POP ≡ POCL, Right? Complexity Results for Partial Order (Causal Link) Makespan Minimization

Pascal Bercher,* **Conny Olz**
Institute of Artificial Intelligence, Ulm University, Germany
pascal.bercher@anu.edu.au
conny.olz@uni-ulm.de

Abstract

We study PO and POCL plans with regard to their makespan – the execution time when allowing the parallel execution of causally independent actions. Partially ordered (PO) plans are often assumed to be equivalent to partial order causal link (POCL) plans, where the causal relationships between actions are explicitly represented via causal links. As a first contribution, we study the similarities and differences of PO and POCL plans, thereby clarifying a common misconception about their relationship: There are PO plans for which there does not exist a POCL plan with the same orderings. We prove that we can still always find a POCL plan with the same makespan in polynomial time. As another main result we prove that turning a PO or POCL plan into one with minimal makespan by only removing ordering constraints (called *deordering*) is *NP-complete*. We provide a series of further results on special cases and implications, such as *reordering*, where orderings can be changed arbitrarily.

Introduction

Although most of today's planning systems produce sequential (i.e., totally ordered) solutions, many applications also use partially ordered ones. Thus, there are many application scenarios that take such a sequential plan as an input and generate a partially ordered one (Aghighi and Bäckström 2017). Such a partial order, which represents an up to exponential set of sequential plans, can be exploited in many ways, for instance to obtain higher flexibility when it comes to plan execution (Muise, Beck, and McIlraith 2016; 2013; Muise, McIlraith, and Beck 2011). One of their main applications is *temporal planning* (Coles et al. 2010; Vidal and Geffner 2006), where the execution time of a plan needs to be optimized. The partial order is here a natural representation, as parallel execution of independent actions can be exploited. By doing so, one is usually interested in a plan's optimal *makespan* – the time needed to execute a partially ordered plan.

Optimizing sequential or partially ordered plans has thus been investigated by many researchers – both in theory by investigating the computational complexity of plan optimizations (Olz and Bercher 2019; Aghighi and Bäckström 2017; Nakhost and Müller 2010; Bäckström 1998; Fink and Yang 1992) and in practice by providing algorithms for optimizing a given plan (Muise, Beck, and McIlraith 2016; Say, Cire, and Beck 2016; Siddiqui and Haslum 2015; Chrpa, Kilani, and Balyo 2014; Nakhost and Müller 2010; Do and Kambhampati 2003; Fink and Yang 1992).

Many algorithms that produce partially ordered plans in the first place (Vidal and Geffner 2006; Younes and Simmons 2003; Penberthy and Weld 1992; McAllester and Rosenblitt 1991) as well as those that infer them from sequential plans (Siddiqui and Haslum 2015) rely on causal links, which explicitly represent causal dependencies between actions. As it turns out, plans with these links, so-called *POCL plans*, have slightly different properties than those without, which are called *PO plans*. Yet, it's often assumed that both kinds of plans are equivalent, i.e., that the solution criteria based on PO vs. POCL plans are interchangeable, which is not correct. We clarify this and explain why in some cases it might become problematic. For this, we (re-)make aware of literature and examples that illustrate these differences. As first technical contribution, we prove that these two kinds of plans can be transformed into each other in polynomial time without incurring undesired increases in the makespan. Our second line of technical contributions is the provision of several novel complexity results related to the PO and POCL makespan optimization problem. Most importantly, we show that optimizing a plan's makespan by *deordering* it, i.e, by removing ordering constraints and changing causal links, is *NP-complete*.

Formal Framework

We consider the problem of optimizing the makespan of a given (solution) plan. We choose STRIPS as the underlying formalism. Here, a planning problem $\mathcal{P}$ is given by a tuple (V, A, s_I, g), where V is a finite set of propositional state variables, $s_I \in 2^V$ is the initial state, and $g \subseteq V$ the goal description. A is a finite set of *actions*, each being a tuple $(prec, add, del)$ consisting of its *precondition*, *add*, and *delete list*, $prec, add, del \subseteq V$. For an action $a \in A$, we also write $prec(a)$, $add(a)$, and $del(a)$ to refer to these

*Pascal Bercher is now at the College of Engineering and Computer Science, the Australian National University

sets. Action execution is defined as usual, i.e., an action $a \in A$ is called executable in a state $s \in 2^V$ if and only if $prec(a) \subseteq s$. If a is executable in s, then the state transition function $\gamma : A \times 2^V \to 2^V$ returns the state resulting from executing a in s, $\gamma(a, s) = (s \setminus del(a)) \cup add(a)$. A sequence of actions $\bar{a} = (a_0 a_1 \dots a_n)$ is called executable to a state s_0 if there exists a state sequence $s_1 \dots s_{n+1}$ such that for all $1 \leq i \leq n+1$ holds $\gamma(a_{i-1}, s_{i-1}) = s_i$. The state s_{n+1} is called the *result* from executing said sequence. A *goal state* is a state s that makes all goals true, i.e., such that $s \supseteq g$.

The standard definition of a (solution) plan is simply such an action sequence leading to a goal state. For the sake of this paper – and in order to be able to speak about the makespan – we need to extend this definition to *partially ordered* plans, where ordering constraints are only given among *some* of the actions but not necessarily all (which is often possible, since normally not all actions depend among each other).

Partially Ordered Plans

There are essentially *two* standard definitions of (partial) partially ordered plans – one relies on causal links and the other does not. Please note that we will always use the term *partial plan* to refer to the plan data structure and *plan* to refer to such partial plans that are solutions.

The practically most widely used formalization comes from partial order causal link (POCL) planning, a plan space-based planning approach (Weld 1994; Penberthy and Weld 1992; McAllester and Rosenblitt 1991) for standard STRIPS problems. The underlying techniques are further exploited in makespan-optimal (constraint-based) planning (Vidal and Geffner 2006), in *state-based* satisficing search (Lipovetzky and Geffner 2011), and in optimal state-based search (Karpas and Domshlak 2012). There is also a vast range of *hierarchical* planning systems that rely on POCL techniques as well (Bercher et al. 2016).

