Fixed-Parameter Tractable Reductions to SAT for Planning*

Ronald de Haan[1], **Martin Kronegger**[1], **and Andreas Pfandler**[1,2]
firstname.lastname@tuwien.ac.at
[1]Vienna University of Technology, Austria
[2]University of Siegen, Germany

Abstract

Planning is an important AI task that gives rise to many hard problems. In order to come up with efficient algorithms for this setting, it is important to understand the sources of complexity. For planning problems that are beyond NP, identifying fragments that allow an efficient reduction to SAT can be a feasible approach due to the great performance of modern SAT solvers. In this paper, we use the framework of parameterized complexity theory to obtain a more fine-grained complexity analysis of natural planning problems beyond NP. With this analysis we are able to point out several variants of planning where the structure in the input makes encodings into SAT feasible. We complement these positive results with some hardness results and a new machine characterization for the intractability class $\exists^*\forall^k$-W[P].

1 Introduction

Like many other formalisms in AI, planning in general is highly intractable (many variants are complete for Σ_2^P or even PSPACE). There are, however, several restricted settings where planning is in NP, and can thus be solved by encoding the problem into SAT and subsequently calling a SAT solver. Due to the great performance of modern SAT solvers in many practical settings [Sakallah and Marques-Silva, 2011; Gomes *et al.*, 2008; Malik and Zhang, 2009], this is often a feasible approach. In the planning literature, the general approach of employing SAT solvers has been used also in planning settings that are beyond NP; e.g., see the approach presented by Palacios and Geffner [2009] to compile away uncertainty.

We extend this SAT-encoding approach by means of the framework of parameterized complexity theory. A classical complexity analysis can only provide a coarse complexity classification, that is based on the input size measured in bits only. However, in virtually all cases one knows more about the structure of the instance than is indicated in the raw bit size. Using parameterized complexity, one can capture information about structure in the input by means of a *parameter*. This parameter can then be used to provide a multivariate complexity analysis, that is sensitive to the structure of the input.

Recently, the idea of applying parameterized complexity to extend the range of cases where SAT solvers can be used to solve the problem has been applied to reasoning problems that are beyond NP [Fichte and Szeider, 2013; Pfandler *et al.*, 2013]. In addition, theoretical tools have been developed that allow us to indicate the boundaries of this approach [De Haan and Szeider, 2014a; 2014b], i.e., to give evidence that the SAT-encoding approach cannot be used in certain cases.

In this paper, we provide parameterized complexity results for natural planning problems that are beyond NP. We point out several variants of planning where structure in the input admits the use of encodings into SAT as an efficient way to solve the problem. In addition, we identify several planning settings where the structure in the input does not suffice to allow this approach of direct SAT-encodings. We hope that our results may help initiate a structured investigation of the use of parameterized complexity methods aimed at obtaining SAT-encodings in the domain of planning. In addition, we expect that our results can be used, as a stepping stone, to obtain further parameterized complexity results for other reasoning problems that are beyond NP, since the formalism of planning is widely-known and intuitive to use. In particular, this holds for showing membership results, as planning is an expressive formalism in which other problems can often be encoded very conveniently. The fact that all complexity classes discussed in this work are characterized by different flavors of planning problems can facilitate this considerably. Moreover, as a side result, we develop a machine characterization of one of the parameterized intractability classes for this setting. We believe that this result will be useful in the future.

Main Contributions:

- We present completeness results for a variety of planning problems and parameterized complexity classes – classically all these problems are harder than NP, e.g., complete for Σ_2^P or PSPACE. We show that two variants of planning under uncertainty and planning with soft goal optimization are complete for the classes para-NP, para-DP, and $\text{FPT}^{\text{NP}[f(k)]}$, respectively. For these classes, the SAT approach offers potential for practical algorithms. We contrast these results by showing completeness for harder classes. Together, this gives a more fine-grained picture of the (sources of) complexity.

*Supported by the Austrian Science Fund (FWF): P25518 and P26200, and the German Research Foundation (DFG): ER 738/2-1.

• This is the first work to characterize the parameterized complexity classes para-NP, para-DP, $\mathsf{FPT}^{\mathsf{NP}[f(k)]}$, $\exists^k\forall^*$, and $\exists^*\forall^k$-W[P] by means of variants of a single formalism, i.e., planning. We hope that our homogeneous characterization of these classes will facilitate future complexity analysis. Additionally, the methodology used in this work may be applied to *any* KR formalism of high complexity.
• In addition, we present a novel machine characterization of the complexity class $\exists^*\forall^k$-W[P]. We expect that this result is especially useful for showing membership in this class.

Organization. After recalling the required basics in the next section, we discuss the idea of parameterized reductions to SAT in Section 3. In each of the Sections 4 to 8, we present a natural variant of planning that captures a particular parameterized complexity class. Finally, we conclude in Section 9.

2 Preliminaries

We assume the reader to be familiar with the basics of complexity theory and logic, such as classes from the Polynomial Hierarchy (PH), e.g., Σ_2^{P}. For details we refer to textbooks on the topic [Papadimitriou, 1994; Arora and Barak, 2009].

Planning. In this work we build upon the SAS$^+$ formalism (see, e.g., [Bäckström and Nebel, 1995]). Let $V = \{v_1, \ldots, v_n\}$ be a finite set of *variables* over a finite *domain* D. Furthermore, let $D^+ = D \cup \{\mathbf{u}\}$, where $\mathbf{u}$ is a special "undefined" value not present in D. Then D^n is the set of *total states* and $(D^+)^n$ is the set of *partial states* over V and D. Clearly, $D^n \subseteq (D^+)^n$. The value of a variable v in a state $s \in (D^+)^n$ is denoted by $s[v]$. A *SAS$^+$ instance* is a tuple $\mathbb{P} = \langle V, D, A, I, G\rangle$ where V is a set of variables, D is a domain, A is a set of *actions*, $I \in D^n$ is the *initial state* and $G \in (D^+)^n$ is the (partial) *goal state*. Each action $a \in A$ has a *precondition* $\mathsf{pre}(a) \in (D^+)^n$ and an *effect* $\mathsf{eff}(a) \in (D^+)^n$.

