2013 IEEE 25th International Conference on Tools with Artificial Intelligence

On Delete Relaxation in Partial-Order Causal-Link Planning

Pascal Bercher, Thomas Geier, Felix Richter, Susanne Biundo
Institute of Artificial Intelligence
Ulm University
Ulm, Germany
e-mail: *firstName.lastName*@uni-ulm.de

***Abstract*—We prove a new complexity result for Partial-Order Causal-Link (POCL) planning which shows the hardness of refining a search node (i.e., a partial plan) to a valid solution given a delete effect-free domain model. While the corresponding decision problem is known to be polynomial in state-based search (where search nodes are states), it turns out to be intractable in the POCL setting. Since both of the currently best-informed heuristics for POCL planning are based on delete relaxation, we hope that our result sheds some new light on the problem of designing heuristics for POCL planning.**
Based on this result, we developed a new variant of one of these heuristics which incorporates more information of the current partial plan. We evaluate our heuristic on several domains of the early International Planning Competitions and compare it with other POCL heuristics from the literature.

I. Introduction

Partial-Order Causal-Link (POCL) planning [1], [2] is a technique for solving classical planning problems via search in the space of plans. The currently most prominent approach for solving such problems is planning as search in the space of *states*, although POCL planning has several advantages compared to state-based planning: POCL planning follows a *least commitment principle*, in which only necessary decisions are performed like maintaining only a partial order on the plan steps in a partial plan and maintaining only a partial variable binding thereby avoiding unnecessary restrictions, which might turn out as wrong later in the search. Since solutions are also only partially ordered and all causal dependencies between plan steps are explicitly represented, POCL planning allows greater flexibility at plan execution time [3] and the explanation of the structure of the solution at hand [4].

Despite these advantages, POCL planning has become less attractive to many researchers, because planning systems based on this approach are currently not competitive with current state-of-the-art (state-based) planning systems in terms of runtime. We assume this can be attributed to two main reasons: first, there is a large number of highly informed heuristics for state-based search [5], and, second, there is a lot of theoretical work which helps designing new heuristics or improving existing ones [5], [6]. For example, Bylander's result showing that the plan existence problem given a state and a delete effect-free domain is polynomial [6] lead to the development of several tractable heuristics based on delete relaxation. The probably most prominent one is the FF heuristic as implemented in the Fast Forward (FF) planning system [7]. The success of these heuristically guided systems lead to a paradigm shift from several approaches like POCL planning and CSP-based approaches such as GraphPlan [8] to state-based planning.

We are only aware of two well-informed heuristics for domain-independent, non-temporal POCL planning: The *Relax heuristic* [9] and the *Additive heuristic for POCL planning* [10]. Both heuristics are based on the same idea of finding a solution using a delete-relaxed planning domain. In contrast, heuristics used in state-based search also rely on *critical paths*, *abstractions*, and *landmarks* [5]. We recently developed an approach which enables the use of state-based heuristics in POCL planning, but it did not yet result in a system competitive with state-based planners [11]. We believe that the small number of heuristics for POCL planning can be attributed to the fact that it seems to be much more complicated to estimate the goal distance for a partial plan than to estimate the goal distance for a state. To shed some light on that difficulty, we study the complexity of the plan existence problem for POCL planning, when delete relaxation is performed – analogously to the proposition of Bylander [6] for state-based planning. Based on our results, we improve the *Relax Heuristic* and present empirical results.

The remainder of the paper is structured as follows: Section II is devoted to the formalization of POCL planning. Section III shows our new complexity result. Section IV presents our new heuristic including an empirical evaluation and, finally, the last section concludes the paper.

II. Problem Formalization

Although POCL planning is ordinarily done in a lifted fashion [10], we base our formalization on a fully ground, propositional representation.

A domain model $\langle \mathcal{V}, \mathcal{A} \rangle$ consists of a finite set of propositional state variables $\mathcal{V}$ implicitly defining the induced state space $\mathcal{S} = 2^{\mathcal{V}}$ and a finite set of available actions $\mathcal{A}$. Each action $a = \langle pre, add, del \rangle \in \mathcal{A}$ is a tuple from the set $2^{\mathcal{V}} \times 2^{\mathcal{V}} \times 2^{\mathcal{V}}$ and defines the action's precondition pre, its add list add, and its delete list del. It is applicable in a state $s \in \mathcal{S}$ iff $pre \subseteq s$ and, if applicable in that state, generates the state $(s \setminus del) \cup add$. The applicability and application of action sequences is defined in the straight-forward way.

Partial plans are abstractions of totally ordered action sequences in the sense of actions being only partially ordered; furthermore, partial plans explicitly model effect/precondition relations between different actions by means of so-called

1082-3409/13 $31.00 © 2013 IEEE
DOI 10.1109/ICTAI.2013.105

causal links. More formally, a partial plan is a tuple $(PS, \prec, CL)$ with PS being a finite set of *plan steps*, $l{:}a \in PS$ consisting of an action $a \in \mathcal{A} \cup \{a_0, a_\infty\}$ and l being a unique label symbol to identify the correct plan step in case the partial plan contains multiple occurrences of a. The two special actions a_0 and a_∞ encode the initial state of the problem and its goal description, respectively. Thus, a_0 has an empty precondition, an empty delete list and the initial state as add effect, and the action a_∞ has an empty add and delete list, and the goal description as precondition. The corresponding plan steps $l_0{:}a_0$ and $l_\infty{:}a_\infty$ are called $init$ and $goal$, respectively. The ordering constraints are represented by the strict partial order $\prec$ defined on the plan steps of PS. Every partial plan satisfies $(0, \infty) \in \prec$ and, furthermore, every plan step in PS different from $init$ and $goal$ is ordered between these two plan steps. The set CL specifies the causal links. A causal link $l \xrightarrow{v} l'$ specifies that the precondition variable $v \in \mathcal{V}$ of the plan step $l'{:}a'$ is provided by the add list of plan step $l{:}a$.