A *partial POCL plan* is a tuple $P = (PS, \prec, CL)$, where PS is a finite set of *plan steps* $ps = (l, a)$ with l being a label unique in PS, $a \in A$ an action; $\prec$ is a partial order on PS; and CL is a finite set of causal links. We need to introduce plan steps, i.e., labeled actions, so that the ordering constraints can differentiate between multiple occurrences of the same action. As for actions, we use $prec(ps)$, $add(ps)$, and $del(ps)$ to refer to the precondition, add list, and delete list of a plan step ps of the respective action. Causal links are used to explicitly represent the causal dependencies between plan steps. Originally, they were introduced as a technical vehicle to document and verify the progress of plan-based search. Formally, the causal links are a set $CL \subseteq PS \times V \times PS$. A causal link $cl = (ps_p, v, ps_c) \in CL$ indicates that the precondition v of the *consumer* plan step ps_c is *supported* by the *producer* plan step ps_p (i.e., it also implies $v \in prec(ps_c) \cap add(ps_p)$). The variable v is also said to be *protected* by its causal link, because it raises a *causal threat (flaw)* in case the condition might be violated. Thus, threats need to be resolved to obtain a solution. Let a partial plan $P = (PS, \prec, CL)$ contain a causal link, $(ps_p, v, ps_c) \in CL$. Then, a causal threat is the situation where a plan step ps_t with $v \in del(ps_t)$ may be ordered between ps_p and ps_c, i.e., the ordering constraints may neither entail $ps_t \prec ps_p$ nor $ps_c \prec ps_t$. The step ps_t is called a *threatening* plan step. To ease definitions, we require that any causal link between two plan steps implies the respective ordering. So, w.l.o.g., we assume the existence of an ordering (ps_p, ps_c) in $\prec$ for every causal link $(ps_p, v, ps_c) \in CL$. The problem's initial state and goal description are encoded by two artificial actions. The first, called *init*, uses exactly the initial state as add effect. The second, called *goal*, uses exactly the goal description as precondition. For each partial plan *init* is ordered before all other steps and *goal* behind all others. We can now define the solution criteria based on causal links.

Definition 1. A partial POCL plan $P = (PS, \prec, CL)$ is called *POCL plan* (also *POCL solution*) to a planning problem if and only if every precondition is supported by a causal link and there are no causal threats.

The solution criteria for POCL plans can obviously be checked in lower polynomial time. It is also easy to see that these criteria are sufficient to imply that every linearization of its steps is a solution in the classical sense given above. These plans can hence be regarded a compact representation of an up to exponential number of sequential solutions. We can also define such solution sets without causal links, though.

We refer to a partial POCL plan $P = (PS, \prec, CL)$ *without* causal links, i.e., $CL = \emptyset$, as a *partial partially ordered (PO) plan*. For such a partial plan, we also write $P = (PS, \prec)$. Due to the absence of causal links, the solution criteria are here defined directly by the desired property of every linearization being executable:

Definition 2. A partial PO plan $P = (PS, \prec)$ is called *PO plan* (also *PO solution*) to a planning problem if and only if every linearization is executable in the initial state and results into a goal state.

Despite the absence of causal links, this solution criterion can still be verified in polynomial time as Chapman (1987) showed on p. 340[1]. Please note that this is *not* done by simply inferring causal links, because there exist plans which still possess causal threats but without violating the property of every linearization being executable (an example is given in Fig. 1 and will be discussed in the next section). In order to check for executability in the sense of Def. 2 we need to check Chapman's modal truth criterion, which ensures that in every possible linearization each precondition of each action holds in its predecessor state. Simplified to a ground setting, it states:

A proposition p is necessarily true in a [state][2] s iff two conditions hold: there is a [state] t equal or necessarily previous to s in which p is necessarily asserted; and for every step C possibly before s and every proposition [...] p which C denies, there is a step W necessarily between C and s which asserts [...] p [...]. (Chapman 1987, p. 340)

[1] Nebel and Bäckström (1994) essentially showed the same in Thm. 12, but based on event systems rather than STRIPS. They also discuss the relation of their results to that of Chapman (Sec. 5).

[2] Chapman (1987) used the term "situation" instead of "state", which is defined as a collection of literals.

Note that a plan step W that can re-introduce a once established and then deleted proposition (i.e., state variable) is called a *White Knight* (Chapman 1987, p. 340). For a further discussion of the modul truth criterion and related work we refer to the work by Kambhampati and Nau (1996).

For the sake of completeness, we would like to note that there also exist formalizations of PO plans that can be regarded hybrids between our formalization of POCL plans as defined in Def. 1 and PO plans as defined in Def. 2. Such formalizations are given by Haslum et al. (2019, Def. 2.11 and below) and Siddiqui and Haslum (2015, Def. 1). There, causal links are used, but the definition of a causal threat now relies on white knights. First, let us consider the following situation: There are two plan steps ps_p and ps_c, a causal link (ps_p, v, ps_c) between them, and a threatening step ps_t with $v \in del(ps_t)$ (that could be ordered between ps_p and ps_c). In standard POCL planning, such a situation is always referred to as a causal threat and the only possibilities to resolve it rely on promotion (i.e., ordering the threatening step before the producer, $ps_t \prec ps_p$) or on demotion (i.e., ordering the threatening step behind the consumer, $ps_c \prec ps_t$). Haslum et al., in contrast, do not consider such situations a causal threat in case a white knight exists, i.e., if there is another step ps_{wk} in the (partial) plan that is ordered between the threatening step ps_t and the consumer ps_c, $ps_t \prec ps_{wk} \prec ps_c$. We will however not use this "hybrid" formalization in the remainder of the paper and instead strictly assume either Def. 1 or Def. 2.

Release Times and Makespan

For our makespan investigations we use terms and definitions resembling the ones by Bäckström (1998) defined in Def. 5.4. Our definition of a makespan coincides with his definition of the *length of a parallel plan* – but in contrast to his definition we only consider actions of unit-duration and we do not make use of a constraint called *non-concurrency*[3]. We discuss the relation to our results in more detail in the section on deordering and reordering (after Def. 6). Informally, the *makespan* is defined as the shortest execution time if plan steps are allowed to be executed in parallel whenever possible. Formally:

Definition 3. Let $P = (PS, \prec, CL)$ be a (PO or POCL) plan. A *parallel execution* of P is a function $r : PS \to \mathbb{N} \cup \{0\}$ denoting release times for the plan steps in PS satisfying for all $a, b \in PS$: $r(a) + 1 \leq r(b)$ if $a \prec b$.