We will frequently use the convention that a variable has value $\mathbf{u}$ in a precondition/effect unless a value is explicitly specified. Furthermore, by slight abuse of notation, we denote actions and partial states such as preconditions, effects, and goals as follows: Let $a \in A$, $p_i \in V$, $d_i \in D$, and $1 \leq i \leq m \leq n$ such that $\mathsf{pre}(a)[p_1] = d_1, \ldots, \mathsf{pre}(a)[p_m] = d_m$. Then we denote the precondition $\mathsf{pre}(a)$ by $\mathsf{pre}(a) = \{p_1 = d_1, \ldots, p_m = d_m\}$ (analogously for effects). For $m' \leq n$ and $e_j \in V$ with $1 \leq j \leq m'$, we use $a : \{p_1 = d_1, \ldots, p_m = d_m\} \to \{e_1 = d'_1, \ldots, e_{m'} = d'_{m'}\}$ as a shorthand to describe that a has $\mathsf{pre}(a) = \{p_1 = d_1, \ldots, p_m = d_m\}$ and $\mathsf{eff}(a) = \{e_1 = d'_1, \ldots, e_{m'} = d'_{m'}\}$.

Let $a \in A$ and $s \in D^n$. Then a is *valid in* s if for all $v \in V$, either $\mathsf{pre}(a)[v] = s[v]$ or $\mathsf{pre}(a)[v] = \mathbf{u}$. The *result of a in s* is a state $t \in D^n$ defined such that for all $v \in V$, $t[v] = \mathsf{eff}(a)[v]$ if $\mathsf{eff}(a)[v] \neq \mathbf{u}$ and $t[v] = s[v]$ otherwise. Let $s_0, s_\ell \in D^n$ and let $\omega = \langle a_1, \ldots, a_\ell\rangle$ be a sequence of actions (of length ℓ). Then ω is a *plan from s_0 to s_ℓ* if either (i) $\omega = \langle\rangle$ and $\ell = 0$, or (ii) there are states $s_1, \ldots, s_{\ell-1} \in D^n$ such that for all $1 \leq i \leq \ell$, a_i is valid in s_{i-1} and s_i is the result of a_i in s_{i-1}. A state $s \in D^n$ is a *goal state* if for all $v \in V$, either $G[v] = s[v]$ or $G[v] = \mathbf{u}$. An action sequence ω is a *plan for* $\mathbb{P}$ if ω is a plan from I to a goal state.

In planning instances often so-called *conditional effects* are permitted as effects. For a planning instance with n variables and domain D, let $1 \leq m_1, m_2 \leq n$. A conditional effect is an effect of the form $\{p_1 = d^1_1, \ldots, p_{m_1} = d^1_{m_1}\} \triangleright \{e_1 = d^2_1, \ldots, e_{m_2} = d^2_{m_2}\}$ with $d^1_i, d^2_j \in D$, $1 \leq i \leq m_1$ and $1 \leq j \leq m_2$. Without defining the semantics formally, a conditional effect (of an executed action) ensures that the effect $\{e_1 = d^2_1, \ldots, e_{m_2} = d^2_{m_2}\}$ only materializes if its condition $\{p_1 = d^1_1, \ldots, p_{m_1} = d^1_{m_1}\}$ is also satisfied.

Parameterized Complexity. Before we turn to parameterized algorithms for SAT-encodings, we introduce some core notions from parameterized complexity theory. For an in-depth treatment we refer to other sources [Downey and Fellows, 1999; 2013; Flum and Grohe, 2006; Niedermeier, 2006]. A *parameterized problem* L is a subset of $\Sigma^* \times \mathbb{N}$ for some finite alphabet Σ. For an instance $(I, k) \in \Sigma^* \times \mathbb{N}$, we call I the *main part* and k the *parameter*. The following generalization of polynomial time computability is commonly regarded as the tractability notion of parameterized complexity theory. A parameterized problem L is *fixed-parameter tractable* if there exists a computable function f and a constant c such that there exists an algorithm that decides whether $(I, k) \in L$ in time $O(f(k)\,|I|^c)$, where $|I|$ denotes the size of I. Such an algorithm is called an *fpt-algorithm*, and this amount of time is called *fpt-time*. FPT is the class of all fixed-parameter tractable decision problems. If the parameter is constant, then fpt-algorithms run in polynomial time where the order of the polynomial is independent of the parameter.

Parameterized complexity also generalizes the notion of polynomial-time reductions. Let $L \subseteq \Sigma^* \times \mathbb{N}$ and $L' \subseteq (\Sigma')^* \times \mathbb{N}$ be two parameterized problems. An *fpt-reduction* from L to L' is a mapping $R : \Sigma^* \times \mathbb{N} \to (\Sigma')^* \times \mathbb{N}$ from instances of L to instances of L' such that there exist some computable function $g : \mathbb{N} \to \mathbb{N}$ such that for all $(I, k) \in \Sigma^* \times \mathbb{N}$: (i) (I, k) is a yes-instance of L if and only if $(I', k') = R(I, k)$ is a yes-instance of L', (ii) $k' \leq g(k)$, and (iii) R is computable in fpt-time.

3 Parameterized Encodings into SAT

Problems that can be solved by means of a fixed-parameter tractable (many-one) encoding into SAT are captured by the class para-NP, that is defined as follows. Let C be a classical complexity class, e.g., NP. The parameterized complexity class para-C is defined as the class of all parameterized problems $L \subseteq \Sigma^* \times \mathbb{N}$, for some finite alphabet Σ, for which there exist an alphabet Π, a computable function $f : \mathbb{N} \to \Pi^*$, and a problem $P \subseteq \Sigma^* \times \Pi^*$ such that $P \in C$ and for all instances $(x, k) \in \Sigma^* \times \mathbb{N}$ of L we have that $(x, k) \in L$ if and only if $(x, f(k)) \in P$ [Flum and Grohe, 2003].

