Parameterized Complexity of Optimal Planning: A Detailed Map

Martin Kronegger, Andreas Pfandler, and Reinhard Pichler
Vienna University of Technology, Austria
{kronegger, pfandler, pichler}@dbai.tuwien.ac.at

Abstract

The goal of this paper is a systematic parameterized complexity analysis of different variants of propositional STRIPS planning. We identify several natural problem parameters and study all possible combinations of 9 parameters in 6 different settings. These settings arise, for instance, from the distinction if negative effects of actions are allowed or not. We provide a complete picture by establishing for each case either paraNP-hardness (i.e., the parameter combination does not help) or W[t]-completeness with $t \in \{1, 2\}$ (i.e., fixed-parameter intractability), or FPT (i.e., fixed-parameter tractability).

1 Introduction

Planning is a classical reasoning task in AI. In general, it is computationally hard. Bylander [1994] provided a comprehensive complexity analysis of propositional STRIPS (below simply referred to as STRIPS, for short) – a fundamental model of planning. Without any restrictions (in particular, imposing no restriction on the plan length), STRIPS is PSPACE-complete. With a polynomial upper bound on the plan length, the problem becomes NP-complete. Bylander also identified several restrictions which make STRIPS planning tractable, e.g., by allowing only positive preconditions and only one fluent (i.e., variable) to be affected by each action.

Many attempts have been made to identify further tractable fragments of planning. Bäckström and Klein [1991] have introduced various restrictions for the more general planning formalism SAS$^+$, where the domain is not necessarily binary. These restrictions include, e.g., P (i.e., every fluent is set to a particular value by at most one action) and U (i.e., only one fluent to be affected by an action). Bäckström and Nebel [1995] investigated the complexity under all combinations of these restrictions. Giménez and Jonsson [2008], Chen and Giménez [2010], and Katz and Domshlak [2008] analyzed planning under various restrictions of the causal graph.

A modern approach for identifying tractable fragments of hard problems comes from parameterized complexity [Downey and Fellows, 1999], which introduces a multivariate view of complexity. The time needed to solve a problem is thus measured not only in terms of the mere size n of a problem instance but also in terms of the size k of some parameter (or combination of parameters) that describes certain characteristics of the given problem instance. The primary goal of a parameterized complexity analysis is to identify *fixed-parameter tractability* (FPT), i.e., the problem can be solved in time $f(k) \cdot n^{\mathcal{O}(1)}$, where n denotes the size of the problem instance and $f(k)$ depends on the parameter k only (but not on n). The exponential time complexity is thus confined to the parameter, i.e., the function $f(k)$. Moreover, an FPT result immediately yields tractability of the problem if the parameter is bounded by a constant. In the area of planning, very few FPT results are known. Downey *et al.* [1999] proved FPT for STRIPS planning when parameterized by the plan length and the treewidth of an instance. Bäckström *et al.* [2012] proved FPT of SAS$^+$ planning with restriction P recalled above when considering the plan length as parameter.

In principle, any (natural) characteristic of problem instances is worth considering as parameter. Of course, there is no guarantee that a particular parameter yields a tractable fragment. If restricting a parameter by a constant does not lead to tractability, we call the problem paraNP-*hard*. Bylander [1994] showed that STRIPS remains NP-hard even if every action has only one precondition and one effect. In other words, the problem is paraNP-hard w.r.t. the parameter combination "precondition size and effect size".

In parameterized complexity, the area between the "nice" case of FPT and the extreme case of paraNP-hardness is even more fine-grained: a whole hierarchy of complexity classes W[1], W[2], etc. lies in between and is used to classify *fixed-parameter intractable* problems. For instance, STRIPS planning parameterized by the plan length was shown W[1]-hard by Downey *et al.* [1999] and actually W[2]-complete by Bäckström *et al.* [2012]. It is a commonly believed complexity-theoretic assumption that FPT $\neq$ W[1], which is also supported by the exponential time hypothesis (ETH). Hence showing hardness for W[1] (or higher classes) rules out the existence of an FPT-algorithm. Indeed, for W[t]-complete problems, only algorithms with run time $\mathcal{O}(n^{f(k)})$ are known, i.e., the parameter k occurs in the exponent of the input size n. This is worse than the upper bound $f(k) \cdot n^{\mathcal{O}(1)}$ for FPT, but it still allows us to identify a PTIME-solvable fragment of the problem by imposing a constant upper bound on the parameter value.

In various areas of reasoning, systematic parameterized

complexity analyses (by considering all combinations of several problem parameters) have been conducted recently. For instance, the parameterized complexity of abduction was studied by Fellows *et al.* [2012], of circumscription by Lackner and Pfandler [2012a], and of handling minimal models by Lackner and Pfandler [2012b]. For planning, very little is known about the parameterized complexity apart from the few results recalled above. The goal of this paper is to close this gap by embarking on a *systematic parameterized complexity analysis of planning*.

We start our analysis by identifying several natural parameters of planning instances. All of these parameters can be simply read off from an instance. Some of them describe the instance, e.g., size of the preconditions, size of the effects, maximum number of occurrences of each fluent, etc. The other parameters refer to desired properties of the solutions: the plan length as has already been studied before and a refinement of it, namely the total number of fluent changes in a plan. We study all possible combinations of 9 parameters in 6 different settings. These settings arise depending if negative preconditions/effects of actions are allowed or not and if we consider the length or the number of fluent changes as a possible restriction on allowed plans. In principle we thus get $6 \cdot 2^9 = 3072$ cases. However, in Section 3 we shall identify various dependencies between these parameters so that we only need to establish the parameterized complexity of a small subset of these cases to cover all of them. For details, see Section 3.

