A Structural Complexity Analysis of Hierarchical Task Network Planning

Cornelius Brand[1], **Robert Ganian**[2], **Fionn Mc Inerney**[3] and **Simon Wietheger**[2]
[1]Algorithms & Complexity Theory Group, Regensburg University, Germany
[2]Algorithms and Complexity Group, TU Wien, Austria
[3]Telefónica Scientific Research, Barcelona, Spain
cornelius.brand@ur.de, {rganian, fmcinern}@gmail.com, swietheger@ac.tuwien.ac.at

Abstract

We perform a refined complexity-theoretic analysis of three classical problems in the context of Hierarchical Task Network planning: the verification of a provided plan, whether an executable plan exists, and whether a given state can be reached. Our focus is to identify structural properties yielding tractability. We obtain new polynomial-time algorithms for all three problems on a natural class of primitive networks, along with corresponding lower bounds. We also obtain an algorithmic meta-theorem for lifting polynomial-time solvability from primitive to general task networks, and prove that its preconditions are tight. Lastly, we analyze the parameterized complexity of the three problems.

1 Introduction

Automated Planning is a core research topic whose goal is to devise computational methods capable of finding solutions to prominent planning tasks. In this work, we consider automated planning in the context of *Hierarchical Task Networks (HTNs)*, which have received significant attention in the artificial intelligence research community. HTNs are capable of incorporating the compound, hierarchical structure of real-world tasks, as opposed to the fine-grained view of classical planning that operates directly on the state space of the system under consideration, and are one of the best-established planning formalisms in the literature [Tsuneto *et al.*, 1996; Li *et al.*, 2009; Sohrabi *et al.*, 2009; Geier and Bercher, 2011; Alford *et al.*, 2015b; Xiao *et al.*, 2017; Behnke *et al.*, 2019; Höller *et al.*, 2019; Lin and Bercher, 2023].

On a high level, HTNs can be informally described and illustrated as follows: at the top level of the task hierarchy is a set of tasks that is to be executed in a (partially specified) order. For instance, the task of going to work may be decomposed into three subtasks: leaving the house, going the actual distance, and getting set up at the office. Now, each of these tasks can in turn be decomposed into a set of smaller tasks, like climbing the stairs or taking a bus, which again can be decomposed, and so forth. This continues until we reach an elementary level of tasks that are no longer decomposable, which correspond to single executable *actions*.

The crucial feature of the model is that it separates the user-facing description of a task (what we want to do) from its internal implementation (how to do it). For instance, leaving the house needs to be done *before* going the actual distance, regardless of how it is done. However, in an HTN, we may have *several* options of how to decompose this task: the agent can, e.g., either climb the stairs or take the elevator.

Perhaps the most natural problem associated with HTNs is PLAN VERIFICATION: can the tasks in a given network be decomposed and then ordered to match a given, executable sequence of actions? A second fundamental problem is PLAN EXISTENCE: can the tasks in a given network be decomposed and then ordered in any way that yields an executable sequence of actions? The literature has investigated the complexity of several variants of the aforementioned problems [Erol *et al.*, 1996; Geier and Bercher, 2011; Alford *et al.*, 2015b; Behnke *et al.*, 2015; Lin and Bercher, 2023]. Moreover, we also consider the question of whether a specified target state can be reached, formalized as the STATE REACHABILITY problem. This is the typical goal in classical planning [Fikes and Nilsson, 1971; Chapman, 1987; Bäckström and Nebel, 1995], and has also been considered in the HTN setting [Höller and Bercher, 2021; Olz *et al.*, 2021].

It is well-established that problems on HTNs such as those listed above remain computationally intractable even in severely restricted settings. These results usually depend on restricting the decompositions, e.g., requiring acyclicity [Alford *et al.*, 2015a; Chen and Bercher, 2021], restricting the impact of actions on the state space (e.g., delete-relaxed HTN planning) [Alford *et al.*, 2014; Höller *et al.*, 2020], or by allowing the insertion of tasks at arbitrary positions [Geier and Bercher, 2011; Alford *et al.*, 2015b].

In this article, we take a different approach and instead focus on structural restrictions of the task network. While there are some complexity results known for the extreme cases of a total ordering or no ordering of the tasks [Geier and Bercher, 2011; Alford *et al.*, 2014; Behnke and Speck, 2021], other restrictions received little to no attention so far. Building on these first results, we identify interesting and natural special cases where pockets of polynomial-time solvability show themselves. Moreover, we prove that these pockets give rise to a surprisingly rich complexity-theoretic landscape.

Our Contributions. We first consider *primitive* task networks, i.e., HTNs in which there are no compound tasks.

While this may seem restrictive, it is a natural place to start from a complexity-theoretic perspective: all three of the considered problems remain NP-hard on primitive task networks. In fact, PLAN VERIFICATION is known to be NP-hard even if the network forms a collection of disjoint stars [Lin and Bercher, 2023]. Moreover, unlike PLAN VERIFICATION, the latter two problems remain NP-hard even on trivial (in particular, edgeless) networks when the state space is not of fixed size [Bylander, 1994; Erol *et al.*, 1996]. Thus, in line with previous related works, we henceforth consider PLAN EXISTENCE and STATE REACHABILITY exclusively in the setting of constant-sized state spaces [Kronegger *et al.*, 2013; Bäckström *et al.*, 2015]. Even in this setting, we establish the NP-hardness of the three considered problems on networks consisting of a collection of independent chains.

While these initial results may seem discouraging, it turns out that all three problems are polynomial-time solvable on networks which are chains (i.e., totally-ordered networks) [Erol *et al.*, 1996; Behnke *et al.*, 2015] or antichains (Theorems 3 and 4). We generalize and unify these tractability results by showing that all three considered problems are polynomial-time solvable on networks of bounded "generalized partial order width", that is, the partial order width of the network when ignoring isolated elements (Theorems 5 and 6).

Building on the above results for the primitive case, in the second part of the paper, we establish an *algorithmic meta-theorem* that allows us to precisely describe the conditions under which polynomial-time tractability for primitive task networks can be lifted to the compound case (Theorem 8). Crucially, we prove that this result is tight in the sense that all of the required conditions are *necessary* (Theorem 9).

In the final part of the paper, we push beyond classical complexity and analyze the considered problems from the perspective of the more fine-grained *parameterized complexity* paradigm [Cygan *et al.*, 2015]. There, one analyzes problems not only w.r.t. the input size n, but also w.r.t. a specified parameter $k \in \mathbb{N}$, where the aim is to obtain algorithms which run in time at most $f(k) \cdot n^{\mathcal{O}(1)}$ for some computable function f (so-called *fixed-parameter algorithms*); problems admitting such algorithms are called *fixed-parameter tractable* (FPT).

