\}$,
- A contains the following actions:
  - $a_y : \{y_1 = 1, y_2 = 1\} \Rightarrow \{y = 1\}$,
  - $a_{y_1} : \{i_1 = 1, i_2 = 0\} \Rightarrow \{y_1 = 1\}$,
  - $a_{y_2} : \{i_1 = 0, i_2 = 1\} \Rightarrow \{y_2 = 1\}$,
  - $a_{i_1} : \{\varnothing\} \Rightarrow \{i_1 = 1\}$,
  - $a_{i_2} : \{\varnothing\} \Rightarrow \{i_2 = 1\}$,
  - $a_{v_1} : \{x_1 = 1\} \Rightarrow \{i_1 = 0\}$,
  - $a_{v_2} : \{x_2 = 1\} \Rightarrow \{i_2 = 0\}$.
- $I[x_1]$ and $I[x_2]$ are arbitrary and $I[v] = 0$ for all other $v \in V$.
- $G[y] = 1$ and G is otherwise undefined.

Theorem 19 If PSAT(SAS$^+$-PUB) can be solved in time $2^{o(v)}$ or time $2^{o(a)}$, then the ETH is false.

Proof sketch. Instance $\mathbb{P}$ in Construction 4 is a SAS$^+$-UB instance, but it is not post-unique since there are three actions with the same effect for each clause variable y_j. The three actions together simulate the disjunction in the clause. Construction 18 computes the logical OR of two variables in a PUB instance, using seven actions and four additional variables. This gadget has the property that it has no plan if both input variables are false, and otherwise it always has a plan of length six that sets the output variable. We can then compute the logical OR of four variables by using three such gadgets, using the outputs of the first two as inputs to the third. Since we only need three variables as input, we can use one of the variables for two inputs. We need three OR gadgets for each clause, and the outputs of the first two must be new variables. That is, we need $3 \cdot 4 + 2 = 14$ new variables for each clause, which yields a total of $2n + 15m$ variables for the instance. We also need $3 \cdot 7 = 21$ new actions for each clause, but the original three ones are not needed, so there are 18 actions per clause, which yields a total of $4n + 18m$ actions. The proof of Theorem 5 can easily be modified to this case. □

Corollary 20 If PSAT(SAS$^+$-PBS) can be solved in time $2^{o(v)}$ or time $2^{o(a)}$, then the ETH is false.

Proof. Immediate from Theorem 19 since there is a polynomial reduction from LOP(SAS$^+$-PUB) to LOP(SAS$^+$-PBS) that increases the number of variables by a factor 2 [6, Proof of Thm. 4.16]. □

It has remained an open question in the literature whether PSAT(SAS$^+$-PUB) and PSAT(SAS$^+$-PBS) are **NP**-hard, while the corresponding LOP problems are known to be **NP**-hard. Since the two preceeding proofs use polynomial reduction from 3-SAT we can settle this question affirmatively as a spin-off result.

Corollary 21 PSAT(SAS$^+$-PUB) and PSAT(SAS$^+$-PBS) are **NP**-hard.

It is further known that PSAT(SAS$^+$-US) is in **P** [6] and corresponding results for all other combinations of the PUBS restrictions follow trivially, so this is a complete classification for PSAT for all combinations of the PUBS restrictions.

All the hardness results for PSAT above immediately apply also to LOP, but the tractable cases are fewer for LOP, it is only known that LOP(SAS$^+$-PUS) is in **P** [6]. It is sufficient to add the following result to get a complete classification also for LOP.

Corollary 22 (To Theorem 9) If LOP(SAS$^+$-UBS) can be solved in time $2^{o(v)}$ or time $2^{o(a)}$, then the ETH is false.

9 CAUSAL GRAPHS

The *causal graph* of a planning instance describes certain types of variable dependencies of a planning instance, and has frequently been exploited for identifying easy subclasses or for classifying the complexity of planning classes [19, 20, 26, 29, 38].

Definition 23 The causal graph for a SAS$^+$ instance $\mathbb{P} = \langle V, A, I, G\rangle$ is the directed graph $CG(\mathbb{P}) = \langle V, E\rangle$ where for all $u, v \in V$, $\langle u, v\rangle \in E$ if and only if both $u \neq v$ and there is some $a \in A$ such that $u \in \text{vars}(\text{pre}(a)) \cup \text{vars}(\text{eff}(a))$ and $v \in \text{vars}(\text{eff}(a))$.

We first show a general hardness result for instances with quite restricted causal graphs.