A POCL planning problem is a tuple $\pi = \langle \mathcal{D}, P \rangle$ with $\mathcal{D}$ being a domain model and P being a partial plan. Typically, P contains only the two plan steps $init$ and $goal$ without causal links, since such a problem corresponds to an ordinary STRIPS planning problem [12], where only an initial state and a goal description is given in addition to the domain model.

A partial plan $P_{sol} = (PS_{sol}, \prec_{sol}, CL_{sol})$ is a solution to a POCL planning problem π, called *plan*, if and only if:

(1) P_{sol} is a refinement of P. That is, if $P = (PS, \prec, CL)$, then $PS \subseteq PS_{sol}$, $\prec \subseteq \prec_{sol}$, and $CL \subseteq CL_{sol}$.

(2) There are no *open preconditions*. That is, for every plan step $l{:}a \in PS_{sol}$, $a = (pre, add, del)$ with $v \in pre$, there is a causal link $l' \xrightarrow{v} l \in CL_{sol}$ with $l'{:}a' \in PS_{sol}$, $a' = (pre', add', del')$, and $v \in add'$.

(3) There are no *causal threats*. That is, if there is a causal link $l \xrightarrow{v} l'' \in CL_{sol}$, then for all plan steps $l'{:}a' \in PS_{sol}$ with $a' = (pre', add', del')$ and $v \in del'$ it holds that the set $\prec_{sol} \cup \{(l, l'), (l', l'')\}$ is no strict partial order.

Criterion (1) relates solutions to the problem specification, i.e., to the initial partial plan P. Criteria (2) and (3) ensure that the solution is *executable* in the sense that any action sequence induced by the ordering constraints is applicable in the initial state and generates a state satisfying the goal condition.

POCL planning procedures perform plan-based search, starting with the initial partial plan and refining it until a solution is generated [10]. To that end, the violation of criteria (2) and (3) is represented by so-called flaws; thus, a partial plan is a solution iff it does not show any flaws. In a first step, a most-promising partial plan is selected from a set of candidates. For that partial plan, first, all its flaws are identified, i.e., all its open preconditions and all causal threats. Then, one of these flaws is selected and resolved using all available possibilities (for instance, an open precondition flaw can be resolved by inserting a causal link rooted either in a plan step from the current partial plan or in a new plan step taken from the domain). All resulting plans are then added to the set of candidates and a new cycle starts over.

This procedure has two decision points: The selection of a most promising partial plan and the selection of a flaw. In this paper, we focus on the former by means of heuristics.

III. Complexity Results

POCL procedures pick a most-promising partial plan from a set of candidates based on some criteria; often, an informed search procedure like A* using a heuristic function is chosen. As noted in the introduction, several well-informed heuristics have been developed for state-based planning [5], but we are only aware of *two* heuristics for POCL planning: the *Relax heuristic* [9] and the *Additive heuristic for POCL planning* [10] (*Add*, for short). Both approximate the number of necessary actions to solve a delete-relaxed version of the planning problem. In state-based planning, ignoring delete lists is a promising idea for constructing heuristics, as the plan existence problem for a delete effect-free domain is known to be solvable in polynomial time [6]. However, in POCL planning, that problem has not yet been investigated in detail.

Let $\pi = \langle \mathcal{D}, P \rangle$ be a POCL planning problem with P being an arbitrary partial plan, i.e., possibly containing more plan steps than just $init$ and $goal$ and possibility containing causal links and ordering constraints. Such problems are generated during planning, as each search node is a new partial plan and hence induces a new planning problem.

In state-based planning, performing delete-relaxation is straight-forward, as simply all actions in the domain need to be relaxed (although there are more elaborated approaches, which ignore only some of the variables in the delete lists [13]). However, in the POCL setting, in addition to the actions in the domain, we have the actions/plan steps in the current partial plan and the question arises whether these should be relaxed as well. If not, the actions from the domain would still *all* be delete-free, but the plan steps in P are not. We can motivate leaving P unaltered by considering the analog question in state-based planning: there, search nodes are states; hence, every state *completely* reflects the entire information about the search path from the initial state up to the current one. This includes all applied actions leading to that state including their negative effects. In POCL planning, this "planning progress" is reflected in the current partial plan. Relaxing its actions would mean to ignore information about the current progress of the search. Thus, the question we would like to have answered is: "Given P, how hard is it to refine it to a solution given an easier *planning domain*?". In the following, we study the hardness of that problem where this "easier" domain is obtained by performing delete relaxation.

Unfortunately, it turns out that relaxing only the actions in the domain is $\mathbf{NP-complete}$, whereas the problem is in $\mathbf{P}$, if also the actions in P are delete-relaxed. We prove the first result formally, but omit a proof for the latter, since it is trivially solvable in polynomial time in the size of $|\mathcal{D}| + |P|$.

Let us first define our notion of delete relaxation. We call a POCL planning problem $\pi = \langle \mathcal{D}, P \rangle$ with $\mathcal{D} = \langle \mathcal{V}, \mathcal{A} \rangle$ *delete-free*, if and only if for each $(pre, add, del) \in \mathcal{A}$, $del = \emptyset$. We hence call a POCL planning problem π' *delete-relaxed* if it is obtained from a POCL planning problem π by ignoring the delete lists of the actions in $\mathcal{A}$ (but leaving P unaltered).

The decision problem for determining whether a partial plan has a solution using a delete-relaxed domain model is then given by $\mathbf{PLANSAT} := \{\pi | \pi$ is a delete-free POCL planning problem and has a solution$\}$.

Theorem 1. **PLANSAT** *is* **NP−complete**.

Proof: Membership. Fix an arbitrary delete-free POCL planning problem $\langle \mathcal{D}, P \rangle$ with domain $\mathcal{D} = \langle \mathcal{V}, \mathcal{A} \rangle$ and $P = (PS, \prec, CL)$. Guess a linearization of the plan steps in PS, which respects the ordering constraints of P. We need to show that it can be verified in polynomial time that such a sequence $init, l_1{:}a_1, \ldots, l_n{:}a_n, goal$ can be extended to an applicable action sequence using (the delete-free) actions from the domain. This is sufficient for membership, as a POCL solution can be obtained from such a sequence by inserting causal links, which can also be done in polynomial time.

