Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICAPS 2021)

A Closer Look at Causal Links: Complexity Results for Delete-Relaxation in Partial Order Causal Link (POCL) Planning

Pascal Bercher
The Australian National University,
College of Engineering and Computer Science,
School of Computing, Canberra, Australia
pascal.bercher@anu.edu.au

Abstract

Partial Order Causal Link (POCL) planning follows the principle of least commitment in that it maintains only a partial order on its actions to prevent unnecessary early commitment during search. This can reduce the search space significantly by systematically representing up to an exponential number of action sequences in just a single search node. Progress on goal achievement is represented fully by this partial order and by causal links, which represent the causal relationships between these actions as well as between the initial state and goal. Plan existence for a state in classical planning thus corresponds to plan existence for a partial plan in POCL planning. Yet almost no theoretical investigations for POCL plan existence were conducted so far. While delete-relaxation makes plan existence tractable in classical planning, we show it to be **NP**-hard in POCL planning unless the current plan is totally ordered or causal links are almost completely ignored.

Introduction

Partial Order Causal Link (POCL) planning is a planning technique that follows the principle of least commitment (Weld 1994). Plans are only partially ordered, thereby representing many possible action sequences in one single search node, which can result into an exponential search space reduction (Minton, Bresina, and Drummond 1994). In lifted POCL planning, least commitment can further prevent unnecessary early variable assignments by only binding those necessary to reach the current subgoals, leading to further search space reductions (Younes and Simmons 2002).

In the early days of AI planning, POCL search was the dominating planning approach, but it was abandoned by many researchers in favor of better-scaling approaches in the late 90s (Weld 2011). Today, undoubtedly the currently most successful approach is heuristic progression search in the space of states. The success of that approach is mainly based upon well-informed heuristics that estimate the distance from a current state to a goal state in terms of missing actions or action costs. However, POCL planning was never theoretically proven to be generally inferior to other approaches, it is simply being less advanced at the moment due to the non-availability of informed heuristics.

Though POCL planning algorithms are currently inferior to other approaches when it comes to classical plan existence, POCL approaches seem attractive when it comes to solving problems where execution time matters – due to POCL plans' inherent parallel action representation (Benedictis and Cesta 2020; Vidal and Geffner 2006, 2004), and there also exists an approach combining POCL techniques with state-based forward search for temporal planning (Coles et al. 2010).

Today, POCL techniques are actively deployed in many *hierarchical* planning approaches (Bercher et al. 2016). E.g., the FAPE planner (Bit-Monnot et al. 2020; Bit-Monnot, Smith, and Do 2016; Dvořák et al. 2014) is a plan-space based hierarchical planner that focuses on dealing with problems with time. HATP (Sebastiani et al. 2017; Lallement, de Silva, and Alami 2018) and CHIMP (Stock et al. 2015) are recent hierarchical planners deployed to robotics. HiPOP (Bechon et al. 2014) and its successor pyHiPOP (Lesire and Albore 2021), as well as the PANDA planner (Bercher et al. 2017; Bercher, Keen, and Biundo 2014) are two further current POCL-based hierarchical planners (which do not feature time). Another current hierarchical planner based on POCL techniques is deployed for the generation of narratives (Winer and Young 2016).

As Weld (2011) pointed out, heuristic development in POCL planning is much more complicated than in state-based progression search, because there is no current state available, but instead a partially ordered set of actions with causal links between them, which encode which preconditions of these actions have already been achieved (by the other actions in the current plan). The same observation holds for POCL-based hierarchical planning, where heuristics are again currently inferior to those developed for state-based progression in hierarchical planning (Höller et al. 2020; 2018). Whereas the complete progress in state-based search is given by a current state, the search progress in POCL planning is given by the current POCL search node. So any heuristic that relaxes the current plan would not just make the "search path to the goal" easier, but it would also throw away information about the current search progress thus making the semantics of the goal distance a bit harder to grasp. In progression search, this would correspond to not using the current state as a basis for heuristic estimates, but "some" superset. Yet all currently exist-

actions of initial plan	causal links respected?	ordering constraints	insertable actions	Computational Complexity	Theorem
*	*	*	original	**PSPACE**-complete	Thm. 1 and Cor. 1
original	*	partial order	delete-relaxed	**NP**-complete	Prop. 2 and Cor. 2
delete-relaxed	no	*	delete-relaxed	in **P**	Thm. 2
*	*	total order	delete-relaxed	in **P**	Thm. 3 and Cor. 3
*	yes	partial order	delete-relaxed	**NP**-complete	Thm. 4
original	*	partial order	none	**NP**-complete	Nebel and Bäckström (1994, Thm. 15) Erol, Hendler, and Nau (1996, Thm. 8)
delete-relaxed	*	*	none	in **P**	Prop. 3

Table 1: The table summarizes our complexity results for POCL planning as well as the most related results from the literature. If a cell contains "*" the results hold for both possibilities. Note that the column on causal links relates to Def. 5, where we define a weaker problem relaxation demanding that the initially given causal links have to be respected even by delete-relaxed actions (it thus strengthens the impact of causal links despite delete relaxation).

ing POCL heuristics do perform some relaxation on the current plan, namely by either relaxing the ordering constraints, the delete effects of the actions already present in the current plan, the current plan's causal links, or any combination thereof[1] (Shekhar and Khemani 2015; Bercher, Geier, and Biundo 2013; Bercher et al. 2013; Younes and Simmons 2003; Nguyen and Kambhampati 2001; Bylander 1997).

Apart from a search node's actions, it's the causal links that encode the search progress. Informally, a causal link connects one action's effect with another action's precondition thus *protecting* it, i.e., it prevents any other action that deletes that condition to be inserted or ordered in between these actions. Thus, any causal link induces a *constraint* on which action may be ordered (or inserted) at which position within a plan. This "pruning power" of a causal link has a severe influence on which actions may remain available in certain intervals. Ignoring this pruning power in heuristics is thus a relaxation on the current *search progress*, yet this is done by both state-of-the art heuristics in POCL planning (Younes and Simmons 2003; Nguyen and Kambhampati 2001). It is however not known whether these relaxations were actually necessary for tractability, i.e., whether it would be possible to exploit the pruning power of causal links without becoming intractable. The same applies to all POCL-based hierarchical planners we mentioned earlier. To the best of our knowledge, none of the deployed heuristics exploits the causal links for pruning or making the heuristics more informed. We believe that this is also a consequence of lacking theoretical investigations of POCL plans – an opinion shared with others (Tan and Gruninger 2014).