In addition to many-one fpt-reductions to SAT, we are also interested in Turing fpt-reductions. A *Turing fpt-reduction from a problem P to SAT* is an fpt-algorithm that has access to a SAT oracle and that decides P. We are mainly interested in fpt-algorithms that only use a small number of queries to the SAT oracle (*SAT calls*). We let $\mathsf{FPT}^{\mathsf{NP}[f(k)]}$ denote the class of all parameterized problems P for which there exists an fpt-algorithm that decides if $(x, k) \in P$ by using at most $f(k)$ many SAT calls, for some computable function f.

In order to provide evidence against the existence of (many-one) fpt-reductions to SAT, one can show hardness

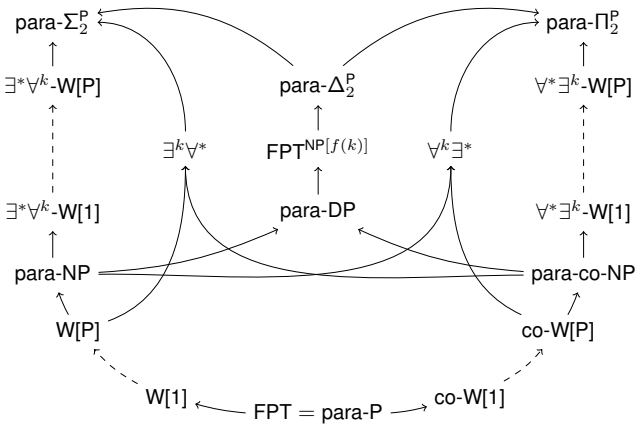

Figure 1: Relevant parameterized complexity classes. Arrows indicate inclusion relations. For a definition of the classes W[t] and co-W[t], see, e.g., [Flum and Grohe, 2006]. For a definition of the class para-Δ_2^P, see, e.g., [Papadimitriou, 1994; Flum and Grohe, 2003].

for para-Σ_2^P. Recent work in parameterized complexity theory has resulted in additional complexity classes that can be used to provide evidence for the non-existence of fpt-reductions to SAT [De Haan and Szeider, 2014b]. The parameterized complexity class $\exists^k\forall^*$ consists of all parameterized problems that can be fpt-reduced to the problem $\exists^k\forall^*$-WSAT, which is a variant of quantified Boolean satisfiability that is based on truth assignments of restricted (Hamming) weight. The input for this problem is a quantified Boolean formula (QBF) $\varphi = \exists X \forall Y \psi$, and an integer k. The parameter is k, and the question is whether there is a truth assignment α to X of weight k such that for all truth assignments β to Y it holds that $\alpha \cup \beta$ satisfies ψ. There is evidence that problems that are hard for $\exists^k\forall^*$ do not allow an fpt-reduction to SAT or UNSAT [De Haan and Szeider, 2014b]. Similarly, the intractability class $\exists^*\forall^k$-W[P] has been defined based on an analogous restriction on the weight of assignments for quantified Boolean circuits, a generalization of QBFs. Formally, the class $\exists^*\forall^k$-W[P] consists of all parameterized problems that can be fpt-reduced to the problem $\exists^*\forall^k$-WSAT(CIRC). The input for this problem is a quantified Boolean circuit $\varphi = \exists X \forall Y C$, and an integer k. The parameter is k, and the question is whether there is a truth assignment α to X such that for all truth assignments β to Y of weight k it holds that $\alpha \cup \beta$ satisfies the circuit C.

For an overview of parameterized complexity classes that are relevant to the results in this paper, we refer to Figure 1. For a more detailed discussion, we refer to previous work on this topic [De Haan and Szeider, 2014b].

4 A para-NP-complete Variant

We start our analysis with a parameterized version of planning with uncertainty in the initial state where the plan length is polynomially bounded in the input size. Classically this problem is known to be Σ_2^P-complete [Baral *et al.*, 2000]. More formally, we analyze the following problem:

PLANNING[UNCERTAINTY]
Instance: A planning instance $\mathbb{P} = \langle V, V_u, D, A, I, G \rangle$ similar to a classical planning instance, but containing additional variables V_u that are *unknown* in the initial state, i.e., for all $v \in V_u$ it holds that $I(v) = \mathbf{u}$.
Parameter: $|V_u| + |D|$.
Question: Is there a plan of polynomial length for $\mathbb{P}$ that works for all complete initial states, i.e., each possible way of completing I with a combination of values for variables in V_u?

PLANNING[UNCERTAINTY] turns out to be para-NP-complete and thus admits an fpt-reduction to SAT. In other words, even though the problem is Σ_2^P-complete from the classical perspective, an encoding into SAT is feasible if the number of unknown variables is moderately large, as long as we are only interested in short, i.e., polynomially bounded, plans. In addition, this means that we can characterize the class para-NP by planning with uncertainty in the initial state if the degree of uncertainty, i.e., the number of variables that are unknown, and the size of the domain are considered as parameter.

Theorem 1. *The problem* PLANNING[UNCERTAINTY] *is* para-NP*-complete if the plan length is bounded by a polynomial in the input size.*

Proof. The basic idea of the membership proof is that we can materialize all initial states in the transformation of the planning instance into SAT by an fpt-reduction. Let n be the input size and $m = p(n)$ be a bound on the plan length where p is an arbitrary polynomial that bounds the plan length. To show membership, recall that classical planning is NP-complete if the plan length is at most m. Thus, for a classical planning instance $\mathbb{P}$ we can construct a propositional formula $\varphi[\mathbb{P}, V_A]$ in polynomial time that is satisfiable if and only if there is a plan of length at most m for $\mathbb{P}$, where V_A contains all variables representing actions. Due to the uncertainty in the initial state in PLANNING[UNCERTAINTY], there is not a single initial state, but rather a set of initial states $\mathcal{I}$ of cardinality $|D|^{|V_u|}$. Now let $\mathbb{P}'$ be an instance of PLANNING[UNCERTAINTY] and $I \in \mathcal{I}$. Then, let $\mathbb{P}'(I)$ denote the classical planning instance obtained by instantiating the unknown variables in $\mathbb{P}'$ with the values of the corresponding variables in I. To show membership, we construct the formula $\psi = \bigwedge_{I \in \mathcal{I}} \varphi[\mathbb{P}'(I), V_A]$ of conjuncts that are variable-disjoint (except V_A). Clearly, ψ is satisfiable if and only if $\mathbb{P}'$ is a yes-instance.