First, we need to verify that the chosen linearization does not violate any causal links present in P. This is the case if and only if it does not have any causal threats, which can be verified in polynomial time. Note that the causal links may only be violated by the plan steps already present, but not by the additional actions from the domain, as these actions do not show delete lists and hence cannot cause new causal threats.

Afterwards, we build a saturated relaxed planning graph [8] starting from $init$. This graph can be built in polynomial time [7]. Furthermore, if the precondition of the plan step $l_1{:}a_1$ is contained in the last fact layer $L \subseteq \mathcal{V}$ of this planning graph, we have verified that we can find a sequence of actions to support the precondition of that plan step (or proved its non-existence, otherwise), since such a sequence can be extracted from the planning graph without backtracking due to the absence of negative effects [7]. Now, we apply that plan step by removing the delete list from L and adding its add list thereby generating a new state. From this state, we build another saturated planning graph to test whether the precondition of the plan step $l_2{:}a_2$ holds in its last fact layer. We repeat that procedure thereby building $|PS| - 1$ planning graphs, one between each tuple of two consecutive plan steps. If the precondition of $goal$ is contained in the last layer of the last planning graph, we have verified that the chosen linearization can be extended to an applicable action sequence containing the plan steps of P in an order compatible with $\prec$ and respecting its causal links.

Hardness. To prove the hardness, we adapt a proof by Nebel and Bäckström [14, Theorem 15], in which they proved the NP completeness of deciding whether there exists an applicable action sequence given a partial plan without causal links and without the capability of inserting actions from the domain; i.e., the problem studied was to find a suitable order of the plan steps. We show that this problem does not become easier when one is allowed to insert delete-relaxed actions, independently of whether causal links are present or not. From that observation follows hardness, since P can be *refined*[1] to a POCL solution if and only if there exists a linearization of the plan steps in P which can be extended to an applicable action sequence using actions from the domain.

The proof is done by reduction from CNF-SAT. Given a set of boolean variables $X = \{x^1, \ldots, x^n\}$ and a set of clauses $C = \{c^1, \ldots, c^m\}$, each clause c^j being a set of literals over X representing a disjunction, we construct a delete-free POCL planning problem, whose solutions are isomorphic to that of the CNF-SAT problem. The general idea is that the initial partial plan (depicted in Fig. 1) contains two plan steps/actions $A_i^\top$ and $A_i^\bot$ for each variable x^i in X. The order in which these plan steps occur in a solution encodes the truth assignment of x^i using the two state variables $x_i{=}\top$ and $x_i{=}\bot$.

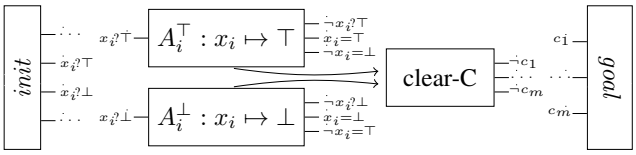

Fig. 1. The initial partial plan used by the hardness proof of Theorem 1.

For example, if $A_i^\top$ appears after $A_i^\bot$ in a solution plan, then $x_i{=}\top$ does hold at the end of any action sequence derived from that plan, and $x_i{=}\bot$ does not. Note that every pair of two actions $A_i^\top$ and $A_i^\bot$ *needs* to be ordered w.r.t. each other, since one plan step would *threaten* the other as soon as a causal link is set over $x_i{=}\top$ or $x_i{=}\bot$, respectively.

Furthermore, adding the delete-relaxed variants of these actions does not change which of these variables finally holds for the following reasons: First, the delete-relaxed variant of $A_i^\top$, for example, cannot be applicable *after* $A_i^\top$ due its precondition and delete effect $x_i?\top$, which encodes whether this action has already been executed. Second, inserting it *before* $A_i^\top$ does not change which of the variables $x_i{=}\top$ and $x_i{=}\bot$ finally holds, because only the last applied *non-relaxed* action determines the final outcome.

We still need to "use" these variables in order to determine whether the resulting variable assignment satisfies the SAT formula. To that end, the domain model contains one action for each literal in any clause which serves the purpose of setting a clause to true if one of its literals is true. Thus, if $x^i \in c^j$ and $c^j \in C$, then $C_{ij}^\top := (\{x_i{=}\top\}, \{c_j\}, \emptyset) \in \mathcal{A}$. The action $C_{ij}^\bot$ is defined analogously for $\neg x^i \in c^j$, $c^j \in C$. Obviously, these actions can be used to support the preconditions of the $goal$ plan step (and thus solve the SAT formula) if and only if a satisfying variable assignment was chosen, i.e., if a correct order of the actions $A_i^\top$ and $A_i^\bot$ was found. However, since these actions may be inserted at an arbitrary position in the partial plan, and in every solution $x_i{=}\top$ and $x_i{=}\bot$ (which are the preconditions of $C_{ij}^\top$ and $C_{ij}^\bot$, respectively) are both true at some point, we need to ensure that the goal's preconditions are only supported by those actions $C_{ij}^\bot$ and $C_{ij}^\top$, which were applied after the very last action from $A := \{A_i^\top, A_i^\bot | i \in \{1, \ldots, n\}\}$. We ensure this by means of the action clear-C, which is ordered after the ones from A and deletes all state variables c_j. Since the actions $C_{ij}^\top$ and $C_{ij}^\bot$ are already delete-relaxed and the delete relaxation of clear-C is a no-op, there are no further cases to consider.

See Appendix A for the formal problem specification. ■

The main result of our theorem is that performing delete-relaxation only for the domain's actions is not sufficient to obtain a tractable problem class. Looking closer to the proof reveals that the complexity lies in finding the correct order of the plan steps. The possibility to insert delete-free actions into a partial plan does not make this problem easier.

[1]Please note that this question is different from verifying that P *already is* a valid solution, which can be done in polynomial time using the POCL solution criteria [14, Theorem 14].