The natural question from the parameterized perspective is whether the aforementioned tractability results for HTNs with constant-sized structural measures (Theorems 5 and 6) can be lifted to fixed-parameter tractability when parameterized by the same measures. While analogous questions have been thoroughly explored—and often answered in the affirmative—in many other areas of artificial intelligence [Eiben *et al.*, 2021; Ganian and Korchemna, 2021; Froese *et al.*, 2022; Heeger *et al.*, 2023], here we prove that the generalized partial order width does not yield fixed-parameter tractability for any of our problems on primitive task networks, under well-established complexity assumptions. Moreover, the aforementioned hardness of the problems on forests of stars already rules out fixed-parameter tractability for essentially all standard structural graph parameters, except for the *vertex cover number* (vcn) [Blazej *et al.*, 2023; Bodlaender *et al.*, 2023; Chalopin *et al.*, 2024]. We complement these lower bounds by designing highly non-trivial fixed-parameter algorithms for all three considered problems on primitive task networks w.r.t. the vertex cover number of the network. For compound HTNs, we then adapt our general algorithmic meta-theorem for polynomial-time solvability to the fixed-parameter paradigm (Theorem 12), allowing us to also extend these results to the non-primitive setting. See Fig. 1 for a schematic overview of our results.

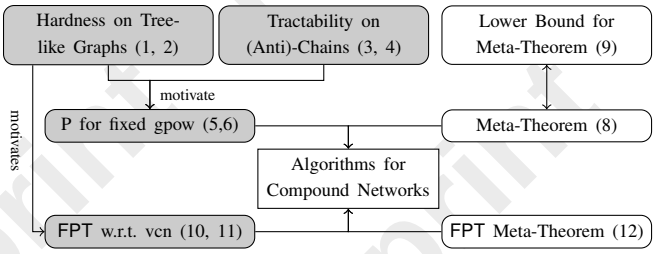


Figure 1: Overview of our results and respective theorems. Generalized Partial Order Width is abbreviated to *gpow*, and Vertex Cover Number to *vcn*. Results in gray boxes refer to primitive instances.

Related Work. HTN planning has been extensively studied in artificial intelligence research and has applications in areas such as healthcare [González-Ferrer *et al.*, 2013b], business processes [González-Ferrer *et al.*, 2013a], human-robot interaction [Hayes and Scassellati, 2016], emergency response [Liu *et al.*, 2016; Zhao *et al.*, 2017], and visualization [Padia *et al.*, 2019]. For an overview of the current state of HTN planning, see the works by Bercher, Alford, and Hoeller [2019] and Georgievski and Aiello [2015].

Questions on the complexity and even computability of many tasks associated with HTNs have been treated in the literature [Erol *et al.*, 1994; Erol *et al.*, 1996; Geier and Bercher, 2011; Alford *et al.*, 2015a; Olz *et al.*, 2021; Lin and Bercher, 2023; Lin *et al.*, 2024a; Lin *et al.*, 2024b]. Albeit carried out in a different context, the work of Kronegger, Pfandler, and Pichler [2013] on propositional STRIPS with negation (PSN) studies a problem (k-PSN) which is related to STATE REACHABILITY. The main difference is that while PSN includes a broader definition of actions, it does not consider any primitive or hierarchical network structure. Finally, the computational complexity of planning has also been studied in settings beyond HTNs [Bylander, 1994; Bäckström *et al.*, 2012; Bäckström *et al.*, 2013; Bäckström *et al.*, 2015].

2 Preliminaries

For $n \in \mathbb{N}$, let $[n] = \{1, \dots, n\}$. We use standard graph terminology [Diestel, 2012]. The *partial order width* w of a partial order $\lesssim$ over a set S is the maximum number of pairwise-disjoint maximal chains contained in $\lesssim$ or, equivalently, the size of its largest antichain. An element $s \in S$ is *isolated* under $\lesssim$ if it is incomparable to all other elements of S. We define the *generalized partial order width* of $\lesssim$ as the partial order width of $\lesssim$ after removing isolated elements from S.

Hierarchical Task Networks. Our formalization of HTNs follows the terminology employed in recent works [Olz *et al.*, 2021; Lin and Bercher, 2023]. The *domain* of a planning problem is a tuple $(F, A, \mathcal{C}, \delta, M)$, where F is a finite set of

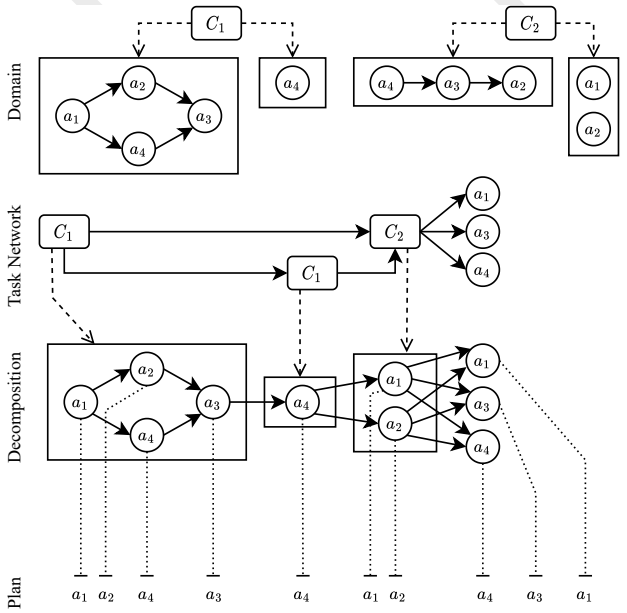
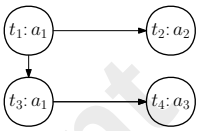


Figure 2: Top: domain with actions $a_1, \ldots, a_4$, compound task names C_1, C_2, and decomposition methods as indicated by dashed arrows; δ and F are not shown. Center: example task network and its decomposition into a primitive task network, where solid arrows indicate $\prec$. Bottom: a plan for the network, where dotted lines indicate the order of linearization (executability would depend on δ).

propositions, A is a set of *actions* (or *primitive task names*), and $\mathcal{C}$ is a set of *compound task names* such that $A \cap \mathcal{C} = \emptyset$. The function $\delta\colon A \to 2^F \times 2^F \times 2^F$ maps each action to a set $\mathrm{prec}(a)$ of preconditions, a set $\mathrm{del}(a)$ of propositions to be removed, and a set $\mathrm{add}(a)$ of propositions to be added to the current state. Here, a state is a subset of the propositions in F. An action a is *executable* in a state $s \subseteq F$ if $\mathrm{prec}(a) \subseteq s$. If executed, it changes the state s to $s' = (s \setminus \mathrm{del}(a)) \cup \mathrm{add}(a)$. A *plan* is a sequence of actions and is executable in a state s if its actions can be executed consecutively starting from s. Lastly, M represents the *decomposition methods*. M is a mapping that assigns to each $c \in \mathcal{C}$ a set $M(c)$ of task networks (defined below) which c can be decomposed into.

A *task network* in such a domain is a tuple $(T, \prec^+, \alpha)$ consisting of a set T of task identifiers (or simply *tasks*) with a partial order $\prec^+ \subseteq T \times T$ and a function $\alpha\colon T \to A \cup \mathcal{C}$.