Theorem 24 If $\mathrm{LOP}(\mathrm{PSN}^{1+}_{1+})$ can be solved in time $2^{o(v)}$ or time $2^{o(a)}$ for instances where the causal graph is acyclic, bipartite and has degree 3 and depth 2, then the ETH is false.

Proof. Consider the construction in the proof of Theorem 9. The causal graph contains contains the following arcs:

- $\langle f_i, s_i\rangle$ and $\langle t_i, s_i\rangle$ for all i $(1 \leq i \leq n)$;
- $\langle f_i, y_j\rangle$ for all i, j $(1 \leq i \leq n, 1 \leq j \leq m)$ such that $\overline{x_i} \in c_j$ and $\langle t_i, y_j\rangle$ for all i, j $(1 \leq i \leq n, 1 \leq j \leq m)$ such that $x_i \in c_j$.

This graph is bipartite and each y_j variable has at most 3 incoming arcs and no outgoing arcs. Similarily, each s_i variable has two incoming arcs and no outgoing arc. It also follows from Lemma 2 that we can restrict the class S_d in the proof of Theorem 5 to instances where each variable occurs at most 3 times. We can assume that the SAT instance is preprocessed so variables which occur with only one polarity are removed. Then each variable of type f_i or t_i has at most two outgoing arcs to variables of type y_j and one arc to variable s_i and no incoming arcs. It follows that the graph has degree 3. □

It can be analogously shown that also Theorems 5 and 7 hold when restricted to instances where the causal graph is acyclic bipartite of degree 3 and depth 2.

We then continue to some special types of causal graphs that have been studied in the literature: out-stars (aka. forks), in-stars (aka. inverted forks), directed-path graphs (aka. chains) and fences. It is known that PSAT remains **NP**-hard when restricted to instances having a causal graph of either of these types [4, 14, 19]. We can sharpen these results by the following explicit lower bounds.

Theorem 25 If $\mathrm{PSAT}(\mathrm{SAS}^{+}{}^{2}_{1})$ can be solved in time $2^{o(v)}$ or time $2^{o(a)}$ for instances where the causal graph is an out-star, then the ETH is false.

Proof sketch. There is a polynomial reduction from 3-SAT to $\mathrm{PSAT}(\mathrm{SAS}^{+}{}^{2}_{1})$ with out-star causal graphs and $m + 1$ variables [4, Lemma 5]. This can be used to make a proof analogous to the one for Theorem 5. □

Theorem 26 If $\mathrm{PSAT}(\mathrm{SAS}^{+}{}^{1}_{1})$ can be solved in time $2^{o(v)}$ or time $2^{o(a)}$ for instances where the causal graph is an in-star, then the ETH is false.

Proof sketch. There is a polynomial reduction from 3-SAT to $\mathrm{PSAT}(\mathrm{SAS}^{+}{}^{1}_{1})$ with in-star causal graphs and n variables [4, Lemma 4]. □

Theorem 27 If $\mathrm{PSAT}(\mathrm{SAS}^{+}{}^{2}_{1})$ can be solved in time $2^{o(v)}$ or time $2^{o(a)}$ for instances where the causal graph is a directed-path graph, then the ETH is false.

Proof sketch. There is a polynomial reduction from 3-SAT to $\mathrm{PSAT}(\mathrm{SAS}^{+}{}^{2}_{1})$ with directed-path causal graphs and $(2m+4)n$ variables [19, Proposition 5.5]. □

Theorem 28 If $\mathrm{PSAT}(\mathrm{SAS}^{+}{}^{1}_{1})$ can be solved in time $2^{o(v)}$ or time $2^{o(a)}$ for instances where the causal graph is a fence graph, then the ETH is false.

Proof sketch. There is a polynomial reduction from 3-SAT to $\mathrm{PSAT}(\mathrm{SAS}^{+}{}^{1}_{1})$ with fence causal graphs and $2m + 1$ variables [4, Lemma 7]. □

10 TIME VS. SPACE

The best upper bounds for hard problems usually assume algorithms that do not run in polynomial space. For instance, the result of Björklund et al. [7] that GRAPH COLOURABILITY can be solved in time $2^n poly(n)$ also requires using space $2^n poly(n)$. They also show an upper bound of time $2.2461^n poly(n)$ under the additional restriction of polynomial space [7, Proposition 7].