To obtain a fundamental understanding of the computational hardness in POCL planning, and the role of causal links in particular, we make the following contributions: We first prove that POCL planning problems, where we are given a POCL plan as input rather than a standard classical planning problem, is **PSPACE**-complete. We then provide a complete picture of the computational complexity of delete-relaxation in POCL planning. Delete relaxation is one of the most important problem relaxations in state-based planning, because it results into a tractable problem class (Bylander 1994). We investigate delete-relaxation for the input plan, the action portfolio (i.e., those actions that can be inserted), and both. We furthermore introduce a novel problem relaxation, where delete-relaxed actions may be inserted into plans, but only in "intervals" where their original, non-relaxed versions do not conflict with existing causal links from the input plan. All these results are studied for partially and totally ordered input plans. The results vary between tractability (**P**) and intractability (**NP**-complete), where tractability can only be achieved for totally ordered input plans or when the pruning power of causal links is ignored completely. Tab. 1 summarizes our results.

Related Work on Complexity Studies in POCL Planning

In the context of Hierarchical Task Network (HTN) planning, Erol, Hendler, and Nau (1996) showed that it is **NP**-complete to decide whether a partially ordered set of actions has an executable linearization. Nebel and Bäckström (1994) studied various questions regarding partially ordered plans, such as possible or necessary truth of properties before or after plan steps. Their most important result related to our studies is, similar to the result by Erol, Hendler, and Nau, the **NP**-completeness of deciding whether a partially ordered set of actions possesses an executable linearization. This can be regarded as a base-case of POCL planning, where all required actions are already inserted, and one only needs to figure out whether they can be arranged in the right way. Kambhampati and Nau (1996) studied related questions under different assumptions. The authors, quote, "believe that the results in this paper complement [the ones by Nebel and Bäckström] and together provide a coherent interpretation of the role of modal truth criteria in planning". Motivated by designing well-informed POCL heuristics, Bercher

[1] To be precise, Bercher et al. (2013) present a heuristic that does not do any of these relaxations, i.e., it respects delete effects of the initial plan as well as all causal links. To remain tractable they relax the problem by guessing a fixed number of linearizations which required them to fall back to a different heuristic in those cases where all guessed linearizations turned out unsolvable.

et al. (2013) generalized Nebel and Bäckström's as well as Erol, Hendler, and Nau's result by showing that this problem does not become easier when one is allowed to insert delete-relaxed actions (cf. Prop. 2). Also motivated by providing theoretical insights for POCL heuristic design, Tan and Gruninger (2014) directly built on Nebel and Bäckström's results and refined them by a more in-depth analysis showing the impact of the interaction of preconditions and effects thereby giving a more detailed picture for when finding such a linearization is possible in polynomial time, and when it remains **NP**-hard.

Only loosely related to our problem are various complexity investigations regarding plan optimization. POCL plans are particularly attractive for reasoning about plan optimization since causal links explicitly represent the causal relationships between actions (Waters, Padgham, and Sardina 2020; Waters et al. 2018; Muise, Beck, and McIlraith 2016). One work investigates the computational hardness for optimizing the action set of a POCL plan (Olz and Bercher 2019), whereas several works investigate the computational hardness of optimizing ordering constraints or execution time (makespan) for partially ordered plans and POCL plans in particular (Bercher and Olz 2020; Aghighi and Bäckström 2017; Bäckström 1998).

Problem Formalization

Often POCL plans are defined in a lifted manner, where the propositions are formalized using first-order predicates rather than (ground) propositions, as this allows for even less commitment due to delayed variable assignment (Younes and Simmons 2002). For the sake of simplicity, we base our complexity investigations on a propositional model. The studies can still be transferred to a formalization with variables, where our results serve as lower bounds.

We first define classical planning problems in the standard STRIPS notation, and later on extend it to the notation of POCL plans and POCL problems. We follow the notation used by Bercher and Olz (2020).

A *classical planning problem* is a tuple (F, A, s_I, g) defined as follows. F denotes a finite set of *fluents*, they describe the propositional world properties. We call any $s \in 2^F$ a state. $A \subseteq 2^F \times 2^F \times 2^F$ is a finite set of *actions*. For an action $a \in A$, $pre(a) \subseteq F$ denotes its *preconditions*, $add(a) \subseteq F$ its *add effects*, and $del(a) \subseteq F$ its delete effects. For the purpose of this paper we restrict to the case where $add(a) \cap del(a) = \emptyset$ holds for all $a \in A$. According to the standard definition of action applicability (see below), this case can be compiled away via removing any elements from $del(a)$ that also occur in $add(a)$ since precedence is given to add effects over delete effects[2]. An action $a \in A$ is called *applicable* (or *executable*) in a state $s \in 2^F$ if and only if $pre(a) \subseteq s$. If a is applicable in s, the state transition function $\gamma : A \times 2^F \to 2^F$ returns its successor state $\gamma(a, s) = (s \setminus del(a)) \cup add(a)$. A sequence of actions $\overline{a} = (a_1, \ldots, a_n)$ is applicable in a state s_0 if there exists a sequence of states $s_1, \ldots, s_n$, such that for all $1 \leq i \leq n$ it holds that a_i is applicable in state $\gamma(a_i, s_{i-1}) = s_i$. The state s_n is called *the state generated* by $\overline{a}$. Finally, $s_I \in 2^F$ is the *initial state* and $g \subseteq F$ is the *goal description* implicitly defining a set of goal states $G = \{s \mid s \supseteq g\}$.

Definition 1 (Classical Solution to Classical Problem)**.** *An action sequence $\overline{a}$ is called a* classical solution *to a (classical) planning problem (F, A, s_I, g) if and only if it is applicable to s_I and generates a goal state $s \supseteq g$.*

When applying POCL search algorithms to solve a classical planning problem, search nodes are so-called *partial plans*, i.e., partially ordered sets of actions, which use *causal links* as the main means to evaluate and achieve executability (Weld 1994). We will also call these partial plans *POCL plans* or just *plans* for short, as it is always clear from context whether we refer to an actual solution (which would usually be referred to as "plan"). More formally, a *plan* is a tuple $P = (PS, \prec, CL)$, where PS is a finite set of *plan steps*, where each plan step $(l, a) \in PS$ consists of an action $a \in A$, labeled with a label l unique in PS. $\prec \subseteq PS \times PS$ is a strict partial order defined on the plan steps. We require action labeling to differentiate between multiple occurrences of an action in the same plan. Just like for actions, we use the notation $pre(ps)$, $add(ps)$, and $del(ps)$ to refer to the precondition, add, and delete effects of a plan step's action.

The set of causal links $CL \subseteq PS \times F \times PS$ is used to annotate the causal relationships between plan step preconditions and effects, and they will be used to define whether or not a partial plan is executable. A *causal link* $(ps, f, ps') \in CL$ indicates that f is both a precondition of ps', the *consumer* of the link, as well as an effect of ps, the *producer* of the link (i.e., such a link implies $f \in add(ps) \cap pre(ps')$). We also demand that every causal link $(ps, f, ps') \in CL$ implies $(ps, ps') \in \prec$. Let $(ps, f, ps') \in CL$ be a causal link. Its fluent f is also called *protected by that link*, because all POCL-based search algorithms will ensure that f will never be deleted between those steps. Formally, we say that a partial plan $P = (PS, \prec, CL)$ raises a *causal threat* if there exists a causal link $(ps, f, ps') \in CL$ as well as a *threatening plan step* $ps'' \in PS$, such that ps'' could be ordered between ps and ps' while violating the protected condition. Formally, ps'' is threatening (ps, f, ps') if and only if $f \in del(ps'')$, and $(\prec \cup \{(ps, ps''), (ps'', ps')\})^+$ (where X^+ denotes the transitive closure of X) is a strict partial order.