The *length of a parallel execution* r is defined as $\max_{ps \in PS} r(ps) + 1$, i.e., the latest finishing time of any plan step. The *makespan* of P is then defined as the length of a parallel execution with minimal length, i.e., $\min_r \max_{ps \in PS} r(ps) + 1$.

[3] A non-concurrency constraint $a \# b$ demands that plan steps a and b are not executed in parallel. Such constraints might be considered when *executing* partially ordered plans. Although all linearizations of plans are executable, one might still want to forbid a *parallel* execution of some, e.g., when actions have contradicting effects to prevent partially undefined states. We refer to Bäckström (1998) for a discussion.

PO Plans vs. POCL Plans: Makespan-preserving Plan Conversion

In this section we investigate the differences and similarities between PO plans and POCL plans. As mentioned in the last section, the solution criteria for POCL plans imply that every linearization of such a plan is an executable action sequence generating a goal state. The converse, however, is not true. That is, there exist PO plans which cannot be turned into POCL plans by inserting causal links without having to insert additional ordering constraints thereby changing the set of ordering constraints and the number of possible linearizations. McAllester and Rosenblitt (1991) provide an example by Kambhampati[4] (reproduced in Fig. 1) illustrating this proposition, which we formally capture next:

Proposition 1. *There are PO plans for which there does not exist a POCL plan with the same plan steps and the same ordering constraints.*

Proof. For this, Kambhampati provided a partial POCL plan, depicted in Fig. 1. Clearly, when removing its causal links, it is a PO plan, as all (of its six) linearizations are executable and end up in a goal state. But to turn it into a POCL plan we need to support the open precondition P of the goal action. There are two possibilities to do so. No matter which one we choose (w1 or w2), the respective causal link will raise a causal threat with either s1 or s2, respectively. To resolve said threat, we will either have to order s2 before w1 or s1 before w2, respectively, thereby adding ordering constraints to the induced PO plan (and reducing the number of linearizations). □

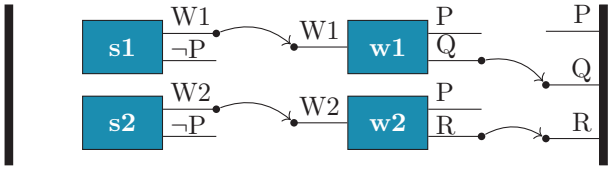

Figure 1: Example demonstrating that there are PO plans for which there is no POCL plan with the same set of ordering constraints (or linearizations).

Note that this is not an issue for the completeness or correctness of any POCL planning system. In the example provided, any POCL planner would simply branch over both options to support the open precondition with a causal link thereby creating two POCL plans (rather than a single PO plan). The union of these two POCL plan's linearizations is still the same as the PO plan's linearizations. Thus the proposition mainly shows that PO plans can represent more linearizations compactly than POCL plans (but regarding makespan-optimization there are still issues as discussed below). We believe that the knowledge about this non-equivalence of PO and POCL plans has been widely

[4] See also Fig. 4 by Kambhampati and Nau (1996). Kambhampati and Nau note (on p. 146) that this example is due to Mark Drummond (conveyed privately in 1991).

lost. This can be noticed from several papers based on POCL planning where the PO and POCL criteria are considered equivalent. Sometimes the POCL criteria and PO criteria are mixed in a non-intuitive way (therefore contributing towards the impression that both are equivalent). For instance, when defining a POCL plan as solution if every linearization is executable (Winer and Young 2016, p. 15; Schattenberg, Bidot, and Biundo 2007, p. 369; Ghallab, Nau, and Traverso 2004, p. 91) we get the situation that partial plans which are normally not considered POCL plans due to the presence of flaws could be considered POCL plans (e.g., the partial plan from Fig. 1 would be considered a POCL plan even though there is an open precondition; it would still be regarded a solution after link insertion although there is still a causal threat flaw). While the previous examples are technically not wrong, others explicitly mention the POCL criteria as equivalent to all linearizations being executable (Muise, Beck, and McIlraith 2016, Thm. 1; Siddiqui and Haslum 2012, p. 805[5]).

In practice, reasoning about partially ordered plans is often based on causal links. But without knowing Prop. 1, one can easily tend to assume that *a PO plan is a solution if and only if for every precondition there exists a plan step with the respective effect that is ordered before it, such that no other step negating that condition can be ordered in between*. However, this criterion is equivalent to stating that a partially ordered set of actions is executable if and only if one can insert a set of causal links (without adding further ordering constraints other than those implied by the links). This, however, is not (always) correct as shown by Kambhampati's example. Instead, if it is important that *all* plans in which every linearization is a solution (e.g., including the one depicted in Fig. 1) are recognized as such, then one needs to check for white knights. Executability checking based on causal links is done in practice, e.g., for planning as SAT (Kautz, McAllester, and Selman 1996) or for SAT-based plan optimization (Muise, Beck, and McIlraith 2016). This, however, can be problematic when aiming at plans that are either minimal in their number of ordering constraints (as shown by the example given in Fig. 1) or when aiming at plans that have a minimal makespan. For the latter, it was – until now – not even clear whether the additional ordering constraints required to turn a PO plan into a POCL plan might influence the plan's makespan. In our (small) example, despite adding an ordering constraint, the makespan remained unaltered. Whether this is *always* possible was yet unknown, however. In other words, Prop. 1 raises the question whether for a given PO plan there exists a POCL plan with the same makespan and whether this decision problem is still tractable or NP-hard. The answer to that question is of high practical relevance as it might either invalidate makespan optimizations that exploit causal links (in case POCL plans turn out to have a potentially larger makespan than their PO counterparts) or it might give evidence on how to find makespan-minimal POCL plans.