Our Observation 3 gives an upper bound of time $O(3^v)$ for monotone planning, but this result also requires space $O(3^v)$, since it is based on Dijkstra's algorithm. Almost all heuristic search algorithms also require exponential space. However, Depth-first search (DFS) runs in time $O(b^d)$ and space $O(bd)$ for implicitly represented graphs [32], where b is the branching factor and d is the search depth. Since the shortest plans are of length v at most, we can solve $\mathrm{LOP}(\mathrm{PSN}^{*}_{*+})$ in time $O(b^v)$ and polynomial space. This is still heavily dependent on the branching factor, so we will present an algorithm that also runs in polynomial space and beats DFS for larger branching factors. To do so, we first need to recapitulate some theory on ordered partitions.

The Stirling number $\left\{ {n \atop k} \right\}$ of the second kind denotes the number of ways we can partition a set of size n into k parts. Each partition of size k can be ordered in $k!$ different ways, so the total number of ordered partitions of all sizes of a set with n elements is $F(n) = \sum_{k=0}^{n} k! \left\{ {n \atop k} \right\} = \frac{1}{2}\sum_{m=0}^{\infty} \frac{m^n}{2^m}$, which is known as the nth *Fubini number* (or the nth *ordered Bell number*).

Theorem 29 $\mathrm{LOP}(\mathrm{PSN}^{*}_{*+})$ can be solved in time $O(F(v) \cdot poly(||\mathbb{P}||))$ using polynomial space.

Proof. Let $\mathbb{P} = \langle V, A, I, G\rangle$ be a PSN^{*}_{*+} instance. Let $a_1, a_2, \ldots, a_\ell$ be a plan from I to some state s_ℓ, and let $s_0, s_1, \ldots, s_\ell$ be its state sequence. Then $I = s_0 \subseteq s_1 \subseteq \cdots \subseteq s_\ell$, since all action effects are positive. Furthermore, if the plan is optimal, then all subset relations are strict, since a_i is redundant if $s_{i-1} = s_i$. It follows that for each optimal plan $\omega = a_1, a_2, \ldots, a_\ell$, there is some sequence $I = s_0 \subset s_1 \subset \cdots \subset s_k = V$ of states such that $s_0, s_1, \ldots, s_\ell$ is the state sequence of ω for some ℓ $(1 \leq \ell \leq k)$. Hence, we can find all plans for $\mathbb{P}$ by enumerating all such state sequences and check which ones have prefixes that correspond to a plan. Also define the sequence $\delta = d_1, d_2, \ldots, d_k$ such that $d_i = s_i \setminus s_{i-1}$ for all i $(1 \leq i \leq k)$. We note that δ is a partition on V, and it is furthermore an ordered partition since different orders on its parts generate different state sequences. That is, there is a one-to-one correspondence between the monotone state sequences and the ordered partitions, so it is sufficient to enumerate the latter.

For each ordered partition $d_1, d_2, \ldots, d_k$, generate the corresponding state sequence $s_0, s_1, \ldots, s_k$, where $s_0 = I$ and $s_i = s_{i-1} \cup d_i$ for all i. For all i from 0 to k do the following: If there is no $a \in A$ such that $\mathrm{pre}(a) \sqsubseteq s_{i-1}$ and $d_i \subseteq \mathrm{eff}(a)$, then break and continue with the next partition. Otherwise, choose any such action as action a_i. If $G \sqsubseteq s_i$, then break and remember i if it is the shortest plan length so far.

Generating all partitions of the set $\{1, \ldots, n\}$ takes $O(1)$ amortized time per partition [35] and generating all permutations of $\{1, \ldots, n\}$ takes $O(1)$ time per permutation [34], both in polynomial space. Hence, all $F(n)$ ordered partitions of $\{1, \ldots, n\}$ can be generated in time $O\big(F(n)\big)$. Checking each partition takes polynomial time in the instance size, so our algorithm runs in time $O\big(F(v) \cdot poly(||\mathbb{P}||)\big)$ and uses only polynomial space, since it considers only one partition at a time. □

It is known that $F(n) \simeq \frac{n!}{2(\ln 2)^{n+1}}$ [36] and that $n! < (\frac{n}{2})^n$,

so this algorithm will beat DFS for branching factors approximately greater than $\frac{v}{2}$. Furthermore, our algorithm does not even depend on the branching factor.

