Makespan Investigations of Sequential, Parallel, PO, and POCL Plans

Harrison Oates, Pascal Bercher

School of Computing,
The Australian National University, Canberra, Australia
harrison.oates@anu.edu.au, pascal.bercher@anu.edu.au

Abstract

Modern planning systems utilize various plan representations — sequential, parallel, partially ordered (PO), and partial-order causal link (POCL) — each with different models for concurrency. These formalisms are often implicitly assumed to have the same base properties, particularly regarding makespan. We challenge this assumption, proving the relationship between them is fundamentally asymmetric. Our analysis shows conversions from plans with rigid concurrency layers (sequential, parallel) to those with flexible partial orders (PO, POCL) can preserve makespan. However, the reverse generally fails; the flexible orderings in PO/POCL plans can yield shorter makespans for solutions that cannot be represented in parallel plans without serialization. We prove that finding an optimal parallel representation for a given POCL plan is NP–complete, resolving a key question about their practical interchangeability. We also provide tight complexity bounds for makespan-bounded plan existence. Notably, our results disprove a claim in the literature that planning graph-based planners maximize concurrency by minimizing the critical path in derived PO plans.

Introduction

Classical planning employs a variety of plan representations — sequential, parallel, partial-order (PO) and partial-order causal link (POCL) — each offering distinct trade-offs in expressivity, the computational cost of plan generation, and their ability to model concurrent execution. These formalisms remain central to diverse areas of contemporary planning, including SAT-based methods that leverage parallel structures (Rintanen 2012), hierarchical planners that directly encode POCL plans (Bercher, Lin, and Alford 2022; Bit-Monnot 2023; Firsov, Fiorino, and Pellier 2023; Bit-Monnot et al. 2020) or generate parallel plans (Lotem, Nau, and Hendler 1999; Cavrel, Pellier, and Fiorino 2023), as well as plan optimization techniques (Bercher, Haslum, and Muise 2024). When actions can be executed concurrently, *makespan*, the minimum time required to execute a plan under parallelism, emerges as a useful performance metric. Despite their structural differences, a persistent and often implicit assumption seems to be that these representations are interchangeable. This view suggests that a makespan-minimal plan in one formalism is optimal across all.

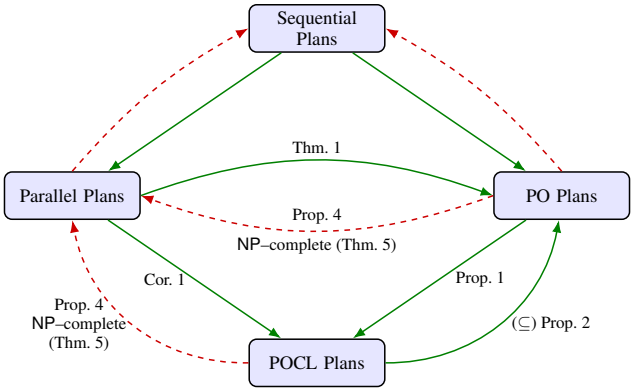

Figure 1: Makespan convertibility. Solid green arrows indicate makespan can always be preserved. Dashed red arrows indicate makespan is not generally preserved.

For instance, Cavrel, Pellier, and Fiorino (2023) find makespan-minimal solutions in a hierarchical setting using parallel plan structures. In their future work, they correctly identify partial-order plans as a "stronger way to represent concurrency," but frame this strength in qualitative terms like being "more resilient and flexible," leaving the potential for quantitative improvement – achieving a shorter makespan – unaddressed. This omission suggests a default assumption that both approaches achieve the same optimal makespan. Similarly, CPT (Vidal and Geffner 2006) claims makespan optimality for temporal POCL planning, but this optimality is defined only over POCL plans encoded as parallel plans, not over standard POCL structures. This again may reflect an implicit assumption that POCL and parallel plan representations are interchangeable with respect to makespan.