To answer that question, we first introduce the concept of *minimum release times*. Let $P = (PS, \prec, CL)$ be a POCL plan. A parallel execution $r : PS \to \mathbb{N} \cup \{0\}$ denoting release times was defined in Def. 3. A function providing *minimum release times* $mr : PS \to \mathbb{N} \cup \{0\}$ is a parallel execution with minimum release times among all possible parallel executions r of P, i.e., it is a minimal release time by which the makespan of a plan is defined. Intuitively, minimum release times can be regarded as the release times where all plan steps are executed as early as possible while taking the ordering constraints and causal links into account. We can compute a minimum release time function mr using Bäckström's DPPL algorithm (cf. Fig. 9), replicated in Alg. 1. It simply performs a progression on the available plan steps "executing" them as early as possible (and in parallel) and assigning the respective release time.

Algorithm 1: Computes the minimum release times.

Input: A POCL plan $P = (PS, \prec, CL)$
Output: The minimum release time mr

```
1 forall ps ∈ PS do mr(ps) ← 0
2 while PS ≠ ∅ do
3 |  Select a step a ∈ PS without predecessors in PS
4 |  forall b ∈ PS with a ≺ b do
5 |  |  mr(b) ← max(mr(b), mr(a) + 1)
6 |  PS ← PS \ {a}
7 return mr
```

Giving these minimal release times – which can be computed in $\mathcal{O}(|PS|^2)$ by Alg. 1 – we can construct a POCL plan from a PO plan with the same makespan thereby answering our open question:

Theorem 1. *Let $P = (PS, \prec)$ be a PO plan with makespan k. Then there exists a POCL plan $P' = (PS, \prec', CL)$ with $\prec' \supseteq \prec$ that also has a makespan of k. Furthermore, P' can be computed in polynomial time.*

Proof. The main idea is as follows: We can observe that the makespan of a plan coincides with the length of a minimal parallel execution mr, i.e., it is $\max_{ps \in PS}\{mr(ps) + 1\}$. We provide an algorithm (Alg. 2) that can always (i.e., for each possible PO plan) insert causal links in such a way that even if it has to alter the ordering constraints to resolve causal threats (cf. our example in Fig. 1), it can do so without increasing the release times responsible for defining the makespan. Thus, the theorem follows inductively when we show that we can always insert a causal link and the consequential ordering constraints (to resolve arising threats that might occur due to situations as depicted in Fig. 1) without increasing the minimum release times.

To do so, we first compute the minimum release times by Alg. 1 in Line 2 of Alg. 2. This step takes $\mathcal{O}(|PS|^2)$ time. The while loop starting in Line 3 selects arbitrarily one open precondition after another. Alg. 2 then picks a producer of

[5]While Siddiqui and Haslum (2012) use the standard POCL formalization *without* white knights (thus the presumed equivalence (p. 805) is incorrect), their follow-up work (Siddiqui and Haslum 2015) relies on a POCL threat formalization *with* white knights (thus the stated equivalence (cf. Def. 1 and Thm. 1) is correct).

the selected open precondition out of a set of possibilities according to their minimum release times and specifies which causal link is to be added. Let c be a plan step with open precondition v as selected in Line 4 of Alg. 2. We only take plan steps for being a producer into account that are already ordered before the consumer. Such a plan step must exist because otherwise not every linearization of the PO plan is a solution.

Let pc be the candidates for being a producer of v as given in Line 5. Pick $p \in pc$ as producer such that $mr(p)$ is maximal among pc (Line 6). We need to protect this causal link and therefore let $T \subseteq PS$ be the set of threatening plan steps. The choice of p implies $mr(p) > mr(t)$ for all plan steps $t \in T$, because for every $t \in T$ there must be a plan step out of pc that is ordered after t. Otherwise there would be a linearization of the PO plan such that there is a step $t \in T$ that is ordered between c and all plan steps out of pc, which is again a contradiction to P being a PO plan. In order to resolve the threats we order every $t \in T$ before p (Line 8). This does not change $mr(p)$ since $mr(t) + 1 \leq mr(p)$ for all $t \in T$ as observed before (more precisely, if $mr(p)$ would increase, then there were a $t \in T$ such that $mr(t) + 1 > mr(p)$, which would be a contradiction to our choice of p with $mr(p) > mr(t')$ for all $t' \in T$). Moreover, we recompute the transitive closure of the orderings, i.e. we add (not already existing) ordering constraints from actions ordered before all $t \in T$ to p and the actions ordered after p. Adding transitive ordering constraints cannot increase the minimum release times. Therefore, in the resulting partial plan the minimum release times have not changed and still every linearization is a solution. We can thus inductively repeat this procedure until every precondition is supported by a causal link. As the minimum release times do not increase, the resulting POCL plan has the same makespan as the input PO plan. Concerning the runtime, we can state that we require at most $|PS|^2$ operations for calling Alg. 1 in Line 2 plus the runtime for the while loop. A single iteration of it needs at most $|PS|$ (Line 5) $+|PS|$ (Line 6) $+$ 1 (Line 7) $+ |PS|^2$ (Line 8) operations. Since one causal link is added per loop, the overall runtime of the algorithm is in $\mathcal{O}(|Pre| \cdot |PS|^2)$ with $|Pre| = \sum_{ps \in PS} |prec(ps)|$ and hence polynomial. □

We showed that we can turn each PO plan into a POCL plan with the same makespan in polynomial time. The converse holds as well, since every POCL plan induces a PO plan with the same makespan (simply remove all causal links). Taken together, we get the insight that it does not make a difference whether we rely on PO plans or on POCL plans when we aim at makespan optimality. Formally, we get the following corollary.

Corollary 1. *Let $\mathcal{P}$ be a planning problem. Then there is a PO plan with makespan k for $\mathcal{P}$ if and only if there is a POCL plan with makespan k for $\mathcal{P}$.*

Deordering & Reordering

Our main endeavor is to find plans having an optimal makespan. For this, we first provide the necessary definitions for *obtaining* a plan with a desired makespan, i.e., we define the allowed plan modifications. Our definitions are straight-forward adaptations of those in Def. 6.1 by Bäckström (1998) to work for POCL plans.

Algorithm 2: Computes a POCL plan from a PO plan with identical makespan.