For the general, non-monotone case, however, we probably cannot hope for nearly as efficient algorithms under the polynomial-space constraint. It may even be difficult to find any algorithm running in polynomial space since the shortest plans may themselves be of exponential length in the general case. Exceptions exist in restricted cases, though. Jonsson and Bäckström [26] report a class of planning problems where the shortest solutions can be of exponential length, but it is always possible to decide in polynomial time if there is a solution or not. Jonsson [25] has further shown that it is even possible to generate a polynomial-size macro representation of a solution in this case. However, deciding if there is a solution of a specified length is **NP**-complete.

We do know, of course, that there must exist an algorithm for LOP(SAS$^+$) that runs in polynomial space, since the problem is in **PSPACE**, but this does not tell us much about the actual time bounds. Even if using an implicit representation of the state-transition graph, most search algorithms may still use an exponential amount of memory. This applies even to depth-first search (since there are planning instances with exponentially long shortest solutions). By using Savitch's theorem [33], the amount of memory can be lowered. Savitch showed that there exists an algorithm $\mathcal{A}_S$ that takes a graph $G = \langle U, E\rangle$ as input and checks whether there exists a path from $u \in U$ to $v \in U$ of length k or less using space $O\big(\log^2(|U|)\big)$ and time $|U|^{O(\log k)}$. The time bound did not appear in Savitch's article, but it is a well-known folklore result. The only thing one has to keep in mind when using this time bound is that we must be able to check whether two vertices are connected or not in polynomial time (in the size of the graph). Problem PSAT can thus be solved by asking if there is a plan of length $k = |S| = 2^{|V|}$, i.e. by solving LOP for this value of k. If we assume an implicit graph representation (where vertex adjacency can be checked in $p(||\mathbb{P}||)$ time for some polynomial p) we can thus solve PSAT in time

$$\begin{aligned}|S|^{O(\log k)} &\cdot p(||\mathbb{P}||)\\ &= |S|^{O(\log |S|)} \cdot p(||\mathbb{P}||) = (d^{|V|})^{O(\log d^{|V|})} \cdot p(||\mathbb{P}||)\\ &= (d^{|V|})^{O(|V|)} \cdot p(||\mathbb{P}||) = d^{O(|V|^2)} \cdot p(||\mathbb{P}||)\end{aligned}$$

using space

$$O(\log^2 |S|) = O(\log^2 d^{|V|}) = O(|V|^2)$$

Savitch's theorem is clearly also useful for problem LOP, since checking whether there exists a plan of length k or less takes time $2^{O(|V| \log k)}$) and uses space $O(|V|^2)$, which can be substantially better than solving PSAT when k is moderately large.

We conclude by noting that the polynomial factor $p(||\mathbb{P}||)$ that we have used to cover the time for checking the action set of an instance $\mathbb{P}$ is sufficient also for verifying an action.

Note that Savitch's theorem has been repeatedly applied to planning in the literature for proving membership in **PSPACE**. However, it has never been used to derive explicit bounds on time and space in the way we do.

11 DISCUSSION

Most of the planning classes that we prove not solvable in subexponential time (unless the ETH is false) are already known to be **NP**-hard. Our results are stronger in the following sense: even if it is the case that **P** $\neq$ **NP**, it is possible that an **NP**-hard problem can be solved in subexponential time (there are superpolynomial subexponential functions). Our results rule out that possibility (assuming the ETH holds).

For problems that are not solvable in polynomial time, one usually resorts to alternative methods, for instance, polynomial-time approximation algorithms or heuristic search, in the latter case hoping that this will perform satisfactorily in practice. However, with modern computers it is becoming increasingly popular to consider also algorithms running in superpolynomial time. Preferrably, such an algorithm should still run in subexponential time. It is then interesting to know whether such an algorithm can exist or not, thus asking for the type of lower-bound results we derive in this paper. There are even cases where one considers algorithms, and even approximation algorithms, that require low-order exponential time [12]. In such cases, the performance is very sensitive to the constant in the exponent, requiring results in the style of our Theorem 12.

Our analysis of LOP is similar in spirit to recent analyses of lower bounds for CSP [27, 13]. It is interesting to note that they prove a case where CSP can be solved in time $2^{o(n)}$ if $m \in o(n)$, but cannot be solved in time $2^{o(n)}$ if $m \in \Omega(n)$ and the ETH holds, where n is the number of variables and m the number of constraint tuples. Although there are no immediate connections, this is a sharp easy-hard transition of the same type indicated by Theorem 10 contrasted with the results in Section 7.