The usual representation of a task network is its cover graph $D_\prec = (T, \prec)$, where $\prec$ is the cover of $\prec^+$ (in particular, $(a, b) \in \prec$ if and only if $a \prec^+ b$ and there is no c such that $a \prec^+ c \prec^+ b$). We denote the underlying undirected graph by $G_\prec$. We include two examples to illustrate these definitions. In Fig. 2, the high-level interplay of the components of the formalism is shown, without an explicit state transition function. The latter is shown in a smaller example in Fig. 3. A subset T' of tasks is a *chain* if $\prec^+$ defines a total order on T', and an *antichain* if all of its tasks are pairwise incomparable. A task t is *primitive* if $\alpha(t) \in A$, and *compound* otherwise. A task network is *primitive* if it contains no compound tasks. For a primitive task network, a total order $\bar{t}$ on T such that $t \prec t'$ implies that t precedes t' in $\bar{t}$ is called a



Action	prec	del	add
a_1	$\emptyset$	$\emptyset$	$\{1\}$
a_2	$\emptyset$	$\emptyset$	$\{2\}$
a_3	$\{2\}$	$\{1\}$	$\emptyset$

Figure 3: The digraph $D_\prec$ for a primitive HTN with tasks $T = \{t_1, t_2, t_3, t_4\}$, actions $A = \{a_1, a_2, a_3\}$, propositions $F = \{1, 2\}$, and initial state $s_0 = \emptyset$. A task's action is in its node. There is an arc from a task t to a task t' if $(t, t') \in \prec$. Here, the only sequences that execute all tasks are t_1, t_2, t_3, t_4 and t_1, t_3, t_2, t_4.

linearization. Enumerating the actions $\alpha(t)$ in order of $\bar{t}$ defines an action sequence $\alpha(\bar{t})$. Non-primitive task networks can be decomposed into primitive ones via the methods in M; decomposing a task t into a subnetwork $tn_m \in M(\alpha(t))$ means replacing t by tn_m and modifying $\prec$ such that for all tasks t' in tn_m and $t_1 \in T - \{t\}$, we have $t' \prec t_1$ or $t_1 \prec t'$ if and only if $t \prec t_1$ or $t_1 \prec t$ were true originally, respectively.

The input of an *HTN problem* contains a domain $D = (F, A, \mathcal{C}, \delta, M)$, an initial task network tn in that domain, and an initial state $s_0 \subseteq F$.[1] In a generalization of the literature, we consider a primitive task network $tn' = (T', \prec^{+\prime}, \alpha')$ to be a *solution* to a triple (D, tn, s_0) if tn' has a linearization $\bar{t}$ such that $\alpha'(\bar{t})$ is executable in s_0 and tn decomposes into a primitive task network $(T^*, \prec^{+*}, \alpha^*)$ with $T' \subseteq T^*$, for each $(t, t') \in \prec^{+*}$ with $t' \in T'$ we have that $t \in T'$ and $(t, t') \in \prec^{+\prime}$, and $\alpha'(t) = \alpha^*(t)$ for all $t \in T'$. The literature usually considers the case where tn can be decomposed into tn', that is, $T^* = T'$, in which we call tn' a *full solution*. Intuitively, non-full solutions respect δ and $\prec^+$, but do not need to execute all tasks. We now define our problems of interest.

PLAN VERIFICATION

Input: A task network $tn = (T, \prec^+, \alpha)$ in a domain $D = (F, A, \mathcal{C}, \delta, M)$, an initial state $s_0 \subseteq F$, and a plan p over A.

Question: Is there a full solution to (D, tn, s_0) with a linearization $\bar{t}$ such that $\alpha(\bar{t}) = p$?

PLAN EXISTENCE

Input: A task network tn in a domain $D = (F, A, \mathcal{C}, \delta, M)$ and an initial state $s_0 \subseteq F$.

Question: Is there a full solution to (D, tn, s_0)?

In addition to the problems mentioned in the introduction, we also define the corresponding problem of constructing a plan executing (at least) a specified set of actions. This ACTION EXECUTABILITY problem is closely related to PLAN EXISTENCE and, in fact, all results that we prove for one of the problems will also hold for the other.

[1] Some works refer to the triple (D, tn, s_0) as an *HTN problem*, while we use that term for the considered algorithmic problems.

ACTION EXECUTABILITY	
Input:	A task network tn in a domain $D = (F, A, \mathcal{C}, \delta, M)$, an initial state $s_0 \subseteq F$, and a multiset S of actions in A.
Question:	Is there a solution to (D, tn, s_0) with a linearization $\bar{t}$ such that $\alpha(\bar{t})$ covers S?

ACTION EXECUTABILITY can be seen as a generalization of PLAN EXISTENCE: any PLAN EXISTENCE instance can be easily reduced to ACTION EXECUTABILITY, e.g., by adding a new task t with new action a, letting $S = \{a\}$, and adapting $\prec^+$ or δ to ensure that all other tasks are executed before t. Such direct reductions can be used to transfer algorithmic and hardness results in the respective directions; in our setting, we establish all of our hardness results for the "simpler" PLAN EXISTENCE, while all of our algorithms can deal with the "harder" ACTION EXECUTABILITY.

Finally, as motivated in the introduction, we consider the following natural problem on HTNs.

STATE REACHABILITY	
Input:	A task network tn in a domain $D = (F, A, \mathcal{C}, \delta, M)$, an initial state $s_0 \subseteq F$, and a target state $s_g \subseteq F$.
Question:	Is there a solution to (D, tn, s_0) with a linearization yielding a state $s \supseteq s_g$?

While the considered problems are stated in their decision variants for complexity-theoretic purposes, all of our obtained algorithms are constructive and can output a solution and a corresponding linearization as a witness. Further, for primitive networks, all the problems are clearly in NP. In some results, we use the *state transition graph*, i.e., the multigraph whose vertices are the states reachable from the initial state s_0 by any sequence of actions, and a transition from one state to another via an action a is represented by an arc labeled a.

3 Primitive Task Networks

Our first set of results focuses on the classical complexity of the targeted problems on primitive task networks. Before proceeding with our analysis, we make a general observation about the state space in our instances.

The State Space. As a basic precondition for solving our problems on more sophisticated task networks, we need to ensure tractability on trivial primitive task networks where $\prec^+ = \emptyset$ (antichains). Unfortunately, all but the first of our problems of interest are intractable even on these degenerate networks, as the hardness of PLAN EXISTENCE in STRIPS planning [Bylander, 1994] transfers to HTN planning by an HTN-to-STRIPS translation [Alford *et al.*, 2016]; see also the related work of Nebel and Bäckström [1994] and Erol et al. [1996]. As discussed above, the hardness of ACTION EXECUTABILITY follows from the hardness of PLAN EXISTENCE. The hardness of STATE REACHABILITY follows as well: there is a trivial reduction from PLAN EXISTENCE that adds a special task t_g and adapts all actions and s_g such that s_g is only reachable if t_g is executed, and t_g is only executable after all other tasks have been executed.

A requirement for the above reductions is that the state space (i.e., the number of states in the state transition graph) can be large, and in particular not bounded by any constant. Hence, we hereinafter analyze the affected problems (STATE REACHABILITY, PLAN EXISTENCE, and ACTION EXECUTABILITY) in the bounded state-space setting. The drop in complexity from PSPACE with unbounded state sets (see [Bylander, 1994]), to NP in the bounded state space setting is abstractly explained as follows: unbounded state sets capture the full tape contents of a non-deterministic Turing machine (NTM) during execution, while bounded state sets can only model the fixed-size set of states that define an NTM.