Input: A PO plan $P = (PS, \prec)$
Output: A POCL plan $P' = (PS, \prec', CL)$ with $\prec' \supseteq \prec$ and the same makespan as P

1 $\prec' \leftarrow \prec$ and $CL \leftarrow \emptyset$
2 compute mr by using Alg. 1
3 **while** $P' = (PS, \prec', CL)$ *is not a POCL plan* **do**
4 select some plan step c with precondition variable v that is not yet supported by a causal link
/* pc (producer candidates) is a set of candidates for being a producer of v */
5 $pc \leftarrow \{p \in PS \mid v \in add(p) \wedge (p, c) \in \prec'\}$
/* select producer out of pc with highest minimum release time */
6 $p = \arg\max_{q \in pc} mr(q)$ and $cl = (p, v, c)$
7 $CL \leftarrow CL \cup \{cl\}$
8 $\prec' \leftarrow (\prec' \cup \{(t, p) \mid t \in PS \text{ threatens } cl\})^+$
9 **return** $P' = (PS, \prec', CL)$

Definition 4. Let $P = (PS, \prec, CL)$ and $Q = (PS, \prec', CL')$ be two (PO or POCL) plans for a planning problem $\mathcal{P}$. Then,

- Q is a *reordering* of P if and only if both P and Q are PO plans or both are POCL plans for $\mathcal{P}$.
- Q is a *deordering* of P if and only if Q is a reordering of P and $\prec' \subseteq \prec$.

Intuitively, *reordering* a plan means that we are allowed to *change* ordering constraints and causal links arbitrarily, as long as the resulting plan is still a solution. By contrast, *deordering* means that we are only allowed to *remove* ordering constraints. Since we do not demand anything for causal links in the previous definition, they might be changed as well as long as this does not result into adding new orderings.

Bäckström (1998) and Aghighi and Bäckström (2017) showed that finding a PO plan with a minimum *number of ordering constraints* via deordering and/or reordering – which he called *minimum-constrained* deordering/reordering – is NP-complete (Thm. 4.12). This minimality does not imply a minimal makespan, however. In Fig. 2 we provide an example showing that a deordering w.r.t. a plan's number of ordering constraints does not always result in a minimal makespan.

The example's PO plan has 5 ordering constraints (6 with transitive orderings) and a makespan of 3. Minimizing the number of ordering constraints can be achieved by removing the orderings from $A2$, $A3$, and $A4$ to $A5$ resulting in a PO plan with 2 ordering constraints (3 with transitives) while the makespan stays 3. Instead, removing the ordering from

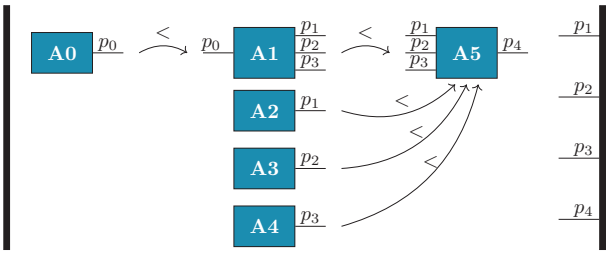

Figure 2: Example demonstrating that minimizing the number of ordering constraints does not imply a minimum makespan.

$A1$ to $A5$ (and from $A0$ to $A5$) will also result in a PO plan, but with a makespan of 2 and 4 ordering constraints.

To study the makespan optimization problem, we adapt Bäckström's definitions of minimum-parallel reordering and deordering (Def. 6.1) to our notation for the makespan:

Definition 5. Let P and Q be two (PO or POCL) plans. Then,

- Q is a *minimum-makespan reordering* of P if and only if Q is a reordering of P and there is no reordering of Q with a smaller makespan.
- Q is a *minimum-makespan deordering* of P if and only if Q is a deordering of P and there is no deordering of Q with a smaller makespan.

Similar to Bäckström's Def. 6.2 and Def. 6.3 for parallel plans with concurrency constraints, we define the two respective decision problems for plans without them:

Definition 6. Let P be a (PO or POCL) plan with makespan k. Then, we define the following decision problems:

- MINIMUM-MAKESPAN REORDERING: Does there exist a reordering of P with makespan $k' < k$?
- MINIMUM-MAKESPAN DEORDERING: Does there exist a deordering of P with makespan $k' < k$?

Before we come to our main result we want to discuss some of the related work and how it differs from ours. As mentioned in the beginning, Bäckström (1998) also investigated the makespan of plans. He, however, allows non-unit execution times of plan steps and plans (called parallel plans) may feature non-concurrency-constraints that specify which plan steps may not be executed in parallel. For such plans, even *computing* the makespan (which he calls execution time) is NP-hard (Bäckström 1998, Thm. 5.8 and Cor. 5.9). His further investigations of optimizing the makespan also depend on this non-concurrency constraint set (as do the proofs) – thus both the results for reordering and deordering were not known in the absence of these constraints. A special case that was investigated by Bäckström are *definite plans* (Def. 5.1), which are plans in which the given non-concurrency constraints are trivially satisfied (as definite plans require that for each non-concurrency constraint $a\#b$ either $a \prec b$ or $b \prec a$ holds). NP-completeness was shown for makespan-minimization via reordering definite plans (Bäckström 1998, Thm. 6.7 and Thm. 6.8). Whether *deordering* definite plans is NP-complete as well was stated as an open question by Bäckström (Sec. 7). We will show NP-completeness of deordering *PO plans*, thus NP-hardness for definite plans follows directly thereby answering this question.

Theorem 2. *Deciding the* MINIMUM-MAKESPAN DEORDERING *problem of a PO plan P is NP-complete.*

Proof. Membership: Compute the current makespan k in polynomial time and guess a deordering P with the same plan steps. This includes verifying that the guessed deordering P is actually a PO plan, which can be done in polynomial time (Nebel and Bäckström 1994, Thm. 12; Chapman 1987, p. 340). Reject if infeasible. Accept if the makespan (computed by Alg. 1) is $< k$, otherwise reject.