To represent the initial state and goal state of a classical planning problem, we further extend the definition of POCL plans. We demand that any POCL plan contains two special plan steps, we call them *init* and *goal*, such that *init* is ordered strictly before all other steps, and *goal* is ordered strictly after each other step. The respective actions (which will not be part of the action set A, so that they cannot be inserted) are also called *init* and *goal*, and defined by $pre(init) = del(init) = add(goal) = del(goal) = \emptyset$, as well as by $add(init) = s_I$ and $pre(goal) = g$. We can

[2] Planning models with such overlappings in the add and delete effects are equivalent regarding the standard notion of applicability for solution *sequences* as defined later. However, such cases might be introduced on purpose when aiming at *partially ordered solutions* to enforce certain ordering constraints. Dealing with these cases will however complicate some theorems and proofs, so we still pose this restriction and leave dealing with it future work.

now define what it means for a POCL plan to be a solution.

Definition 2 (POCL Solution to Classical Problem)**.** *A POCL plan P is called a* POCL solution *to a (classical) planning problem if and only if every precondition of its plan steps is protected by a causal link and it does not raise any causal threat.*

Please note the following (commonly known) observation relating Def. 1 to Def. 2.

Proposition 1 (Solution Correspondence)**.** *Let $\mathcal{P}$ be a classical planning problem. Then, $\mathcal{P}$ has a classical solution if and only if it has a POCL solution.*

This proposition is easy to see because any POCL solution is just a compact representation of up to an exponential number of classical solutions, so any linearization of its plan steps compatible with its ordering constraints is a classical solution. The other way round, any action sequence can be turned into a (totally ordered) POCL plan by simply inserting the required causal links, which can be done in polynomial time (Kambhampati 1994; Muise, Beck, and McIlraith 2016).

To be able to express the decision problem whether a current search node (i.e., POCL plan) has a solution, we now provide a definition of a *POCL planning problem* that generalizes classical planning problems in the sense that the "problem to solve" is not just given by an initial state and goal description, but by an arbitrary POCL plan. Note that every search node produced during plan-based POCL search can be regarded as a POCL planning problem thus motivating our definition.

A *POCL planning problem* is a tuple $(F, A, P_{s_I,g})$, consisting of a finite set of fluents F, a finite set A of actions, and an initial partial plan $P_{s_I,g} = (PS_I, \prec_I, CL_I)$. The POCL plan $P_{s_I,g}$ contains the two designated plan steps $\{init, goal\} \subseteq PS_I$ as defined above, which are set according to the problem's initial state $s_I \in 2^F$ and goal description $g \subseteq F$. We call the problem *totally ordered* if $\prec_I$ is a total order, and *partially ordered* otherwise.

We can now define a solution to a POCL problem as any solution plan that is a refinement of $P_{s_I,g}$, i.e., it may contain additional plan steps, causal links, and ordering constraints, but it may not *change* any of the contents of $P_{s_I,g}$. This reflects the idea of "refinement planning" (Kambhampati 1997) and any existing POCL search procedure, where decisions taken can never be taken back.

Definition 3 (POCL Solution to POCL Problem)**.** *Let $\mathcal{P} = (F, A, P_{s_I,g})$ be a POCL planning problem and P be a POCL plan. P is a POCL solution to $\mathcal{P}$ if and only if:*

- *Let $P = (PS, \prec, CL)$ and $P_{s_I,g} = (PS_I, \prec_I, CL_I)$. Then, $PS \supseteq PS_I$, $\prec \supseteq \prec_I$, and $CL \supseteq CL_I$.*
- *Every precondition of P's plan steps is protected by a causal link and P does not raise any causal threat.*

The first criterion demands that any solution is a refinement of the initial plan, meaning that it is a superset of the input with regard to the ordering constraints, causal links, and plan steps. The second criterion ensures the plan's executability, which, because it contains the *init* and *goal* steps, implies that the original classical problem is solved.

Whenever a POCL algorithm produces some POCL search node P, we can now consider it as a POCL planning problem, and thus investigate its computational complexity.

Complexity Results: Plan Existence

In this section we study the computational complexity of the *plan existence problem*, i.e., we are interested in the complexity of deciding whether there is a solution that can be reached from the current POCL plan. We start with the general case and investigate a range of results related to delete relaxation later on.

General Case

It is well known that deciding (ground) classical planning problems is **PSPACE**-complete (Bylander 1994). For POCL problems, we don't just want to achieve some goal fluents from an initial state, but we must make sure that the initially given POCL plan is respected, which poses various further restrictions due to the actions/plan steps already present as well as the causal links defined among them, which further limit the amount of actions that can be used at certain positions within a final plan. This, however, will not increase the computational hardness as stated next.

Theorem 1. *Let $\mathcal{P}$ be a totally or partially ordered POCL planning problem. Deciding whether $\mathcal{P}$ has a solution is* **PSPACE***-complete.*

Proof. Membership: If the problem is partially ordered, guess a linearization of the plan steps of $P_{s_I,g} = (PS_I, \prec_I, CL_I)$ that is compatible with its ordering constraints $\prec_I$. Let that sequence be $ps_0, \ldots, ps_{n+1}$, where $|PS_I| = n + 2$, $ps_0 = init$, and $ps_{n+1} = goal$. We will create $n + 1$ classical planning problems, i.e., one for the goal and one for each of the "inner" plan steps of $P_{s_I,g}$. We will construct them in a way such that the solutions for the different subproblems can be concatenated later on.

For this, we simply have to guess a sequence of n states, $s_1, \ldots, s_n$ satisfying $s_i \supseteq pre(ps_i)$ for all $1 \leq i \leq n$. Each of these states will be used as the goal description for the problem defined between ps_{i-1} and ps_i. The required initial states of the *next* subproblem can be obtained by applying the respective steps to the guessed states. With s'_i we represent the initial state *after* a step ps_i. Also define $s'_0 = s_I$ and $s_{n+1} = g$ thus defining $n + 1$ classical planning problems of the form (s'_{i-1}, s_i), for all $1 \leq i \leq n + 1$.

We need to make sure that available actions between ps_{i-1} and ps_i respect causal links, defined by:

$$A_{i-1,i} = A \setminus \{a \in A \mid f \in del(a), (ps_j, f, ps_k) \in CL, \text{ and } j < i \leq k\} \text{ for all } 1 \leq i \leq n + 1$$

This gives us the $n + 1$ classical problems $(F, A_{i-1,i}, s'_{i-1}, s_i)$, for $1 \leq i \leq n + 1$. Their solutions, if they exist, can easily be concatenated without invalidating the resulting complete plan. All intermediate (classical) problems can be solved in **PSPACE** (Bylander 1994), and we know that there exists a classical solution if and only if there exists a POCL solution (Prop. 1). We furthermore had to guess the linearization and n states.