The Intractability of Tree-like HTNs. We begin with a series of lower bounds which rule out polynomial-time algorithms even for simple "tree-like" networks (unless P = NP). PLAN VERIFICATION is known to be NP-complete on primitive networks such that $D_\prec$ consists of disjoint stars, even when $\mathrm{prec}(a) = \mathrm{del}(a) = \mathrm{add}(a) = \emptyset$ for all actions $a \in A$ [Lin and Bercher, 2023, Theorem 3]. As our first result of this section, we show that our problems remain intractable even if $\prec^+$ describes a set of total orders that are independent from one another (i.e., $D_\prec$ is a disjoint union of paths). We reduce from SHUFFLE PRODUCT, which asks, given words $u, c_1, \ldots, c_w$ over an alphabet Σ, to decide if u can be generated by interlacing $c_1, \ldots, c_w$ while preserving the order of the letters as they appear in each c_i, $i \in [w]$. This problem is NP-hard even if $|\Sigma| = 2$ [Warmuth and Haussler, 1984].

Theorem 1. PLAN VERIFICATION *is* NP*-complete, even when restricted to primitive task networks such that* $\prec^+$ *is a collection of chains,* $|A| = 2$*, and* $\mathrm{prec}(a) = \mathrm{del}(a) = \mathrm{add}(a) = \emptyset$ *for all actions* $a \in A$*.*

Proof. We reduce from SHUFFLE PRODUCT with the alphabet $\Sigma = \{a, b\}$. Let $A = \Sigma$. Construct a task for each letter in the words $c_1, \ldots, c_w$; let the task $t_{i,j}$ refer to the j^{th} letter of the word c_i, $i \in [w]$. Let $\prec^+$ be such that the tasks for each word form a chain in the respective order, i.e., let $\prec$ consist of the edges $(t_{i,j}, t_{i,j+1})$, for all $i \in [w]$ and $j \in [|c_i| - 1]$. Then, $\prec^+$ partitions T into w chains (LEFT side of Fig. 4). Now, set $\alpha(t_{i,j})$ to the action representing the j^{th} letter of c_i. Let the input plan p be such that $p = u$.

The correctness of the reduction follows since verifying whether the tasks can be arranged to match the input plan corresponds to interlacing the words to form u. □

Theorem 2. STATE REACHABILITY *and* PLAN EXISTENCE *are* NP*-complete, even when restricted to primitive task networks such that* $\prec^+$ *is a collection of chains,* $|A| \leq 5$*, and the state transition graph has at most* 4 *states.*

Proof Sketch. We use a similar reduction as for Theorem 1, (see Fig. 4), but encode u as a separate chain and adapt the actions so that executable plans must alternate between LEFT and RIGHT tasks, and executed RIGHT tasks must be preceded by executed LEFT tasks of the same color. □

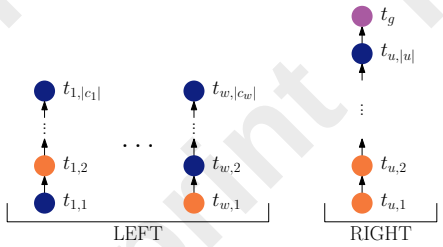


Figure 4: Constructed task network for reductions from SHUFFLE PRODUCT with words $c_1, \ldots, c_w$ and u. Each word is represented by a chain of tasks, where each task represents a letter in that word. The two letters in the alphabet are represented in orange and blue.

Chains and Antichains. Given the seemingly discouraging lower bounds obtained so far, one might wonder whether there are natural structural restrictions on the network which can be exploited to obtain polynomial-time algorithms. Here, we chart the route towards identifying these by first establishing the tractability of the considered problems (under the state-space restrictions justified in Sec. 3) in the extremes of $\prec^+$, where T is a chain (total order) or antichain ($\prec^+ = \emptyset$).

When T is a primitive chain, it is easy to observe that all of our problems are polynomial-time solvable (see, e.g., [Erol *et al.*, 1996; Behnke *et al.*, 2015]). PLAN VERIFICATION restricted to primitive antichains is also trivial, as it suffices to check whether the plan is executable in the initial state and the multiset of actions defined by the tasks in the network matches the actions in the plan. Handling antichains for the other problems requires more effort, and no tractability results exist that only assume the state space to be bounded.

Theorem 3. STATE REACHABILITY *is polynomial-time solvable on primitive antichain task networks with constantly many states in the state transition graph.*

Proof. Let $\mathcal{I}$ be an instance of STATE REACHABILITY, k the number of states, s_0 the initial state, and s_g the target state. We first compute the state transition graph $\mathcal{G}$ of $\mathcal{I}$ in quadratic time. In $\mathcal{G}$, if there are more than k edges between any pair of states, select any k of them and discard the rest. Then, consider all simple edge paths from s_0 to s_g. Note that there are $\mathcal{O}(2^k \cdot k^{k-1})$ such paths, as there are $\mathcal{O}(2^k)$ simple vertex paths, each of length at most $k-1$, and each step might choose between up to k different edges. For each simple edge path, check whether there is any action that is used more times in the path than there are tasks of that action in T. If there exists a simple edge path for which the number of required tasks does not exceed the number of available ones, then give a positive answer, and otherwise reject.

If such a path is found, the instance is indeed a yes-instance as there is a task sequence that results in a path to s_g in $\mathcal{G}$. For the other direction, let there be a task sequence that leads to the state s_g. Consider the subsequence obtained by exhaustively removing cycles in $\mathcal{G}$ from that sequence. The resulting task sequence describes a simple edge path from s_0 to s_g. It might employ edges that were discarded from $\mathcal{G}$ by the algorithm. However, in this case there are k other edges connecting the respective states in $\mathcal{G}$. Thus, one can choose to replace deleted edges by existing ones in such a way that each replacement action occurs at most once in the path, which ensures that there are enough tasks to actually walk this path. Hence, there is a path from s_0 to s_g that only uses edges that are present in $\mathcal{G}$ after the deletion, and for which the number of required tasks does not exceed the number of available ones. As the algorithm will find this path, it will correctly give a positive output. Finally, the overall running time can be upper-bounded by $\mathcal{O}(|\mathcal{I}|^2 + |A|k + |T| + 2^k \cdot k^k)$. □

To obtain an analogous result for ACTION EXECUTABILITY, a more sophisticated approach is necessary, since a solution may be required to visit a state multiple times.

Theorem 4. ACTION EXECUTABILITY *is polynomial-time solvable on primitive antichain task networks with constantly many states in the state transition graph.*

Proof Sketch. ACTION EXECUTABILITY on primitive antichains can be expressed by deciding whether there is a walk in the state transition graph such that each action is used in the path at least as many times as required by S and at most as many times as there are respective tasks in T. We bound the number of actions by a function of the number $k = O(1)$ of states by identifying *equivalent* actions that have the same impact on the states in the state transition graph. Now we enumerate all possible *types* of walks in constant time by branching on a "main-path" from s_0 to any $s \supseteq s_g$ and which sets of simple cycles the walk uses. For all branches where the main-path and all selected cycles are connected, it remains to determine the number of times each cycle is traversed. We do so by solving an *Integer Linear Program* (ILP) where there are variables stating the number of traversals of each of the $O(1)$ selected cycles, and a constraint for each action requiring that the number of times it is executed (as defined by the variables and the number of times it occurs in the main-path and the respective cycles) is admissible w.r.t. S and T. □

Generalized Partial Order Width. We have seen that the considered problems are polynomial-time solvable on primitive chains and antichains. Here, we unify and extend these results by establishing tractability for primitive HTNs of constant generalized partial order width (chains and antichains have generalized partial order width 1 and 0, respectively).

