On the Computational Complexity of Correcting HTN Domain Models

Songtuan Lin, Pascal Bercher
School of Computing, College of Engineering and Computer Science
The Australian National University
{songtuan.lin, pascal.bercher}@anu.edu.au

Abstract

Incorporating user requests into planning processes is a key concept in developing flexible planning technologies. Such systems may be required to change its planning model to adapt to certain user requests. In this paper, we assume a user provides a non-solution plan to a system and asks it to change the planning model so that the plan becomes a solution. We study the computational complexity of deciding whether such changes exist in the context of Hierarchical Task Network (HTN) planning. We prove that the problem is NP-complete in general independent of what or how many changes are allowed. We also identify several conditions which make the problem tractable when they are satisfied.

1 Introduction

Incorporating humans into planning processes has emerged as the frontier of the research in automatic planning for its potential to accomplish highly complicated tasks, e.g., see the works by Ferguson, Allen, and Miller (1996), Ferguson and Allen (1998), Ai-Chang et al. (2004), Bresina et al. (2005), and Behnke et al. (2016). One major challenge faced by the community in this direction is how to deal with the situation where a planning agent acts different from what a user expects. For instance, an agent may find a planning problem being unsolvable under its model whereas a user thinks it is not the case, or an agent offers a plan which differs from the one produced by a user himself/herself. The treatment for this problem varies in the role a user plays in the planning process. An end user may be curious about why the system's behavior is not in line with his/her expectation, namely looking for the explanations about the questions like "*why the problem is unsolvable?*" and "*why my plan is not a solution?*". Such explanations might be formulated either via transforming the planning model accordingly, e.g., changing the initial state (Göbelbecker et al. 2010) and abstracting the planning model to a certain level (Sreedharan, Srivastava, and Kambhampati, 2018; 2019), or via adjusting the user's expectation, e.g., correcting the plan the user has in mind (Barták et al. 2021a) and model reconciliation (Chakraborti et al., 2017; 2020). On the other hand, if the human involved is a domain writer, he/she may want to modify the planning model so that the agent's behavior can align with his/her anticipation. To this end, providing *modeling assistance* to help the domain writer comprehend the planning domain (Olz et al. 2021) or identify possible modeling errors via model transformations (Keren et al. 2017; Sreedharan et al. 2020) is vital especially when the planning domain is rather complicated.

In this paper, we re-visit a scenario we previously studied (Lin and Bercher 2021) where a user provides a plan and claims that it is supposed to be a solution to some planning problem, though it is actually not, and transformations on the planning model are required so that it will be. In our earlier work, we investigated the computational complexity of deciding whether such transformations can be found in the framework of totally ordered HTN (TOHTN) planning, which is a hierarchical approach of planning. Here we will extend those results. Our contributions are twofold. 1) We generalize our study to cover partially ordered HTN (PO-HTN) planning. 2) We consider the scenario with regard to different forms of the user input. For instance, a user could provide a partially ordered or sequential potential solution plan. The main results are summarized in Tab. 1

2 HTN Planning

We start with an introduction to the HTN formalism, which is based on the one by Bercher, Alford, and Höller (2019) and by Geier and Bercher (2011). We first give the definition of task networks.

Definition 1. A task network tn is a tuple $(T, \prec, \alpha)$ where T is a set of task identifiers, $\prec \subseteq T \times T$ specifies the partial order defined over T, and α is a function that maps a task identifier to a task name.

Definition 2. Two task networks $tn = (T, \prec, \alpha)$ and $tn' = (T', \prec', \alpha')$ are said to be isomorphic, written $tn \cong tn'$, if and only if there exists a one-to-one mapping $\phi : T \to T'$ such that for all $t \in T$, $\alpha(t) = \alpha'(\phi(t))$, and for all $t_1, t_2 \in T$, if $(t_1, t_2) \in \prec$, $(\phi(t_1), \phi(t_2)) \in \prec'$.

The task names in a task network are further categorized as being primitive or compound. Primitive task names are mapped to respective actions by a function δ. The action of a primitive task name p, $\delta(p) = (prec, add, del)$, consists of p's precondition, add, and delete list, respectively. We also write $(prec(p), add(p), del(p))$ for short. On the other hand, a compound task name c can be refined (decomposed) into a task network tn by some method $m = (c, tn)$.

Complexity	Changes	Theorems	
		Any Changes	*k Changes*
NP-complete	Action	Cor. 2	Cor. 4
	Order	Thm. 2	

(a) The complexity of changing planning models provided with a PO task network that is supposed to be a solution.

Complexity	Changes	Theorems	
		Any Changes	*k Changes*
NP-complete	Action	Cor. 6	Cor. 8
	Order	Thm. 6	
P (Conditioned)	Action	Thm. 3 & 7	?

(b) The complexity of changing planning models provided with a PO/TO task network and a method sequence that is supposed to generate it. Special cases with changing actions being allowed that cover both totally ordered and partially ordered HTN planning are in P. Whether similar cases exist for the bounded version remains open (marked with '?').

Complexity	Changes	Theorems	
		Any Changes	*k Changes*
NP-complete	Action	Cor. 10	Cor. 13
	Order	Cor. 11	

(c) The complexity of changing planning models provided with an action sequence that is supposed to be a linearisation of a non-given solution task network.

Table 1: The computational complexity of the problems studied in this paper and the respective theorems (corollaries). The column 'Changes' specifies the target that changes are imposed to, i.e., changing actions or ordering constraints. The column 'Any Changes' refers to the case where an arbitrary number of changes can be applied, and 'k Changes' refers to the case where at most k changes can be applied.

Given a task network tn, the notations $T(tn)$, $\prec(tn)$, and $\alpha(tn)$ refer to the task identifier set, the partial order, and the identifier-name mapping function of tn, respectively. For a method m, we use $tn(m)$ to refer to its task network.

For convenience, we also define a restriction operation.

Definition 3. Let D and V be two arbitrary sets, $R \subseteq D \times D$ be a relation, $f : D \to V$ be a function and tn be a task network. The restrictions of R and f to some set X are defined by

- $R|_X = R \cap (X \times X)$
- $f|_X = f \cap (X \times V)$
- $tn|_X = (T(tn) \cap X, \prec(tn)|_X, \alpha(tn)|_X)$

A planning problem is then defined as follows.

Definition 4. An HTN planning problem P is a tuple (D, tn_I, s_I) where D is called the domain of P. It is a tuple (F, N_p, N_c, δ, M) in which F is a finite set of facts, N_p is a finite set of primitive task names, N_c is a finite set of compound task names with $N_c \cap N_p = \emptyset$, $\delta : N_p \to 2^F \times 2^F \times 2^F$ is a function that maps primitive task names to their actions, and M is a set of (decomposition) methods. tn_I is the initial task network, and $s_I \in 2^F$ is the initial state.