For para-NP-hardness, notice that the construction used in a proof by Kronegger *et al.* [2013, Theorem 11] for SAS$^+$ planning even holds if $V_u = \emptyset$. □

5 A para-DP-complete Variant

The complexity class DP consists of all problems that can be reduced to the problem SAT-UNSAT $= \{ (\varphi, \psi) \mid \varphi \in \text{SAT}, \psi \in \text{UNSAT} \}$. The class para-DP is interesting as at most two calls to a SAT solver are required to solve any instance. Therefore, showing membership in para-DP can be a positive result for hard problems. In what follows we will describe a natural planning problem complete for this class.

We consider a variant of planning with uncertainty in the initial state as described in Section 4. The crucial difference is

that we ask whether an action a_0 is *essential*. More formally, we consider the following problem.

PLANNING[ESSENTIAL ACTION]
Instance: A planning instance with uncertainty in the initial state $\mathbb{P} = \langle V, V_u, D, A, I, G\rangle$ and an action $a_0 \in A$.
Parameter: $|V_u| + |D|$.
Question: Is a_0 essential, i.e., is there a plan of polynomial length for $\mathbb{P}$ that uses a_0 and works for all complete initial states, but there is no such plan for $\mathbb{P}$ without using a_0?

We now show that this problem is para-DP-complete and thus can be solved by two SAT-calls for polynomial plan length.

Theorem 2. *The problem* PLANNING[ESSENTIAL ACTION] *is* para-DP*-complete if the plan length is bounded by a polynomial in the input size.*

Proof. To establish hardness, we will show that we can encode an instance of the SAT-UNSAT problem into an instance of PLANNING[ESSENTIAL ACTION]. For this, let (φ, ψ) be a SAT-UNSAT instance. Recall that classical planning is NP-complete if the plan length is bounded by a polynomial in the input size. Therefore, there are two planning instances $\mathbb{P}_\varphi = \langle V_\varphi, V_u, D_\varphi, A_\varphi, I_\varphi, G_\varphi\rangle$ and $\mathbb{P}_\psi = \langle V_\psi, V_u, D_\psi, A_\psi, I_\psi, G_\psi\rangle$ with $V_u = \emptyset$ and with disjoint variables, actions, and domain such that $\mathbb{P}_\varphi$ (resp., $\mathbb{P}_\psi$) has a plan of polynomial length if and only if φ (resp., ψ) is satisfiable. We use now the action a_0 of verify that $\mathbb{P}_\psi$ has indeed no plan.

From $\mathbb{P}_\psi$ we construct an instance $\mathbb{P}'_\psi$ as follows: Let the set of actions $A'_\psi = A_\psi \cup \{a_0 : \{\} \to G_\psi\}$ be defined by adding to A_ψ an additional action with empty precondition that immediately fulfills the goal G_ψ. Further, let $\mathbb{P}'_\psi = \langle V_\psi, V_u, D_\psi, A'_\psi, I_\psi, G_\psi\rangle$. We now combine $\mathbb{P}_\varphi$ and $\mathbb{P}'_\psi$ to a single planning instance $\mathbb{P}^*$ (with uncertainty in the initial state). Notice that this can always be done as the instances are disjoint. The instance of PLANNING[ESSENTIAL ACTION] is then given by $(\mathbb{P}^*, a_0)$. It is now easy to verify that (i) there is a plan for $\mathbb{P}^*$ and (ii) a_0 is essential if and only if φ is satisfiable and ψ is unsatisfiable.

Membership in para-DP can be shown as follows. Let $(\mathbb{P} = \langle V, V_u, D, A, I, G\rangle, a_0)$ be an instance of PLANNING[ESSENTIAL ACTION] and k be a bound on the plan length that is polynomial in the input size. W.l.o.g., we assume that $0, 1 \in D$. We have to check whether there is a plan that uses a_0 and that there is no plan without using a_0. Recall from the proof of Theorem 1 that we can construct in fpt-time for an arbitrary instance $\mathbb{P}$ of PLANNING[UNCERTAINTY] (parameterized by $|V_u| + |D|$) a propositional formula $\varphi[\mathbb{P}]$ that is satisfiable if and only if $\mathbb{P}$ is a yes-instance.

To ensure that action a_0 is indeed used, we create the instance $\mathbb{P}' = \langle V \cup \{v_{a_0}\}, V_u, D, A', I', G'\rangle$ where we introduce the new variable v_{a_0}. Let $I'(v_{a_0}) = 0$, $G'(v_{a_0}) = 1$, $I'(v) = I(v)$ and $G'(v) = G(v)$ for each $v \in V$. We obtain A' from A by adding $v_{a_0} = 1$ to the effect of action a_0. Furthermore, we create for the case where a_0 must not be used the instance $\mathbb{P}'' = \langle V, V_u, D, A \setminus \{a_0\}, I, G\rangle$. The instance of SAT-UNSAT is then given by $(\varphi[\mathbb{P}'], \varphi[\mathbb{P}''])$. In the construction we ensure that there is a plan of (of length at most k) that uses action a_0, and that there is no plan (of length at most k) without using action a_0. □

6 An FPT$^{\mathsf{NP}[f(k)]}$-complete Variant

In this section, we investigate the following planning problem with two different types of goals: (i) a *hard goal* G_h that needs to be satisfied, and (ii) a *soft goal* G_s for which the number of variables satisfied according to G_s is to be maximized. We call a plan *optimal* with respect to some given bound k on the plan length if there does not exist another plan with length at most k that satisfies more variables according to the soft goal G_s. We analyze the problem of finding an optimal plan given a planning instance $\mathbb{P} = \langle V, D, A, I, G_h, G_s\rangle$ and a bound k on the plan length, parameterized by $|G_s|$.