Theorem 5. PLAN VERIFICATION *is polynomial-time solvable on primitive task networks of constant generalized partial order width.*

Proof Sketch. We use dynamic programming to decide, for each index i in the plan and all combinations of indices $h_1, \ldots, h_w$ in the $w = O(1)$ many chains partitioning the non-isolated tasks, whether there is a solution matching the first i actions in the plan and using exactly the first $h_1, \ldots, h_w$ tasks of each chain, respectively. Clearly, this is possible for $i = h_1 = \ldots = h_w = 0$ and we iteratively add a next admissable task from a chain or the isolated tasks. □

Theorem 6. STATE REACHABILITY *and* ACTION EXECUTABILITY *are polynomial-time solvable on primitive task networks with constantly many states in the state transition graph and constant generalized partial order width.*

Proof Sketch. We note that the number $|E|$ of action equivalence classes that have the same impact on the state graph is constant. We use this in a dynamic programming approach similar to the one used for Theorem 5. Here, we decide for each state s, indices $h_j, j \in [w]$ in the w chains, and numbers of actions $r_i, i \in [|E|]$ in the equivalence classes, whether there is a solution with a linearization that uses exactly the first h_j tasks of each chain c_j, results in state s, and uses exactly r_i tasks of each action equivalence class e_i. We initialize all variables to be FALSE except for the one with $s = s_0$ and all h_j and r_i are 0. When a variable is set to TRUE (including the initial one), for each next index in a chain and each equivalence class for which the h_j and r_i imply that there is still an unused isolated task, we test whether this task can be appended to a sequence witnessing the TRUE variable. If so, the corresponding variable is set to TRUE as well. □

We recall that Theorem 6 also immediately implies the analogous result for PLAN EXISTENCE.

4 Hierarchical Task Networks

Considering HTNs with compound tasks adds another, seemingly opaque layer of difficulty to the considered planning problems. In this section, we show that—at least in terms of classical complexity—we can cleanly characterize the jump from primitive to compound through three measures on compound task networks: we obtain an algorithmic meta-theorem that allows to lift polynomial-time solvability from primitive to compound task networks if all three of these measures are bounded. Moreover, this result is tight in the sense that none of the three measures can be dropped. The three measures we need are $C_\#$, C_s, and C_d:

Definition 1. For an initial task network $tn = (T, \prec^+, \alpha)$ in a domain $D = (F, A, \mathcal{C}, \delta, M)$, we refer to the number of compound tasks in tn by $C_\#(tn) = |\{t \in T \mid \alpha(t) \in \mathcal{C}\}|$. The maximum size of a network that a task can be decomposed into is $C_s(D) = \max\{|T_m| \mid (c, (T_m, \prec_m^+, \alpha_m)) \in M\}$. The maximum depth $C_d(tn)$ of a network is defined recursively: $C_d(tn) = 0$ for primitive task networks, and a compound task network tn' has $C_d(tn') = i$ if i is the smallest integer such that every decomposition of all compound tasks in the initial network T results in a new network tn'' where $C_d(tn'') \leq i - 1$. We let $C_d(tn) = \infty$ if no such i exists (cyclic decomposition).

For our later considerations, it will be useful to additionally define the maximum number of pairwise non-isomorphic networks that a task can be decomposed into; we denote that by $C_c(D) = \max_{c \in \mathcal{C}} |\{tn_m \mid (c, tn_m) \in M\}|$. The next technical lemma shows how these measures help us bound the "decomposition complexity" of compound HTNs. We drop the references to tn and D in these parameters when they are clear from context.

Lemma 7. *If $C_d \neq \infty$, then a task network $tn = (T, \prec^+, \alpha)$ can be decomposed into at most $C_c{}^{\sum_{i=0}^{C_d-1} C_\# \cdot C_s{}^i}$ pairwise non-isomorphic primitive task networks $tn' = (T', \prec^{+\prime}, \alpha')$. For each of these, $|T'| \leq |T| + C_\# \cdot (C_s{}^{C_d} - 1)$.*

Proof. We first show the second part of the statement. The $|T| - C_\#$ primitive tasks in tn are still present in any network tn' that tn can be decomposed into. A compound task in tn can be decomposed into at most C_s compound tasks in one decomposition step and there are at most C_d decomposition steps. Thus, each compound task in tn can create at most $C_s{}^{C_d}$ tasks in tn'. Hence, $|T'| \leq |T| - C_\# + C_\# \cdot C_s{}^{C_d}$.

For the first statement, note that decomposing each of the $C_\#$ compound tasks once in a task network of depth C_d can result in at most $C_c{}^{C_\#}$ pairwise non-isomorphic task networks of depth at most $C_d - 1$. Further, each of these resulting networks has at most $C_\# \cdot C_s$ compound tasks. Thus, tn can be decomposed into at most

$$C_c{}^{C_\#} \cdot C_c{}^{C_\# \cdot C_s} \cdot \ldots \cdot C_c{}^{C_\# \cdot C_s{}^{C_d-1}} = C_c{}^{\sum_{i=0}^{C_d-1} C_\# \cdot C_s{}^i}$$

pairwise non-isomorphic task networks. □

Having defined our measures of interest, we proceed to formalize the set of problems our meta-theorem will capture. Intuitively, the aim is to formalize the set of all problems which deal with compound HTNs by decomposing them.

Definition 2. Let PR be an arbitrary HTN problem. Then, PR is *decomposable* if and only if, for all non-primitive initial task networks tn, PR on tn is a yes-instance if and only if there is a primitive task network tn' such that tn can be decomposed into tn' and tn' is a yes-instance of PR.

Note that all problems discussed in this work are decomposable. Next, in order to capture the general notions of "tractability" for various HTN problems on networks where certain measures are bounded, we need to slightly restrict the measures in question to avoid entirely degenerate measures.

Definition 3. Let κ be any numerical measure of HTNs in a domain D, i.e., a mapping that assigns each HTN a non-negative number. We say that κ is *stable* if there exists a computable function f with the following property: for each primitive HTN tn^* consisting of c tasks and each HTN tn containing some compound task t, the HTN tn' obtained from tn by decomposing t into tn^* satisfies $\kappa(tn', D) \leq f(\kappa(tn, D), c)$.

The computability of f is a mild technical requirement relevant for the fixed-parameter setting later on. We say that a stable measure κ is a *stable tractability measure* for a decomposable HTN problem PR if, for each constant c, PR can be solved in polynomial time on all primitive HTNs tn such that $\kappa(tn, D) \leq c$. Notably, the (generalized) partial order width measure fits in that definition, where its stability is due to the fact that the addition of c vertices to a graph can only increase its (generalized) partial order width by at most c.

Observation 1. (Generalized) partial order width is a stable tractability measure for PLAN VERIFICATION. The sum of the (generalized) partial order width and the number of states in the state space is a stable tractability measure for ACTION EXECUTABILITY and STATE REACHABILITY.

We are now ready to establish the first meta-theorem.