Contribution. We provide a complete picture of the parameterized complexity of three variants of propositional STRIPS planning. For each of the 3072 cases we establish either paraNP-hardness (i.e., the parameter combination does not help) or W[t]-completeness with $t \in \{1, 2\}$ (i.e., fixed-parameter intractability), or FPT (i.e., fixed-parameter tractability). FPT is achieved, e.g., for the combined parameters plan length and maximum number of occurrences of each fluent. This underlines the importance of studying combinations of parameters, since the first parameter alone yields W[2]-completeness and the second parameter alone even leads to paraNP-hardness.

2 Preliminaries

We assume familiarity with the basics of complexity theory and logic. For an introduction we refer, e. g., to the book of Papadimitriou [1994]. We use the following notation. We denote by $\mathit{var}(\varphi)$ the set of variables of a propositional formula φ. For $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{1, \ldots, n\}$.

Planning. We consider a set of *state variables* (or *fluents*) $V = \{v_1, \ldots, v_n\}$ over the Boolean domain $D = \{0, 1\}$. A *state* s is a mapping $s : V \to D$. Let I be the *initial state* and the *goal* G be a satisfiable conjunction of literals over V. An *action* a consists of a *precondition* $\mathsf{pre}(a)$ and an *effect* $\mathsf{eff}(a)$, both of which are satisfiable conjunctions of literals over V. We denote such an action by $a : \mathsf{pre}(a) \to \mathsf{eff}(a)$. Let A be a set of actions. A PSN (propositional STRIPS with negation) *specification* is given by the tuple (V, A, I, G). An MPSN (monotone-PSN) specification is a PSN specification where only positive literals are allowed in the effect of the actions. The special case of PSN where only positive literals are allowed in the precondition of the actions and in the goal is usually referred to as (classical propositional) STRIPS.

An action $a \in A$ is *applicable* in state s if $\mathsf{pre}(a)$ is satisfied by s, when s is considered as an interpretation. A conjunction of literals has a unique model when considering only the variables which actually occur in this conjunction. We shall refer to this unique model when we speak of the *model of the goal*, the *model of an effect*, etc. Let m be the model of $\mathsf{eff}(a)$ for some action a. For the resulting state s' after executing an applicable action $a \in A$ in state s we have $s'(v) = m(v)$ if $v \in \mathit{var}(\mathsf{eff}(a))$ and $s'(v) = s(v)$ otherwise.

Let $\pi = [a_1, ..., a_l]$ be a sequence of actions and $s_0, \ldots, s_l$ be states. Then π is called a *plan from* s_0 *to* s_l if $a_i \in \pi$ is applicable in state s_{i-1} and s_i is the resulting state after executing a_i in state s_{i-1} for $i \in [l]$. A state s' is called *reachable* from a state s if there is a plan from s to s'. Let m be the model of G and s a state. Then s is called a *goal state* if $s(v) = m(v)$ for all $v \in \mathit{var}(G)$. We call π a *plan* for the planning specification (V, A, I, G) if $s_0 = I$, and s_l is a goal state. The *plan length* of π is its number of actions l.

Parameterized Complexity. Parameterized algorithmics (cf. [Downey and Fellows, 1999; Flum and Grohe, 2006; Niedermeier, 2006]) is a promising approach to obtain efficient algorithms for fragments of NP-hard problems. In a parameterized complexity analysis the runtime of an algorithm is studied with respect to a parameter $k \in \mathbb{N}$ and input size n. The basic idea here is to find a parameter that describes the structure of the instance such that the combinatorial explosion can be confined to this parameter. The most favourable class is FPT (*fixed-parameter tractable*) which contains problems that can be decided by an algorithm running in $f(k) \cdot n^{\mathcal{O}(1)}$ time, where f is a computable function. We call such an algorithm fixed-parameter tractable (fpt).

Formally, a *parameterized problem* is a subset of $\Sigma^* \times \mathbb{N}$, where Σ is the input alphabet. If a combination of parameters $k_1, \ldots, k_l$ is considered, the second component of an instance (x, k) is given by $k = \sum_{1 \leq i \leq l} k_i$. Problem reductions now also have to take the parameter into account. Let L_1 and L_2 be parameterized problems, with $L_1 \subseteq \Sigma_1^* \times \mathbb{N}$ and $L_2 \subseteq \Sigma_2^* \times \mathbb{N}$. A *parameterized reduction* (or fpt-reduction) from L_1 to L_2 is a mapping $P : \Sigma_1^* \times \mathbb{N} \to \Sigma_2^* \times \mathbb{N}$ s.t. (1) $(x, k) \in L_1$ iff $P(x, k) \in L_2$; (2) the mapping can be computed by an fpt-algorithm w.r.t. parameter k; (3) there is a computable function g such that $k' \leq g(k)$, where $(x', k') = P(x, k)$.

To define the most important complexity classes for fixed-parameter intractability, we first define the model-checking problem over $\Sigma_{t,u}$ formulas, MC[$\Sigma_{\mathsf{t,u}}$]. The class $\Sigma_{t,u}$ contains all first-order formulas of the form $\exists \overline{x}_1 \forall \overline{x}_2 \exists \overline{x}_3 \ldots Q \overline{x}_t \varphi(\overline{x}_1, \ldots, \overline{x}_t)$, where the formula φ is quantifier free and the quantifier Q is an $\exists$ if t is odd and a $\forall$ if t is even, and the quantifier blocks – with the exception of the first $\exists$ block – are of length at most u. We write Σ_1 to denote $\Sigma_{1,u}$ for arbitary $u \geq 1$. Given a finite structure $\mathcal{A}$ and a formula $\psi \in \Sigma_{t,u}$, the problem MC[$\Sigma_{\mathsf{t,u}}$] asks whether $\mathcal{A}$ is a model of ψ. The so-called W-hierarchy can be defined with help of MC[$\Sigma_{\mathsf{t,u}}$]. For $t \geq 1, u \geq 1$, the class

W[t] contains all problems that are fpt-reducible to $\mathrm{MC}[\Sigma_{t,u}]$ when parameterized by the length of ψ [Downey *et al.*, 1998; Flum and Grohe, 2005]. The class paraNP [Flum and Grohe, 2003] is defined as the class of problems that are solvable by a nondeterministic Turing-machine in fpt-time. Notice that hardness for paraNP is defined in terms of fpt-reductions. In our paraNP-hardness proofs, we will make use of the following characterization of paraNP-hardness given by Flum and Grohe [2006], Theorem 2.14: any parameterized problem that remains NP-hard when the parameter is set to some constant is paraNP-hard. The following relations between the parameterized complexity classes hold: $\mathsf{FPT} \subseteq \mathsf{W[1]} \subseteq \mathsf{W[2]} \subseteq \mathsf{W[3]} \subseteq \cdots \subseteq \mathsf{paraNP}$.