Definition 5. Let $tn = (T, \prec, \alpha)$ be a task network, $t \in T$ be a task identifier, c be a compound task name with $(t, c) \in \alpha$, and $m = (c, tn_m)$ be a method. We say m decomposes tn into another task network $tn' = (T', \prec', \alpha')$, written $tn \to_m tn'$, if and only if there exists a task network $tn'_m = (T_m, \prec_m, \alpha_m)$ with $tn'_m \cong tn_m$ such that

- $T' = (T \backslash \{t\}) \cup T_m$.
- $\prec' = (\prec \cup \prec_m \cup \prec_X)|_{T'}$, where $\prec_X = \{(t_1, t_2) \mid (t_1, t) \in \prec, t_2 \in T_m\} \cup \{(t_2, t_1) \mid (t, t_1) \in \prec, t_2 \in T_m\}$.
- $\alpha' = (\alpha \backslash \{(t, c)\}) \cup \alpha_m$.

Additionally, a task network tn is decomposed into another task network tn' by a sequence of methods $\overline{m} = m_1 \cdots m_n$ ($n \in \mathbb{N}^0$ with $\mathbb{N}^0 = \mathbb{N} \cup \{0\}$), written $tn \to^*_{\overline{m}} tn'$, if and only if there exists a sequence of task networks $tn_0 \cdots tn_n$ such that $tn_0 = tn$, $tn_n = tn'$, and for each $1 \leq i \leq n$, $tn_{i-1} \to_{m_i} tn_i$. Particularly, $tn \to^*_{\overline{m}} tn$ if $\overline{m}$ is empty.

The solution criteria of a planning problem are then defined as follows.

Definition 6. Let $P = (D, tn_I, s_I)$ be an HTN planning problem. A solution to P is a task network tn such that all tasks in it are primitive, there exists a method sequence $\overline{m}$ that decomposes tn_I into it, i.e., $tn_I \to^*_{\overline{m}} tn$, and it possesses a linearisation of the tasks that is executable in s_I.

A linearisation $t_1 \cdots t_n$ of a (primitive) task network is executable in a state s if there exists a sequence of states $s_0 \cdots s_n$ such that $s_0 = s$, and for each $1 \leq i \leq n$, $s_{i-1} \subseteq prec(\alpha(t_i))$ and $s_i = (s_{i-1} \backslash del(\alpha(t_i))) \cup add(\alpha(t_i))$.

The presented definition is standard in HTN planning as proposed by Erol, Hendler, and Nau (1996) and used in subsequent publications as well (Bercher, Alford, and Höller 2019). Other formalizations of hierarchical planning such as hybrid planning (Bercher et al. 2016) which fuses HTN planning with Partial Order Causal Link (POCL) where in solution plans *every* linearization is executable. We will also provide this alternative solution criterion.

Definition 7. Let $P = (D, tn_I, s_I)$ be an HTN planning problem. A solution to P is a task network tn such that all tasks in it are primitive, there exists a method sequence that decomposes tn_I into tn, and every linearisation of tn is executable in s_I.

The reason for including the more restricted solution criterion is to be able to identify the cause of computational hardness when model changes are required, though it is somehow unrealistic. A more practical one would be 'a task network tn is a solution *iff* it can be obtained via decompositions, and by adding some ordering constraints, every linearisation of it is executable'. However, the requirement of asking for additional ordering constraints has the same algorithmic lower bound as deciding whether tn has an executable linearisation, which itself is NP-hard already (Nebel and Bäckström 1994; Erol, Hendler, and Nau 1996)[1], because if such extra ordering constraints can be found,

[1] See Bercher (2021) for a discussion and further related work.

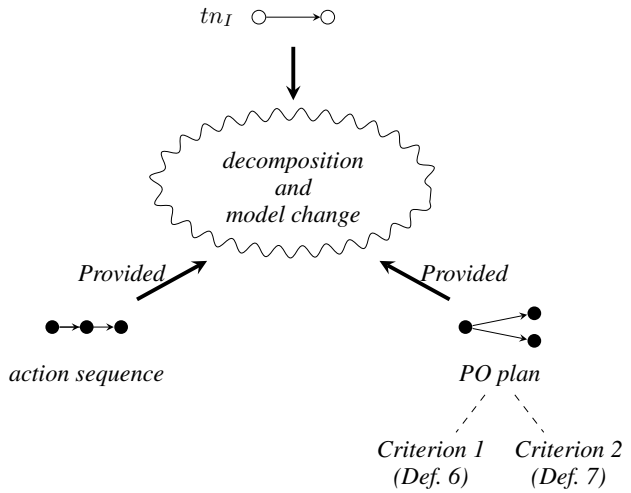

Figure 1: The scenarios where model change is involved. Criterion 1 states that a task network is a solution *iff* it is a refinement of tn_I and there exists a linearisation of it which is executable, whereas Criterion 2 requires that every linearisation is executable.

tn must have an executable linearisation. Thus, if we demand the solution criterion given by Def. 6 or the one requiring extra ordering constraints, it would not be clear where NP-hardness comes from. On the other hand, verifying whether all linearizations of a task network are executable was shown to be tractable (Nebel and Bäckström 1994; Chapman 1987).[2] Consequently, we list Def. 6 only for the sake of completeness, and we will adhere to Def. 7 throughout the paper in order to eliminate the ambiguous hardness source, unless otherwise indicated.

Fig. 1 previews what scenarios will be considered next. In the right branch we assume that a partially ordered plan is provided that is supposed to be a solution. Although we provide two solution criteria with regard to this case, we will primary focus on the one given by Def. 7. In the left branch we consider the case where an action sequence is provided rather than a partially ordered plan.

3 Changing the Model

For the purpose of changing planning models, we shall first define the allowed changes. We have introduced several model-change operations in the context of totally ordered HTN planning in our earlier work (Lin and Bercher 2021), which is a restricted version of HTN planning where the tasks in each task network in a planning model are totally ordered. For such a task network tn, its definition can be simplified by regarding it as a sequence of task names, i.e., $tn \in (N_p \cup N_c)^*$. We first reproduce the definitions of those operations since we will require them later on.

Definition 8. Let p be a primitive task name, $m = (c, tn)$ with $tn = t_1 \cdots t_n$ be a method, and $1 \leq i \leq n+1$ be an integer. The operation $\text{ACT}^{\dagger}_{\text{TO}}$ is a function that takes as inputs p, m, and i and outputs a new method $m' = (c, tn')$ such that $tn' = tn_1\, p\, tn_2$ where $tn_1 = t_1 \cdots t_{i-1}$ and $tn_2 = t_i \cdots t_n$.

Definition 9. Let $m = (c, tn)$ be a method where $tn = tn_1\, p\, tn_2$ with $tn_1 = t_1 \cdots t_{i-1}$ and $tn_2 = t_{i+1} \cdots t_n$ be two sequences of task names, and p be a primitive task name. The operation $\text{ACT}^{\bar{}}_{\text{TO}}$ is a function that takes as inputs m and i and outputs a new method $m' = (c, tn')$ such that $tn' = tn_1\, tn_2$.

We use C_{TO} to refer to the set of changes allowed in totally ordered HTN planning. On top of those operations, we define several new operations that are targeted at partially ordered HTN planning problems. We first consider the operations that change the ordering constraints in a method.

Definition 10. Let $m = (c, tn)$ with $tn = (T, \prec, \alpha)$ be a method, and $t_1, t_2 \in T$ be two task identifiers. The operation ORD^{+} is a function that takes as inputs m and (t_1, t_2) and outputs a new method $m' = (c, tn')$ with $tn' = (T', \prec', \alpha')$ such that $T' = T$, $\prec' = (\prec \cup \{(t_1, t_2)\})^{+}$[3], and $\alpha' = \alpha$.