Theorem 8. *Let* PR *be a decomposable HTN problem and κ a stable tractability measure. Then,* PR *is polynomial-time solvable when restricted to any initial network tn in domain D where $C_d, C_\#, C_s$, and $\kappa(tn, D)$ are constants.*

Proof. Let $\mathcal{I}$ have task network $tn = (T, \prec^+, \alpha)$. If there are no compound tasks in T, then we apply an algorithm for primitive networks. Thus, assume $C_d \geq 1$. Then, PR in tn can be solved by listing all the primitive instances tn can be decomposed into and solving PR in each of these. By Lemma 7, there are at most ${C_c}^{g(C_d,C_\#,C_s)} \leq |\mathcal{I}|^{g(C_d,C_\#,C_s)}$ such instances $\mathcal{I}_p$ for $g(C_d, C_\#, C_s) = \sum_{i=0}^{C_d-1} {C_\#} \cdot {C_s}^i$ and, for each of these, $|\mathcal{I}_p| \leq |\mathcal{I}| + f''(C_d, C_\#, C_s)$ for some computable function f''. Furthermore, for each initial task network tn_p of an instance $\mathcal{I}_p$, we have $\kappa(tn_p, D) \leq f'''(\kappa(tn, D), C_d, C_\#, C_s)$, for some computable function f''' as κ is a stable measure and tn_p is obtained from tn by sequentially decomposing $C_\#$ compound tasks into a total of $f''(C_d, C_\#, C_s)$ primitive tasks. Thus, if primitive instances $\mathcal{I}_p$ can be solved in time at most $|\mathcal{I}_p|^{f(\kappa(tn_p,D))}$, then the total runtime (for some computable function f') is

$$\mathcal{O}\Big((|\mathcal{I}| + f''(C_d, C_\#, C_s))^{f(f'''(\kappa(tn,D),C_d,C_\#,C_s))} \cdot {C_c}^{g(C_d,C_\#,C_s)}\Big) = \mathcal{O}\Big(|\mathcal{I}|^{f'(\kappa(tn,D),C_d,C_\#,C_s)}\Big). \quad \square$$

We complement Theorem 8 via lower bounds which show that bounding C_d, $C_\#$, and C_s is necessary. Specifically, we prove that if any one of these measures is not bounded by a constant, then some of the considered problems become NP-hard on compound networks, even for settings that were easily solvable on primitive networks. While other reductions show the hardness deriving from decomposing non-primitive networks (see, e.g., [Erol *et al.*, 1996; Alford *et al.*, 2015a; Behnke *et al.*, 2015]), our reduction is stronger in the sense that it bounds all but any one of the parameters $C_d, C_\#, C_s$.

Theorem 9. *For each $\iota \in \{C_d, C_\#, C_s\}$,* PLAN EXISTENCE, ACTION EXECUTABILITY, *and* STATE REACHABILITY *remain* NP*-hard when restricted to instances which have (generalized) partial order width at most* 1 *and where $\{C_d, C_\#, C_s, C_c\} \setminus \{\iota\}$ are upper-bounded by a constant.*

Proof Sketch. We reduce from MULTICOLORED-CLIQUE, which is NP-hard and asks whether a given properly k-vertex-colored graph G contains a k-clique [Fellows *et al.*, 2009]. The reduction creates a compound task for each vertex and edge of G. These can all either be decomposed into a primitive vertex- or edge-task, or an empty "dummy" network. The actions and state space of the output instance are set up in a way which ensures that, for each color class, only a single primitive vertex task can be executed. The vertices for which we execute the corresponding primitive vertex task are considered as "chosen", and a primitive edge task can only be executed if both of its incident vertices are chosen. Based on the targeted HTN problem, we ensure the instance admits a solution if and only if it is possible to execute $\binom{k}{2}$ primitive edge tasks, matching each of the $\binom{k}{2}$ pairs of colors.

The above completes the reduction for the case where $\iota = C_\#$. For the case where $\iota = C_s$, we slightly adapt the construction by having the whole network (as described above) be the result of decomposing a single initial compound task t_0. Finally, when $\iota = C_d$ we build on the case of $\iota = C_s$ by making the decomposition from t_0 "gradual" as opposed to only occurring in a single step. $\square$

5 A Parameterized Analysis of HTNs

In this final section, we ask whether—and for which parameterizations—our problems on primitive HTNs admit fixed-parameter algorithms. While there are many graph-theoretic parameters typically used to achieve fixed-parameter tractability, none of the "usual suspects" in that regard will help with our problems due to the hardness on forests of stars [Lin and Bercher, 2023] and sets of chains (Theorems 1 and 2). Indeed, this applies to *treewidth* [Robertson and Seymour, 1986], the *feedback edge number* [Ganian and Korchemna, 2021], and similar measures defined on digraphs [Ganian *et al.*, 2016].

On the other hand, our problems are polynomial-time solvable for constant generalized partial order width and so we need a more refined notion of hardness to rule out fixed-parameter tractability in this case. In particular, hardness w.r.t. a parameter k for the complexity classes W[1] or W[2] rules out any fixed-parameter algorithm under the well-established assumption that W[1] $\neq$ FPT. As the SHUFFLE PRODUCT problem is W[2]-hard when parameterized by the width w [van Bevern *et al.*, 2016], Theorems 1 and 2 also imply the W[2]-hardness of the considered problems.

Fixed-Parameter Tractability via Vertex Cover. We contrast these lower bounds with a fixed-parameter algorithm for primitive networks that exploits a different parameterization, namely the vertex cover number (vcn) of $G_\prec$. A vertex cover of $G_\prec$ is a subset of tasks $V \subseteq T$ such that, for all $t \prec t'$, $t \in V$ and/or $t' \in V$; the vcn of $G_\prec$ is the minimum size of a vertex cover of $G_\prec$. While the vcn has been used to obtain fixed-parameter algorithms for many other problems [Balko *et al.*, 2022; Blazej *et al.*, 2023; Bodlaender *et al.*, 2023; Chalopin *et al.*, 2024], the techniques from previous works do not easily translate to HTN problems. Instead, our algorithms use a delicate branching routine whose correctness requires a careful analysis of the state space.

Theorem 10. PLAN VERIFICATION *is fixed-parameter tractable on primitive task networks parameterized by the vcn of $G_\prec$.*

Proof Sketch. We must order the tasks so that they respect $\prec$ and correspond to the input plan p. We branch over all orderings of the tasks in a minimal vertex cover V and discard those which do not respect $\prec$. Let the *priority* of a task be higher the earlier one of its out-neighbors in V is placed in the ordering. Tasks in V have priority just below the last of their in-neighbors, and tasks outside V with no out-edges have no priority. We iterate through p and, at each step, select a task for that position that has the respective action and has all of its predecessors in $\prec$ already selected. If there are multiple candidates, we choose a task with highest priority, ensuring that order constraints are met as early as possible. $\square$

Theorem 11. STATE REACHABILITY *and* ACTION EXECUTABILITY *on primitive networks are fixed-parameter tractable when parameterized by the vcn of $G_\prec$ plus the number of states in the state transition graph.*

Proof Sketch. We branch over the order of vertices in the vertex cover as in Theorem 10, but need more involved argu-

ments for constructing the task sequence. Inspired by the proof of Theorem 4, we carefully branch on which paths and cycles the sequence takes in the state transition graph in between visiting the tasks of the vertex cover, and use an ILP to determine the number of times each cycle is traversed. □

A Meta-Theorem for Fixed-Parameter Tractability. As our final contribution, we show that the algorithmic meta-theorem for polynomial-time solvability can be lifted to the setting of fixed-parameter tractability if we also restrict the maximum "breadth" of a compound task, i.e., C_c. To do this, we first need to define a suitable notion of stability. We call κ a *stable fixed-parameter tractability measure* for a decomposable HTN problem PR if κ is a stable measure and PR is fixed-parameter tractable on primitive task networks parameterized by κ. The vcn is an example of this notion as the addition of c vertices to a graph increases its vcn by at most c.