Hardness: Proof by reduction from 3-SAT, which is NP-complete (Hopcroft and Ullman 1979). Let ϕ be an instance of 3-SAT, which consists of a set of atoms $X = \{x_1, \ldots, x_n\}$ and a set of clauses $C = \{C_1, \ldots, C_m\}$ over X, such that for all $1 \leq j \leq m$, $C_j = \{l_{j,1}, l_{j,2}, l_{j,3}\}$ is a clause of three literals over X. We construct a planning problem $\mathcal{P} = (V, A, s_I, g)$ and PO plan $P = (PS, \prec)$ with makespan 4, such that it can be deordered to a PO plan with makespan 3 if and only if the SAT formula is satisfiable.

Therefore, every $x_i \in X$ is represented by two state variables x_i^T and x_i^F encoding their truth assignment to *true* and *false*, respectively. Moreover, there is a state variable g_i for each $x_i \in X$, which we use to enforce a truth assignment. For every clause $C_j \in C$ we also introduce a state variable c_j. Let

$$V = \bigcup_{1 \leq i \leq n} \{x_i^T, x_i^F, g_i\} \cup \bigcup_{1 \leq j \leq m} \{c_j\},$$
$$s_I = \emptyset, \text{ and } g = \{g_i \mid 1 \leq i \leq n\}.$$

For each $x_i \in X$ we define three actions and for each clause $C_j \in C$ we define four – all according to Tab. 1.

action	*prec*	*add*	*del*	action	*prec*	*add*	*del*
T_i	$\emptyset$	$\emptyset$	$\{g_i\}$	B_j^1	$\{l_{j,1}^*\}$	$\{c_j\}$	$\emptyset$
A_i^T	$\emptyset$	$\{g_i, x_i^T\}$	$\emptyset$	B_j^2	$\{l_{j,2}^*\}$	$\{c_j\}$	$\emptyset$
A_i^F	$\emptyset$	$\{g_i, x_i^F\}$	$\emptyset$	B_j^3	$\{l_{j,3}^*\}$	$\{c_j\}$	$\emptyset$
				D_j	$\{c_j\}$	$\emptyset$	$\emptyset$

Table 1: Actions for each atom x_i (left) and clause C_j (right). We define $l_{j,r}^* \leftarrow x_k^T$ if $l_{j,r} = x_k$ and $l_{j,r}^* \leftarrow x_k^F$ if $l_{j,r} = \neg x_k$.

Each action appears exactly once in P, which is why we label the plan steps according to the actions and refer to them as actions as well. Furthermore, we introduce the following ordering constraints for each $1 \leq i \leq n$ and $1 \leq j \leq m$:

$$T_i \prec \left\{ \begin{array}{c} A_i^T \\ A_i^F \end{array} \right\} \prec \left\{ \begin{array}{c} B_j^1 \\ B_j^2 \\ B_j^3 \end{array} \right\} \prec D_j.$$

The previous notation is a compact representation for ordering constraints: Actions within the same brackets are unordered with respect to each other, but ordered to all actions depicted before them or behind them. For example, it holds $T_i \prec A_i^{T/F}$, A_i^T is unordered to A_i^F, and both $A_i^{T/F}$ are ordered before all $B_j^{1/2/3}$. Obviously, P is a PO plan to $\mathcal{P}$ with makespan 4 as the state variables g_i in the goal description are established by the actions $A_i^{T/F}$ and T_i, which delete g_i, are ordered before the $A_i^{T/F}$. The preconditions $x_k^{T/F}$ of the actions $B_j^{1/2/3}$ are satisfied because of the $A_i^{T/F}$. Finally, the actions D_j can be executed since the $B_j^{1/2/3}$ are executed before and establish all c_j. Note that actually just one of A_i^T and A_i^F needs to be ordered after T_i in order to guarantee g_i in the goal description. Hence, one of these ordering constraints can be deleted such that the corresponding action, say A_i^T for example, can be executed at time point 0. Then, those $B_j^{1/2/3}$ that have a precondition fitting the chosen ordering to the respective $A_i^{T/F}$ (e.g., B_j^1 if it had the precondition x_i^T) may be executed already at time point 1. As a consequence, D_j could also be executed earlier, i.e., at time point 2. If we can find the right choice concerning the removal of ordering constraints including the T_i and $A_i^{T/F}$, all D_j will be executable at time point 2 such that the makespan shrinks to 3. Hence, this choice may be interpreted as assignment to the variables in X, which causes the respective clauses to become true (reflected by the respective actions $B_j^{1/2/3}$ as well as D_j becoming applicable earlier). Thus, it remains to show that ϕ is satisfiable if and only if P can be deordered to a plan Q with makespan at most 3.

$\Rightarrow$ Suppose ϕ is satisfiable. Let I be a truth assignment to X that satisfies ϕ. For all i with $I(x_i) = \mathit{true}$ delete the ordering $T_i \prec A_i^T$ and for all i with $I(x_i) = \mathit{false}$ delete the ordering $T_i \prec A_i^F$. As ϕ is satisfiable, for each $1 \leq j \leq m$ at least one of the $\{l_{j,1}, l_{j,2}, l_{j,3}\}$ of clause C_j must be true. We pick the corresponding action B_j^r and remove ordering constraints of the form $A_i^{T/F} \prec B_j^r$, specified more precisely below, such that B_j^r can be executed at time point 1. The precondition-establishing plan step $A_i^{T/F}$ is still ordered before B_j^r at time point 0 as we have ensured in the first step. D_j can then also be executed earlier at time point 2, directly after B_j^r. So, for each $1 \leq j \leq m$ choose r_1 such that $l_{j,r_1} \in C_j$ is satisfied by I. Let l_{j,r_2} and l_{j,r_3} be the remaining two literals in C_j. Remove $A_k^F \prec B_j^{r_1}$ if $l_{j,r_1} = x_k$, otherwise (i.e., if $l_{j,r_1} = \neg x_k$) remove $A_k^T \prec B_j^{r_1}$. Moreover, remove $B_j^{r_2} \prec D_j$ and $B_j^{r_3} \prec D_j$. For example, suppose $I(x_i) = \mathit{true}$ for all i. Then the remaining ordering can be written as

$$\left\{ \begin{array}{c} T_i \\ A_i^T \end{array} \right\} \prec \left\{ \begin{array}{c} A_i^F \\ B_j^{r_1} \end{array} \right\} \prec \left\{ \begin{array}{c} B_j^{r_2} \\ B_j^{r_3} \\ D_j \end{array} \right\}.$$

Thus, $mr(D_j) = 2$ for all $1 \leq j \leq m$. For all remaining actions, other than these D_j, their minimum release time was already at most 2 in the first place. Therefore Q has makespan 3. It is still executable because one of the actions A_i^T, A_i^F is ordered after T_i and establishes g_i for the goal description. Moreover, the actions B_{j,r_2}, B_{j,r_3} are still ordered after all A_i^T, A_i^F, so their preconditions are satisfied and B_{j,r_1} is ordered after the corresponding $A_k^{T/F}$.