Definition 11. Let $m = (c, tn)$ with $tn = (T, \prec, \alpha)$ be a method, and $t_1, t_2 \in T$ be two task identifiers with $(t_1, t_2) \in \prec$. The operation ORD^{-} is a function that takes as inputs m and (t_1, t_2) and outputs a new method $m' = (c, tn')$ with $tn' = (T', \prec', \alpha')$ such that $T' = T$, $\prec' = \prec \backslash \{(t_1, t_2)\}$, and $\alpha' = \alpha$.

We then consider the operations that change the actions (primitive tasks) in a method. We start with the operation which adds an action to a method's task network.

Definition 12. Let $m = (c, tn)$ with $tn = (T, \prec, \alpha)$ be a method, $T_A = \{t_1, \cdots, t_n\}$ and $T_B = \{t'_1, \cdots t'_m\}$ with $n, m \in \mathbb{N}$ and $T_A \cap T_B = \emptyset$ be two subsets of T, and $p \in N_p$ be a primitive task name. The operation $\text{ACT}^{\ddagger}_{\text{PO}}$ is a function that takes as inputs m, T_A, T_B, and p and outputs a new method $m' = (c, tn')$ with $tn' = (T', \prec', \alpha')$ such that $T' = T \cup \{t\}$ with $t \notin T$ be a new task identifier, $\prec' = (\prec \cup \prec_A \cup \prec_B)^{+}$ with $\prec_A = \bigcup_{i=1}^{n} \{(t_i, t)\}$ and $\prec_B = \bigcup_{i=1}^{m} \{(t, t'_i)\}$, and $\alpha' = \alpha \cup \{(t, p)\}$.

Informally, the above operation inserts a primitive task to a position in tn that is after the tasks listed in T_A and before those in T_B. For instance, a new task is placed before all tasks in tn if $T_A = \emptyset$ and $T_B = T$. On the other hand, when removing an action from a method, we should delete all ordering constraints associated with this action.

Definition 13. Let $m = (c, tn)$ with $tn = (T, \prec, \alpha)$ be a method, and $t \in T$ be a task identifier. The operation $\text{ACT}^{\bar{}}_{\text{PO}}$ is a function that takes as inputs m and t and outputs a new method $m' = (c, tn')$ with $tn' = tn|_{T \setminus \{t\}}$.

Similarly, we use C_{PO} to refer to the set of change operations allowed in a partial order setting. Given two methods m, m' and a sequence of model-change operations

[2] Nebel and Baeckstroem did not show this in the context of HTN planning, but for unconditional *event systems*. These, however, perfectly coincide with a partially ordered set of actions such as in primitive task networks. A more detailed discussion can be found in the work by Bercher and Olz (2020).

[3] The superscript $^{+}$ refers to the transitive closure.

$\mathcal{X} = x_1(m_1, *) \cdots x_n(m_n, *)$ where for each $1 \leq i \leq n$, $x_i \in C_{\text{TO}}$ if a total order setting is given, otherwise $x_i \in C_{\text{PO}}$, m_i is a method, and $*$ refers to the remaining parameters in the operation. We write $m \rightarrow^*_{\mathcal{X}} m'$ if $m = m_1$, $m' = x_n(m_n, *)$, and for each $1 \leq i \leq n-1$, $m_{i+1} = x_i(m_i, *)$.

Definition 14. Let $P = (D, tn_I, s_I)$ with $D = (F, N_p, N_c, \delta, M)$ and $M = \{m_1, \cdots, m_n\}$ be a planning problem, and $\mathcal{X}$ be a sequence of method-changes. A problem $P' = (D', tn_I, s_I)$ with $D' = (F, N_p, N_c, \delta, M')$ and $M' = \{m'_1, \cdots, m'_n\}$ is obtained from P by applying $\mathcal{X}$, written $P \rightarrow^*_{\mathcal{X}} P'$ if and only if for each $1 \leq i \leq n$, either $m'_i = m_i$ or there exists a sub-sequence X_i of $\mathcal{X}$ such that $m_i \rightarrow^*_{X_i} m'_i$.

The definition is applied to both partially ordered and totally ordered HTN planning, and it implies that the method set in P maintains a one-to-one mapping to that in P'. We use $\beta_{\mathcal{X}} : M \rightarrow M'$ to denote this mapping, where for each method m_i with $1 \leq i \leq n$, $\beta_{\mathcal{X}}(m_i) = m'_i$.

Now we have defined all necessary model changes, we can move on to investigate the computational complexity of checking whether a change sequence exists that turns the given task network into a solution.

4 Complexity of Correcting the Model – Given Just A Task Network

We start by considering the question asking whether there exists a sequence of model-change operations with arbitrary length that turns a given partially ordered task network into a solution. We formulate the decision problem as follows, which generalizes the old one we gave for totally ordered HTN planning.

Definition 15. Let $X \subseteq \{\text{ACT}^+_{\text{SET}}, \text{ACT}^-_{\text{SET}}, \text{ORD}^+, \text{ORD}^-\}$ and $|X| \geq 1$, $\text{SET} \in \{\text{TO}, \text{PO}\}$, P be a planning problem, and tn be a task network. The problem $\text{FIXMETHS}^X_{\text{SET}}$ with SET specifying whether it is in a TO or a PO setting is to decide whether there is a sequence of change operations $\mathcal{X}$ consisting of the operations restricted by X such that $P \rightarrow^*_{\mathcal{X}} P'$, and tn is a solution to P'.

The hardness of the problem in a PO setting can be immediately obtained under the solution criterion given by Def. 6 (because deciding whether a partially ordered task network has an executable linearisation is already NP-hard). Thus, the question of interest is whether NP-hardness (henceforth NP-completeness) holds when we employ the solution criterion given by Def. 7. For this, we first consult our old result (Lin and Bercher 2021) that the problem is NP-complete in totally ordered HTN planning.

Proposition 1 (Lin and Bercher (2021, Thm. 1–4))**.** *Given an $X \subseteq \{\text{ACT}^+_{\text{TO}}, \text{ACT}^-_{\text{TO}}\}$ and $|X| \geq 1$, $\text{FIXMETHS}^X_{\text{TO}}$ is NP-complete.*

This proposition holds for both solution criteria given by Def. 6 and 7 because every task network in totally ordered HTN planning has only one linearisation. Since totally ordered HTN planning is a restricted version of partially ordered HTN planning, the hardness of the variants in the context of partially ordered HTN planning where only changing actions is allowed follows directly.

Corollary 1. *$\text{FIXMETHS}^X_{\text{PO}}$ with $X \subseteq \{\text{ACT}^+_{\text{PO}}, \text{ACT}^-_{\text{PO}}\}$ and $|X| \geq 1$ is NP-hard.*

Next we show that these variants are in NP as well. To this end, we first prove that there always exists a polynomial upper bound of the length of the *shortest* change sequence that turns the given task network into a solution independent of what changes are allowed.