Observation 2. The vcn is a stable fixed-parameter tractability measure for PLAN VERIFICATION. The sum of the vcn and the number of states in the state space is a stable fixed-parameter tractability measure for ACTION EXECUTABILITY and STATE REACHABILITY.

We now establish the parameterized analog of Theorem 8, which extends Theorems 10 and 11 to compound networks:

Theorem 12. *Let* PR *be a decomposable HTN problem and κ a stable fixed-parameter tractability measure. If $C_d \neq \infty$, then* PR *is fixed-parameter tractable by the combined parameters $C_c, C_d, C_\#, C_s$, and $\kappa(tn, D)$ for the initial network tn.*

Proof. Let $tn = (T, \prec^+, \alpha)$. If there are no compound tasks in T, i.e., $C_d = 0$, the statement immediately holds by applying an algorithm for primitive networks. Thus, assume $C_d \geq 1$. Then, PR in tn is solved by listing all primitive instances tn can be decomposed into and solving PR in each of these. By Lemma 7, there are at most $C_c^{\sum_{i=0}^{C_d-1} C_\# \cdot {C_s}^i}$ such instances $\mathcal{I}_p$ and, for each of these, $|\mathcal{I}_p| \leq |\mathcal{I}| + f''(C_d, C_\#, C_s)$ for some computable function f''. Further, for each initial task network tn_p of an instance $\mathcal{I}_p$, we have $\kappa(tn_p, D) \leq f'''(\kappa(tn, D), C_d, C_\#, C_s)$, for some computable function f''' as κ is a stable measure. Thus, if primitive instances $\mathcal{I}_p$ can be solved in time $f(\kappa(tn_p, D))|\mathcal{I}_p|^{\mathcal{O}(1)}$, then the total running time can be upper-bounded by $\mathcal{O}\Big(C_c^{\,g(C_d, C_\#, C_s)} \cdot f'(\kappa(tn, D), C_d, C_\#, C_s) \cdot |\mathcal{I}|^{\mathcal{O}(1)}\Big)$ for computable functions g and f'. □

6 Concluding Remarks

This article provides a comprehensive understanding of the complexity-theoretic landscape of several fundamental problems on HTNs. Our results include strong algorithmic lower bounds and complementary positive results—not only for specific problems, but also in the form of meta-theorems that can be used for other HTN problems. For further work, while we have provided lower bounds justifying all the restrictions applied in the algorithmic meta-theorem for polynomial-time solvability (Theorem 8), in the more refined fixed-parameter tractability setting we leave it open whether the "hierarchical breadth" C_c needs to be part of the parameterization in order to establish Theorem 12.

Let us close with a brief discussion of the practical relevance of the results contained in this article. One indicator for the existence of practical algorithms is whether efficient algorithms should even be possible from a theoretical point of view. Our work provides such indicators, which may guide the development of algorithms that are efficient in practice as well. This is an exciting direction for future work, but it lies beyond the scope of this paper. That said, we deem it plausible that real-world planning problems exhibit some of the structural restrictions for which we have theoretically efficient algorithms. For example, the fixed-parameter algorithms in Theorems 10 and 11 could be applied to instances where only a few central tasks have precedence over others, while the remainder are essentially independent of each other. Moreover, the polynomial-time algorithms from Theorems 5 and 6 could be applied to instances where the tasks can be partitioned into isolated tasks and a small number of sets with internally predefined orders. For a concrete example, consider airport maintenance scheduling. Here, one may need to schedule a sequence of tasks for a small number of aircraft (where, for each aircraft, the order is predefined by FAA standards), while also dealing with a potentially large number of auxiliary tasks which are all pairwise independent.

Acknowledgements

This work was funded by the Austrian Science Fund (FWF) [10.55776/Y1329 and 10.55776/COE12], the WWTF Vienna Science and Technology Fund (Project 10.47379/ICT22029), and the EU Horizon Europe project TaRDIS (grant agreement 101093006).

References

[Alford *et al.*, 2014] Ron Alford, Vikas Shivashankar, Ugur Kuter, and Dana Nau. On the feasibility of planning graph style heuristics for HTN planning. In *ICAPS*, 2014.

[Alford *et al.*, 2015a] Ron Alford, Pascal Bercher, and David Aha. Tight bounds for htn planning. In *ICAPS*, 2015.

[Alford *et al.*, 2015b] Ron Alford, Pascal Bercher, and David W. Aha. Tight bounds for HTN planning with task insertion. In *IJCAI*, 2015.

[Alford *et al.*, 2016] Ron Alford, Gregor Behnke, Daniel Höller, Pascal Bercher, Susanne Biundo, and David W. Aha. Bound to plan: Exploiting classical heuristics via automatic translations of tail-recursive HTN problems. In *ICAPS*, 2016.

[Bäckström and Nebel, 1995] Christer Bäckström and Bernhard Nebel. Complexity results for SAS^+ planning. *Computational Intelligence*, 11(4):625–655, 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 *AAAI*, 2012.

[Bäckström *et al.*, 2013] Christer Bäckström, Peter Jonsson, Sebastian Ordyniak, and Stefan Szeider. Parameterized

complexity and kernel bounds for hard planning problems. In *CIAC*, 2013.

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

[Balko *et al.*, 2022] Martin Balko, Steven Chaplick, Robert Ganian, Siddharth Gupta, Michael Hoffmann, Pavel Valtr, and Alexander Wolff. Bounding and computing obstacle numbers of graphs. In *ESA*, 2022.

[Behnke and Speck, 2021] Gregor Behnke and David Speck. Symbolic search for optimal total-order HTN planning. In *AAAI*, 2021.

[Behnke *et al.*, 2015] Gregor Behnke, Daniel Höller, and Susanne Biundo. On the complexity of HTN plan verification and its implications for plan recognition. In *ICAPS*, 2015.

[Behnke *et al.*, 2019] Gregor Behnke, Daniel Höller, and Susanne Biundo. Finding optimal solutions in HTN planning - A SAT-based approach. In *IJCAI*, 2019.

[Bercher *et al.*, 2019] Pascal Bercher, Ron Alford, and Daniel Höller. A survey on hierarchical planning – one abstract idea, many concrete realizations. In *IJCAI*, 2019.

[Blazej *et al.*, 2023] Václav Blazej, Robert Ganian, Dusan Knop, Jan Pokorný, Simon Schierreich, and Kirill Simonov. The parameterized complexity of network microaggregation. In *AAAI*, 2023.

[Bodlaender *et al.*, 2023] Hans L. Bodlaender, Carla Groenland, and Michal Pilipczuk. Parameterized complexity of binary CSP: vertex cover, treedepth, and related parameters. In *ICALP*, 2023.

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

[Chalopin *et al.*, 2024] Jérémie Chalopin, Victor Chepoi, Fionn Mc Inerney, and Sébastien Ratel. Non-clashing teaching maps for balls in graphs. In *COLT*, 2024.

[Chapman, 1987] David Chapman. Planning for conjunctive goals. *Artif. Intel.*, 32(3):333–377, 1987.