3 Overview of Results

In our parameterized complexity analysis of STRIPS, MPSN and PSN, we shall study the parameters listed in Table 2. For the definition of c, let V be a set of fluents, let s_0 denote the initial state, let $\pi = [a_1, ..., a_l]$ be a plan, and let s_i with $i \in [l]$ denote the state after action a_i. For $x \in V$, we set $c(x, \pi, i) = 1$ if $s_{i-1}(x) \neq s_i(x)$ to denote that x is *changed* in the i-th step of π and $c(x, \pi, i) = 0$ if $s_{i-1}(x) = s_i(x)$. Then parameter c is defined as $\sum_{x \in V, i \in [l]} c(x, \pi, i)$, i.e., c can be seen as a refinement of the plan length k. For the definition of the remaining parameters we need the following terminology. Let (V, I, A, G) be a planning specification. The *size* of a conjunction of literals φ is given by $|\mathit{var}(\varphi)|$. Analogously, the *size* of an action $a \in A$ is defined as $|\mathit{var}(\mathsf{pre}(a))| + |\mathit{var}(\mathsf{eff}(a))|$. We say that a variable $v \in V$ *occurs* in a conjunction of literals φ if v or $\neg v$ occurs in φ. Let $\mathsf{occ}_v(\varphi)$ be 1 if v occurs in φ and 0 otherwise. The *number of occurrences* of a variable $v \in V$ is defined as $\sum_{a \in A}(\mathsf{occ}_v(\mathsf{pre}(a)) + \mathsf{occ}_v(\mathsf{eff}(a)))$. Finally, the *overlap* of $a \in A$ is given by $|\mathit{var}(\mathsf{pre}(a)) \cap \mathit{var}(\mathsf{eff}(a))|$.

For the formal treatment of the parameters k and c, there is a subtle issue: the parameter value should be computable in polynomial time from the input. While all other parameters in Table 2 can be computed from the input in linear time, the parameters k and c refer to properties of the desired solutions. We thus include k and c explicitly in the input. But then another issue has to be solved: combining k and c may make the planning problem artificially more complex in an unintended and unintuitive way, e.g.: if we consider k (but not c) as parameter, then a restrictive value of c in the input may render possible plans invalid even though they satisfy the upper bound on the plan length k. We thus further distinguish *variants* of STRIPS, MPSN and PSN planning depending on whether k or c should be used as possible restriction on plans. Of course, if one is not interested in specifying upper bounds on k or c, then one simply chooses a sufficiently big (i.e., single exponential) value for them. In total, we thus study the following six settings:

k-STRIPS/k-MPSN/k-PSN
Instance: A STRIPS/MPSN/PSN planning specification (V, A, I, G) and an integer k.
Question: Is there a plan for (V, A, I, G) of length $\leq$ k?

k	length of the plan
c	number of fluent changes in the plan
p	maximum size of a precondition
e	maximum size of an effect
a	maximum size of an action
na	number of actions
g	size of the goal
nv	number of state variables
vo	maximum number of occurrences of a variable
o	maximum overlap of the actions

Table 2: List of considered parameters.

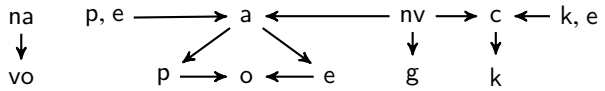

Figure 1: Parameter dependencies. An arc $x \to y$ denotes that $y \leq f(x)$ holds, where f is a computable function.

c-STRIPS/c-MPSN/c-PSN
Instance: A STRIPS/MPSN/PSN planning specification (V, A, I, G) and an integer c.
Question: Is there a plan for (V, A, I, G) with at most c fluent changes?

Our goal is a complete parameterized complexity classification in the above 6 settings by analyzing all combinations of either k or c with the remaining 8 parameters in Table 2. In this analysis, we may make use of dependencies between these parameters, e.g., if we take a as parameter, then p, o, e are implicitly also bounded. More specifically, we can bound the size of parameter a in terms of the parameters p, e via the inequation $\mathsf{a} \leq \mathsf{p} + \mathsf{e}$. For parameter c, the following bounds hold: $\mathsf{c} \leq \mathsf{k} \cdot \mathsf{e}$ as well as $\mathsf{c} \leq 2^{\mathsf{nv}} \cdot \mathsf{nv}$. The following theorem, which summarizes the relevant dependencies, is easily verified.

Theorem 1. *The parameter dependencies depicted in Figure 1 are correct.*

In Tables 1 (a)–(c) we give a summary of all parameterized complexity results that are needed for a full classification of all 2^9 parameter combinations in all MPSN and PSN settings. Indeed, these tables contain the *strongest* hardness- and membership results, from which the complexity classification of the remaining cases can be derived. To formalize the derivation of further complexity results, let P denote a combination of parameters from Table 2. We define the *closure* of P (denoted as P^*) as the set of parameters reachable from P in the directed graph in Figure 1. Moreover, we write "x-PSN (resp. x-MPSN) w.r.t. P" to denote that we consider the x-PSN (resp. x-MPSN) problem parameterized by the parameter combination P. With the following rules, which allow us to derive further complexity results from Tables 1 (a)–(c), we will indeed get a complete parameterized complexity classification of all parameter combinations in all settings.