Since $\mathbf{NPSPACE} = \mathbf{PSPACE}$ (Savitch 1970), we obtain $\mathbf{PSPACE}$ membership in total.

Hardness: POCL problems generalize classical problems, which are $\mathbf{PSPACE}$-complete (Bylander 1994). □

Delete Relaxation

We will now investigate the plan existence complexity for *delete relaxation*, where action delete effects are ignored. Other than in progression search in the space of states it is not completely clear how delete relaxation should be defined in the POCL setting. The core difference is that in our case the input plan also contains actions as well as causal links, so the question is whether they should be delete-relaxed as well, and what the impact on the computational complexity will be if we do or don't. As argued previously, not relaxing anything that is part of the initially given plan would be motivated by not throwing away information encoding the current search progress. We will consider delete-relaxation for either the input plan, the action portfolio, or both.

Definition 4 (Delete-relaxed POCL Problem)**.** *Let $\mathcal{P} = (F, A, P_{s_I,g})$ be a POCL planning problem and $P_{s_I,g} = (PS_I, \prec_I, CL_I)$ its initial partial plan.*

- *We call $\mathcal{P}^{A^+} = (F, A^+, P_{s_I,g})$ the POCL planning problem with* delete-relaxed actions *and define the set A^+ as $\{(pre(a), add(a), \emptyset) \mid a \in A\}$. Note that the actions in the plan steps of $P_{s_I,g}$ are still taken from A.*
- *We call $\mathcal{P}_{P^+} = (F, A, P^+_{s_I,g})$ the POCL planning problem with* delete-relaxed plan *and define the partial plan $P^+_{s_I,g} = (PS'_I, \prec_I, CL_I)$[3] with $PS'_I = \{(l, a^+) \mid (l, a) \in PS_I,$ and $a^+ = (pre(a), add(a), \emptyset)\}$.*
- *Consequently, $\mathcal{P}^{A^+}_{P^+} = (F, A^+, P^+_{s_I,g})$ denotes the POCL planning problem with* delete-relaxed actions and plan*.*

We start our investigations with the case where the input plan is delete-relaxed, but the actions to insert are not. As we have seen from our previous proof, the impact of the initial partial plan is rather limited in this case, since it merely defines a linear sequence of classical problems, one for each interval between two consecutive plan steps. So it is not surprising that the problem stays hard if we do not relax the action set that is used to solve these problems:

Corollary 1. *Let $\mathcal{P}_{P^+}$ be a totally or partially ordered POCL planning problem with delete-relaxed plan. Deciding whether $\mathcal{P}_{P^+}$ has a solution is $\mathbf{PSPACE}$-complete.*

So it is clear that we need to relax the available action set as well. Let us start our investigations with the most restricted case, where both the initial partial plan is delete-relaxed, as well as all actions that can be inserted. This problem class is the underlying one that's currently used by the state-of-the art POCL heuristics *Add Heuristic for POCL planning* (Younes and Simmons 2003) and *Relax* (Nguyen and Kambhampati 2001).

Theorem 2. *Let $\mathcal{P}^{A^+}_{P^+}$ be a totally or partially ordered POCL planning problem with delete-relaxed actions and plan. Deciding whether $\mathcal{P}^{A^+}_{P^+}$ has a solution is in $\mathbf{P}$.*

[3]Technically, we also had to alter $\prec_I$ so that it's defined upon PS'_I instead upon PS_I. Omitted to improve readability.

Proof. It is commonly known that delete-relaxed classical problems can be decided in polynomial time (Bylander 1994). In our case, however, we also have to respect the available partial order as well as existing causal links. Causal links can effectively be ignored, because the only way they would impose a constraint is if there were any delete effects that could cause causal threats, which is not the case due to the full delete-relaxation. The following procedure, running in $\mathbf{P}$, can decide the problem:

1. Construct the delete-relaxed planning graph (RPG) until a fixed point is reached. This can be done in polynomial time (Hoffmann and Nebel 2001).
2. Note that all actions that could possibly by applied are now part of the RPG, and that the final layer of fluents can not grow anymore, even after applying the actions in the plan. So we only need to check whether the actions in the plan can be applied in this final layer. Also note that ordering constraints do not matter anymore. We can thus iterate over the plan's actions in any order and check whether their preconditions are part of the final RPG fluent layer. Return "success" if and only if this is the case.

□

We now ask whether it is required to also ignore the delete effects of the input plan as well as its causal links (on top of delete-relaxing the action portfolio) to obtain tractability. A partial answer to this question was given previously. It shows that delete-relaxing the actions, but not the current plan is $\mathbf{NP}$-complete (Bercher et al. 2013, Thm. 1). We re-state it here for the sake of completeness and to have it available in our formalization.

Proposition 2. *Let $\mathcal{P}^{A^+}$ be a POCL planning problem with delete-relaxed actions. Deciding whether $\mathcal{P}^{A^+}$ has a solution is $\mathbf{NP}$-complete.*

This result was shown via a reduction from SAT, adapting the proof of Thm. 15 by Nebel and Bäckström (1994), which states that it is $\mathbf{NP}$-complete to decide whether a partially ordered set of actions has an executable linearization.[4] Thus, this result gets generalized by Prop. 2 in the sense that finding an executable linearization of a partially ordered set of actions does not become easier when we are allowed to insert delete-relaxed actions. This result is quite interesting because it shows that it is not sufficient to delete-relax the actions in the action portfolio to obtain tractability – which is in contrast to state-based search (Bylander 1994), where the tractability result is exploited by the influential FF heuristic (Hoffmann and Nebel 2001) and many more.

[4]Note that Nebel and Bäckström and Erol, Hendler, and Nau (1996) showed the same result independently. Nebel and Bäckström showed this for unconditional event systems, which coincides with standard STRIPS. Erol, Hendler, and Nau showed it for a special case of HTN planning, where the problem is given by a *primitive* task network, i.e., a partially ordered set of actions. Erol, Hendler, and Nau's proof exploits a constraint formula, which is not available to us. However, when restricting the SAT formula to be in conjunctive normal form (as Nebel and Bäckström did), the proof can easily be altered to work without a constraint formula.

If the input plan is restricted to be totally ordered, we obtain a lower complexity:

Theorem 3. *Let $\mathcal{P}^{A^+}$ be a totally ordered POCL planning problem with delete-relaxed actions. Deciding whether $\mathcal{P}^{A^+}$ has a solution is in* $\mathbf{P}$.