This issue escalates to flawed foundational claims in other work. In an analysis of loosely-coupled planning and scheduling using PO plans, Pecora, Rasconi, and Cesta (2004) claim in their Theorem 1 that planning graph (PG) planners like Graphplan (Blum and Furst 1997), which naturally produce parallel plans, 'maximize concurrency with respect to the causal model of the problem' by minimizing the critical path of the resulting PO plan. We prove this theorem is incorrect, as a PO plan can achieve strictly shorter makespans than any valid parallel plan for the same

problem. One immediate consequence is that heuristics that are admissible for makespan-optimal parallel planning, such as those of Haslum and Geffner (2000), are inadmissible for PO/POCL search (Bercher, Geier, and Biundo 2013; Sapena, Onaindia, and Torreño 2015) as their cost estimates derived from PGs are no longer guaranteed lower bounds. This finding is particularly significant as it directly challenges the makespan-optimality claims of influential planners that rely on such cost estimations, such as CPT.

This paper resolves these foundational issues in the literature by systematically investigating the makespan relationships between these four key plan types. Our contributions are twofold. First, we prove a foundational asymmetry in makespan-preserving convertibility along a conceptual hierarchy from more rigid to more flexible representations. While moving 'up' this hierarchy (e.g., from parallel to PO plans) can preserve makespan, the reverse generally fails. The flexible orderings in PO/POCL plans can encode solutions with shorter makespans that cannot be represented in parallel plans without serialization. We prove that converting a POCL plan to a makespan-minimal parallel plan is NP–complete, highlighting a key barrier to efficient interchange between plan representations; we also provide nontrivial upper bounds on makespan. These relationships are summarised in Figure 1.

Second, we establish the first comprehensive complexity bounds for the makespan-bounded plan existence problem for non-sequential plans. We show the problem is PSPACE–complete when the makespan bound is encoded in binary, but drops to NP–complete for a unary encoding. This distinction reflects the common search strategy in practical systems like SAT-based planners, where the makespan bound increases incrementally until a solution is found. Together, these results refute the assumption of makespan equivalence and provide a formal basis for understanding the computational trade-offs between plan representations.

Classical Planning Framework

A classical planning problem is a tuple $(\mathcal{F}, \mathcal{A}, \mathcal{I}, \mathcal{G})$ where $\mathcal{F}$ is a finite set of fluents, a *state* is any $s \in 2^{\mathcal{F}}$, and $\mathcal{A} \subseteq (2^{\mathcal{F}})^3$ is a finite set of *actions*. An action $a = (pre(a), add(a), del(a))$ is a tuple with preconditions $pre(a) \subseteq \mathcal{F}$, add effects $add(a) \subseteq \mathcal{F}$, and delete effects $del(a) \subseteq \mathcal{F}$. $\mathcal{I} \in 2^{\mathcal{F}}$ is the initial state, and $\mathcal{G} \subseteq \mathcal{F}$ is the goal description. An action a is applicable in a state s if and only if $pre(a) \subseteq s$. If a is applicable in s, the state transition function $\gamma : \mathcal{A} \times 2^{\mathcal{F}} \to 2^{\mathcal{F}}$ returns the successor state $\gamma(a, s) = (s \setminus del(a)) \cup add(a)$.

A *plan* is a structure of actions intended to transform the initial state $\mathcal{I}$ into a state satisfying $\mathcal{G}$. We assume all actions take unit duration to complete. Informally, the *makespan* of a plan is the minimum number of time steps required to execute the plan, allowing actions to be performed in parallel whenever possible according to dependencies. A precise definition depends on the considered plan representation.

Sequential Plans

An action sequence $\bar{a} = a_1, a_2, \ldots, a_n$ is applicable in a state s_0 if and only if there exists a sequence of states $s_0, s_1, \ldots, s_n$ such that for all $1 \leq i \leq n$, it holds that a_i is applicable in state s_{i-1} and generates state $s_i = \gamma(a_i, s_{i-1})$. s_n is called the state generated by $\bar{a}$.

Definition 1 (Sequential Plan)**.** *An action sequence $\bar{a}$ is called a sequential plan to a classical planning problem $(\mathcal{F}, \mathcal{A}, \mathcal{I}, \mathcal{G})$ if and only if it is applicable to $\mathcal{I}$ and generates a goal state $s \supseteq \mathcal{G}$.*

The makespan of a sequential plan $\bar{a} = a_1, \ldots, a_n$ under the unit duration assumption is simply its length, n.