Lemma 2. *Let $x \in \{\mathsf{c}, \mathsf{k}\}$ and let P, S be parameter combinations. Further, let $\mathcal{P} \in \{x\text{-MPSN}, x\text{-PSN}\}$ denote a prob-*

Setting	Complexity
k-MPSN, k, a, g, (p, e, o)	W[1] (Thm. 9)
k-MPSN, k, p, (o)	W[2] $*$
k-MPSN, k, g, o	W[2] (Thm. 10)
k-MPSN, a, g, vo, (p, e, o)	paraNP (Thm. 11)
c-MPSN, c, a, g, (p, e, o)	W[1] (Thm. 9)
c-MPSN, a, g, vo, (p, e, o)	paraNP (Thm. 11)

(a) Hardness results for k-MPSN/c-MPSN

Setting	Complexity
k-PSN, nv	FPT (Prop. 6)
k-PSN, k, vo	FPT (Thm. 5)
k-PSN, na	FPT (Thm. 4)
k-PSN, k, e	W[1] (Thm. 8)
k-PSN, k, p, g	W[1] (Thm. 7)
k-PSN, k	W[2] $*$

(b) Membership results for k-PSN

Setting	Complexity
c-PSN, nv	FPT (Prop. 6)
c-PSN, c, vo	FPT (Thm. 5)
c-PSN, na	FPT (Thm. 4)
c-PSN, c	W[1] (Thm. 12)

(c) Membership results for c-PSN

Table 1: Complexity Map; The results marked with $*$ are shown in the work of Bäckström *et al.* [2012].

lem and let $\mathcal{C}$ denote a complexity class. Then we can derive new complexity results from known ones as follows:

- *$\mathcal{C}$-membership of x-PSN w.r.t. P implies $\mathcal{C}$-membership of x-MPSN w.r.t. P.*
- *If $S \subseteq P^*$, then $\mathcal{C}$-membership of $\mathcal{P}$ w.r.t. S implies $\mathcal{C}$-membership of $\mathcal{P}$ w.r.t. P.*
- *$\mathcal{C}$-hardness of x-MPSN w.r.t. P implies $\mathcal{C}$-hardness of x-PSN w.r.t. P.*
- *If $P \subseteq S^*$ then $\mathcal{C}$-hardness of $\mathcal{P}$ w.r.t. S implies $\mathcal{C}$-hardness of $\mathcal{P}$ w.r.t. P.*

Note that in Table 1a (and in the theorems in Section 5) we present the hardness results for the *maximal* combinations of parameters s.t. the corresponding hardness still holds. Some parameters are put in parentheses to emphasize that we could have left them out, since they can be obtained from the remaining parameters via the last derivation rule in Lemma 2.

Theorem 3. *The results in Tables 1 (a)–(c) together with the derivation rules in Lemma 2 give a complete parameterized complexity classification, i.e., for each of the 4 settings* k-MPSN/k-PSN/c-MPSN/c-PSN *and for each combination of parameter* k *respectively* c *with the remaining 8 parameters in Table 2, we can either derive (1) membership in* FPT*, (2) completeness for* W[1] *or* W[2]*, or (3)* paraNP*-hardness.*

Proof. For every parameter combination in every setting one has to apply the rules from Lemma 2 to derive all possible membership and hardness results. By retaining only the strongest membership and hardness results, we get one of the 3 possible classifications according to the theorem. We have automatically verified this claim by means of a simple script which loops through all cases. □

For instance, suppose that we want to identify the complexity of k-PSN w.r.t. the parameter combination $P = \{\mathsf{k}, \mathsf{a}\}$. For the membership, first compute $P^* = \{\mathsf{k}, \mathsf{a}, \mathsf{p}, \mathsf{e}, \mathsf{o}\}$. In Table 1b we get W[1]- and W[2]-membership for the subsets $\{\mathsf{k}, \mathsf{e}\}$ respectively $\{\mathsf{k}\}$ of P^*. We retain W[1]-membership as the stronger result. For the hardness, we recall from Lemma 2 that hardness results for k-MPSN carry over to k-PSN. By the first row of Table 1a, we may derive the matching W[1]-hardness since $P \subseteq S^*$ with $S = S^* = \{\mathsf{k}, \mathsf{a}, \mathsf{g}, \mathsf{p}, \mathsf{e}, \mathsf{o}\}$. In total, we have thus established W[1]-completeness of k-PSN w.r.t. the parameter combination P.

Strips. Notice that PSN can be easily reduced to STRIPS via the following reduction. For each variable $v \in V$ we introduce a new "dual" variable v' with the intended meaning that v' is true whenever v is false and vice versa. Next, we replace each negative literal $\neg v$ by v' in all preconditions of actions and in the goal. Furthermore, whenever a variable occurs in the effect of an action we add the "dual" variable as follows: For v we add $\neg v'$ and for $\neg v$ we add v' to the effect. Finally, in the initial state each primed variable is set to false if the unprimed variable is initialized to true (and vice versa). After these steps all variables occurring in preconditions and the goal are positive, i. e., we have obtained a STRIPS specification. Notice that this reduction is an fpt-reduction w.r.t. all parameters listed in Table 2 since it can be carried out in fpt-time and all parameters in Table 2 are increased at most by a factor of two. Therefore, all hardness results obtained for k-PSN/c-PSN will carry over to k-STRIPS/c-STRIPS. Furthermore, all membership results for k-PSN/c-PSN trivially also hold for k-STRIPS/c-STRIPS.

Roadmap. The remainder of this paper is organized as follows. We first prove the membership results for k-PSN in Table 1b (see Section 4) and then the hardness results for k-MPSN in Table 1a (see Section 5). Finally, in Section 6, we show how most results can be carried over from parameter k to c. A separate proof is needed only when considering parameter c alone: in this case, the complexity drops from W[2]- (for k) to W[1]-completeness (for c).