Proof. The following is a polytime decision procedure:

1. Check whether there are causal threats in the input plan. If so, return *fail*, because such a threat cannot be repaired since the plan is already totally ordered. If not, continue. Causal links can be ignored from now on since no further threats can be raised.
2. Create a series of delete-relaxed classical planning problems (F, A^+, s'_{i-1}, s_i), $1 \leq i \leq n+1$, similar to the membership proof of Thm. 1. As before, define $s'_0 = s_I$ and $s_{n+1} = g$. This proof, however, relied on guessing the goal states s_i, which we obtain differently:
   - Construct the RPG in s'_{i-1} until a fixed point is reached, which can be done in $\mathbf{P}$ (Hoffmann and Nebel 2001). This fixed point will serve as s_i.
   - Compute s'_i via the application of ps_i, the ith plan step in the sequence, to s_i.

   All subproblems are delete-relaxed and can thus be decided in $\mathbf{P}$ (Bylander 1994), and there exists a solution to $\mathcal{P}^{A^+}$ if and only if each subproblem has a solution.

□

We can still slightly improve on this result by yet another problem relaxation the literature did not consider so far.

Delete-Relaxation Respecting Causal Links

We have seen in the previous proof that for totally ordered input plans and delete-relaxed actions to be inserted, the subproblems can be decided in polynomial time. The proof furthermore "ignored" the influence of causal links present in this plan on the actions to be inserted. Those causal links did not play any role for the inserted actions because they were all delete-relaxed and thus could not possibly conflict with the causal links already present.

It is however easy to see that in those $n+1$ subproblems we also could have respected these links by simply disallowing delete-relaxed actions if their original, non-relaxed counterpart would violate the respective link. This effectively acts as a simple "filter" that can be performed in polynomial time. And it makes perfect sense to disallow these actions because we know that no such action could possibly be applied in the non-relaxed plan – any heuristic estimate based on such actions makes a relaxation of which we know *for sure* that the estimate is wrong in the sense that it might return a finite estimate although the perfect heuristic based upon such an action is infinity.

We thus define a novel problem relaxation, where, despite delete-relaxation, causal links that are already present in the input plan are respected in the way that inserted delete-relaxed actions may still raise a causal threat if their original non-relaxed counterpart would raise a threat (with those links already present). Note that we define this constraint for the general case of partially ordered input plans.

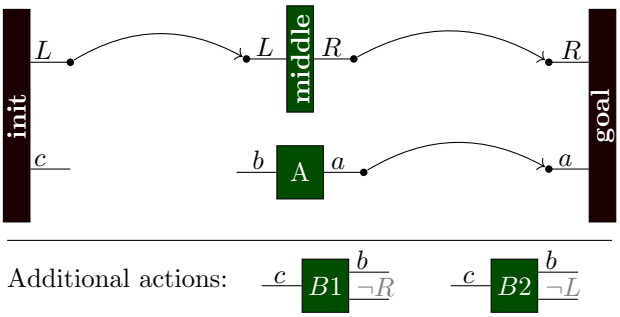


Figure 1: An example plan with initially given causal links. In addition to the actions that are already in the plan, the additional actions depicted below can be added as well. Note that their delete effects are grayed out because they are partially ignored (they are only considered to restrict the intervals in which they can be inserted).

Fig. 1 shows a simple example plan to illustrate this concept. This plan has one open precondition, b, of the plan step A. There are two further actions, $B1$ and $B2$, that are suitable to support that precondition. Although their delete-relaxed version will be inserted, their delete effects are still respected with regard to the causal links already present. That means that if $B1$ is inserted, it may only be added before the plan step *middle* because its delete effect R would raise a threat with the causal link protecting R. If we insert $B2$ instead, then we may only do so after the *middle* step, which in turn also restricts the ordering constraints involving A (since A requires the effect of $B2$ and must thus be ordered behind it). So the question might arise whether delete relaxation even makes the problem easier in this case. To see this, consider the altered example in which the precondition c of $B1$ and $B2$ is not already true in the initial state, but must be achieved first by some additional series of actions. For this, we can again rely on delete relaxation thus simplifying the problem (we only need to make sure again that these actions do not threaten causal links that are already present in the input plan).

Definition 5 (Delete-relaxed POCL Problem Respecting Causal Links)**.** *Let $\mathcal{P}$ be POCL planning problem with initial partial plan $P_{s_I,g} = (PS_I, \prec_I, CL_I)$. We denote by $\mathcal{P}^{A^+}_{links}$ and $\mathcal{P}^{A^+}_{P^+,links}$ the POCL planning problem with* delete-relaxed actions (and plan, respectively) respecting causal links. *They are both syntactically defined like $\mathcal{P}^{A^+}$ and $\mathcal{P}^{A^+}_{P^+}$, respectively, but with altered solution criteria. A partial plan $P = (PS, \prec, CL)$ is a solution to $\mathcal{P}^{A^+}_{links}$ or $\mathcal{P}^{A^+}_{P^+,links}$, respectively, if and only if:*

- *$PS \supseteq PS_I$, $\prec \supseteq \prec_I$, and $CL \supseteq CL_I$. (Unchanged)*
- *All preconditions of P's plan steps are protected by a causal link and there are no causal threats. (Unchanged)*
- *Let $ps^{a^+} = (l, a^+) \in PS \setminus PS_I$ be a delete-relaxed plan step. Let $a \in A$ be the non-relaxed version of $a^+ \in A^+$*

and define $ps^a = (l, a)$. Then, there may not be a causal link $(ps, f, ps') \in CL_I$, such that ps^a would threaten it, i.e., PS and $\prec$ substituting ps^a for ps^{a^+} may not induce a causal threat (see footnote[5] for a formal definition). (New)

The first two solution criteria remained unchanged, and in addition we demand that exactly those causal links of the input plan are respected in the sense that even delete-relaxed actions may not threaten those initially given causal links (which can be achieved by additional ordering insertions). This means that some (delete-relaxed) actions may only be available in certain "positions" of the partial plan thus making heuristic estimates more informed since relaxed actions would be forbidden at positions where their original versions could not possibly be used.

By exploiting this stronger notion of causal links in the presence of delete relaxation, we can also generalize Prop. 2 by stating that it remains **NP**-complete to decide a POCL problem with non-delete-relaxed initial partial plan even if we respect causal links. Since one problem is a special case of the other, hardness follows directly. And since membership can be shown by guessing a total ordering, respecting the causal links is again just applying a series of filters, which can be done in polynomial time.

Corollary 2. *Let $\mathcal{P}^{A^+}_{links}$ be a POCL problem with delete-relaxed actions respecting causal links. Deciding whether $\mathcal{P}^{A^+}_{links}$ has a solution is* **NP***-complete.*

We can now state the corollary mentioned in the beginning of this subsection, which provides a tighter result than Thm. 3 (and Thm. 2) because it relies on a problem with fewer relaxations.

Corollary 3. *Let $\mathcal{P}^{A^+}_{links}$ and $\mathcal{P}^{A^+}_{P^+,links}$ be a totally ordered POCL problem with delete-relaxed actions (and plan, respectively) respecting causal links. Deciding whether $\mathcal{P}^{A^+}_{links}$ and $\mathcal{P}^{A^+}_{P^+,links}$ have a solution is in* **P**.