IV. SAMPLE-FF

In this section, we introduce *Sample-FF*, a heuristic greatly inspired by the constructive proof of the **NP** membership proof of Theorem 1 presented in the last section. *Sample-FF* is based on the same ideas as the *Relax heuristic* [9], which is an adaptation of the FF heuristic [7] for state-based planning. However, *Sample-FF* incorporates more information about the currently considered partial plan. In particular, it does not ignore the negative effects of its plan steps and is able to use the constraints implied by the causal links. Before we explain how *Sample-FF* works in detail, we briefly review the *Relax heuristic* to pinpoint the differences.

Given a partial plan P, the *Relax heuristic* estimates the distance from the initial state (i.e., the postcondition of $init$) to an "artificial" goal description, which is obtained by building the union of all open preconditions of the plan steps of P. This goal distance is obtained in the same way the FF heuristic estimates the distance from a state to the goal description: it solves a delete-relaxed version of the problem and uses the number of actions of that solution as heuristic estimate[2]. Thus, the *Relax heuristic* can be calculated very efficiently, since the problems solved are in **P**. However, while tractable on the one hand, the performed relaxation is quite severe on the other: The heuristic uses only the set of open preconditions of the plan steps in P, thereby ignoring their quantity, their negative effects, and the constraints posed by the causal links.

However, although delete-relaxed planning with a non-relaxed partial plan is $\mathbf{NP-complete}$, there is no need to ignore the negative effects and the causal links altogether. In particular the causal links are of interest, since even a single wrongly placed causal link can prevent that the respective partial plan may be refined to a valid solution. Looking at the **NP** membership part of the proof of Theorem 1 though, it becomes clear that the "hard" part of extracting a solution is only guessing a suitable linearization. Given such a linearization, the rest of the proof can be directly translated into a deterministic *polytime* program consisting of the following three phases.

First, we generate some linearizations of the partial plan at hand. This is done using sampling and simulates the "guessing" part of the **NP** membership proof.

Second, we estimate the cost of completing the linearizations sampled in the first phase into relaxed solution plans. We look at each linearization separately and for each create a sequence of relaxed planning graphs that reflect the argumentation in the membership part of the proof of Theorem 1. The number of required additional relaxed actions inserted in this process yields a heuristic estimate for each linearization.

Last, we derive a heuristic estimate for a partial plan by combining heuristic estimates for its linearizations, i.e., by taking the minimum of the computed linearization estimates. We will next take a closer look at the individual phases and some possible optimizations of the basic idea.

[2]More precisely, only a subset of these actions is used: If an action in the delete-relaxed solution corresponds to a (non-relaxed) plan step in the partial plan, it does not count towards the cost estimate.

A. Sampling Linearizations

Since it is infeasible to consider *all* linearizations of a given partial plan, we need to limit the number of considered linearizations to receive a practical heuristic, e.g., by a constant. We also want to avoid choosing linearizations that are too similar to each other to avoid introducing too much bias into the heuristic. Therefore, we use a Markov Chain Monte Carlo approach for approximately *uniformly* sampling a constant number of linearizations [15].

We define a *linearization graph* whose nodes are linearizations consistent with the partial order. There is an edge between two nodes when their corresponding linearizations can be converted into each other by swapping two adjacent plan steps. Performing a random walk in the linearization graph thus corresponds to a sequence of swappings thereby generating a new linearization consistent with the ordering constraints. Let $d(z)$ be the degree of the node representing a linearization z of the plan steps PS in the linearization graph. Thus, it holds $d(z) \leq |PS|-1$. We define the probability of proceeding from a linearization z to its neighbor z' as follows:

$$p(z,z') = \begin{cases} \frac{1}{2(|PS|-1)} & \text{if } z \text{ and } z' \text{ are adjacent,} \\ 1 - \frac{d(z)}{2(|PS|-1)} & \text{if } z = z', \\ 0 & \text{otherwise.} \end{cases}$$

In other words, the fewer neighbors a linearization has (i.e., the lower the number of swaps consistent with the underlying partial order), the more likely staying at the current linearization becomes. It can be shown that a uniform distribution over all linearizations is reached after polynomially many random walks. By choosing an arbitrary consistent initial linearization and doing enough random walks, we can thus uniformly sample from the set of possible linearizations.

B. Estimating the Cost for Linearizations

Let $z = init, l_1{:}a_1, \ldots, l_n{:}a_n, goal$ be a sampled linearization of a partial plan which is not yet a solution. This linearization represents a totally ordered partial plan with "gaps" where some steps are still missing. Filling the gaps with appropriate non-relaxed plan steps will create a solution plan. Therefore, the number of relaxed plan steps required for filling the gaps can serve as a heuristic estimate for that linearization. We begin by building a saturated planning graph starting at the postcondition of $init$. We then apply $l_1{:}a_1$ in its last fact layer, yielding a "state" s_1. If $l_1{:}a_1$ cannot be applied in the last fact layer, the linearization can be discarded, since it cannot be completed into a solution. Otherwise a new saturated planning graph is constructed starting in s_1, in whose last layer $l_2{:}a_2$ is applied, yielding s_2, and so on. The last such planning graph is built after applying $l_n{:}a_n$. When $goal$ is applicable in the last fact layer of the last planning graph, we know that a relaxed solution exists and we can proceed with extracting it.

For this, we traverse the constructed planning graphs last to first. For the last graph, this amounts to doing standard relaxed solution extraction in the same fashion of *Relax*. Then, the non-relaxed plan step $l_n{:}a_n$ is applied in reverse, the second-last graph is considered, and so on until the postcondition of $init$ is reached. The heuristic value for z is then defined as the total number of relaxed plan steps required in all constructed planning graphs.

C. Combining Estimates for Linearizations

The last phase is conceptually simple: take the minimum of all estimates for the sampled linearizations.