Parallel Plans

Parallel plans, as introduced in the Graphplan algorithm (Blum and Furst 1997) are structured as a sequence of *action sets*, often called timesteps. Actions within the same timestep are intended to be executed concurrently. For an action set A to be applicable in a state s, two conditions must be met. The first is that all preconditions of all actions in A must hold in s: $\bigcup_{a \in A} pre(a) \subseteq s$. The second criterion is that all pairs of distinct actions $a_i, a_j \in A$ must be non-interfering (Blum and Furst 1997). This holds if no action deletes another's preconditions $(pre(a_i) \cap del(a_j) = \emptyset)$ and no action deletes another's positive effects $(add(a_i) \cap del(a_j) = \emptyset)$. The first condition for non-interference guarantees that all actions are individually applicable in the current state, regardless of their execution order. The second ensures that the successor state is unique and independent of the order in which actions are applied.

Assuming a non-interfering action set A, the state transition function γ is generalised to apply the combined effects: $\gamma(A, s) = \big(s \setminus \bigcup_{a \in A} del(a)\big) \cup \big(\bigcup_{a \in A} add(a)\big)$. Analogous to the serial case, a sequence of action sets $\bar{A} = A_1, \ldots, A_m$ is applicable in state s_0 if and only if there exists a sequence of states $s_0, \ldots, s_m$ such that each A_i is applicable in s_{i-1} and generates state $s_i = \gamma(A_i, s_{i-1})$. s_m is called the state generated by $\bar{A}$.

Definition 2 (Parallel Plan)**.** *An action set sequence $\bar{A}$ is called a parallel plan to a classical planning problem $(\mathcal{F}, \mathcal{A}, \mathcal{I}, \mathcal{G})$ if and only if it is applicable to $\mathcal{I}$ and generates a goal state $s \supseteq \mathcal{G}$.*

The makespan of a parallel plan $\bar{A} = A_1, \ldots, A_m$ is m.

Partial Order Plans

A partial partial–order (PO) plan P is a tuple $(PS, \prec)$ consisting of a finite set of plan steps PS, with each step $(l, a) \in PS$ consisting of an action a and a unique label l. This labelling is required in order to differentiate between multiple occurrences of an action in the same plan. As we did for actions, we use $pre(ps), add(ps)$, and $del(ps)$ to refer to the preconditions and effects of a plan step ps's action. $\prec \subseteq PS \times PS$ is a strict partial order over the plan steps.

Definition 3 (PO Plan)**.** *A partial PO plan $P = (PS, \prec)$ is called a PO plan to a classical planning problem $(\mathcal{F}, \mathcal{A}, \mathcal{I}, \mathcal{G})$ if and only if every linearization is executable in $\mathcal{I}$ and generates a goal state $s \supseteq \mathcal{G}$.*

The makespan for PO plans represents the minimum time required to execute the plan, assuming steps can run in parallel whenever allowed by the ordering constraints.

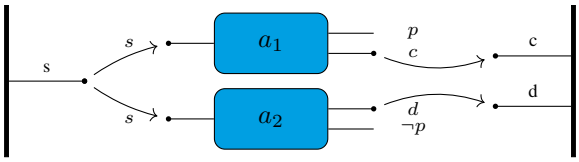

Figure 2: A POCL plan where the order between a_1 and a_2 is unconstrained. Arrows depict causal links and implied ordering constraints are not shown.

Definition 4 (Parallel Execution (Bercher and Olz 2020))**.** *Let $P = (PS, \prec)$ be a PO 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 constraint $r(a) + 1 \leq r(b)$ ensures that if step a precedes step b, then b cannot start until at least the time step after a starts. The total duration of a parallel execution r is determined by the completion time of the last step: $\text{Duration}(r) = \max_{ps \in PS} r(ps) + 1$.

Definition 5 (Makespan)**.** *The makespan of a PO plan P is the minimum possible duration over all valid parallel executions of P:* $\min_r \{\max_{ps \in PS} r(ps) + 1\}$.

The minimization in this definition is necessary because a single PO plan can admit multiple valid parallel executions whenever actions are unordered relative to one another. Conceptually, this minimum duration is equivalent to the plan's critical path length (its longest chain of dependent steps).