Whereas this result for totally ordered plans was easy to discover, it raises the question whether the problem remains in **P** or becomes **NP**-complete if we assume a *partially ordered* delete-relaxed input plan, but demand that its causal links are respected despite delete-relaxation.

We can see from the example illustrated in Fig. 1 that there is no obvious greedy strategy anymore on how to achieve open conditions. The reason is that the partial order of the existing plan steps dictates over which intervals their causal links span, which influences which delete-relaxed actions can be inserted at specific locations. For example, if we move A before *middle*, its causal link protecting a becomes active earlier and thus poses more constraints. But depending on which actions we add, we might *have to* move existing steps like A thus changing where specific causal links become active. This observation is exploited by our proof, which shows that respecting existing causal links in a partial-order setting is **NP**-complete.

Theorem 4. *Let $\mathcal{P}^{A^+}_{P^+,links}$ and $\mathcal{P}^{A^+}_{links}$ be a POCL planning problem with delete-relaxed actions (and plan, respectively) respecting causal links. Deciding whether $\mathcal{P}^{A^+}_{P^+,links}$ and $\mathcal{P}^{A^+}_{links}$ have a solution is* **NP***-complete.*

Proof. Membership: We can adapt the procedure for Thm. 2. That procedure assumes a totally ordered plan, which we can guess in a first step (and check whether it contradicts existing causal links). There are just two steps we need to adapt. One is the application of the steps in the plan, which now might also delete fluents, whereas Thm. 2 assumes only delete-relaxed steps. Furthermore we need to make sure that in each interval between two plan steps of the initial plan no delete-relaxed actions are inserted that have a non-relaxed counterpart conflicting with any of the links spanning over the current interval.

Hardness: We show hardness for $\mathcal{P}^{A^+}_{P^+,links}$, since $\mathcal{P}^{A^+}_{links}$ will follow directly. We reduce from CNF-SAT, i.e., the decision problem whether a formula in conjunctive normal form has a satisfying valuation, which is **NP**-complete (Hopcroft and Ullman 1979). Let $C = \{C_1, \dots, C_m\}$ be a finite set of clauses, where for each $1 \leq i \leq m$, $C_i = \{l^i_1, l^i_2, \dots\}$ represents a conjunct (i.e., disjunction) consisting of a finite set of literals. The literals are based on the finite set of variables $X = \{x_1, \dots, x_n\}$, where each literal l is of the form x or $\neg x$ for some $x \in X$.

We construct the initial plan depicted in Fig. 2. As required by the problem definition, this plan contains only delete-relaxed steps.

The initial state and goal description are given by:

$$\textit{init}\colon (\emptyset, \bigcup_{1\leq i\leq n} \{x_i{=}\top{-}L, x_i{=}\bot{-}L\}, \emptyset)$$

$$\textit{goal}\colon (\bigcup_{1\leq i\leq n} \{x_i{=}\top{-}R, x_i{=}\bot{-}R\} \cup \bigcup_{1\leq i\leq m} \{C_i\}, \emptyset, \emptyset)$$

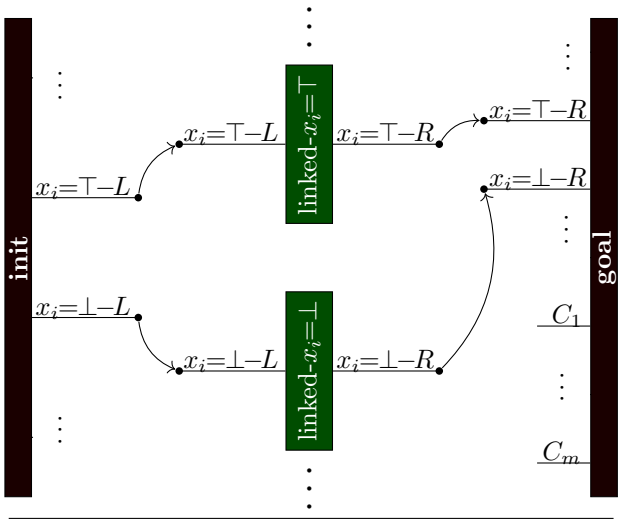
The C_i preconditions encode that every clause will be made true. The purpose of the other fluents will be explained later.

We introduce two plan steps and two further actions for each variable $x_i \in X$. We call the plan steps *linked-*$x_i{=}\top$ and *linked-*$x_i{=}\bot$ to differentiate them from the additional actions, which will be called $x_i{=}\top$ and $x_i{=}\bot$. The plan steps' actions are defined as follows:

$$\textit{linked-}x_i{=}\top\colon (\{x_i{=}\top{-}L\}, \{x_i{=}\top{-}R\}, \emptyset)$$
$$\textit{linked-}x_i{=}\bot\colon (\{x_i{=}\bot{-}L\}, \{x_i{=}\bot{-}R\}, \emptyset)$$

We refer to the plan steps of these actions by $ps_{i\top}$ and $ps_{i\bot}$, respectively. The preconditions and effects of these steps serve as consumer and producer of causal links with *init* and *goal*, respectively, thus spanning over the entire plan (hence their name *linked-*$x_i{=}\top$ and *linked-*$x_i{=}\bot$). Their

[5] Formally, if $P_{s_I,g} = (PS_I, \prec_I, CL_I)$ is the input plan, $P = (PS, \prec, CL)$ the solution candidate, $(ps_f, f, ps'_f) \in CL_I$, and $ps^{a^+} = (l, a^+) \in PS \setminus PS_I$ with $f \in del(a)$, let $PS'' = (PS \setminus \{(l, a^+)\}) \cup \{(l, a)\}$ and $\prec'' = (\prec \setminus \{(ps, (l, a^+)), ((l, a^+), ps) \mid ps \in PS\}) \cup \{(ps, (l, a)) \mid (ps, (l, a^+)) \in \prec\} \cup \{((l, a), ps) \mid ((l, a^+), ps) \in \prec\}$. Then, $(\prec'' \cup \{(ps_f, (l, a)), ((l, a), ps'_f)\})^+$ is a strict partial order.