Obviously the ratio a/v is crucial here. This has similarities to the phenomenon of phase transitions for **NP**-complete problems, which was pioneered by Cheeseman et al. [10] and has remained an active research area ever since. For instance, in the case of k-SAT, the phase transition occurs at a particular value of the ratio m/n for each k such that instances around this ratio are likely to be hard and the probability of hard instances is very low for other values of the ratio. While the vast majority of work in this area has been empirical, the exact values of the phase transitions for k-SAT have been determined analytically [2]. There is also a previous result on this type of phase transitions for planning, but with a very broad transition region rather than a sharp transition [9]. However, these are all transitions of the type easy-hard-easy. Our transition is of the type easy-hard and is, thus, more similar to the type of transitions for resolution proofs studied by Achlioptas et al. [1].

The upper bounds of time 3^v for monotone planning and time 4^v for the general case might, perhaps, suggest that the latter case is much harder, especially since monotone planning is **NP**-complete but the general case is **PSPACE**-complete. However, it is dangerous to draw any such conclusions, as Stearns [37] has pointed out:

> Although **PSPACE**-completeness is stronger evidence of hardness than **NP**-completeness, there is no reason to believe that **PSPACE**-complete problems are harder in the sense that they require more time.

While upper-bound results sometimes take space into account, lower-bound results generally refer to time only, making no additional restrictions on space. Having seen in Sec. 10 how additional space restrictions can affect the upper bound, it is a valid question to ask if additional space bounds could also strengthen the lower-bound results upwards.

Acknowledgements

Bäckström is partially supported by the Swedish Research Council (VR) under grant 621-2014-4086.

REFERENCES

[1] Dimitris Achlioptas, Paul Beame, and Michael S. O. Molloy, 'A sharp threshold in proof complexity yields lower bounds for satisfiability search', *J. Comput. Syst. Sci.*, **68**(2), 238–268, (2004).

[2] Dimitris Achlioptas and Cristopher Moore, 'Random k-SAT: Two moments suffice to cross a sharp threshold', *SIAM J. Comput.*, **36**(3), 740–762, (2006).

[3] Meysam Aghighi and Christer Bäckström, 'Cost-optimal and net-benefit planning - A parameterised complexity view', in *Proc.24th Int'l Joint Conf. Artif. Intell. (IJCAI 2015), Buenos Aires, Argentina*, pp. 1487–1493, (2015).

[4] Christer Bäckström and Peter Jonsson, 'A refined view of causal graphs and component sizes: SP-closed graph classes and beyond', *J. Artif. Intell. Res.*, **47**, 575–611, (2013).

[5] Christer Bäckström, Peter Jonsson, Sebastian Ordyniak, and Stefan Szeider, 'A complete parameterized complexity analysis of bounded planning', *J. Comput. Syst. Sci.*, **81**(7), 1311–1332, (2015).

[6] Christer Bäckström and Bernhard Nebel, 'Complexity results for SAS^+ planning', *Comput. Intell.*, **11**, 625–656, (1995).

[7] Andreas Björklund, Thore Husfeldt, and Mikko Koivisto, 'Set partitioning via inclusion-exclusion', *SIAM J. Comput.*, **39**(2), 546–563, (2009).

[8] Tom Bylander, 'The computational complexity of propositional STRIPS planning', *Artif. Intell.*, **69**(1-2), 165–204, (1994).

[9] Tom Bylander, 'A probabilistic analysis of propositional STRIPS planning', *Artif. Intell.*, **81**(1-2), 241–271, (1996).

[10] Peter Cheeseman, Bob Kanefsky, and William M. Taylor, 'Where the really hard problems are', in *Proc. 12th Int'l Joint Conf. Artif. Intell. (IJCAI 1991) Sydney, Australia*, pp. 331–340, (1991).

[11] Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström, 'On problems as hard as CNF-SAT', in *Proc. 27th Conf. Computational Complexity, (CCC 2012), Porto, Portugal*, pp. 74–84, (2012).

[12] Marek Cygan, Lukasz Kowalik, and Mateusz Wykurz, 'Exponential-time approximation of weighted set cover', *Inf. Process. Lett.*, **109**(16), 957–961, (2009).

[13] Ronald de Haan, Iyad A. Kanj, and Stefan Szeider, 'On the subexponential-time complexity of CSP', *J. Artif. Intell. Res.*, **52**, 203–234, (2015).

[14] Carmel Domshlak and Yefim Dinitz, 'Multi-agent off-line coordination: Structure and complexity', in *Proceedings of the 6th European Conference on Planning (ECP'01), Toledo, Spain*, (2001).