An important corner case, however, is the situation where none of the sampled linearizations can be completed to a relaxed solution, because then the minimum of all estimates does not represent a finite heuristic value. For other heuristics, an infinite heuristic value does not pose a problem: It usually means that a partial plan cannot be completed to a relaxed solution, let alone a real solution, and can therefore be safely discarded. In our setting, however, we cannot be sure of this. It might just be coincidence that none of the sampled linearizations could be completed to a relaxed solution, and that the right linearization was missed in the sampling phase. The question which value should be returned in this case is hence not trivial. As pointed out, returning a high value might be too pessimistic and, contrarily, returning zero might be too optimistic; in fact, the partial plan might even be doomed to become invalid, but given the non-exhaustive number of samples, the heuristic was not able to prove that. Hence, we choose a compromise and return the number of open preconditions as estimate. Alternatively, we could return the value of the *Relax heuristic* instead, but we did not evaluate that variant.

D. Optimizations.

In the following, we present a few directions in which the basic algorithm can be improved.

a) Enumerating all linearizations: In cases where the number of linearizations is small, enumerating all of them is desirable, as it yields more accurate heuristic values than sampling. Additionally, this allows for safe pruning, as the corner case described before cannot occur: A partial plan *can* be discarded when all possible linearizations are proved unsolvable in the relaxed setting, i.e., completeness is guaranteed.

We therefore want an estimate on the number of linearizations a partial plan has and use it to decide whether it is deemed feasible to look at all linearization for it. Unfortunately, determining the exact number of linearizations is $\#\mathbf{P}-\mathbf{complete}$ [15], i.e., hard in the sense that there is no known method substantially better than enumerating all possible linearizations. Hence, we take the direct approach and start enumerating linearizations until we have reached a predefined maximum number of linearizations. If we have not enumerated all linearizations at this point, we switch to sampling, throwing away the linearizations enumerated thus far. Experiments performed in a pre-evaluation indicate that the benefits in precision and pruning power outweigh the effort wasted for generating unused linearizations.

The impact of that optimization heavily depends on the chosen flaw selection function. For example, always preferring "old" flaws, i.e., flaws detected early in the given partial plan, will produce partial plans with a high number of linearizations, since new plan steps are inserted level-wise starting from $goal$, as the oldest flaws in the initial partial plan are the open precondition flaws of $goal$ (given that plan contains only the representatives of the initial state and goal description, but no other initial plan steps). If the converse strategy is applied, i.e., if always a flaw is preferred that was detected last, the number of linearizations stays constantly 1 until a sequence of actions has been found, which supports at least one state variable of the goal description and roots in the initial state. Thus, up to this point, enumerating $n \geq 1$ linearizations will exhaustively enumerate all possible linearizations.

b) Precomputing a fixed point for the initial state: Since the forward phase of computing a heuristic value for a linearization always starts at the initial state, the fixed point reached by applying relaxed actions will always be the same. This first fixed point can thus be precomputed once and be reused each time the cost of a linearization is estimated. Note that it is also likely that the extracted solutions contain more relaxed steps before the first non-relaxed step than between two later non-relaxed steps, because typically only a few facts are deleted by applying a non-relaxed step. This makes precomputing a fixed point for the initial state an attractive idea for optimization.

c) Respecting causal links: We can also take a closer look at the relaxed solutions generated for a given linearization. It then becomes apparent that in many cases, these solutions contain relaxed plan steps at places where the POCL planning algorithm would not put non-relaxed plan steps due to the presence of causal links. To illustrate this, let P be a partial plan that contains a causal link $l \xrightarrow{v} l'$ between l and l' over variable v. Let furthermore a be an action whose delete list contains v. Adding a plan step $l''{:}a$ to P creates a causal threat if l'' can be ordered between l and l' and thus prevents P from being a solution. On the other hand, the relaxed version of a can of course be inserted between l and l' when a relaxed solution is constructed. When this happens, the heuristic value is a poor estimate of the true remaining search effort, as it is certain that the relaxed solution conflicts with every potential solution plan that can be generated from P.

We therefore modify the planning graph generation to respect causal links. Let $z = l_0{:}a_0, \ldots, l_{n+1}{:}a_{n+1}$ with $l_0{:}a_0 = init$ and $l_{n+1}{:}a_{n+1} = l_\infty{:}a_\infty = goal$ be a linearization of the plan steps of the partial plan $P = (PS, \prec, CL)$. We define the *active* causal links between l_i and l_{i+1} to be the set $\{l_n \xrightarrow{v} l_m \in CL \mid n \leq i \text{ and } m \geq i+1\}$, i.e., the causal links whose arcs cross an imaginary line drawn between l_i and l_{i+1} in a graphical representation of z. The set of active causal links contains exactly the causal links that can potentially cause a causal threat when an action is put between l_i and l_{i+1}. To be more precise, an action put between l_i and l_{i+1} will lead to a causal threat if its delete list contains a variable mentioned in an active causal link. Such actions are called *threatening*, and we modify relaxed planning graph generation to not use threatening actions. Identifying the active causal links and filtering out actions which are threatening them can obviously be done in polynomial time.

In summary, the improvement works by calculating the set of threating actions each time before the relaxed planning graph between two non-relaxed plan steps is built, and using only non-threatening actions for building the relaxed planning graph. Unfortunately, this optimization can not be used in conjunction with *precomputing* the first fixed point: The causal links that begin in $init$ can change between linearizations for different partial plans, and so can the computed fixed point if only non-threatening actions are used. Our implementation

can therefore incorporate the causal links rooting in $init$ independently of the remaining causal links, s.t. one can choose between the per-node runtime-improvement obtained by precalculation of the first saturated planning graph versus more informed heuristic values while still being able to incorporate the remaining causal links independently of that choice.

d) Reusing parent linearizations: In our early experiments (before we implemented the following optimization), we observed cases where linearizations with an estimated cost of zero were found for some partial plan visited during search, yet the planner was unable to generate a solution. This is an undesirable and strange situation, since such a linearization can easily be transformed into a POCL solution: Applying the actions of the zero-cost linearization starting in the initial state generates a state satisfying the goal description; that is, there is no need to insert any additional action, neither relaxed nor non-relaxed, only missing causal links and ordering constraints need to be inserted, which can be done very efficiently.