4 Membership Results for Plan Length

We start with Table 1b. To this end, we first prove the FPT-results and then move on to the new W[1]-membership results.

Theorem 4. k-PSN *parameterized by* na *is in* FPT.

Proof. Let (V, A, I, G) be a PSN specification with at most na actions. The crucial observation is that there are only $\mathcal{O}(2^{\mathsf{na}} \cdot \mathsf{na}!)$ states reachable from the initial state I. To see this, let π be a plan and let $A(\pi) = \{a_1, \ldots, a_l\}$ denote the set of actions executed in π. We write $a_i < a_j$ to denote that the last execution of a_j in plan π is after the last execution of a_i in π. W.l.o.g., suppose $a_1 < a_2 < \cdots < a_l$. Then the value of a fluent $v \in V$ after the execution of π is as follows.

- if v is not in the effect of any of the actions $a_1, \ldots, a_l$ then v has the value according to the initial state;
- if $v \in \mathit{var}(\mathsf{eff}(a_i))$ for some a_i but not in the effect of $a_l, a_{l-1}, \ldots, a_{i+1}$, then v has the value according to a_i.

In other words, the state after the execution of an arbitrary plan π is uniquely determined by the set $A(\pi)$ and the order on the actions in $A(\pi)$ defined by π. The desired upper bound $\mathcal{O}(2^{\mathsf{na}} \cdot \mathsf{na}!)$ on the number of reachable states follows from the fact that there are at most 2^{na} possible sets of actions and for each such set there are at most $\mathsf{na}!$ orderings.

To decide k-PSN, we thus set up the state transition graph with $\mathcal{O}(2^{\mathsf{na}} \cdot \mathsf{na}!)$ nodes and check if a state satisfying the goal-conditions is reachable in k steps from the initial state. □

Theorem 5. k-PSN *parameterized by* k *and* vo *is in* FPT.

Proof. Let (V, A, I, G) be a PSN specification and let $\pi = [a_1, \ldots, a_j]$ be a sequence of actions with $j \leq \mathsf{k}$. We say there is a *gap before* i with $i \in [j+1]$ if (1) after the execution of $[a_1, \ldots, a_{i-1}]$, the precondition of a_i is not satisfied or (2) $i = j+1$ and after executing π the goal G is not satisfied.

Consider a sequence of actions $\pi = [a_1, \ldots, a_j]$ with $j \in [k]$ and assume that there is a gap before i in π for some $i \in [j+1]$. Otherwise π is already a plan from I to G. We only work out the details of case (1) in the definition of a gap, i.e., $i \in [j]$, s.t. in state s_{i-1}, the precondition of a_i is not satisfied. Case (2) (i.e., $i = j+1$) is shown analogously.

Let m be the model of $\mathsf{pre}(a_i)$. Then there is at least one variable $v \in \mathit{var}(\mathsf{pre}(a_i))$, s.t. $s_{i-1}(v) \neq m(v)$. There are at most vo actions with v in the effect. Hence, one of these vo actions has to be inserted at a position p with $p \leq i$ in the plan. Obviously, there are $\leq \mathsf{vo} \cdot i \leq \mathsf{vo} \cdot j$ possibilities to (1) choose an action with variable v in the effect and (2) to insert this action into the current plan at some position $p \leq i$.

By enumerating all these options and checking recursively whether a gap is left, we obtain a search tree of depth $\leq \mathsf{k}$ whose branching factor increases from vo (at the root) to $\mathsf{k} \cdot \mathsf{vo}$ (at the parent nodes of the leaves). This yields a running time of $\mathcal{O}(\mathsf{k}! \cdot \mathsf{vo}^k \cdot \mathit{poly}(n))$, where $\mathit{poly}(n)$ denotes a polynomial in the input size n. □

The third FPT-result is obvious since the number of states (and hence the plan length) as well as the number of actions are bounded by a (single exponential) function of nv.

Proposition 6. k-PSN *parameterized by* nv *is in* FPT.

We now turn our attention to the W[1]-membership results in Table 1b. Recall that Bäckström *et al.* [2012] have shown the W[2]-completeness of k-PSN w.r.t. parameter k (and implicitly also w.r.t. the parameter combination k, p, o). In Theorem 10, we will show W[2]-hardness of k-PSN if k is combined with g and o. However, if k is combined with both p and g together, then we obtain W[1]-membership.

Theorem 7. k-PSN *parameterized by* k, p, g *is in* W[1].

Proof. We proceed by a reduction from k-PSN to MC[Σ_1]. For an arbitrary k-PSN instance $\langle (V, I, A, G), \mathsf{k} \rangle$, we construct an MC[$\Sigma_1$] instance $(\mathcal{A}, \varphi)$ as follows. Let A' be the set of the identifiers of the actions in A, let $V' \subseteq V$ contain the variables that occur in a precondition or in the goal, and let $u = \mathsf{k} \cdot \mathsf{p} + \mathsf{g}$. To simplify the notation we will view conjunctions of literals also as sets of literals. We create the structure $\mathcal{A}$ with domain $V \cup A'$ and the following relations: the unary relations $\mathsf{Var} := V'$, $\mathsf{Act} := A'$, $\mathsf{init}_t := \{v \in V \mid I(v) = 1\}$, $\mathsf{goal}_t := \{v \in V \mid v \in G\}$, and the binary relations $\mathsf{pre}_t := \{(a, v) \in A' \times V \mid v \in \mathsf{pre}(a)\}$, $\mathsf{eff}_t := \{(a, v) \in A' \times V \mid v \in \mathsf{eff}(a)\}$. For each relation with subscript t, we also introduce a relation with subscript f for the dual case with negated variables. Furthermore, we add g unary relations $\mathsf{goalvar}_i$ with $i \in [\mathsf{g}]$, s.t. $\mathsf{goalvar}_i$ contains the variable corresponding to the i-th goal element. Similarly, we add p binary relations prevar_i with $i \in [p]$ where $\mathsf{prevar}_i(a, x)$ indicates that variable x is contained in the precondition of action a at position i. In case the size of the precondition is less than p, we can assume that the last variable is repeated until p is reached. To simplify the presentation, we use the macro $\mathsf{t}_i(x)$ with the intuition that the variable x is 1 at step i. $\mathsf{t}_i(x)$ is replaced by $\mathsf{init}_t(x)$ if $i = 0$ and by the subformula $\mathsf{eff}_t(a_i, x) \vee \bigvee_{i' < i} \left(\mathsf{t}_{i'}(x) \wedge \bigwedge_{i' < j \leq i} \neg \mathsf{eff}_f(a_j, x) \right)$ otherwise. Its dual version $\mathsf{f}_i(x)$ is defined analogously. The Σ_1-formula φ is defined as follows:

$$\varphi := \exists c_1 \cdots c_u \, \exists a_1 \cdots a_k \bigwedge_{i \in [u]} \mathsf{Var}(c_i) \wedge \bigwedge_{i \in [k]} \Big[\mathsf{Act}(a_i) \wedge \bigwedge_{j \in [u]} \big((\mathsf{pre}_t(a_i, c_j) \rightarrow \mathsf{t}_{i-1}(c_j)) \wedge (\mathsf{pre}_f(a_i, c_j) \rightarrow \mathsf{f}_{i-1}(c_j)) \big) \Big] \wedge \bigwedge_{i \in [u]} \Big[(\mathsf{goal}_t(c_i) \rightarrow \mathsf{t}_k(c_i)) \wedge (\mathsf{goal}_f(c_i) \rightarrow \mathsf{f}_k(c_i)) \Big] \wedge \bigwedge_{i \in [\mathsf{g}]} \bigvee_{j \in [u]} \mathsf{goalvar}_i(c_j) \wedge \bigwedge_{i \in [k]} \bigwedge_{l \in [p]} \bigvee_{j \in [u]} \mathsf{prevar}_l(a_i, c_j)$$

Clearly $\varphi \in \Sigma_1$ holds and the length of φ is bounded in terms of k, p, and g. The crucial idea in this reduction is the following. Since e is not considered as a parameter here, we cannot iterate over all literals in the effects. (This would require a $\forall$-quantifier or increase the size of the formula by a function depending on the input size.) We can, however, restrict the attention to those variables which are going to be "read" later on. These are exactly the variables which are either contained in the goal or in the precondition of one of the executed actions. Observe that there are $u = \mathsf{p} \cdot \mathsf{k} + \mathsf{g}$ many of them. Since u depends only on parameters we can "guess" these critical variables $c_1, \ldots, c_u$ by means of $\exists$-quantifiers. Now it suffices to use a conjunction over $c_1, \ldots, c_u$ each time the preconditions of an action are verified or the goal is checked (all other state variables can safely be ignored). Notice that φ encodes only plans of length exactly k. By using a "nop-action" for padding the plan, this construction can be easily extended to allow for plans of length at most k. □

For the sake of a simpler presentation of the proof we defined the macro $t_i(\cdot)$ as above. We remark, however, that a recursive definition of $t_i(\cdot)$ would have the advantage that the expanded size reduces from exponential to polynomial.

We postpone the proof of the following theorem, since it follows immediately from the proof of Theorem 12.

Theorem 8. k-PSN *parameterized by* k, e *is in* W[1].

5 Hardness Results for Plan Length

In this section, we prove the new parameterized hardness results for k-MPSN from Table 1a. We proceed from W[1]-hardness via W[2]-hardness to paraNP-hardness.

Theorem 9. k-MPSN *parameterized by* k, a, g, *(*p, e, o*) is* W[1]*-hard.*

Proof. We show this by a reduction from the well-known W[1]-complete problem CLIQUE, parameterized by the clique size s. Let $\langle (N, E), s\rangle$ be an instance of CLIQUE with $N = \{v_1, \ldots, v_n\}$ and $l = \frac{s(s-1)}{2}$, i.e., l is the number of edges in a clique of size s. We construct a k-MPSN instance $\langle (V, A, I, G), \mathsf{k}\rangle$ as follows. We set $V := V' \cup E' \cup H \cup \{f\}$ with $V' = \{x_1, \ldots, x_n\}$ representing the vertices, $E' = \{e_{ij} \mid 1 \leq i < j \leq n\}$ representing the edges, and $H = \{g_1, \ldots, g_l\}$ containing the goal variables. In the initial state I, we set the variables $e_{ij} \in E'$ with $\{v_i, v_j\} \notin E$ to 1 and all other variables to 0. The goal is $G := g_1 \wedge \ldots \wedge g_l$. We create $A := A_v \cup A_e \cup \{f : \neg f \to f\}$, where A_v and A_e are defined as follows:

$$A_v = \bigcup_{i \in [n]} \{a_{x_i} : \neg f \to x_i\}$$

$$A_e = \bigcup_{\substack{1 \leq i < j \leq n,\\ m \in [l]}} \{a^m_{e_{ij}} : f \wedge \neg g_m \wedge x_i \wedge x_j \wedge \neg e_{ij} \to g_m \wedge e_{ij}\}$$

Finally, we define the plan length k as $\mathsf{k} := \frac{s(s+1)}{2} + 1$.

The intuition of this reduction is as follows. As long as f is set to 0, the actions in A_v allow us to choose variables from V' (corresponding to the vertices in N) to be contained in the clique. Eventually action f is executed, and the state of the variables in V' is now "fixed". Basically, it remains to check whether for each pair $x_i, x_j \in V$ with $x_i = x_j = 1$, the edge $\{v_i, v_j\}$ is present in E. This is done with help of the actions in A_e. For each $m \in [l]$ an unused edge variable $e_{ij} \in E'$ such that x_i and x_j are set to 1 has to be chosen. This is done by executing $a^m_{e_{ij}}$ which sets both g_m and e_{ij} to 1. As a consequence, all actions $a^m_{e_{ij}}$ with $m \in [l]$ are inapplicable in the future. Here the intended meaning is that v_i and v_j are connected by an edge in E and thus do not violate the clique property. The goal is reached if all $g_1, \ldots, g_l$ are set to 1, which means that l edges between the chosen vertices have been found.