Lemma 1. *Let P and tn be a planning problem and a task network given by an instance of the $\text{FIXMETHS}^X_{\text{PO}}$ problem with $X \subseteq \{\text{ACT}^+_{\text{PO}}, \text{ACT}^-_{\text{PO}}, \text{ORD}^+, \text{ORD}^-\}$ and $X \geq 1$. There must exist a change sequence $\mathcal{X}$ consisting of changes restricted by X such that $P \rightarrow^*_{\mathcal{X}} P'$, tn is a solution to P', and $|\mathcal{X}| \leq (\sum_{(c,tn_m) \in M} |T(tn_m)| + |\prec(tn_m)|) + |T(tn)| + |\prec(tn)|$ provided that any change sequence exists that meets the restriction of X and turns tn into a solution.*

Proof. We first consider the variant where all changes are allowed. We need to show that the upper bound presented is sufficient for the *shortest* change sequence. In such a change sequence, the number of action deletions must *not* exceed the total number of tasks in all methods, which is $\sum_{(c,tn_m) \in M} |T(tn_m)|$, otherwise, there must exist some action that is added first and removed afterward, and thus a shorter change sequence exists. For the same reason, the number of ordering constraint deletions is smaller or equal to $\sum_{(c,tn_m) \in M} |\prec(tn_m)|$, which is the total number of ordering constraints in all methods. On the other hand, the number of action insertions in the shortest change sequence cannot exceed the total number of tasks in tn (i.e., $|T(tn)|$), otherwise, some inserted actions must be deleted, and thus a shorter change sequence exists. The same argument holds for the number of ordering constraint insertions, which cannot exceed $|\prec(tn)|$. Thus, the presented upper bound holds.

For the remaining variants, the length of the shortest change sequence is strictly smaller than the presented upper bound because some changes are forbidden, e.g., if only adding actions is allowed, the length of the shortest change sequence must not exceed $|T(tn)|$. Thereby, the upper bound holds for all variants. □

The presented lemma not only reveals the NP-membership of the variants where only changing actions is allowed, but the fact that all classes are in NP.

Theorem 1. *Let $X \subseteq \{\text{ACT}^+_{\text{PO}}, \text{ACT}^-_{\text{PO}}, \text{ORD}^+, \text{ORD}^-\}$ and $X \geq 1$. $\text{FIXMETHS}^X_{\text{PO}}$ is in NP.*

Proof. For *each* $X \subseteq \{\text{ACT}^+_{PO}, \text{ACT}^-_{PO}, \text{ORD}^+, \text{ORD}^-\}$ and $X \geq 1$, we can guess a change sequence of length smaller or equal to the upper bound stated in Lem. 1 which turns P into P' and consists of operations restricted by X. This step can be done in poly-time because the change sequence is bounded in length by a polynomial. Afterward, we verify whether *every* linearisation of tn is executable, which can be accomplished in polynomial time as well (Nebel and Bäckström 1994; Chapman 1987). Lastly, we employ the non-deterministic VERIFYTN algorithm (Behnke, Höller,

and Biundo 2015) to check whether tn_I can be decomposed into tn under the modified domain. Although the VERIFYTN algorithm is developed under the solution criterion given by Def. 6, it can be employed here because it is exploited in the sense that we do not need to consider the executability of tn (which has been verified previously). Thus, $\text{FIXMETHS}^{\text{X}}_{PO}$ is in NP. □

The NP-completeness of the variants where only changing actions in methods is allowed is thus a direct corollary of the previous results.

Corollary 2. $\text{FIXMETHS}^{X}_{\text{PO}}$ *with* $X \subseteq \{\text{ACT}^{+}_{\text{PO}}, \text{ACT}^{-}_{\text{PO}}\}$ *and* $|X| \geq 1$ *is NP-complete.*

What is new compared to totally ordered HTN planning are the operations that change ordering constraints in methods. It turns out that deciding whether we can transform a plan into a solution via changing ordering constraints in methods is NP-complete as well.

Theorem 2. $\text{FIXMETHS}^{\text{ORD}^{+}}_{\text{PO}}$ *is NP-complete.*

Proof. Membership has been given by Thm. 1. For hardness, we reduce from the *independent set* problem. The independent set problem is that given a graph $G = (V, E)$ and an integer $k \in \mathbb{N}$, we want to decide whether there is a subset $V' \subseteq V$ such that $|V'| = k$, and there are no two vertices in V' which are connected to each other by an edge in E. Suppose $k \in \mathbb{N}$ and $G = (V, E)$ with $V = \{v_1, \cdots v_n\}$ and $E = \{e_1, \cdots, e_m\}$ are the integer and the graph given by an instance of the independent set problem. The key idea of the reduction is constructing a planning problem P whose initial task network tn_I encodes the structure of G. To this end, we construct one compound task v^c_i $(1 \leq i \leq n)$ for each vertex v_i and one primitive task e^p_i $(1 \leq i \leq m)$ for each edge e_i. The initial task network tn_I consists of two parts as shown by Fig. 2. The first part contains the *unordered* tasks $v^c_1, \cdots, v^c_n$. The second part is m continuous blocks $E_1 \cdots E_m$[4]. A block E_i $(1 \leq i \leq m)$ consists of the primitive task e^p_i, two compound tasks $v^c_{i_1}$ and $v^c_{i_2}$ $(1 \leq i_1, i_2 \leq n)$ whose respective vertices v_{i_1} and v_{i_2} in G are connected by the edge e_i, and one additional compound task h^c_i. Further, the block also has the ordering constraints $(e^p_i, v^c_{i_1})$, $(e^p_i, v^c_{i_2})$, and (e^p_i, h^c_i) which are drawn by thin arrows. Each thick arrow in the figure represents a set of ordering constraints specifying that all tasks in the left-hand side are ordered before those in the right-hand side. Afterward, we construct one method $m_{v_i} = (v^c_i, tn_{v_i})$ with $tn_{v_i} = (\{t_1, t_2\}, \emptyset, \{(t_1, s), (t_2, s)\})$ for each v^c_i in which s is an action. Additionally, for each h^c_i $(1 \leq i \leq m)$, we construct a method $m_{h_i} = (h^c_i, tn_{h_i})$ such that $tn_{h_i} = (\{t_1, t_2\}, \emptyset, \{(t_1, s), (t_2, s)\})$ as well. Finally, we construct the target task network tn as shown by Fig. 2. By construction, each compound task in tn_I has only one method that can decompose it. Adding an ordering constraint to some method m_{v_i} with $1 \leq i \leq n$ is now equivalent to selecting the respective vertex into the independent set. Next we show that an independent set of size k exists if and only if tn can be turned into a solution by adding ordering constraints to methods.

($\Longrightarrow$): Suppose V' is an independent set of size k. The change sequence that turns tn into a solution can be found as follows. For each $v_i \in V'$, we add the ordering constraint (t_1, t_2) to the method m_{v_i}. Afterward, we examine whether there exists some edge e_j of which two endpoints are not in V', and if it is the case, we add the ordering constraint (t_1, t_2) to the method m_{h_j}. By accomplishing this procedure, tn can now be obtained from tn_I.