The problem of that situation is the randomized nature of the heuristic. Since in each node a fixed number of samples is generated independently of the linearizations obtained by its parent node, heuristic values may strongly vary between each two consecutive partial plans. To obtain more "stable" heuristic estimates, a partial plan P tries to reuse the best linearization of its parent, "best" being defined as the first linearization for which the smallest heuristic value was obtained. Whether such a parent linearization z can be reused depends on the last applied modification: In case of an insertion of an ordering constraint or a causal link, it must be tested whether z is compatible with the inserted ordering constraints. If it is, $n+1$ samples are used with n being the predefined fixed number of samples, otherwise just n (new) samples are considered. In case of an action insertion, the linearization z is extended to a linearization of size $|z|+1$ by inserting the new action at an arbitrary suitable position. Since z proved being successful for the parent node, we do not only create one reused linearization, but three – with randomly chosen positions for the new action. Thus, in general, each partial plan uses $n+m$ samples with n new sampled linearizations and $m \in \{0,\dots,3\}$ linearizations obtained from the best linearization of its parent node.

The described improvement "stabilizes" heuristic estimates, since good linearizations remain being used for heuristic estimation. Furthermore, it solves the problem concerning the zero-cost linearizations: Since the POCL algorithm is complete and zero-cost linearizations are applicable in the initial state and satisfy the goal condition (otherwise, it would not have cost zero), at least one modification m^* compatible with that linearization must exist for each remaining flaw. Thus, reusing that linearization to compute the heuristic value for the child plan created by applying m^*, the child plan will in turn have a heuristic value of zero; hence, after a partial plan with heuristic zero is found, a solution is obtained shortly afterwards. Note that this does not mean that the planner will return the linearization itself as a solution. Since only a sufficiently small number of causal links and ordering constraints is added in order to receive a solution, the least-commitment principle of POCL planning is preserved.

E. Evaluation

We implemented the proposed heuristic within our POCL planner, which is implemented in Java®. As search strategy, we used weighted A* using a weight of 2. That is, in each cycle, a partial plan p is selected with minimal f value, f given by $f(p) = g(p) + 2 * h(p)$, g being the unit cost of the partial plan and h being its heuristic estimate. In case two partial plans have the same f value, we break ties by preferring a partial plan with higher cost, thereby preferring smaller heuristic values. Remaining ties are broken using the $LIFO$ strategy thereby preferring newest partial plans. Concerning the flaw selection strategy, we always select a newest flaw, where all flaws detected in the same partial plan are regarded equally new/old. Among these flaws, we break ties by pursuing the *Least Cost Flaw Repair* selection [16], which prefers a flaw for which there are the least number of modifications. Remaining ties are broken by chance.

We compare the *Sample-FF heuristic* with the two currently best-informed heuristics for POCL planning: the *Relax heuristic* [9] and the *Additive heuristic for POCL planning* [10] (*Add*, for short). In addition to these heuristics from the literature, we implemented a new variant of the *Relax heuristic*. It only differs from the original version by a small detail: *All* actions of a relaxed solution count towards the heuristic estimate, whereas the original version ignores actions, which already exist in the given partial plan. This variant, called *Relax** dominates the original version while being "more inadmissible". Performance is measured in terms of size of the produced search space and number of solved problem instances based on several benchmarks taken from the early International Planning Competitions (IPCs).

The evaluated domains and problems are taken from the IPC 1 to IPC 5 (cf. Tab. I). For each domain, we used n consecutive problem instances, starting with the smallest ones. We omitted domains for which all configurations timed out on all problem instances. We used a time limit of 15 minutes CPU time and a memory limit of 2 GB. We run our experiments on a machine with two Intel Xeon® processors, each having 8 physical cores running at 2,6 GHz.

When comparing the performance of *Relax*, *Relax**, and *Sample-FF* with *Add* in terms of solved problems in total, we see that *Add* clearly dominates all other heuristics. This comes to our very surprise, as *Add* may heavily overestimate the optimal relaxed goal distance. In fact, *Relax* can be regarded as an *improvement* over *Add*, which avoids this overestimation.

As opposed to the other evaluated heuristics, *Sample-FF* has parameters which need to be specified. To achieve a polytime-bounded procedure, we fix the number of sampled linearizations to a predefined constant. We evaluated 1, 3, 10, and 30 samples per search node. The more samples are chosen, the more accurate the heuristic becomes. However, clearly, the calculation time becomes much more expensive as it scales linearly with the number of samples. The optimizations of trying to enumerate all linearizations and to reuse parent linearizations are always turned on, as we observed a clear improvement in terms of solved problem instances in a small pre-evaluation. Concerning respecting causal links, the heuristic features to respect no causal links at all, all causal links, or just the ones which are not rooting in $init$ (as was motivated in

TABLE I. THIS TABLE COMPARES THE DIFFERENTLY PARAMETRIZED VERSIONS OF *Sample-FF*. "*front:*" AND "*end:*" SPECIFY WHETHER CAUSAL LINKS ROOTING IN $init$ (OR *not* ROOTING IN $init$, RESPECTIVELY) WERE USED TO REDUCE THE SET OF APPLICABLE ACTIONS IN THE RESPECTIVE LAYERS OF THE SAMPLED LINEARIZATIONS. THE DOMAINS ARE ORDERED BY THE IPC, IN WHICH THEY WERE FIRST USED. THE NUMBER n SPECIFIES THE NUMBER OF USED PLANNING PROBLEMS IN THE RESPECTIVE DOMAIN. THE ENTRIES SPECIFY THE NUMBER OF SOLVED INSTANCES OF THE RESPECTIVE DOMAIN. BOLD ENTRIES SPECIFY THE CONFIGURATION WITH THE LARGEST NUMBER AMONG ALL CONFIGURATIONS OF *Sample-FF*.