Clearly, this transformation can be carried out in fpt-time and the parameters a, g, p, e, o are bounded in terms of the clique size s. It remains to verify that also the length k of valid plans is bounded in terms of s. First, s actions are required to select the s vertices into the clique. Then, a single action is required to activate the flag f. During the checking phase, l actions from A_e have to be executed to reach the goal. In total, we obtain $\mathsf{k} = s + 1 + \frac{s(s-1)}{2} = \frac{s(s+1)}{2} + 1$. □

Bäckström *et al.* [2012] have shown the W[2]-hardness of k-PSN w.r.t. parameter k by a reduction from the HITTING SET problem. An inspection of that proof reveals that the authors have actually shown a slightly stronger result, namely the W[2]-hardness of k-PSN w.r.t. the parameter combination {k, p, o}. Below we show a further W[2]-hardness result of k-PSN by adding different parameters to parameter k.

Theorem 10. k-MPSN *parameterized by* k, g, o *is* W[2]*-hard.*

Proof. We show this by a reduction from DOMINATING SET, which is known to be W[2]-complete when parameterized by the size s of the dominating set. Let $\langle H = (V, E), s\rangle$ be an instance of DOMINATING SET where $V = \{v_1, \ldots, v_n\}$. By $N[v]$ we denote the closed neighbourhood of vertex v (i.e., the set containing all vertices adjacent to v and v itself) in the graph H. The k-MPSN instance $\langle (V \cup \{g\}, A, I, G), \mathsf{k} = s + 1\rangle$ is constructed as follows. In I, we set all variables to 0. Let $G := g$. We create the actions $A := \{a_v : \top \to N[v] \mid v \in V\} \cup \{g : v_1 \wedge \cdots \wedge v_n \to g\}$. Clearly, the reduction works in fpt-time. Moreover, the parameters k, g, o are indeed bounded in terms of s or by a constant. □

Bylander [1994] showed that PSN remains NP-hard even if every action has only one precondition and one effect. In other words, the problem is paraNP-hard w.r.t. the parameters p and e. Actually, it is thus also paraNP-hard w.r.t. p, e, a, o. Below, we further strengthen this result by adding also the parameters g and vo.

Theorem 11. k-MPSN *parameterized by* a, g, vo, *(*p, e, o*) is* paraNP*-hard.*

Proof. The hardness proof by Bylander [1994], (Theorem 3.5 and Corollary 3.6), is by reduction from the NP-hard 3-SAT problem. Actually, 3-SAT remains NP-hard even if we restrict the propositional formulas such that every variable may occur at most three times (see [Papadimitriou, 1994], Proposition 9.3). By allowing only such instances of 3-SAT in the reduction to k-MPSN, we make sure that also the parameter vo is bounded by a constant.

It remains to modify the reduction by Bylander such that also the goal size g is bounded without increasing p arbitrarily. The idea here is that we can always replace a "big goal" $G = g_1 \wedge \cdots \wedge g_n$ in a planning specification (V, A, I, G) as follows. Let $V' = V \cup \{g, h_1, \ldots, h_{n-2}\}$ and $G' = g$. Finally we add $n-1$ new actions to obtain $A' = A \cup \{a_1 : g_1 \wedge g_2 \to h_1\} \cup \{a_i : g_{i+1} \wedge h_{i-1} \to h_i | 2 \leq i \leq n-2\} \cup \{a_{n-1} : g_n \wedge h_{n-2} \to g\}$. Notice that the increase of the plan length caused by this transformation is not critical since k is not considered as a parameter here. Furthermore, in the new MPSN specification (V', A', I, G'), the parameters a, g, vo, p, e, o are bounded. □

6 Results for Limited Fluent Changes

Instead of imposing a limit on the plan length, we now consider the possible restriction of another "resource" – the number of changes that may be performed on the fluents by the execution of a plan. Formally, we study the problems c-MPSN and c-PSN defined in Section 3. While the plan length k intuitively measures the time needed to execute a plan, the number of fluent changes c somehow measures the effort required by the plan. In settings where changing the state of a variable is expensive or variables cannot be changed arbitrarily often, this parameter seems more appropriate. Moreover, the choice of the parameter has an interesting influence on the complexity: below we show that c-MPSN/c-PSN is W[1]-complete

w.r.t. c while k-MPSN/k-PSN was shown to be W[2]-complete w.r.t. k by Bäckström *et al.* [2012].

The paraNP-hardness result for k-MPSN in Theorem 11 easily carries over to c-MPSN, since both settings do not involve the parameters k or c. Furthermore, the proof of the W[1]-hardness result for k-MPSN in Theorem 9 can be modified to yield also W[1]-hardness for c-MPSN.

Another nice property is that all FPT results for k-PSN presented in Section 4 can be extended to the c-PSN setting. Here we use the fact that $c \geq k$ for any non-redundant plan and that the number of fluent changes can be computed in polynomial time for a given plan. Hence, to obtain a complete picture in the c-MPSN/c-PSN setting, it remains to show the following theorem.

Theorem 12. c-PSN *parameterized by* c *is in* W[1].

Proof (sketch). The proof is by reduction to MC[Σ_1]. The general approach is similar to the proof of Theorem 7. The formula, however, becomes much more intricate since we now have no upper bound on the values of p and g. On the other hand, taking c as parameter instead of k means that we have control over the fluents that are indeed changed by executing a plan.

In a preprocessing step we compute the Hamming-distance between the goal and the initial state. We can then split the set of variables in the goal into "matching" and "non-matching" variables. In case there are more than c non-matching variables, we can immediately stop with answer "no". Similarly, we check the distance of the precondition and of the effect of every action from the initial state. Actions for which the distance in the precondition or in the effect exceeds c may never be executed and can therefore safely be dropped. We keep track of all the non-matching variables (of the goal and of the remaining actions) in the finite structure $\mathcal{A}$.