($\Longleftarrow$): Suppose $\mathcal{X}$ is a change sequence that turns tn into a solution. An independent set of size k can be found by examining each operation in $\mathcal{X}$ iteratively and checking whether it adds the ordering constraint (t_1, t_2) to some method m_{v_i} $(1 \leq i \leq n)$. If so, the respective vertex v_i is in the set. The remaining operations that adds (t_1, t_2) to some m_{h_i} $(1 \leq i \leq m)$ can be simply ignored. □

Note that the only difference between the solution (which is uniquely defined) to the (unmodified) planning problem P and the task network tn in the presented proof is their ordering constraints. Thus, the proof still holds when the operations that change actions in methods are allowed. Moreover, since each method constructed in the proof does not have any ordering constraint at the beginning, allowing ordering constraint deletions is redundant as well. The following result is then a direct corollary.

Corollary 3. *Let* $X \subseteq \{\text{ACT}^{+}_{\text{PO}}, \text{ACT}^{-}_{\text{PO}}, \text{ORD}^{+}, \text{ORD}^{-}\}$ *and* $X \geq 1$. $\text{FIXMETHS}^{X}_{\text{PO}}$ *is NP-complete.*

Instead of asking whether there exists a change sequence of arbitrary length that transforms a task network into a solution, we are also interested in finding an optimal one. The decision problem asking for that is formulated in terms of an additional integer k.

Definition 16. Let $X \subseteq \{\text{ACT}^{+}_{\text{SET}}, \text{ACT}^{-}_{\text{SET}}, \text{ORD}^{+}, \text{ORD}^{-}\}$ and $X \geq 1$, $\text{SET} \in \{\text{TO}, \text{PO}\}$, and $k \in \mathbb{N}$, the problem $\text{FIXMETHS}^{X,k}_{\text{SET}}$ is identical to $\text{FIXMETHS}^{X}_{\text{SET}}$ except that any change sequence should be limited in length by k.

We have shown in our previous work that the problem is NP-complete in a total order setting (Lin and Bercher, Cor. 1). In a partial order setting, any given $\text{FIXMETHS}^{X}_{\text{PO}}$ instance can be reduced to a $\text{FIXMETHS}^{X,k}_{\text{PO}}$ instance by replicating the planning problem and the target task network given and setting k to the upper bound given by Lem. 1. Hardness thus follows immediately. For membership, although the given k can be exponentially large via logarithmic encoding, we can always guess a change sequence of length smaller than the minimum of k and the polynomial bound given by Lem. 1. Thereby, the problem is in NP as well.

Corollary 4. *Let* $X \subseteq \{\text{ACT}^{+}_{\text{PO}}, \text{ACT}^{-}_{\text{PO}}, \text{ORD}^{+}, \text{ORD}^{-}\}$ *and* $X \geq 1$. $\text{FIXMETHS}^{X,k}_{\text{PO}}$ *is NP-complete.*

5 Complexity of Fixing the Model – Given A Task Network and A Method Sequence

So far our investigation only consider a given planning problem and a task network which is supposed to be a solution.

[4]Note that each E_i $(1 \leq i \leq m)$ is *not* a compound task but an abbreviation of a component in tn_I.

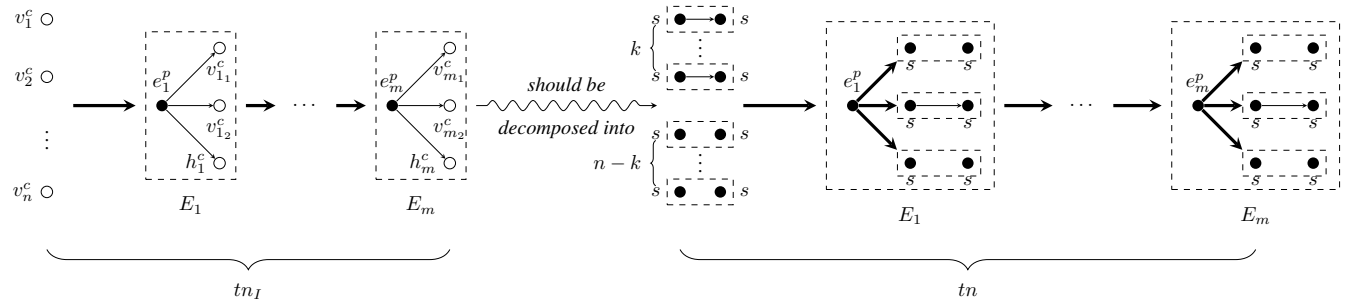

Figure 2: The constructions of tn_I and tn in the proof of Thm. 2. Each thick arrow represents a set of ordering constraints specifying that the tasks in the *lhs* are ordered before those in the *rhs*. Each thin arrow denotes a single ordering constraints. Each dashed rectangle represents nothing but a group of tasks that is part of tn_I or tn.

One can identify that one possible source of hardness is that we do not know which methods should be applied to generate the task network in question. To eliminate this source, we consider another scenario where we are given not only a task network and a planning problem, but a decomposition method sequence that is supposed to decompose the initial task network into the given one. For the practical motivation for this scenario, consider, e.g., a scenario in the context of modeling assistance where a user provides a plan as well as a method sequence to a planning system and argues that the plan must be generated by the given method sequence, whereas a plan verification system (Behnke, Höller, and Biundo 2017; Barták, Maillard, and Cardoso 2018; Barták et al. 2020, 2021b) rejects the plan. Thus, there must be some methods in the planning model that are incorrectly implemented. Correcting the model and identifying which methods are flawed can not only satisfy user requests but serve as counter-factual explanations (Ginsberg 1986; Chakraborti et al. 2017; Chakraborti, Sreedharan, and Kambhampati 2020) telling users what are the implementation errors in the case that plan verification fails, e.g., in a hierarchical planning competition.

Definition 17. Let $X \subseteq \{\text{ACT}^+_{\text{SET}}, \text{ACT}^-_{\text{SET}}, \text{ORD}^+, \text{ORD}^-\}$ and $|X| \geq 1$, $\text{SET} \in \{\text{TO}, \text{PO}\}$, P be a planning problem, $\overline{m} = m_1 \cdots m_n$ ($n \in \mathbb{N}^0$) be a sequence of methods, and tn be a task network. The problem $\text{FIXMSEQ}^X_{\text{SET}}$ with SET specifying whether it is in a TO or a PO setting is to decide whether there is a sequence of change operations $\mathcal{X}$ consisting of the operations restricted by X such that $P \to^*_{\mathcal{X}} P'$, and $tn_I \to^*_{\overline{m}'} tn$ with $\overline{m}' = \beta_{\mathcal{X}}(m_1) \cdots \beta_{\mathcal{X}}(m_n)$.

In our early study (Lin and Bercher 2021) we have shown that the problem is NP-complete in general in the context of totally ordered HTN planning. Here we will extend this result by showing that the presence of the method sequence does make the problem become easier when certain conditions are satisfied.

Proposition 2 (Lin and Bercher (2021, Thm. 5))**.** $\text{FIXMSEQ}^X_{\text{TO}}$ *with* $X \subseteq \{\text{ACT}^+_{\text{TO}}, \text{ACT}^-_{\text{TO}}\}$ *is NP-complete.*

Theorem 3. *Let* $X = \{\text{ACT}^+_{\text{TO}}, \text{ACT}^-_{\text{TO}}\}$. *The problem* $\text{FIXMSEQ}^X_{\text{TO}}$ *can be decided in constant time if* tn_I *contains no primitive tasks and there exists at least one method* $m_i = (c_i, tn_i)$ *(*$1 \leq i \leq n$*) in* $\overline{m}$ *such that for all* $m_j = (c_j, tn_j)$ *with* $1 \leq j \leq n$ *and* $j \neq i$, $c_i \neq c_j$.