Domain	n	**Add**	**Relax***	**Relax**	**Sample-FF** front: $\bot$ end: $\bot$				front: $\bot$ end: $\top$				front: $\top$ end: $\top$			
					1	3	10	30	1	3	10	30	1	3	10	30
grid	5	0	0	0	0	0	0	0	0	0	0	0	1	**1**	**1**	**1**
gripper	20	14	20	7	1	1	1	1	1	2	1	1	2	**3**	**3**	2
logistics	20	12	8	7	**8**	5	6	6	6	7	6	5	0	0	1	1
movie	30	30	30	30	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**
mystery	20	8	8	9	10	11	9	9	10	11	10	9	**12**	**12**	11	11
mystery-prime	20	3	3	3	3	4	**6**	5	5	4	4	4	**6**	**6**	**6**	**6**
blocks	21	4	5	7	5	5	**6**	**6**	4	3	3	3	5	3	2	0
logistics	28	28	28	27	22	23	23	**24**	21	19	20	21	15	13	14	15
miconic	100	100	49	39	**40**	**40**	37	35	39	41	37	32	15	16	18	20
depot	22	2	2	1	1	1	1	1	0	1	1	1	2	2	**3**	2
driverlog	20	7	9	7	**11**	9	10	9	9	10	9	8	8	7	9	7
rover	20	20	18	19	13	14	**15**	**15**	11	11	12	11	7	9	9	9
zeno-travel	10	4	5	3	3	**5**	4	**5**	4	3	4	4	1	1	1	1
airport	20	18	15	9	10	**11**	**11**	**11**	7	10	10	10	7	8	6	4
pipesworld-noTankage	10	8	1	2	2	3	**5**	3	2	1	2	1	1	4	3	**5**
pipesworld-Tankage	10	1	1	1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
satellite	20	16	7	7	**7**	5	6	6	5	5	7	5	1	2	3	3
pipesworld	10	1	1	1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
storage	20	7	6	4	6	7	9	8	6	7	7	6	9	9	**10**	**10**
tpp	20	19	11	11	**8**	7	6	7	6	6	6	6	5	6	6	6
total	446	292	227	194	182	183	187	183	168	173	171	159	127	132	136	133

the last section). Again, we obtain a trade-off between accuracy and calculation speed. While respecting all causal links is the most informed variant, it is clearly the slowest one, as it has to calculate the set of applicable actions between each two plan steps in a sampled linearization, as explained in the last section. We evaluated all possible combinations of the two parameters leading to 12 variants of *Sample-FF*.

Investigating the use of causal links, our experiments seem to suggest that the additional overhead incurred by respecting them does not pay off in total: We clearly see that the variant which does not respect causal links dominates the other two configurations of *Sample-FF*. While this is true taking into account the total number of solved problems, there are also domains where the configuration respecting all causal links solved more problems than *all other* heuristics. In one particular unsolvable problem instance, *all Sample-FF* configurations respecting all causal links were able to prove that problem to be unsolvable, whereas all other heuristics including *Relax*, *Relax** and *Add* incurred time-outs. Concerning the optimal number of samples, there is not a clear result, but we can observe that an optimal choice lies between 1 and 10.

Taking a look at the overall number of solved problems, the best configuration is the one which does not respect causal links and uses 10 samples. That configuration is clearly competitive with *Relax* in terms of the number of solved problems, as it solved 187 out of 446 problems, whereas *Relax* solved 194 problems. Investigating the number of domains in which one heuristic performed better than another, the data reveals that the configuration using no causal links and just a single sample is even better than *Relax*. In 6 out of 20 domains it solved strictly more problems than *Relax*, whereas *Relax* strictly dominated *Sample-FF* in only 5 domains. While it seems discouraging that using causal links seems not to pay off, we see a potential in further improving the heuristic: We evaluated the ratio of created partial plans to the ones for which no heuristic value could be obtained due to infeasible samples. As stated in the previous section, we return the number of open preconditions in these cases to prevent being blind. That is, a ratio of 100% would correspond to a heuristic which is *entirely* based on the number of open preconditions. We discovered that the mean ratio ranges from 4% to 1% (for increasing number of samples) for the configurations which do not respect causal links, from 7% to 3% for the configurations using only causal links not rooting in $init$, and from astonishing 46% to 36% for the configurations which respect all causal links. This clearly indicates the impact of the constraints posed by the causal links, as even for 30 samples, in 36% of all created nodes there does not exist a delete-relaxed solution. While this proves our assumption correct that causal links have a major impact on the set of valid solutions which can be derived from partial plans, we still need to find a way how to cope with these cases. Note that it might also be that in many of these cases the corresponding partial plans could actually have been pruned from the search space given *all* linearizations but we cannot decide this being the case in polynomial time.

Considering only the number of solved problem instances does not tell how well-informed the respective heuristics are. A larger number may also be attributed to the time a heuristic needs to be evaluated, as faster heuristics allow for a larger search space within the time limit. We hence investigated the number of solved problem instances given the number of generated search nodes. Heuristics which have a larger number of solved instances given the same number of generated search nodes can thus be regarded more accurate. Our data reveals that adding more samples improves heuristic accuracy and that the variant without respecting causal links is the most informed one among all *Sample-FF* configurations. However, the last result can be attributed to the large number of partial plans for which no heuristic value could be calculated. If we figure out how to solve this problem, the variant respecting causal links will probably improve its performance significantly.

V. Conclusion

In this paper, we made two contributions to the field of POCL planning: We proved that the plan existence problem given a search node in standard POCL planning (i.e., an arbitrary non-relaxed partial plan) and a delete-relaxed planning domain is $\mathbf{NP-complete}$. This is an interesting observation, since the corresponding decision problem in state-based planning is in $\mathbf{P}$. Based on the constructive proof of our main complexity result, we developed a new heuristic for POCL planning and presented empirical results.

The presented heuristic can still be improved. In particular, we want to solve the problem that for many plans no plan step sequence was found that could be extended to a relaxed solution. Also, we want to adapt our heuristic to lifted planning, s.t. it can evaluate partial plans which are not fully ground.