Example 1. *The plan in Figure 2 illustrates this with two unordered actions, a_1 and a_2. The lack of an ordering constraint permits multiple executions: a sequential one (in either order) or a concurrent one. Since makespan is the minimum possible duration, the plan's makespan is just 1.*

Partial Order Causal Link Plans

A partial partial–order causal link (POCL) plan P is a tuple $(PS, \prec, CL)$ with PS and $\prec$ defined as in PO plans, and $CL \subseteq PS \times \mathcal{F} \times PS$ is a finite set of causal links.

To encode the initial and goal states of a classical problem within a POCL plan, we must introduce two special plan steps: *init* and *goal*. The *init* step precedes all other steps in the partial ordering, while the *goal* step follows all other steps, establishing the boundaries of the plan execution. These special steps are associated with corresponding actions that have specific properties. The *init* action has no preconditions ($pre(init) = \emptyset$) and no delete effects ($del(init) = \emptyset$). Its add effects correspond exactly to the initial state ($add(init) = \mathcal{I}$). The *goal* action's preconditions represent the goal conditions ($pre(goal) = \mathcal{G}$), while having no effects ($add(goal) = \emptyset$ and $del(goal) = \emptyset$).

A causal link $(ps, f, ps') \in CL$ indicates that $f \in add(ps) \cap pre(ps')$: that is, the precondition f of ps' is to be fulfilled by an effect of ps. To ease definitions, we require that any causal link $(ps, f, ps') \in CL$ implies an ordering constraint $(ps, ps') \in \prec$. Then, the definitions for parallel execution and makespan in the context of PO plans apply directly to POCL plans. We call f *protected* by that link, as f will not be deleted between these steps in any derived solution. A plan step ps'' *threatens* a causal link (ps, f, ps') iff $f \in del(ps'')$ and the transitive closure of $\prec \cup \{(ps, ps''), (ps'', ps')\}$ is a strict partial order. A plan satisfying such a condition raises a *causal threat*.

Although peripheral to our primary focus, it is nonetheless useful to briefly examine how causal threats are resolved in POCL plans through promotion or demotion of plan steps (Younes and Simmons 2003). Suppose a plan step ps'' threatens a causal link (ps, f, ps'). Promotion enforces an ordering constraint $ps' \prec ps''$ to ensure that ps'' occurs after the consumer of the link, preventing ps'' from deleting f before ps' can use it. Similarly, demotion enforces an ordering constraint $ps'' \prec ps$ to ensure that ps'' occurs before the producer ps, so ps'' does not interfere with the causal link[1]. We can now define POCL plans.

Definition 6 (POCL plan)**.** *A partial POCL plan P is called a POCL plan to a classical planning problem if and only if every precondition of its plan steps is protected by a causal link and there are no causal threats.*

Therefore, a POCL plan is essentially a PO plan where the causal structure for plan validity is made explicit, ensuring that all precondition fulfilments are deliberate and protected.

With the framework in place, we now examine how makespan is preserved across different plan representations.

Makespan Relationships Among Plan Representations

In this section we investigate the time complexity of converting between plan representations. Along the hierarchy *Sequential* $\to$ *Parallel* $\to$ *PO* $\to$ *POCL*, conversion upward is makespan-preserving, but the converse does not hold.

We begin our investigations with conversions from parallel plans to PO and POCL plans.

Theorem 1. *Let $\bar{A} = A_1, \ldots, A_k$ be a parallel plan of n actions with makespan k. Then, there exists a PO plan P with makespan k that can be computed from $\bar{A}$ by adding exactly $\frac{n^2 - \sum_{i=1}^{k} |A_i|^2}{2}$ ordering constraints.*

Proof. A parallel plan can be converted into a PO plan by labelling each action as a step and adding ordering constraints between layers: for all $i < j$, constrain each $a_i \in A_i$ to precede each $b \in A_j$. This introduces exactly $\sum_{1 \leq i < j \leq k} |A_i||A_j| = \frac{n^2 - \sum_{i=1}^{k} |A_i|^2}{2}$ constraints. □

Bercher and Olz (2020) showed that obtaining a POCL plan from a PO plan with the same makespan is always possible if additional ordering constraints are able to be added.