Additional actions:

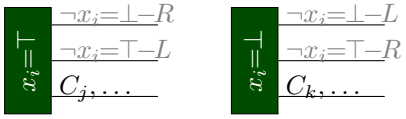


Figure 2: Constructed plan for the proof of Thm. 4. The delete-relaxed actions that may be inserted are depicted below the plan. Note that these steps only *depict* delete effects (in gray), but technically they are only available in the original planning problem, not after inserting them. They are, however, used to ensure that they do not violate any causal link of the initial plan. The effects $C_j, \ldots$ and $C_k, \ldots$ of these actions are derived from the clauses that become true by setting the respective variable accordingly.

causal links are given by:

$$\begin{aligned} CL_I : \bigcup_{1 \le i \le n} & \{(\mathit{init}, x_i{=}\top{-}L, ps_{i\top})\} \cup \\ & \{(\mathit{init}, x_i{=}\bot{-}L, ps_{i\bot})\} \cup \\ & \{(ps_{i\top}, x_i{=}\top{-}R, \mathit{goal})\} \cup \\ & \{(ps_{i\bot}, x_i{=}\bot{-}R, \mathit{goal})\} \end{aligned}$$

We now define the set of actions that can achieve the goal fluents that are not yet protected by a causal link. For each $x_i \in X$ we add two actions (only to the action portfolio, not to the plan), each representing setting x_i to either true or false. Each of these actions makes all clause fluents $C_i \in pre(\mathit{goal})$ true that correspond to a clause $C_i \in C$ that becomes true by setting x_i accordingly:

$$x_i{=}\top : (\emptyset, \{C_j \mid x_i \in C_j\}, \{x_i{=}\bot{-}R, x_i{=}\top{-}L\})$$
$$x_i{=}\bot : (\emptyset, \{C_j \mid \neg x_i \in C_j\}, \{x_i{=}\bot{-}L, x_i{=}\top{-}R\})$$

This finishes our construction. We claim: *The constructed POCL problem has a solution if and only if the given SAT formula has a satisfying valuation.*

"⇒": Assume $P = (PS, \prec, CL)$ is a solution plan. Due to the goal precondition(s) $\{C_1, \ldots, C_m\}$, we know that for each $C_j, 1 \le i \le m$, at least one action with the corresponding effect (i.e., $x_i{=}\top$ or $x_i{=}\bot$) is in the plan. If such a step represents setting a variable x_i to true, $x_i{=}\top$ must be in the plan. Because the delete effects of its original version are $x_i{=}\top{-}L$ and $x_i{=}\bot{-}R$, it will have to be ordered behind $ps_{i\top}$ as well as before $ps_{i\bot}$ thus enforcing the total order $(ps_{i\top}, ps_{i\bot}) \in \prec$. Note that the (delete-relaxed variant of the) other action, $x_i{=}\bot$, can not be in the plan, as otherwise it would raise a threat: It has delete effects $x_i{=}\bot{-}L$ and $x_i{=}\top{-}R$, which violates a causal link no matter where it resides. The case if x_i is set false works analogously with the inverted order of $ps_{i\bot}$ and $ps_{i\top}$. We can thus construct a satisfying variable assignment from the respective orderings, where the left action of each pair $(ps_{i\bot}, ps_{i\top}) \in \prec$ or $(ps_{i\top}, ps_{i\bot}) \in \prec$ corresponds to the assignment.

"⇐": Let's assume we have a satisfying variable assignment. We design a solution $P = (PS, \prec, CL)$ based on such a truth assignment. Let $x_i \in X$ and assume it's set true in the assignment (the case for setting x_i false works analogously). Choose the ordering $(ps_{i\top}, ps_{i\bot}) \in \prec$ and insert $x_i{=}\top$ between these steps (and set its causal links) so that there are no causal threats raised by the third (new) criterion of Def. 5. $x_i{=}\top$ will satisfy all goal fluents C_j, such that clause C_j is made true by x_i. Since all clauses can be made true by assumption, we can repeat this process for all $x_i \in X$ to make the plan executable. □

We might also be interested in the special case arising from disallowing action insertion, i.e., asking how hard it is to decide whether a linearization for a delete-relaxed input plan becomes when we allow for our notion of causal links. This is clearly polynomial because the causal links already present cannot raise any causal threats (causal threats might only be raised by plan steps that got *inserted*).

Proposition 3. *Let $\mathcal{P}^{linearize}_{P^+,links}$ denote a POCL planning problem with delete-relaxed plan respecting causal links, where we forbid action insertion. Deciding whether $\mathcal{P}^{linearize}_{P^+,links}$ has a solution is in* **P**.

The two major **NP**-completeness results, i.e., Prop. 2 with Cor. 2 and Thm. 4 are somehow disappointing as they show that we can only decide the plan existence problem for delete relaxation in polynomial time (assuming **P**≠**NP**) if we delete-relax *all* actions, including those in the initial plan, and furthermore ignore the pruning power of existing causal links. On the other hand this motivates the relaxations done in previous POCL heuristics (Younes and Simmons 2003; Nguyen and Kambhampati 2001), which can be computed in **P**, so now we know that these relaxations were indeed necessary to obtain a delete-relaxation POCL heuristic that runs in polynomial time.

Discussion and Conclusion

We conducted a comprehensive complexity analysis for the POCL plan existence problem. Apart from studying the general case, which is **PSPACE**-complete, we studied the influence of delete-relaxation by analyzing plan existence for delete-relaxing the current plan, the actions that can be inserted, or both. We furthermore proposed a weaker notion

of delete-relaxation where delete-effects are ignored except regarding causal links that already exist in the given partial plan – since they encode the current progress of the search leading to the current search node. Our investigations reveal that despite delete relaxation for the action portfolio, no matter whether we allow delete effects in the current partial plan or even relax them as well and only respect causal links (despite doing delete-relaxation), we are still $\mathbf{NP}$-complete. Only if, on top of doing delete-relaxation for the action portfolio, we ignore both – delete effects in the given partial plan and its causal links – we end up being in $\mathbf{P}$.

However, our investigations also point towards promising future work regarding heuristic development for POCL planning. We have shown that for totally ordered input plans, delete-relaxed POCL problems can be decided in $\mathbf{P}$, even if delete-relaxed actions have to respect existing causal links (Thm. 3). This result might seem insignificant at first glance, given that POCL planning is inherently about *partial orders*. However, every single solution has always at least one chain of plan steps connecting the goal description step to the initial state step with a chain of causal links thus inducing a total order. Heuristics could thus solve the intermediate problems with tighter action sets (similar to our proof of Thm. 3) rather than considering the entire problem at once thus performing severe problem relaxations and therefore having much less informed heuristics.

References

Aghighi, M.; and Bäckström, C. 2017. Plan Reordering and Parallel Execution — A Parameterized Complexity View. In *Proc. of AAAI*, 3540–3546. AAAI Press.

Bechon, P.; Barbier, M.; Infantes, G.; Lesire, C.; and Vidal, V. 2014. HiPOP: Hierarchical Partial-Order Planning. In *Proc. of the 7th European Starting AI Researcher Symposium (STAIRS)*, 51–60. IOS Press.

Benedictis, R. D.; and Cesta, A. 2020. Lifted Heuristics for Timeline-based Planning. In *Proc. of ECAI*, 2330–2337. IOS Press.

Bercher, P.; Behnke, G.; Höller, D.; and Biundo, S. 2017. An Admissible HTN Planning Heuristic. In *Proc. of IJCAI*, 480–488. IJCAI.

Bercher, P.; Geier, T.; and Biundo, S. 2013. Using State-Based Planning Heuristics for Partial-Order Causal-Link Planning. In *Proc. of the 36th German Conference on Artificial Intelligence (KI)*, 1–12. Springer.