Proof. Suppose m_i is the method in $\overline{m}$ that satisfies those conditions. A change sequence that turns tn into a solution can always be found by first removing every action from each method in $\overline{m}$ and then inserting the tasks in tn in turn into m_i. Thus, the problem is constant time decidable. □

Unfortunately, Thm. 3 does not hold in the case where we are only allowed to add or remove actions.

Theorem 4. $\text{FIXMSEQ}^{\text{ACT}^-_{\text{TO}}}_{\text{TO}}$ *is NP-complete even if* tn_I *and* $\overline{m}$ *satisfy the conditions presented in Thm. 3.*

Proof. Let $X = \text{ACT}^-_{\text{TO}}$. Membership follows from Prop. 2. For hardness, we reduce from the general $\text{FIXMSEQ}^{\text{X}}_{\text{TO}}$ problem. Let $P = (D, tn_I, s_I)$ with $D = (F, N_p, N_c, \delta, M)$, $\overline{m}$, and tn be the planning problem, the method sequence, and the task network given by an instance of the general $\text{FIXMSEQ}^{\text{X}}_{\text{TO}}$ problem, respectively. We construct an equivalent instance as follows. We first construct a planning problem $P' = (D', tn'_I, s_I)$ with $D' = (F, N_p, N_c \cup \{c'\}, \delta, M \cup \{m'\})$ where c' is an additional compound task name, $m' = (c', \varepsilon)$ decomposes c' into an empty task network, and $tn'_I = tn_I\, c'$. Afterwards, we construct the method sequence $\overline{m}' = \overline{m}\, m'$ and keep tn unchanged. Since m' results in an empty task network, we implicitly forbid ACT^-_{TO} being applied to it. Thus, the general problem has a 'yes' answer if and only if the problem we constructed has one. □

Theorem 5. $\text{FIXMSEQ}^{\text{ACT}^+_{\text{TO}}}_{\text{TO}}$ *is NP-complete even if* tn_I *and* $\overline{m}$ *satisfy the conditions presented in Thm. 3.*

Proof. Let $X = \text{ACT}^+_{\text{TO}}$. Membership follows from Prop. 2. For hardness, we reduce from the general $\text{FIXMSEQ}^{\text{X}}_{\text{TO}}$ problem. Let $P = (D, tn_I, s_I)$ with $D = (F, N_p, N_c, \delta, M)$, $\overline{m}$, and tn be the planning problem, the method sequence, and the task network given by an instance of the general $\text{FIXMSEQ}^{\text{X}}_{\text{TO}}$ problem, respectively. To complete the reduction, we first construct the planning problem $P' = (D', tn'_I, s_I)$ with $D' = (F, N_p \cup \{p'_1, p'_2\}, N_c \cup \{c'_1, c'_2\}, \delta, M \cup \{m'_1, m'_2\})$ where p'_1 and p'_2 are two additional primitive tasks, c'_1 and c'_2 are additional compound

tasks, $m'_1 = (c'_1, p'_1)$ and $m'_2 = (c'_2, p'_2)$ decompose c'_1 and c'_2 to p'_1 and p'_2, respectively, and $tn'_I = tn_I\, c'_1\, c'_1\, c'_2$. Next we construct the method sequence $\overline{m}' = \overline{m}\, m'_1\, m'_1\, m'_2$ and the task network $tn' = tn\, p'_1\, p'_1\, p'_2$ that should be a solution to the modified planning problem. The existence of p'_1 and p'_2 ensures that actions cannot be added to m'_1 and m'_2. Thus, the general $\text{FIXMSEQ}^{\text{X}}_{\text{TO}}$ instance has a *yes* answer if and only if the problem we construct has one. □

Next we extend our investigation to partially ordered HTN planning. We again consider the problem under the solution criterion given by Def. 7. Note that the polynomial upper bound presented in Lem.1 still holds because the methods in $\overline{m}$ is a subset of M, i.e., the number of methods that need to be changed is smaller than the size of M, and thus the minimal number of changes required must not exceed that upper bound. NP-membership thus follows immediately.

Corollary 5. *Let* $X \subseteq \{\text{ACT}^+_{\text{PO}}, \text{ACT}^-_{\text{PO}}, \text{ORD}^+, \text{ORD}^-\}$ *and* $X \geq 1$. $\text{FIXMSEQ}^{X}_{\text{PO}}$ *is in NP.*

On the other hand, the NP-hardness of the variants in a PO setting where only changing actions is allowed is a direct corollary of Prop. 2. Taken together, we immediately have the following result.

Corollary 6. $\text{FIXMSEQ}^{X}_{\text{PO}}$ *with* $X \subseteq \{\text{ACT}^+_{\text{PO}}, \text{ACT}^-_{\text{PO}}\}$ *and* $|X| \geq 1$ *is NP-complete.*

Next we consider the complexity of the problem when changing ordering constraints is allowed.

Theorem 6. $\text{FIXMSEQ}^{\text{ORD}^+}_{\text{PO}}$ *is NP-complete.*

Proof. Membership has been given by Cor. 5. For hardness, we again reduce from the independent set problem. Given any instance of the independent set problem, we first construct a planning problem P and a target task network tn that are identical to those presented in the proof of Thm. 2. We have argued there that any compound task in the constructed initial task network has only one method which can decompose it. Thus, we can choose any method sequence that results in a solution to P as $\overline{m}$, and the proof still holds. □

The presented proofs also imply that allowing any combination of the defined changes will not make the problem easier because of the same argument made for Cor. 3.

Corollary 7. *Let* $X \subseteq \{\text{ACT}^+_{\text{PO}}, \text{ACT}^-_{\text{PO}}, \text{ORD}^+, \text{ORD}^-\}$ *and* $X \geq 1$. $\text{FIXMSEQ}^{X}_{\text{PO}}$ *is NP-complete.*

Additionally, Thm. 3 can be further generalized in the framework of partially ordered HTN planning since a change sequence can always be constructed by following the same procedure if the conditions described there hold.

Theorem 7. $\text{FIXMSEQ}^{X}_{\text{PO}}$ *can be decided in constant time if* $\{\text{ACT}^+_{\text{PO}}, \text{ACT}^-_{\text{PO}}\} \subseteq X$, tn_I *does not contain any primitive task and there exists at least one unique method in* $\overline{m}$.

We now proceed to study the complexity of finding the minimum number of changes required. We again define the problem by introducing an extra integer k.

Definition 18. Let $k \in \mathbb{N}$. The problem $\text{FIXMSEQ}^{\text{X},k}_{\text{SET}}$ with $X \subseteq \{\text{ACT}^+_{\text{SET}}, \text{ACT}^-_{\text{SET}}, \text{ORD}^+, \text{ORD}^-\}$ and $X \geq 1$ and $\text{SET} \in \{\text{TO}, \text{PO}\}$ is identical to $\text{FIXMSEQ}^{\text{X}}_{\text{SET}}$ except that we demand that *any* change sequence is limited in size by k.