[1] As explained by Weld (1994) in footnote 8, 'the names stem from the fact that demotion moves the threat lower in the temporal ordering, but promotion moves it higher.' We follow this original terminology. Some authors instead use the term *promotion* where we use *demotion*, and vice–versa (Bercher and Olz 2020; Olz and Bercher 2019; Bercher, Geier, and Biundo 2013). This stems from a perspective where 'the earlier, the better'.

Proposition 1 ((Bercher and Olz, 2020, Thm. 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.*

However, in some special cases, no additional ordering constraints are needed during the conversion. Interestingly, the PO plan constructed in the proof of Theorem 1 is threat-free, so no additional ordering constraints need to be added to yield a POCL plan. We prove this in the following lemma by adapting Algorithm 2 of Bercher and Olz (2020).

Lemma 1. *Let $P = (PS, \prec)$ be a PO plan obtained by layering a valid parallel plan over k timesteps as in the proof of Theorem 1 (so each step s sits in a unique layer $\ell(s)$, and $\ell(s) < \ell(t) \implies s \prec t$). Define a POCL plan $P' = (PS, \prec, CL)$ by, for every step c and every precondition $f \in pre(c)$, choosing a producer $p = \max\{\ell(q) \mid q \prec c, f \in add(q)\}$ and adding the single causal link (p, f, c) to CL. Then for every $(p, f, c) \in CL$ and every other step $t \in PS$ with $f \in del(t)$, we have either $t \prec p$ or $c \prec t$, so there are no causal threats and no additional ordering constraints are required.*

Proof. We show that threatening plan steps are not possible. Suppose by way of contradiction there were a step t with $f \in del(t)$ and $\ell(p) < \ell(t) < \ell(c)$. Then, in the original parallel plan, layer $\ell(p)$ would have produced f, layer $\ell(t)$ would have deleted it, and yet layer $\ell(c)$ would still require it. This contradicts plan validity, hence no such t exists. Therefore, any step t with $f \in del(t)$ must lie either in a layer $< \ell(p)$ (so $t \prec p$) or in a layer $> \ell(c)$ (so $c \prec t$). Therefore, no causal threats exist, and by layering we already have all required ordering constraints. □

Therefore, we can construct a POCL plan from a parallel plan with no ordering constraints beyond what was required for the PO plan. Based on this observation, we obtain:

Corollary 1. *Let $\bar{A} = A_1, \dots, A_k$ be a parallel plan of n actions with makespan k. Then there exists a POCL plan P that also has a makespan of k. Computing P requires exactly $\sum_{1 \le i < j \le k} |A_i||A_j| = \frac{n^2 - \sum_{i=1}^{k} |A_i|^2}{2}$ additional ordering constraints.*

We can now consider transformations going in the opposite direction, from 'higher' plans to 'lower' ones. Trivially, every POCL plan induces a PO plan with the same makespan, as any causal link (ps, f, ps') implies an ordering constraint $ps \prec ps'$.

Proposition 2. *Let $P = (PS, \prec, CL)$ be a POCL plan with makespan k. Then, $P' = (PS, \prec)$ is a PO plan with makespan k.*

A subtle issue arises in POCL-to-parallel conversion: POCL plan steps are unique labeled objects (l, a), so multiple steps with the same action a but different labels can execute simultaneously. However, parallel plans use *sets* of actions, which cannot contain duplicates.

Example 2 (Counting 1 to N)**.** *Consider $\mathcal{G} = \{g_1, \dots, g_n\}$, $\mathcal{I} = \emptyset$, and action $a_g = (\emptyset, \{g_1, \dots, g_n\}, \emptyset)$. A valid POCL plan can contain n distinct steps $(l_1, a_g), \dots, (l_n, a_g)$, each fulfilling a different goal, all executing in parallel (Figure 3).*

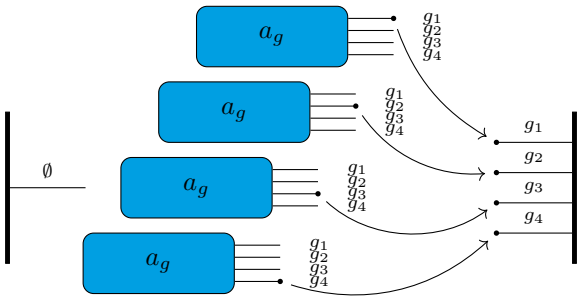

Figure 3: A POCL plan with four identical actions executing at once.