Bercher, P.; Geier, T.; Richter, F.; and Biundo, S. 2013. On Delete Relaxation in Partial-Order Causal-Link Planning. In *Proc. of ICTAI*, 674–681. IEEE Computer Society.

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 *Proc. of ECAI*, 225–233. IOS Press.

Bercher, P.; Keen, S.; and Biundo, S. 2014. Hybrid Planning Heuristics Based on Task Decomposition Graphs. In *Proc. of SoCS*, 35–43. AAAI Press.

Bercher, P.; and Olz, C. 2020. POP $\equiv$ POCL, right? Complexity Results for Partial Order (Causal Link) Makespan Minimization. In *Proc. of AAAI*, 9785–9793. AAAI Press.

Bit-Monnot, A.; Ghallab, M.; Ingrand, F.; and Smith, D. E. 2020. FAPE: a Constraint-based Planner for Generative and Hierarchical Temporal Planning. Technical report, arXiv.org. ID: 2010.13121.

Bit-Monnot, A.; Smith, D. E.; and Do, M. 2016. Delete-Free Reachability Analysis for Temporal and Hierarchical Planning. In *Proc. of ECAI*, 1698–1699. IOS Press.

Bylander, T. 1994. The Computational Complexity of Propositional STRIPS Planning. *AIJ* 94(1-2): 165–204.

Bylander, T. 1997. A Linear Programming Heuristic for 0ptimal Planning. In *Proc. of AAAI*, 694–699. AAAI Press.

Bäckström, C. 1998. Computational Aspects of Reordering Plans. *JAIR* 9: 99–137.

Coles, A.; Coles, A.; Fox, M.; and Long, D. 2010. Forward-Chaining Partial-Order Planning. In *Proc. of ICAPS*, 42–49. AAAI Press.

Dvořák, F.; Barták, R.; Bit-Monnot, A.; Ingrand, F.; and Ghallab, M. 2014. Planning and Acting with Temporal and Hierarchical Decomposition Models. In *Proc. of ICTAI*, 115–121. IEEE.

Erol, K.; Hendler, J. A.; and Nau, D. S. 1996. Complexity results for HTN planning. *Annals of Mathematics and Artificial Intelligence (AMAI)* 18(1): 69–93.

Hoffmann, J.; and Nebel, B. 2001. The FF Planning System: Fast Plan Generation Through Heuristic Search. *JAIR* 14: 253–302.

Höller, D.; Bercher, P.; Behnke, G.; and Biundo, S. 2018. A Generic Method to Guide HTN Progression Search with Classical Heuristics. In *Proc. of ICAPS*, 114–122. AAAI Press.

Höller, D.; Bercher, P.; Behnke, G.; and Biundo, S. 2020. HTN Planning as Heuristic Progression Search. *JAIR* 67: 835–880.

Hopcroft, J. E.; and Ullman, J. D. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

Kambhampati, S. 1994. A Unified Framework for Explanation-Based Generalization of Partially Ordered and Partially Instantiated Plans. *AIJ* 67(1): 29–70.

Kambhampati, S. 1997. Refinement Planning as a Unifying Framework for Plan Synthesis. *AI Magazine* 18(2): 67–98.

Kambhampati, S.; and Nau, D. S. 1996. On the nature and role of modal truth criteria in planning. *AIJ* 82(1): 129–155.

Lallement, R.; de Silva, L.; and Alami, R. 2018. HATP: Hierarchical Agent-based Task Planner. In *Proc. of AAMAS*, 1823–1825. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS).

Lesire, C.; and Albore, A. 2021. pyHiPOP – Hierarchical Partial-Order Planner. In *Proceedings of 10th International Planning Competition: Planner and Domain Abstracts – Hierarchical Task Network (HTN) Planning Track (IPC 2020)*.

Minton, S.; Bresina, J.; and Drummond, M. 1994. Total-Order and Partial-Order Planning: A Comparative Analysis. *JAIR* 2: 227–262.

Muise, C.; Beck, J. C.; and McIlraith, S. A. 2016. Optimal Partial-Order Plan Relaxation via MaxSAT. *JAIR* 57: 113–149.

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

Nguyen, X.; and Kambhampati, S. 2001. Reviving Partial Order Planning. In *Proc. of IJCAI*, 459–466. Morgan Kaufmann.

Olz, C.; and Bercher, P. 2019. Eliminating Redundant Actions in Partially Ordered Plans – A Complexity Analysis. In *Proc. of ICAPS*, 310–319. AAAI Press.

Savitch, W. J. 1970. Relationships between nondeterministic and deterministic tape complexities. *Journal of computer and system sciences* 4(2): 177–192.

Sebastiani, E.; Lallement, R.; Alami, R.; and Iocchi, L. 2017. Dealing with On-Line Human-Robot Negotiations in Hierarchical Agent-Based Task Planner. In *Proc. of ICAPS*, 549–557. AAAI Press.

Shekhar, S.; and Khemani, D. 2015. Extending and Tuning Heuristics for a Partial Order Causal Link Planner. In *Proc. of the 3rd International Conference on Mining Intelligence and Knowledge Exploration (MIKE 2015)*, 81–92. Springer.

Stock, S.; Mansouri, M.; Pecora, F.; and Hertzberg, J. 2015. Online Task Merging with a Hierarchical Hybrid Task Planner for Mobile Service Robots. In *Proc. of IROS*, 6459–6464. IEEE.

Tan, X.; and Gruninger, M. 2014. The Complexity of Partial-Order Plan Viability Problems. In *Proc. of ICAPS*, 307–313. AAAI Press.

Vidal, V.; and Geffner, H. 2004. CPT: An optimal Temporal POCL Planner based on Constraint Programming. Booklet of the IPC-4.

Vidal, V.; and Geffner, H. 2006. Branching and pruning: An optimal temporal POCL planner based on constraint programming. *AIJ* 170: 298–335.

Waters, M.; Nebel, B.; Padgham, L.; and Sardina, S. 2018. Plan Relaxation via Action Debinding and Deordering. In *Proc. of ICAPS*, 278–287. AAAI Press.

Waters, M.; Padgham, L.; and Sardina, S. 2020. Optimising Partial-Order Plans Via Action Reinstantiation. In *Proc. of IJCAI*, 4143–4151. IJCAI.

Weld, D. S. 1994. An Introduction to Least Commitment Planning. *AI Magazine* 15(4): 27–61.

Weld, D. S. 2011. Systematic Nonlinear Planning: A commentary. *AI Magazine* 32(1): 101–103.

Winer, D. R.; and Young, R. M. 2016. Discourse-Driven Narrative Generation with Bipartite Planning. In *Proc. of the Ninth International Natural Language Generation Conference (INLG)*, 11–20. ACL.

Younes, H. L. S.; and Simmons, R. G. 2002. On the Role of Ground Actions in Refinement Planning. In *Proc. of AIPS*, 54–61. AAAI Press.

Younes, H. L. S.; and Simmons, R. G. 2003. VHPOP: Versatile heuristic partial order planner. *JAIR* 20: 405–430.