In general, finding an optimal plan for $\mathbb{P}$ given a bound k on the plan length in polynomial time requires $\mathcal{O}(\log |\mathbb{P}|)$ many SAT calls. We can find such a plan by performing binary search on the number of fulfilled variables in the soft goal, asking whether $\leq \ell$ variables in the soft goal can be fulfilled. We show that we can restrict the number of SAT calls to a function of the number of variables in the soft goal, i.e., $|G_s|$. We believe that this result indicates that the SAT approach is still feasible for this (parameterized) problem. First, we show that planning with soft goals can be solved in fpt-time using $\lceil \log |G_s| \rceil$ SAT calls. Second, we prove that the number of required SAT calls cannot be bounded by a constant.

Proposition 3. *Let $\mathbb{P} = \langle V, D, A, I, G_h, G_s\rangle$ be a planning instance with a hard goal G_h and a soft goal G_s, and let k be an integer that is polynomial in the input size. Then finding a plan of length at most k that is optimal for $\mathbb{P}$ with respect to k (if it exists) can be done in fpt-time using $\lceil \log |G_s| \rceil$ SAT calls.*

Proof (sketch). The problem of deciding whether there is a plan of length at most k that reaches some state s' satisfying the hard goal G_h and that agrees with the soft goal G_s on at least u variables is in NP. Namely, one can guess such a plan, and verify whether it satisfies the requirements. Therefore, any instance of this problem can be encoded into an instance of SAT in polynomial time. Moreover, from a satisfying assignment (if one exists) for such a SAT instance, we can extract in polynomial time a plan that satisfies the requirements. Then we can find the maximum number u of variables contained in the soft goal that can be fulfilled using $\lceil \log |G_s| \rceil$ SAT calls, by using binary search. Moreover, using the satisfying assignments that are given by the SAT solver, we can find a plan that is optimal for $\mathbb{P}$ with respect to length k. □

In order to show that we cannot find such an optimal plan in fpt-time using a constant number of SAT calls, we consider the following parameterized decision problem.

PLANNING[SOFT GOAL]-PARITY
Instance: A planning instance $\mathbb{P} = \langle V, D, A, I, G_h, G_s\rangle$ with a hard goal G_h and a soft goal G_s, and an integer k.
Parameter: $|G_s|$.
Question: Does there exist a plan of length at most k that is optimal for $\mathbb{P}$ (w.r.t. k), and that satisfies an odd number of variables of the soft goal?

We show that this problem is complete for the class FPT$^{\mathsf{NP}[f(k)]}$. It follows that if we could find an optimal plan in fpt-time using a constant number of SAT calls (where the

parameter is $|G_s|$), then the Polynomial Hierarchy would collapse [Endriss *et al.*, 2014, Proposition 18].

Theorem 4. *The problem* PLANNING[SOFT GOAL]-PARITY *is* $\mathsf{FPT}^{\mathsf{NP}[f(k)]}$*-complete.*

Proof (sketch). Membership in $\mathsf{FPT}^{\mathsf{NP}[f(k)]}$ follows directly from Proposition 3. To show hardness, we reduce from the problem SMALL-MAX-MODEL-PARITY. For this problem, the input is a CNF formula φ, and a subset $X \subseteq \mathsf{Var}(\varphi)$, where $\mathsf{Var}(\varphi)$ denotes the set of propositional variables of φ. The parameter is $k = |X|$. The question is whether the satisfying assignment that sets the maximum number of variables $x \in X$ to true, satisfies an odd number of variables in X. This problem can straightforwardly be shown to be $\mathsf{FPT}^{\mathsf{NP}[f(k)]}$-complete, using results by Endriss *et al.* [2014, Theorem 15]. (Due to space limitations, we omit the proof.)

Let (φ, X) specify an instance of SMALL-MAX-MODEL-PARITY, where φ has u clauses. We construct a planning instance $\mathbb{P} = \langle V, D, A, I, G_h, G_s \rangle$, with a hard goal G_h and a soft goal G_s, and an integer k' bounding the plan length as follows. We let $V = \mathsf{Var}(\varphi) \cup \{ z_x \mid x \in \mathsf{Var}(\varphi) \} \cup \{ y_\delta \mid \delta \text{ is a clause of } \varphi \}$, and $D = \{0, 1\}$. For each variable $x \in \mathsf{Var}(\varphi)$, we introduce two actions: $a_0^x : \{\} \to \{x = 0, z_x = 1\}$ and $a_1^x : \{\} \to \{x = 1, z_x = 1\}$. Moreover, for each clause δ in φ and each literal l in δ, we introduce an action: a_l^δ whose precondition requires all variables z_x to have value 1 and requires the variable of l to be set in such a way that δ is satisfied, and whose effect ensures that y_δ is set to 1. We let $I(v) = 0$ for all $v \in V$, $G_h = \{ y_\delta = 1 \mid \delta \text{ is a clause of } \varphi \}$, and $G_s = \{ x = 1 \mid x \in X \}$. Finally, we let $k' = |\mathsf{Var}(\varphi)| + u$.