$\Leftarrow$ Suppose there is a deordering Q of P with makespan at most 3. Note that D_j must be ordered after at least one of the $B_j^{1/2/3}$ for all $1 \leq j \leq m$, since they establish the precondition c_j. In order to guarantee $mr(D_j) \leq 2$, there must be a B_j^r with $mr(B_j^r) \leq 1$. This implies that there is an i, such that $mr(A_i^T) = 0$ or $mr(A_i^F) = 0$ and such that $l_{j,r} = x_i$ or $\neg x_i$ because the precondition $l_{j,r}^*$ of B_j^r must be established. So define an interpretation I such that for all $1 \leq i \leq n$ holds $I(x_i) \leftarrow \mathit{true}$ if $mr(A_i^T) = 0$ and $I(x_i) \leftarrow \mathit{false}$ otherwise. All linearizations are executable as either A_i^T or A_i^F must be ordered after T_i for all $1 \leq i \leq n$ because otherwise it is not guaranteed that g_i is true in the end. The B_j^r with $mr(B_j^r) = 1$ indicate which literal is *true* in clause C_j. This assignment satisfies ϕ by construction. □

There is an interesting relationship between the proofs for deciding the existence of a *minimum-makespan* deordering shown here and Bäckström's hardness proof for *minimum-constrained* plans (plans with a minimal number of ordering constraints). His proof does not require negative (i.e., delete) effects, meaning that deciding minimum-constrained deordering remains NP-hard in their absence, as stated in Cor. 4.10 (Bäckström 1998). Our proof *does* make use of negative effects, however. And in fact, without them, the problem becomes tractable. This is easy to see, since it is optimal for the makespan to (greedily) apply every action as early as possible. A simple polynomial-time algorithm thus checks for all actions all its predecessors (including the initial state) starting with the right-most actions and checks which ordering constraints to its predecessors can be deleted without violating executability of the plan – again in a right-to-left fashion. This way, every action gets executed as early as possible without sacrificing plan executability and without adding ordering constraints. Compute its makespan and check whether it is smaller than the one of the input. Accept or reject accordingly.

Corollary 2. *Let P be PO or POCL plan* without negative effects*. Deciding the* MINIMUM-MAKESPAN DEORDERING *problem for P can be done in polynomial time.*

We can also generalize our main result, minimum-makespan deordering being NP-complete for PO plans, to POCL plans due to Cor. 1. We still have to show membership, which is trivial since we can simply remove the causal links and use the same membership test as in the proof of Thm. 2. Thus:

Corollary 3. *Deciding the* MINIMUM-MAKESPAN DEORDERING *problem of a POCL plan P is NP-complete.*

Recall that deordering of a POCL plan allowed us to change causal links arbitrarily, as long as no new orderings get inserted as a consequence. Choosing the "best" producer for an action's precondition variable was one of the sources of hardness. As it turns out, this is also the *only* source

of hardness, meaning that finding a makespan-optimal deordering becomes decidable in polynomial time if we prohibit changing causal links. Since the problem then becomes tractable, it also means that we might not find the makespan-optimal deordering that might be possible if we would allow changing them. In other words: The optimal makespan of an induced deordered PO plan might be smaller.

Theorem 3. *Let $P = (PS, \prec, CL)$ be a POCL plan for some planning problem $\mathcal{P}$. Deciding whether there is a POCL plan $Q = (PS, \prec', CL)$ for $\mathcal{P}$ with makespan $\leq k$ such that $\prec' \subseteq \prec$ can be done in polynomial time.*

Proof. Let P and k be given. Construct a POCL plan Q in the following way. Consider P and remove all ordering constraints. (By definition the result of this is not even a *partial* POCL plan because the orderings implied by the causal links are now missing, but we will extend it to a POCL plan throughout this proof.) For all causal links $cl \in CL$ add the corresponding ordering constraint and extend it to the transitive closure afterwards. We now resolve all causal threats by demoting or promoting threatening plan steps. However, we may only choose adding an ordering in accordance to the ordering constraints originally present in P (since otherwise the resulting plan would not be a deordering). So let $(ps_p, v, ps_c) \in CL$ be a causal link and ps_t a threatening plan step. We know that either $(ps_t, ps_p) \in \prec$ or $(ps_c, ps_t) \in \prec$ must be true as otherwise P would not have been a POCL plan. We also know that only one of them can hold as otherwise $\prec$ would not have been a partial order. We thus add the respective ordering to $\prec'$ that was already in $\prec$. After all threats are resolved we complete the set of ordering constraints by computing the transitive closure. The result, Q, is then a POCL plan because all preconditions are supported by causal links and there are no causal threats. This can be done in time $\mathcal{O}(|PS|^2)$.

We claim that Q is a deordering of P with minimal makespan (provided we are not allowed to change causal links). By definition Q must contain all ordering constraints implied by its causal links. Furthermore, all causal threats must be resolved and the choice between promoting and demoting is predefined by P because we may not add additional ordering constraints – we thus did not add any unnecessary ordering constraints. If Q has makespan $\leq k$ the answer is "yes" and "no" otherwise. This can be checked in polynomial-time using Algorithm 1. □

Coming back to our main results (Thm. 2 and Cor. 3), there is yet another implication. Since every deordering is also a reordering, we can also conclude that reordering is as hard as deordering. Membership can again be tested as before.