The NP-completeness of the problem in the context of totally ordered HTN planning has been given by our previous work (Lin and Bercher 2021). For partially ordered HTN planning, since the polynomial upper bound given by Lem. 1 still holds, the arguments made for Cor. 4 is still valid, which implies NP-completeness.

Corollary 8. *Let* $X \subseteq \{\text{ACT}^+_{\text{PO}}, \text{ACT}^-_{\text{PO}}, \text{ORD}^+, \text{ORD}^-\}$ *and* $X \geq 1$. $\text{FIXMSEQ}^{X,k}_{\text{PO}}$ *is NP-complete.*

One may ask whether there exist some conditions that make the problem easier once they are satisfied. For example, in totally ordered HTN planning, the problem can be decided in polynomial time if each method in the given method sequence decomposes a unique compound task (Lin and Bercher 2021). However, we cannot guarantee that the same argument holds in a partial order setting due to the existence of isomorphic task networks. Thus, whether such conditions exist in the context of partially ordered HTN planning is still an open question and will be studied in the future.

6 Complexity of Fixing the Methods – Given An Action Sequence

Our previous discussion over partially ordered HTN planning is based on the solution criterion given by Def. 7 because deciding whether a partially ordered task network has an executable linearisation is intractable. The remaining question is whether changing planning models becomes easier under the solution criterion given by Def. 6 if an executable linearisation of a task network is already provided. We formally define this problem as follows.

Definition 19. Let $X \subseteq \{\text{ACT}^+_{\text{PO}}, \text{ACT}^-_{\text{PO}}, \text{ORD}^+, \text{ORD}^-\}$ and $X \geq 1$, P be a partially ordered HTN planning problem, and π be an action sequence. We define the problem $\text{FIXTSEQ}^{X}_{\text{PO}}$ as: Is there a sequence of method-change operations $\mathcal{X}$ such that $P \rightarrow^*_{\mathcal{X}} P'$, P' has a solution that possesses a linearisation which is identical to π, and $\mathcal{X}$ consists of the operations with respect to the value of X.

Clearly, Lem. 1 still holds for this problem because an action sequence π is actually a totally ordered task network which itself is a special partially ordered task network. It then follows that all variants are in NP.

Corollary 9. *Let* $X \subseteq \{\text{ACT}^+_{\text{PO}}, \text{ACT}^-_{\text{PO}}, \text{ORD}^+, \text{ORD}^-\}$ *and* $X \geq 1$. $\text{FIXTSEQ}^{X}_{\text{PO}}$ *is in NP.*

If we restrict ourselves to totally ordered HTN planning and only consider the operations that change actions in methods, then the problem is identical to the one we studied before (Lin and Bercher 2021), which implies the NP-completeness of these variants.

Corollary 10. *Let* $X \subseteq \{\text{ACT}^+_{\text{PO}}, \text{ACT}^-_{\text{PO}}\}$ *and* $|X| \geq 1$. $\text{FIXTSEQ}^{X}_{\text{PO}}$ *is NP-complete.*

For the variants where only changing ordering constraints is allowed, it turns out that they are NP-complete as well.

Corollary 11. *Let* $X \subseteq \{\text{ORD}^+, \text{ORD}^-\}$ *and* $|X| \geq 1$. $\text{FIXTSEQ}^{X}_{\text{PO}}$ *is NP-complete.*

Proof. Membership has been given by Cor. 9. Hardness follows from that fact that VERIFYSEQ is NP-hard for the

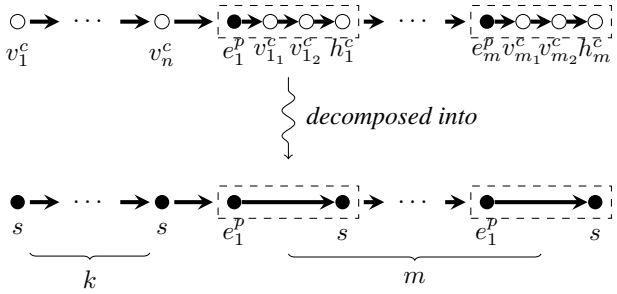

Figure 3: The construction for the proof of Prop. 1 in our earlier work (Lin and Bercher 2021). The constructed domain contains only one primitive task (action) s.

class $\text{HTN}_{\text{unordered}}$ (Behnke, Höller, and Biundo 2015, Cor. 5) where $\text{HTN}_{\text{unordered}}$ refers to the class of totally unordered HTN planning problems. If a planning problem is in $\text{HTN}_{\text{unordered}}$, deleting ordering constraints is clearly redundant. Adding ordering constrains is also pointless because those operations will only increase the possibility that a given action sequence is not a valid linearisation of a task network into which can be decomposed from the initial task network of a planning problem. Therefore, any VERIFY-SEQ instance with the input planning problem belonging $\text{HTN}_{\text{unordered}}$ can be reduced to a FIXTSEQ^{X}_{PO} instance with an arbitrary $X \subseteq \{\text{ORD}^+, \text{ORD}^-\}$ and $|X| \geq 1$. □

When it comes to the combination of changing actions and changing ordering constrains, we shall first consult the proof of Prop. 1 presented in our earlier work (Lin and Bercher 2021). The reduction we constructed is similar to the one shown in Thm. 2 except that tn_I is totally ordered, and each compound task in tn_I is now decomposed into an empty task network by the respective method, see Fig. 3. By construction, the only way to reach the target action sequence is by adding s to some methods. Thus, the operation that deletes an action immediately becomes pointless. Although our original proof is not concerned with changing ordering constraints, those are pointless as well because we can neither change the existed ordering constraints in tn_I nor add new ones to methods. The following result thus follows immediately.

Corollary 12. *Let* $X \subseteq \{\text{ACT}^+_{\text{PO}}, \text{ACT}^-_{\text{PO}}, \text{ORD}^+, \text{ORD}^-\}$ *and* $X \geq 1$. $\text{FIXTSEQ}^{X}_{\text{PO}}$ *is NP-complete.*

The decision problem aiming at finding the minimal number of changes required is formulated as follows.

Definition 20. Let $k \in \mathbb{N}$, P be a partially ordered HTN planning problem, and π be an action sequence. For each $X \subseteq \{\text{ACT}^+_{\text{PO}}, \text{ACT}^-_{\text{PO}}, \text{ORD}^+, \text{ORD}^-\}$ and $X \geq 1$, the problem $\text{FIXTSEQ}^{X,k}_{\text{PO}}$ is identical to $\text{FIXTSEQ}^{X}_{\text{PO}}$ except that we demand that *any* change sequence is bounded by k.

Both membership and hardness are implied by the presence of the polynomial upper bound of the minimal number of changes required.