The intuition behind this reduction is that the actions a_i^x can be used to enforce a truth assignment over the variables in $\mathsf{Var}(\varphi)$, and that the actions a_l^δ can be used to check that such a truth assignment satisfies all clauses δ of φ. It is straightforward to verify that those plans consisting of k' actions – an action a_i^x (for some $i \in \{0, 1\}$) for each $x \in \mathsf{Var}(\varphi)$, and an action a_l^δ (for some $l \in \delta$) for each $\delta \in \varphi$ – correspond to truth assignments over $\mathsf{Var}(\varphi)$ that satisfy φ. Moreover, only plans of this form can satisfy the hard goal. Then, finding such a plan that maximizes the number of fulfilled variables in the soft goal corresponds to a satisfying truth assignment that maximizes the number of satisfied variables in X. From this, the correctness of the reduction follows. □

7 An $\exists^k\forall^*$-complete Variant

In this section we reconsider planning with uncertainty in the initial state as discussed in Section 4 and 5 with a different parameter. This time we allow for an unbounded number of unknown variables and parameterize by the bound k on the plan length. It turns out that the problem does not retain its full hardness although a feasible translation to SAT still seems to be unlikely. To be more precise, we show that this problem is $\exists^k\forall^*$-complete.

For membership we build upon the $\exists\forall$-encoding of Rintanen [2007] for planning instances with uncertainty in the initial state and binary domain. There, a QBF of the form $\exists X \forall Y \psi$, where ψ is a quantifier-free formula, is satisfiable if and only if there is a plan whose length is bounded by a given integer. Observe that X only contains variables representing actions. Furthermore, for any satisfying truth assignment the weight of X is equal to the plan length. Notice that this encoding assumes binary domains. Here, we consider planning instances with an arbitrarily large domain. However, since the domain is part of the input, the reduction also works for arbitrarily large domains. Thus, the aforementioned encoding together with the observations above allows to show membership in $\exists^k\forall^*$. Note that other encodings could in principle also be used for this purpose.

Corollary 5. *Planning with uncertainty in the initial state parameterized by the plan length is in* $\exists^k\forall^*$.

Next we show hardness. Note that this reduction makes use of conditional effects.

Theorem 6. *Planning with uncertainty in the initial state parameterized by the plan length is* $\exists^k\forall^*$*-hard.*

Proof. We reduce from $\exists^k\forall^*$-WSAT over QBFs in 3DNF, which is known to be $\exists^k\forall^*$-complete [De Haan and Szeider, 2014b]. Let (φ, k') be an instance of $\exists^k\forall^*$-WSAT, where k' is an integer, $\varphi = \exists X \forall Y \psi$, and $\psi = \bigvee_{1 \le i \le m} \bigwedge_{1 \le j \le 3} l_{i,j}$.

The variables of the planning instance $\mathbb{P}$ are $V = X \cup \{c_1, \ldots, c_{k'}\} \cup \{e, g\}$. For every $v \in X$ and $1 \le i \le k'$ we introduce an action $a_v^i : \{v = c_i = e = 0\} \to \{v = c_i = 1\}$. Furthermore, we introduce two additional actions: $a_e : \{c_1 = 1, \ldots, c_{k'} = 1\} \to \{e = 1\}$, and $a_g : \{e = 1\} \to \big\{\{l_{1,1} = l_{1,2} = l_{1,3} = 1\} \triangleright \{g = 1\}, \ldots, \{l_{m,1} = l_{m,2} = l_{m,3} = 1\} \triangleright \{g = 1\}\big\}$. The set of actions A is then given by $A = \bigcup_{v \in X} \bigcup_{1 \le i \le k'} \{a_v^i\} \cup \{a_e, a_g\}$. To obtain the instance $\mathbb{P}$, we set $V_u = Y$, $D = \{0, 1\}$, $I = 0^{|V|}$ and $G = \{g = 1\}$.

The intuition of this encoding is to first guess an assignment α of weight k' using k' distinct actions a_v^i, then to fix this assignment using action a_e and finally to evaluate α according to φ with the action a_g. It is straightforward to check that (φ, k') is a yes-instance if and only if there is a plan of length $k = k' + 2$ for $\mathbb{P}$. □

8 An $\exists^*\forall^k$-W[P]-complete Variant

We now turn to a variant of planning with uncertainty in the initial state, i.e., planning with bounded deviation. Let $\mathbb{P} = \langle V, V_u, D, A, I, G \rangle$ be such a planning instance. W.l.o.g., we assume $0 \in D$ which we call the *base value* for variables in V_u. Here, the parameter describes the maximum number of unknown variables that deviate from the base value. In other words, in this setting, there can be many unknown variables but only few of them have unexpected values – we however, do not know which. More formally, the problem is defined as follows.

PLANNING[BOUNDED DEVIATION]
Instance: A planning instance $\mathbb{P} = \langle V, V_u, D, A, I, G \rangle$ and two integers k and d.
Parameter: d.
Question: Does there exist a plan of length at most k that works for all complete initial states where at most d unknown variables deviate from the base value?

Before we present the main result of this section we provide a way of showing membership in $\exists^*\forall^k$-W[P] by means of Turing

machines. In particular, we show that $\exists^*\forall^k$-W[P] contains those parameterized decision problems that can be decided by a certain class of alternating Turing machines.

An *alternating Turing machine (ATM)* with m tapes is a 6-tuple $\mathbb{M} = (S_\exists, S_\forall, \Sigma, \Delta, s_0, F)$, where: $S_\exists$ and $S_\forall$ are disjoint sets; $S = S_\exists \cup S_\forall$ is the finite set of *states*; Σ is the alphabet; $s_0 \in S$ is the *initial state*; $F \subseteq S$ is the set of *accepting states*; and $\Delta \subseteq S \times (\Sigma \cup \{\$, \square\})^m \times S \times (\Sigma \cup \{\$\})^m \times \{\mathbf{L}, \mathbf{R}, \mathbf{S}\}^m$ is the *transition relation*. We use the same notation as Flum and Grohe [Flum and Grohe, 2006, Appendix A.1]; for further details, we refer to their textbook.