Corollary 4. *Deciding the* MINIMUM-MAKESPAN REORDERING *problem of a (PO or POCL) plan P is NP-complete.*

The previous result – restricted to PO plans – also follows from a proof from the literature (but not from its theorem). Said proof shows Thm. 7.10 by Bäckström (1998) stating several additional criteria (totally ordered input plans with toggling effects (every effect requires its negation as precondition)) under which deciding the MINIMUM-MAKESPAN REORDERING problem remains NP-hard.

Conclusion

To finally answer the paper's initial question whether partial order (PO) plans are equivalent to partial order causal link (POCL) plans, we can say: Yes and No. *No*, because there exist PO plans for which there does not exist a POCL plan with the same set of ordering constraints. However, we can state *Yes* because we showed that for each PO plan we can construct a POCL plan with the same makespan. For this, we provided an algorithm running in polynomial time. We further proved that the *makespan deordering problem* for PO and POCL plans (i.e., removing ordering constraints to find a plan with minimal makespan) is NP-complete. Previous related investigations made use of a set of additional constraints so it was not clear whether the problem remains as hard without them. Other results focused on optimizing the *number of ordering* constraints, which was used as an approximation to optimizing the makespan. From our deordering results we also concluded that *reordering* is NP-complete for PO and POCL plans. We further showed that deordering becomes decidable in P if there are no negative effects or if we are not allowed to change the causal links present in a POCL plan.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable feedback.

This work was partly funded by the technology transfer project "Do it yourself, but not alone: *Companion*-Technology for DIY support" of the CRC/TRR 62 funded by the German Research Foundation (DFG). The industrial project partner is the Corporate Research Sector of the Robert Bosch GmbH.

References

Aghighi, M., and Bäckström, C. 2017. Plan reordering and parallel execution – a parameterized complexity view. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI 2017)*, 3540–3546. AAAI Press.

Bäckström, C. 1998. Computational aspects of reordering plans. *Journal of Artificial Intelligence Research (JAIR)* 9:99–138.

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 *Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI 2016)*, 225–233. IOS Press.

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

Chrpa, L.; Kilani, A.; and Balyo, T. 2014. On different strategies for eliminating redundant actions from plans. In *Proceedings of the 7th Annual Symposium on Combinatorial Search (SOCS 2014)*, 10–18. AAAI press.

Coles, A.; Coles, A.; Fox, M.; and Long, D. 2010. Forward-chaining partial-order planning. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS 2010)*, 42–49. AAAI Press.

Do, M. B., and Kambhampati, S. 2003. Improving the temporal flexibility of position constrained metric temporal plans. In *Proceedings of the 13th International Conference on Automated Planning and Scheduling (ICAPS 2003)*, AAAI Press. 42–51.

Fink, E., and Yang, Q. 1992. Formalizing plan justifications. In *Proceedings of the 9th Conference of the Canadian Society for Computational Studies of Intelligence (CSCSI 1992)*, 9–14.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2004. *Automated Planning: Theory and Practice*. Morgan Kaufmann.

Haslum, P.; Lipovetzky, N.; Magazzeni, D.; and Muise, C. 2019. *An Introduction to the Planning Domain Definition Language*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool.

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

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

Karpas, E., and Domshlak, C. 2012. Optimal search with inadmissible heuristics. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS 2012)*, 92–100. AAAI Press.

Kautz, H. A.; McAllester, D. A.; and Selman, B. 1996. Encoding plans in propositional logic. In *Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning (KR 1996)*, 374–384. Morgan Kaufmann Publishers Inc.

Lipovetzky, N., and Geffner, H. 2011. Searching for plans with carefully designed probes. In *Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS 2011)*, 154–161. AAAI Press.

McAllester, D., and Rosenblitt, D. 1991. Systematic nonlinear planning. In *Proceedings of the 9th National Conference on Artificial Intelligence (AAAI 1991)*, 634–639. AAAI Press.

Muise, C.; Beck, J. C.; and McIlraith, S. A. 2013. Flexible execution of partial order plans with temporal constraints. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, 2328–2335. AAAI Press.

Muise, C.; Beck, J. C.; and McIlraith, S. A. 2016. Optimal partial-order plan relaxation via maxsat. *Journal of Artificial Intelligence Research (JAIR)* 57:113–149.

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

Nakhost, H., and Müller, M. 2010. Action elimination and plan neighborhood graph search: Two algorithms for plan improvement. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS 2010)*. AAAI Press.

Nebel, B., and Bäckström, C. 1994. On the computational complexity of temporal projection, planning, and plan validation. *Artificial Intelligence* 66(1):125–160.

Olz, C., and Bercher, P. 2019. Eliminating redundant actions in partially ordered plans – a complexity analysis. In *Proceedings of the 28th International Conference on Automated Planning and Scheduling (ICAPS 2019)*, 310–319. AAAI Press.

Penberthy, J. S., and Weld, D. S. 1992. UCPOP: A sound, complete, partial order planner for ADL. In *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR 1992)*, 103–114. Morgan Kaufmann.

Say, B.; Cire, A. A.; and Beck, J. C. 2016. Mathematical programming models for optimizing partial-order plan flexibility. In *Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI 2016)*, 1044–1052. IOS Press.

Schattenberg, B.; Bidot, J.; and Biundo, S. 2007. On the construction and evaluation of flexible plan-refinement strategies. In *Proceedings of the 30th German Conference on Artificial Intelligence (KI 2007)*, 367–381. Springer.

Siddiqui, F. H., and Haslum, P. 2012. Block-structured plan deordering. In *Proceedings of the 25th Australasian Joint Conference on Artificial Intelligence (AI 2012)*, 803–814. Springer.

Siddiqui, F. H., and Haslum, P. 2015. Continuing plan quality optimisation. *Journal of Artificial Intelligence Research (JAIR)* 54:369–435.

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

Weld, D. S. 1994. An introduction to least commitment planning. *AI Magazine* 15(4):27–61.

Winer, D. R., and Young, R. M. 2016. Discourse-driven narrative generation with bipartite planning. In *Proceedings of the Ninth International Natural Language Generation Conference (INLG 2016)*, 11–20. The Association for Computer Linguistics.

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