[15] Ariel Felner, 'Position paper: Dijkstra's algorithm versus uniform cost search or a case against Dijkstra's algorithm', in *Proc. 4th Ann'l Symp. Combinatorial Search,(SoCS 2011), Castell de Cardona, Barcelona, Spain*, (2011).

[16] Michael L. Fredman and Robert E. Tarjan, 'Fibonacci heaps and their uses in improved network optimization algorithms', in *Proc. 25th Ann. Symp. Foundations Comput. Sci. (FOCS'84), West Palm Beach, FL, USA*, pp. 338–346, (1984).

[17] Hana Galperin and Avi Wigderson, 'Succinct representations of graphs', *Inform. Control*, **56**(3), 183–198, (1983).

[18] Michael R. Garey and David S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, 1979.

[19] Omer Giménez and Anders Jonsson, 'Planning over chain causal graphs for variables with domains of size 5 is NP-hard', *J. Artif. Intell. Res.*, **34**, 675–706, (2009).

[20] Malte Helmert, 'The fast downward planning system', *J. Artif. Intell. Res.*, **26**, 191–246, (2006).

[21] Jörg Hoffmann, 'Where 'ignoring delete lists' works: Local search topology in planning benchmarks', *J. Artif. Intell. Res.*, **24**, 685–758, (2005).

[22] Russell Impagliazzo and Ramamohan Paturi, 'On the complexity of k-SAT', *J. Comput. Syst. Sci.*, **62**(2), 367–375, (2001).

[23] Russell Impagliazzo and Ramamohan Paturi, 'Exact complexity and satisfiability - (invited talk)', in *Parameterized and Exact Computation - 8th Int'l Symp., (IPEC 2013), Sophia Antipolis, France, Revised Selected Papers*, pp. 1–3, (2013).

[24] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane, 'Which problems have strongly exponential complexity?', *J. Comput. Syst. Sci.*, **63**(4), 512–530, (2001).

[25] Anders Jonsson, 'The role of macros in tractable planning', *J. Artif. Intell. Res.*, **36**, 471–511, (2009).

[26] Peter Jonsson and Christer Bäckström, 'Tractable plan existence does not imply tractable plan generation', *Annals Math. Artif. Intell.*, **22**(3-4), 281–296, (1998).

[27] Iyad A. Kanj and Stefan Szeider, 'On the subexponential time complexity of CSP', in *Proc. 27th AAAI Conf. Artif. Intell. (AAAI 2013), Bellevue, WA., USA*, (2013).

[28] Michael Katz and Carmel Domshlak, 'New islands of tractability of cost-optimal planning', *J. Artif. Intell. Res.*, **32**, 203–288, (2008).

[29] Michael Katz and Carmel Domshlak, 'Optimal admissible composition of abstraction heuristics', *Artif. Intell.*, **174**(12-13), 767–798, (2010).

[30] Martin Kronegger, Andreas Pfandler, and Reinhard Pichler, 'Parameterized complexity of optimal planning: A detailed map', in *Proc. 23rd Int'l Joint Conf. Artif. Intell. (IJCAI-13), Beijing, China*, pp. 954–961, (2013).

[31] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh, 'Lower bounds based on the exponential time hypothesis', *Bulletin of the EATCS*, **105**, 41–72, (2011).

[32] Stuart J. Russell and Peter Norvig, *Artificial intelligence - a modern approach: the intelligent agent book*, Prentice Hall series in artificial intelligence, Prentice Hall, 1995.

[33] W. Savitch, 'Relationships between nondeterministic and deterministic tape complexities', *J. Comput. Syst. Sci.*, **4**(2), 177–192, (1970).

[34] Robert Sedgewick, 'Permutation generation methods', *ACM Comput. Surv.*, **9**(2), 137–164, (1977).

[35] Ichiro Semba, 'An efficient algorithm for generating all partitions of the set $\{1, 2, \ldots, n\}$', *J. Inf. Process.*, **7**(1), 41–42, (1984).

[36] Renzo Sprugnoli, 'Riordan arrays and combinatorial sums', *Disc. Math.*, **132**(1-3), 267–290, (1994).

[37] Richard Edwin Stearns, 'Turing award lecture: It's time to reconsider time', *Commun. ACM*, **37**(11), 95–99, (1994).

[38] Brian C. Williams and P. Pandurang Nayak, 'A reactive planner for a model-based executive', in *Proc. 15th Int'l Joint Conf. Artif. Intell. (IJCAI'97), Nagoya, Japan*, pp. 1178–1185, (1997).