We consider the following restrictions on ATMs. An $\exists\forall$-*Turing machine* (or simply $\exists\forall$-machine) is a 2-alternating ATM $(S_\exists, S_\forall, \Sigma, \Delta, s_0, F)$, where $s_0 \in S_\exists$. Let P be a parameterized problem. An $\exists^*\forall^k$-W[P]-*machine for* P is a $\exists\forall$-machine $\mathbb{M}$ such that there exists a computable function f and a polynomial p such that: (i) $\mathbb{M}$ decides P in time $f(k)p(|x|)$; and (ii) for all instances (x, k) of P and each computation path R of $\mathbb{M}$ with input (x, k), at most $f(k)\log|x|$ of the universal configurations of R are nondeterministic. We say that a parameterized problem P *is decided by some* $\exists^*\forall^k$-W[P]-*machine* if there exists a $\exists^*\forall^k$-W[P]-machine for P.

Theorem 7. *If a parameterized problem Q is decided by some $\exists^*\forall^k$-W[P]-machine, then $Q \in \exists^*\forall^k$-W[P].*

Proof (sketch). We describe a way of constructing, for each instance (x, k) of Q, an instance (φ, k') of $\exists^*\forall^k$-WSAT(CIRC) that is a yes-instance if and only if $(x, k) \in Q$. Our construction is based on the proof of Cook's Theorem [Cook, 1971].

We begin with some observations. Let $\mathbb{M}$ be an $\exists^*\forall^k$-W[P]-machine for Q. We may assume without loss of generality that for any nondeterministic transition of $\mathbb{M}$, there are exactly two possible ways of proceeding. Any run of $\mathbb{M}$ on input (x, k) can be specified entirely by indicating what nondeterministic choices $\mathbb{M}$ makes. Given (a representation of) these nondeterministic choices, determining whether this run of $\mathbb{M}$ is an accepting run can be done in fpt-time in (x, k) – simply by simulating $\mathbb{M}$ using the given choices. In other words, to decide whether $\mathbb{M}$ accepts an input (x, k), we need to decide whether there exists some sequence s_1 of nondeterministic choices for the existential phase of the computation, such that for all sequences s_2 of nondeterministic choices for the universal phase of the computation it holds that the run of $\mathbb{M}$ on (x, k) that is specified by (s_1, s_2) is an accepting run.

By definition of $\exists^*\forall^k$-W[P]-machines, we know that there is some computable function f and some constant c such that for each input (x, k) with $|x| = n$, (i) $\mathbb{M}$ runs in time $f(k)n^c$ and (ii) $\mathbb{M}$ makes at most $f(k)\log n$ nondeterministic choices in the universal phase of the computation. Therefore, in particular, in the existential phase of the computation $\mathbb{M}$ makes at most $f(k)n^c$ nondeterministic choices. We can encode all possibilities for the $2^{f(k)n^c}$ many different possible combinations of choices that $\mathbb{M}$ makes in the existential phase of the computation using $f(k)n^c$ many existential variables of φ. Moreover, since there are at most $2^{f(k)\log n} = n^{f(k)}$ many different possible combinations of choices that $\mathbb{M}$ makes in the universal phase of the computation, we can encode these possibilities using n universal variables, whose assignments are restricted to set only $k' = f(k)$ many variables to true.

The circuit φ then simulates the behavior of $\mathbb{M}$ on input (x, k) with the nondeterministic behavior given by (s_1, s_2) as specified by the assignment to the variables. Since such a simulation can be done in fpt-time, we know we can encode this simulation in a circuit φ that can be constructed in fpt-time. Then, (φ, k') is a yes-instance of $\exists^*\forall^k$-WSAT(CIRC) if and only if $\mathbb{M}$ accepts (x, k), which is the case if and only if $(x, k) \in Q$. □

The result for the other direction, i.e., if a parameterized problem $Q \in \exists^*\forall^k$-W[P] then Q is decided by some $\exists^*\forall^k$-W[P]-machine, is omitted due to space limitations. Now we are ready to present the main result of this section.

Theorem 8. *The problem* PLANNING[BOUNDED DEVIATION] *is $\exists^*\forall^k$-W[P]-complete if the plan length k is bounded by a polynomial in the input size.*

Proof. We describe an algorithm to solve the problem that can be implemented by an $\exists^*\forall^k$-W[P]-machine. The algorithm first guesses a sequence ω of m actions from A using $m\log|A|$ (binary) nondeterministic choices in the existential phase of the computation. Then, in the universal phase, it verifies whether this plan works for all cases where k of the unknown variables deviate from the base value in the initial state. Each such initial state can be specified using $k\log(|V_u| \cdot |D|)$ nondeterministic choices (in the universal phase), and for any such initial state I, checking whether the ω reaches a goal state from I can be done in polynomial time. This algorithm is correct and can be implemented by an $\exists^*\forall^k$-W[P]-machine. Then, by Proposition 7, the result follows.

To show hardness, we reduce from $\exists^*\forall^k$-WSAT(CIRC). Intuitively, in the reduction, we will emulate the evaluation of the circuit by a planning problem. Here, the goal is to set the output gate to true and the unknown variables are used to model the universally quantified variables of the circuit. With help of the actions we sequentially evaluate the output of the gates in a fixed order to finally compute the value of the output gate of the circuit. Recall that an instance of $\exists^*\forall^k$-WSAT(CIRC) consists of a Boolean circuit C over two sets of disjoint input variables X and Y, and an integer k'.