Corollary 13. *Let* $X \subseteq \{\text{ACT}^+_{\text{PO}}, \text{ACT}^-_{\text{PO}}, \text{ORD}^+, \text{ORD}^-\}$ *and* $X \geq 1$. $\text{FIXTSEQ}^{X,k}_{\text{PO}}$ *is NP-complete.*

7 Conclusion

We investigated the computational complexity of deciding whether there exists a sequence of model change operations (could be of limited length) that transforms a planning problem into another one that has a given task network as a solution in the context of partially ordered HTN planning. Our results indicate that the problem is NP-complete unless additional constraints are specified, e.g., having no primitive task in the initial task network of a planning problem and having no duplicate methods in a decomposition method sequence that is supposed to generate a solution. Our results can be exploited in the future by transforming the decision problems into some well-studied NP-complete problems which can be solved by efficient solvers, e.g., SAT, and fully integrating model-change into MIP systems.

References

Ai-Chang, M.; Yglesias, J.; Chafin, B.; Dias, W.; Maldague, P.; Bresina, J.; Charest, L.; Chase, A.; Hsu, J.-J.; Jonsson, A.; Kanefsky, B.; Morris, P.; and Rajan, K. 2004. MAPGEN: mixed-initiative planning and scheduling for the Mars Exploration Rover mission. *IEEE Intelligent Systems* 19: 8–12.

Barták, R.; Maillard, A.; and Cardoso, R. C. 2018. Validation of Hierarchical Plans via Parsing of Attribute Grammars. In *Proceedings of the 28th International Conference on Automated Planning and Scheduling, ICAPS 2018*, 593–600. AAAI.

Barták, R.; Ondrčková, S.; Behnke, G.; and Bercher, P. 2021a. Correcting Hierarchical Plans by Action Deletion. In *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021*. IJCAI.

Barták, R.; Ondrčková, S.; Behnke, G.; and Bercher, P. 2021b. On the Verification of Totally-Ordered HTN Plans. In *Proceedings of the 4th ICAPS Workshop on Hierarchical Planning, HPlan 2021*.

Barták, R.; Ondrčková, S.; Maillard, A.; Behnke, G.; and Bercher, P. 2020. A Novel Parsing-based Approach for Verification of Hierarchical Plans. In *Proceedings of the 32th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2020*, 118–125. IEEE.

Behnke, G.; Höller, D.; Bercher, P.; and Biundo, S. 2016. Change the Plan - How Hard Can That Be? In *Proceedings of the 26th International Conference on Automated Planning and Scheduling, ICAPS 2016*, 38–46. AAAI.

Behnke, G.; Höller, D.; and Biundo, S. 2015. On the Complexity of HTN Plan Verification and Its Implications for Plan Recognition. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling, ICAPS 2015*, 25–33. AAAI.

Behnke, G.; Höller, D.; and Biundo, S. 2017. This is a solution! (... but is it though?) - Verifying solutions of hierarchical planning problems. In *Proceedings of the 27th International Conference on Automated Planning and Scheduling, ICAPS 2017*, 20–28. AAAI.

Bercher, P. 2021. A Closer Look at Causal Links: Complexity Results for Delete-Relaxation in Partial Order Causal Link (POCL) Planning. In *Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling, ICAPS 2021*, 36–45. AAAI.

Bercher, P.; Alford, R.; and Höller, D. 2019. A Survey on Hierarchical Planning - One Abstract Idea, Many Concrete Realizations. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI 2019*, 6267–6275. IJCAI.

Bercher, P.; Höller, D.; Behnke, G.; and Biundo, S. 2016. More than a Name? On Implications of Preconditions and Effects of Compound HTN Planning Tasks. In *The 22nd European Conference on Artificial Intelligence, ECAI 2016*, 225–233. IOS.

Bercher, P.; and Olz, C. 2020. POP $\equiv$ POCL, Right? Complexity Results for Partial Order (Causal Link) Makespan Minimization. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence, AAAI 2020*, 9785–9793. AAAI.

Bresina, J. L.; Jónsson, A. K.; Morris, P. H.; and Rajan, K. 2005. Activity Planning for the Mars Exploration Rovers. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling, ICAPS 2005*, 40–49. AAAI.

Chakraborti, T.; Sreedharan, S.; and Kambhampati, S. 2020. The Emerging Landscape of Explainable Automated Planning & Decision Making. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence, IJCAI 2020*, 4803–4811. IJCAI.

Chakraborti, T.; Sreedharan, S.; Zhang, Y.; and Kambhampati, S. 2017. Plan Explanations as Model Reconciliation: Moving Beyond Explanation as Soliloquy. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI 2017*, 156–163. IJCAI.

Chapman, D. 1987. Planning for Conjunctive Goals. *Artificial Intelligence* 32(3): 333–377.

Erol, K.; Hendler, J.; and Nau, D. S. 1996. Complexity results for HTN planning. *Annals of Mathematics and Artificial Intelligence* 18: 69–93.

Ferguson, G.; and Allen, J. F. 1998. TRIPS: An Integrated Intelligent Problem-Solving Assistant. In *Proceedings of the 15th AAAI Conference on Artificial Intelligence, AAAI 1998*, 567–572. AAAI.

Ferguson, G.; Allen, J. F.; and Miller, B. W. 1996. TRAINS-95: Towards a Mixed-Initiative Planning Assistant. In *Proceedings of the 3rd International Conference on Artificial Intelligence Planning Systems, ICAPS 1996*, 70–77. AAAI.

Geier, T.; and Bercher, P. 2011. On the Decidability of HTN Planning with Task Insertion. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI 2011*, 1955–1961. AAAI.

Ginsberg, M. L. 1986. Counterfactuals. *Artificial Intelligence* 30: 35–79.

Göbelbecker, M.; Keller, T.; Eyerich, P.; Brenner, M.; and Nebel, B. 2010. Coming Up With Good Excuses: What to do When no Plan Can be Found. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling, ICAPS 2010*, 81–88. AAAI.

Keren, S.; Pineda, L. E.; Gal, A.; Karpas, E.; and Zilberstein, S. 2017. Equi-Reward Utility Maximizing Design in Stochastic Environments. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI 2017*, 4353–4360. IJCAI.

Lin, S.; and Bercher, P. 2021. Change the World – How Hard Can that Be? On the Computational Complexity of Fixing Planning Models. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence, IJCAI 2021*.

Nebel, B.; and Bäckström, C. 1994. On the Computational Complexity of Temporal Projection, Planning, and Plan Validation. *Artificial Intelligence* 66(1): 125–160.

Olz, C.; Wierzba, E.; Bercher, P.; and Lindner, F. 2021. Towards Improving the Comprehension of HTN Planning Domains by Means of Preconditions and Effects of Compound Tasks. In *Proceedings of the 10th Workshop on Knowledge Engineering for Planning and Scheduling, KEPS 2021*.

Sreedharan, S.; Chakraborti, T.; Muise, C.; Khazaeni, Y.; and Kambhampati, S. 2020. – D3WA+ – A Case Study of XAIP in a Model Acquisition Task for Dialogue Planning. In *Proceedings of the 30th International Conference on Automated Planning and Scheduling, ICAPS 2020*, 488–497. AAAI.

Sreedharan, S.; Srivastava, S.; and Kambhampati, S. 2018. Hierarchical Expertise Level Modeling for User Specific Contrastive Explanations. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI 2018*, 4829–4836. IJCAI.

Sreedharan, S.; Srivastava, S.; Smith, D.; and Kambhampati, S. 2019. Why Can't You Do That HAL? Explaining Unsolvability of Planning Tasks. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI 2019*, 1422–1430. IJCAI.