[Chen and Bercher, 2021] Dillon Chen and Pascal Bercher. Fully observable nondeterministic HTN planning – Formalisation and complexity results. In *ICAPS*, 2021.

[Cygan *et al.*, 2015] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, Cham, 2015.

[Diestel, 2012] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in math.* Springer, 2012.

[Eiben *et al.*, 2021] Eduard Eiben, Robert Ganian, Iyad Kanj, Sebastian Ordyniak, and Stefan Szeider. The parameterized complexity of clustering incomplete data. In *AAAI*, 2021.

[Erol *et al.*, 1994] Kutluhan Erol, James A. Hendler, and Dana S. Nau. HTN planning: Complexity and expressivity. In *AAAI*, 1994.

[Erol *et al.*, 1996] Kutluhan Erol, James Hendler, and Dana Nau. Complexity results for HTN planning. *Annals of Mathematics and Artificial Intelligence*, 18:69–93, 1996.

[Fellows *et al.*, 2009] Michael R. Fellows, Danny Hermelin, Frances Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theor. Comput. Sci.*, 410(1):53–61, 2009.

[Fikes and Nilsson, 1971] Richard E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artif. Intel.*, 2(3):189–208, 1971.

[Froese *et al.*, 2022] Vincent Froese, Leon Kellerhals, and Rolf Niedermeier. Modification-fair cluster editing. In *AAAI*, 2022.

[Ganian and Korchemna, 2021] Robert Ganian and Viktoriia Korchemna. The complexity of Bayesian network learning: Revisiting the superstructure. In *NeurIPS*, 2021.

[Ganian *et al.*, 2016] Robert Ganian, Petr Hlinený, Joachim Kneis, Daniel Meister, Jan Obdrzálek, Peter Rossmanith, and Somnath Sikdar. Are there any good digraph width measures? *J. Comb. Theory, Ser. B*, 116, 2016.

[Geier and Bercher, 2011] Thomas Geier and Pascal Bercher. On the decidability of HTN planning with task insertion. In *IJCAI*, 2011.

[Georgievski and Aiello, 2015] Ilche Georgievski and Marco Aiello. HTN planning: Overview, comparison, and beyond. *Artif. Intel.*, 222:124–156, 2015.

[González-Ferrer *et al.*, 2013a] Arturo González-Ferrer, Juan Fernández-Olivares, and Luis A. Castillo. From business process models to hierarchical task network planning domains. *Knowl. Eng. Rev.*, 28(2):175–193, 2013.

[González-Ferrer *et al.*, 2013b] Arturo González-Ferrer, Annette ten Teije, Juan Fernández-Olivares, and Krystyna Milian. Automated generation of patient-tailored electronic care pathways by translating computer-interpretable guidelines into hierarchical task networks. *Artif. Intel. Medicine*, 57(2):91–109, 2013.

[Hayes and Scassellati, 2016] Bradley Hayes and Brian Scassellati. Autonomously constructing hierarchical task networks for planning and human-robot collaboration. In *ICRA*, 2016.

[Heeger *et al.*, 2023] Klaus Heeger, Danny Hermelin, George B. Mertzios, Hendrik Molter, Rolf Niedermeier, and Dvir Shabtay. Equitable scheduling on a single machine. *J. Sched.*, 26:209–225, 2023.

[Höller *et al.*, 2019] Daniel Höller, Pascal Bercher, Gregor Behnke, and Susanne Biundo. On guiding search in HTN planning with classical planning heuristics. In *IJCAI*, 2019.

[Höller and Bercher, 2021] Daniel Höller and Pascal Bercher. Landmark generation in HTN planning. In *AAAI*, 2021.

[Höller *et al.*, 2020] Daniel Höller, Pascal Bercher, and Gregor Behnke. Delete- and ordering-relaxation heuristics for htn planning. In *IJCAI*, 2020.

[Kronegger *et al.*, 2013] Martin Kronegger, Andreas Pfandler, and Reinhard Pichler. Parameterized complexity of optimal planning: A detailed map. In *IJCAI*, 2013.

[Li *et al.*, 2009] Nan Li, Subbarao Kambhampati, and Sung Wook Yoon. Learning probabilistic hierarchical task networks to capture user preferences. In *IJCAI*, 2009.

[Lin and Bercher, 2023] Songtuan Lin and Pascal Bercher. Was fixing this really that hard? On the complexity of correcting HTN domains. In *AAAI*, 2023.

[Lin *et al.*, 2024a] Songtuan Lin, Daniel Höller, and Pascal Bercher. Modeling assistance for hierarchical planning: An approach for correcting hierarchical domains with missing actions. In *SOCS*, 2024.

[Lin *et al.*, 2024b] Songtuan Lin, Conny Olz, Malte Helmert, and Pascal Bercher. On the computational complexity of plan verification, (bounded) plan-optimality verification, and bounded plan existence. In *AAAI*, 2024.

[Liu *et al.*, 2016] Dian Liu, Hongwei Wang, Chao Qi, Peng Zhao, and Jian Wang. Hierarchical task network-based emergency task planning with incomplete information, concurrency and uncertain duration. *Knowl. Based Syst.*, 112:67–79, 2016.

[Nebel and Bäckström, 1994] Bernhard Nebel and Christer Bäckström. On the computational complexity of temporal projection, planning, and plan validation. *Artif. Intel.*, 66(1):125–160, 1994.

[Olz *et al.*, 2021] Conny Olz, Susanne Biundo, and Pascal Bercher. Revealing hidden preconditions and effects of compound HTN planning tasks – A complexity analysis. In *AAAI*, 2021.

[Padia *et al.*, 2019] Kalpesh Padia, Kaveen Herath Bandara, and Christopher G. Healey. A system for generating storyline visualizations using hierarchical task network planning. *Comput. Graph.*, 78:64–75, 2019.

[Robertson and Seymour, 1986] Neil Robertson and Paul D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322, 1986.

[Sohrabi *et al.*, 2009] Shirin Sohrabi, Jorge A. Baier, and Sheila A. McIlraith. HTN planning with preferences. In *IJCAI*, 2009.

[Tsuneto *et al.*, 1996] Reiko Tsuneto, Kutluhan Erol, James A. Hendler, and Dana S. Nau. Commitment strategies in hierarchical task network planning. In *AAAI*, 1996.

[van Bevern *et al.*, 2016] René van Bevern, Robert Bredereck, Laurent Bulteau, Christian Komusiewicz, Nimrod Talmon, and Gerhard J. Woeginger. Precedence-constrained scheduling problems parameterized by partial order width. In *DOOR*, 2016.

[Warmuth and Haussler, 1984] Manfred K. Warmuth and David Haussler. On the complexity of iterated shuffle. *J. Comput. Syst. Sci.*, 28(3):345–358, 1984.

[Xiao *et al.*, 2017] Zhanhao Xiao, Andreas Herzig, Laurent Perrussel, Hai Wan, and Xiaoheng Su. Hierarchical task network planning with task insertion and state constraints. In *IJCAI*, 2017.

[Zhao *et al.*, 2017] Peng Zhao, Chao Qi, and Dian Liu. Resource-constrained hierarchical task network planning under uncontrollable durations for emergency decision-making. *J. Intel. Fuzzy Syst.*, 33(6):3819–3834, 2017.