Since a circuit can be seen as an acyclic directed graph, we may assume that the gates $g_1, \ldots, g_m$ are numbered in such a way that for each gate g_i, its inputs $g_{j_1}, \ldots, g_{j_\ell}$ are numbered in such a way that $j_1, \ldots, j_\ell < i$. This numbering gives a natural ordering of the gates, which ensures that all input values for a gate are already computed if the output value is to be determined. We create the set $L = \{l_1, \ldots, l_m\}$ of variables. Then we introduce a new variable f and the action $a_f : \{f = 0\} \to \{f = 1\}$. Furthermore, for each variable $x \in X$, we create the action $a_x : \{f = 0\} \to \{x = 1\}$. For each gate g_i, we create an action a_{g_i} where:

$$\mathsf{pre}(a_{g_i}) = \{f = 1, l_{i-1} = 1\}$$
$$\mathsf{eff}(a_{g_i}) = \{l_i = 1\} \cup \Gamma_{g_i}$$

Here Γ_{g_i} depends on the type of gate g_i and is obtained as follows. For each gate g_i we introduce the variable $o(g_i)$, representing the unnegated output of gate g_i. Let the unnegated input gates of gate g be $g'_1, \ldots, g'_p$, and the negated ones be $g''_1, \ldots, g''_l$. To simplify the presentation, we assume that each

variable in Y is also represented by a gate g, whose output is represented as $o(g)$. If g is an AND-gate we define:

$$\Gamma_{g_i} = \big\{\{o(g'_1) = \cdots = o(g'_p) = 1, o(g''_1) = \cdots = o(g''_l) = 0\} \triangleright \{o(g) = 1\}\big\}$$

If g is an OR-gate, Γ_{g_i} is defined analogously. Notice that the effect in the set Γ_{g_i} are conditional. Intuitively, executing the actions $a_{g_1}, \ldots, a_{g_m}$ in order corresponds to evaluating the circuit using the given values for the variables in X and Y. Moreover, executing these actions is the only way to set $o(g_o)$ to 1, where g_o is the output gate of C.

To put things together, we set $V = X \cup L \cup \{f\} \cup \{o(g) \mid g \text{ is a gate in } C\}$, $V_u = Y$, $A = \{a_f\} \cup \{a_x \mid x \in X\} \cup \{a_g \mid g \text{ is a gate in } C\}$, $I(v) = 0$ for each $v \in V$, $G = \{o(g_o) = 1\}$, where g_o is the output gate of C, $k = m + |X|$, and $d = k'$. The correctness of this reduction can be checked straightforwardly. □

9 Conclusion

For a variety of planning problems we have investigated the limits of a reduction-based approach to SAT. In particular, classes such as para-NP, para-DP, and $\mathsf{FPT}^{\mathsf{NP}[f(k)]}$ give hope for an efficient transformation to SAT (as long as the parameters are of moderate size). We believe that our results for different variants of planning provide a versatile toolbox for showing membership in these classes. This is desirable because planning is an expressive formalism that allows for easy encodings of other problems. Thus, our results help to explore the limits of efficient reducibility to SAT also for other important AI problems different to planning.

For future work we plan to extend this complexity map to other problems from different fields. It is also interesting to investigate how the performance of problems from different classes behaves in practice. Furthermore, it is desirable to find natural problems for the more exotic classes in this complexity landscape.

References

[Arora and Barak, 2009] Sanjeev Arora and Boaz Barak. *Computational Complexity – A Modern Approach*. Cambridge University Press, 2009.

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

[Baral *et al.*, 2000] Chitta Baral, Vladik Kreinovich, and Raul Trejo. Computational complexity of planning and approximate planning in the presence of incompleteness. *Artificial Intelligence*, 122(1-2):241–267, 2000.

[Cook, 1971] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proc. 3rd Annual Symp. on Theory of Computing*, pages 151–158, 1971.

[Downey and Fellows, 1999] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999.

[Downey and Fellows, 2013] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[Endriss *et al.*, 2014] Ulle Endriss, Ronald de Haan, and Stefan Szeider. Parameterized complexity results for agenda safety in judgment aggregation. In *Proc. COMSOC 2014*. Carnegie Mellon University, 2014.

[Fichte and Szeider, 2013] Johannes Klaus Fichte and Stefan Szeider. Backdoors to normality for disjunctive logic programs. In *Proc. AAAI 2013*, pages 320–327. AAAI Press, 2013.

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

[Flum and Grohe, 2006] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006.

[Gomes *et al.*, 2008] Carla P. Gomes, Henry Kautz, Ashish Sabharwal, and Bart Selman. Satisfiability solvers. In *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*, pages 89–134. Elsevier, 2008.

[de Haan and Szeider, 2014a] Ronald de Haan and Stefan Szeider. Fixed-parameter tractable reductions to SAT. In *Proc. SAT 2014*, volume 8561 of *Lecture Notes in Computer Science*, pages 85–102. Springer, 2014.

[de Haan and Szeider, 2014b] Ronald de Haan and Stefan Szeider. The parameterized complexity of reasoning problems beyond NP. In *Proc. KR 2014*, pages 82–91. AAAI Press, 2014.

[Kronegger *et al.*, 2013] Martin Kronegger, Andreas Pfandler, and Reinhard Pichler. Parameterized complexity of optimal planning: A detailed map. In *Proc. IJCAI 2013*, pages 954–961. AAAI Press, 2013.

[Malik and Zhang, 2009] Sharad Malik and Lintao Zhang. Boolean satisfiability from theoretical hardness to practical success. *Communications of the ACM*, 52(8):76–82, 2009.

[Niedermeier, 2006] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Mathematics and its Applications. Oxford University Press, 2006.

[Palacios and Geffner, 2009] Héctor Palacios and Hector Geffner. Compiling uncertainty away in conformant planning problems with bounded width. *Journal of Artificial Intelligence Research*, 35:623–675, 2009.

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

[Pfandler *et al.*, 2013] Andreas Pfandler, Stefan Rümmele, and Stefan Szeider. Backdoors to abduction. In *Proc. IJCAI 2013*, pages 1046–1052. AAAI Press, 2013.

[Rintanen, 2007] Jussi Rintanen. Asymptotically optimal encodings of conformant planning in QBF. In *Proc. AAAI 2007*, pages 1045–1050. AAAI Press, 2007.

[Sakallah and Marques-Silva, 2011] Karem A. Sakallah and João Marques-Silva. Anatomy and empirical evaluation of modern SAT solvers. *Bulletin of the European Association for Theoretical Computer Science*, 103:96–121, 2011.