Appendix
Problem Formalization of Hardness Proof

Given a CNF-SAT problem (i.e., a SAT formula given in conjunctive normal form) with variables $X = \{x^1, \ldots, x^n\}$ and a set of clauses $C = \{c^1, \ldots, c^m\}$, we construct the delete-free POCL planning problem $\pi = \langle \mathcal{D}, P \rangle$ with $\mathcal{D} = \langle \mathcal{V}, \mathcal{A} \rangle$:

$$\begin{aligned}
\mathcal{V} &= \{x_i?\top, x_i?\bot, x_i{=}\top, x_i{=}\bot \mid i \in \{1,\ldots,n\}\} \cup \\
&\quad \{c_i \mid i \in \{1,\ldots,m\}\} \\
\mathcal{A} &= \{\textit{dr}\text{-}A_i^\top, \textit{dr}\text{-}A_i^\bot \mid i \in \{1,\ldots,n\}\} \cup \{\textit{dr}\text{-clear-C}\} \cup \\
&\quad \{C_{ij}^\top \mid c^j \in C, x^i \in c^j\} \cup \{C_{ij}^\bot \mid c^j \in C, \neg x^i \in c^j\} \\
P &= (PS, \prec, CL) \text{ and} \\
PS &= \{l_0{:}a_0, l_\infty{:}a_\infty, l_C{:}\text{clear-C}\} \cup \\
&\quad \{l_i^\top{:}A_i^\top, l_i^\bot{:}A_i^\bot \mid i \in \{1,\ldots,n\}\} \\
\prec &= \{(l_i^\top, l_C), (l_i^\bot, l_C) \mid i \in \{1,\ldots,n\}\} \\
CL &= \emptyset
\end{aligned}$$

The actions are given as follows:

$$\begin{aligned}
a_0 &= (\emptyset, \{x_i?\top, x_i?\bot \mid i \in \{1,\ldots,n\}\}, \emptyset) \\
a_\infty &= (\{c_i \mid i \in \{1,\ldots,m\}\}, \emptyset, \emptyset) \\
A_i^\top &= (\{x_i?\top\}, \{x_i{=}\top\}, \{x_i?\top, x_i{=}\bot\}) \\
\textit{dr}\text{-}A_i^\top &= (\{x_i?\top\}, \{x_i{=}\top\}, \emptyset) \\
A_i^\bot &= (\{x_i?\bot\}, \{x_i{=}\bot\}, \{x_i?\bot, x_i{=}\top\}) \\
\textit{dr}\text{-}A_i^\bot &= (\{x_i?\bot\}, \{x_i{=}\bot\}, \emptyset)
\end{aligned}$$

$$\begin{aligned}
\text{clear-C} &= (\emptyset, \emptyset, \{c_1, \ldots, c_m\}) \\
\textit{dr}\text{-clear-C} &= (\emptyset, \emptyset, \emptyset) \\
C_{ij}^\top &= (\{x_i{=}\top\}, \{c_j\}, \emptyset) \\
C_{ij}^\bot &= (\{x_i{=}\bot\}, \{c_j\}, \emptyset)
\end{aligned}$$

Acknowledgment

This work is done within the Transregional Collaborative Research Centre SFB/TRR 62 "Companion-Technology for Cognitive Technical Systems" funded by the German Research Foundation (DFG).

References

[1] D. McAllester and D. Rosenblitt, "Systematic nonlinear planning," in *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI 1991)*. AAAI Press, 1991, pp. 634–639.

[2] J. S. Penberthy and D. S. Weld, "UCPOP: A sound, complete, partial order planner for ADL," in *Proceedings of the third International Conference on Knowledge Representation and Reasoning*. Morgan Kaufmann, 1992, pp. 103–114.

[3] C. Muise, S. A. McIlraith, and J. C. Beck, "Monitoring the execution of partial-order plans via regression," in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*. AAAI Press, 2011, pp. 1975–1982.

[4] B. Seegebarth, F. Müller, B. Schattenberg, and S. Biundo, "Making hybrid plans more clear to human users – a formal approach for generating sound explanations," in *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS 2012)*. AAAI Press, 6 2012, pp. 225–233.

[5] M. Helmert and C. Domshlak, "Landmarks, critical paths and abstractions: What's the difference anyway?" in *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS 2009)*, vol. 9, 2009, pp. 162–169.

[6] T. Bylander, "The computational complexity of propositional STRIPS planning," *Artificial Intelligence*, vol. 94, no. 1-2, pp. 165–204, 1994.

[7] J. Hoffmann and B. Nebel, "The FF planning system: Fast plan generation through heuristic search," *Journal of Artificial Intelligence Research (JAIR)*, vol. 14, pp. 253–302, May 2001.

[8] A. L. Blum and M. L. Furst, "Fast planning through planning graph analysis," *Artificial Intelligence*, vol. 90, pp. 281–300, 1997.

[9] X. Nguyen and S. Kambhampati, "Reviving partial order planning," in *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI 2001)*. Morgan Kaufmann, 2001, pp. 459–466.

[10] H. L. S. Younes and R. G. Simmons, "VHPOP: Versatile heuristic partial order planner," *Journal of Artificial Intelligence Research (JAIR)*, vol. 20, pp. 405–430, 2003.

[11] P. Bercher, T. Geier, and S. Biundo, "Using state-based planning heuristics for partial-order causal-link planning," in *Advances in Artificial Intelligence, Proceedings of the 36nd German Conference on Artificial Intelligence (KI 2013)*, 2013, pp. 1–12.

[12] R. E. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, vol. 2, pp. 189–208, 1971.

[13] M. Katz, J. Hoffmann, and C. Domshlak, "Who said we need to relax all variables?" in *Proceedings of the 23d International Conference on Automated Planning and Scheduling (ICAPS 2013)*, 2013.

[14] B. Nebel and C. Bäckström, "On the computational complexity of temporal projection, planning, and plan validation," *Artificial Intelligence*, vol. 66, no. 1, pp. 125–160, 1994.

[15] G. Brightwell and P. Winkler, "Counting linear extensions," *Order*, vol. 8, no. 3, pp. 225–242, 1991. [Online]. Available: http://dx.doi.org/10.1007/BF00383444

[16] D. Joslin and M. E. Pollack, "Least-cost flaw repair: A plan refinement strategy for partial-order planning," in *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI 1994)*. AAAI Press, 1994, pp. 1004–1009.