Let us call those variables which are indeed altered by a plan "dirty". Intuitively, the Σ_1 formula guesses (via existentially quantified variables) for a plan π the $\leq$ c dirty variables and the sequence of actions of the plan. Moreover, we guess for each action a_i in π the fluents which are indeed changed by a_i. A $\forall$-quantifier is not needed, since we can restrict our attention to the dirty variables. Further, the Σ_1 formula checks whether the goal and the preconditions of the guessed actions are satisfied and whether all fluent changes caused by these actions are indeed contained in the guessed dirty variables. All this can be encoded into a formula whose length is bounded in terms of c. The details are omitted due to lack of space.

Since it is easy to ensure that at most k actions are executed and $k \cdot e \geq c$ holds, we also obtain Theorem 8. □

7 Conclusion

We have presented a complete parameterized complexity analysis for STRIPS, MPSN and PSN where either the plan length or the number of fluent changes may be used to define constraints on the allowed plans. In all settings, we have found three new FPT results. Especially the combination of k and vo could be a starting point for the development of efficient algorithms on tractable fragments. Considering *combinations of parameters* rather than just single parameters was critical since, for instance, PSN is W[2]-complete for k and even paraNP-hard for vo, but becomes FPT for the combination of both. It has turned out that the restriction to positive effects does not affect the parameterized complexity w.r.t. the considered parameters. The distinction between the problem variants c-PSN and k-PSN (and, likewise, between c-MPSN and k-MPSN) does have an influence on the complexity since c-PSN is W[1]-complete w.r.t. parameter c, while k-PSN is W[2]-complete w.r.t. parameter k.

Notice that all our membership results for MPSN and PSN can be easily extended to the more general setting with a multi-valued domain if the size of the domain is bounded by a constant or considered as an additional parameter. While we considered the problem of finding plans of length at most k (i. e., optimal planning) in this work, the question of satisficing planning (i. e., asking whether there is *any plan* to reach the goal) remains future work. Related planning formalisms such as delete-relaxed planning (i.e., STRIPS restricted to positive effects only) are also worth studying. Another important direction of future work is to investigate further constraints on the plans as parameter such as the number of allowed changes of each fluent individually or the notion of local depth introduced by Brafman and Domshlak [2006]. Evaluating the parameter values on real (benchmark) instances is also left for future work.

Acknowledgments

This research was supported by the Austrian Science Fund (FWF): P25518-N23. We thank the anonymous IJCAI-13 reviewers for the very detailed and constructive remarks.

References

[Bäckström and Klein, 1991] Christer Bäckström and Inger Klein. Planning in polynomial time: the SAS-PUBS class. *Computational Intelligence*, 7:181–197, 1991.

[Bäckström and Nebel, 1995] Christer Bäckström and Bernhard Nebel. Complexity results for SAS^+ planning. *Computational Intelligence*, 11:625–656, 1995.

[Bäckström *et al.*, 2012] Christer Bäckström, Yue Chen, Peter Jonsson, Sebastian Ordyniak, and Stefan Szeider. The complexity of planning revisited – a parameterized analysis. In *Proc. AAAI 2012*, pages 1735–1741. AAAI Press, 2012.

[Brafman and Domshlak, 2006] Ronen I. Brafman and Carmel Domshlak. Factored planning: How, when, and when not. In *Proc. AAAI 2006*, pages 809–814. AAAI Press, 2006.

[Bylander, 1994] Tom Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1–2):165–204, 1994.

[Chen and Giménez, 2010] Hubie Chen and Omer Giménez. Causal graphs and structurally restricted planning. *J. Comput. Syst. Sci.*, 76(7):579–592, 2010.

[Downey and Fellows, 1999] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Springer, 1999.

[Downey *et al.*, 1998] Rodney G. Downey, Michael R. Fellows, and Kenneth W. Regan. Descriptive complexity and the W hierarchy. In *Proof Complexity and Feasible Arithmetic*, volume 39 of *AMS-DIMACS Volume Series*, pages 119–134. AMS, 1998.

[Downey *et al.*, 1999] Rodney G. Downey, Michael R. Fellows, and Ulrike Stege. Parameterized complexity: A framework for systematically confronting computational intractability. In *Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future*, volume 49 of *DIMACS Series in Disc. Math. Theor. Comput. Sci.*, pages 49–99. DIMACS, 1999.

[Fellows *et al.*, 2012] Michael R. Fellows, Andreas Pfandler, Frances A. Rosamond, and Stefan Rümmele. The parameterized complexity of abduction. In *Proc. AAAI 2012*, pages 743–749. AAAI Press, 2012.

[Flum and Grohe, 2003] Jörg Flum and Martin Grohe. Describing parameterized complexity classes. *Inf. Comput.*, 187(2):291–319, 2003.

[Flum and Grohe, 2005] Jörg Flum and Martin Grohe. Model-checking problems as a basis for parameterized intractability. *Logical Methods in Computer Science*, 1(1):1–36, 2005.

[Flum and Grohe, 2006] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[Giménez and Jonsson, 2008] Omer Giménez and Anders Jonsson. The complexity of planning problems with simple causal graphs. *J. Artif. Intell. Res. (JAIR)*, 31:319–351, 2008.

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

[Lackner and Pfandler, 2012a] Martin Lackner and Andreas Pfandler. Fixed-parameter algorithms for closed world reasoning. In *Proc. ECAI 2012*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 492–497. IOS Press, 2012.

[Lackner and Pfandler, 2012b] Martin Lackner and Andreas Pfandler. Fixed-parameter algorithms for finding minimal models. In *Proc. KR 2012*, pages 85–95. AAAI Press, 2012.

[Niedermeier, 2006] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[Papadimitriou, 1994] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.