Naively serializing such steps destroys parallelism and inflates makespan. Instead, we observe that concurrent duplicate actions are functionally redundant.

Proposition 3 (Redundant Concurrent Steps)**.** *Let $P = (PS, \prec, CL)$ be a POCL plan with parallel execution r. If $S_{dup} \subseteq PS$ contains multiple steps $\{(l_1, a), (l_2, a), \dots\}$ with the same release time in r, then we can replace all steps in S_{dup} with a single representative step $ps^* = (l^*, a)$, rerouting all outgoing causal links to ps^*, yielding a valid POCL plan P' with the same parallel execution.*

Proof. Since all steps in S_{dup} execute concurrently with identical preconditions and effects, the representative ps^* satisfies the same preconditions, provides the same causal support (by rerouting links), and creates no new threats. Thus P' is valid. □

Thus, we can assume each action appears at most once per release time.

We now consider converting POCL plans to parallel plans. A natural approach is to group POCL plan steps into timesteps based on their optimal release times in a parallel execution. However, this direct conversion can fail to preserve makespan. We provide a counterexample POCL plan in Figure 2. While a_1 and a_2 are unordered and thus share the same optimal release time, they have inconsistent effects. a_1 adds p, while a_2 deletes p. This violates the non-interference criterion for parallel plans, and so a_1 and a_2 cannot execute in the same action set. Therefore, a_1 and a_2 *must* be serialized in any valid parallel plan, even though the POCL structure allowed them to be concurrent. This leads to the following observation:

Proposition 4. *There exist POCL plans which cannot be converted to a parallel plan with the same makespan.*

This result refutes Theorem 1 of Pecora, Rasconi, and Cesta (2004), which claims that PG-based planners 'maximize concurrency' for PO scheduling. Because parallel plans cannot represent plans like those in Figure 2 while maintaining makespan, PG-based planners do not guarantee makespan optimality for the general class of partially ordered plans. This has important implications for deployment

of state-based heuristics for PO/POCL planning (Bercher, Geier, and Biundo 2013; Sapena, Onaindia, and Torreño 2015).

For a heuristic to be useful in guaranteeing optimality, it must be *admissible*. A heuristic is admissible if it does not overestimate the cost of achieving the goal. Consider the h_p^m family of admissible heuristics for parallel planning (Haslum and Geffner 2000). We first observe that h_p^1 remains admissible for POCL planning. This heuristic computes the cost of achieving the most expensive *individual* goal fluent, assuming parallel execution. Since any valid plan must achieve a goal fluent, the total makespan must be at least the time required to achieve the most difficult *individual* fluent. Thus, h_p^1 provides a valid lower bound.

However, for $m \geq 2$, the heuristic accounts for interactions between subsets of up to m fluents using mutual exclusion reasoning based on non-interference constraints. In particular, h_p^2 corresponds to the Graphplan heuristic h_G, which identifies the earliest planning-graph layer at which all goal fluents in the set appear together without any mutual exclusions. In our example from Figure 2, $h_G(\{c, d\}) = 2$, yet the POCL plan achieves both goals within a single timestep. Therefore, h_p^2 overestimates the true makespan and is inadmissible for POCL plans. Since any h_p^m with $m \geq 2$ reduces to h_p^2 when the goal contains only two fluents, it immediately follows that all such h_p^m heuristics are also inadmissible for POCL planning:

Corollary 2. *For $m \geq 2$, the h_p^m heuristic is inadmissible for makespan-optimal PO/POCL planning.*

This finding directly impacts how optimality claims for partial-order planners should be interpreted. A prominent example is CPT (Vidal and Geffner 2006), which is presented as a makespan-optimal temporal POCL planner. CPT simultaneously employs h_p^2 as a heuristic and enforces the non-interference constraints of parallel planning at each time point. While this ensures optimality under those parallel-plan semantics, our results clarify that such semantics restrict the search space compared to standard POCL planning. For example, CPT could not represent the plan in Figure 2, even if a POCL-admissible heuristic were employed. Consequently, CPT's optimality guarantee does not extend to the general class of POCL plans.

Having established that a makespan-preserving conversion is not generally possible, the challenge becomes how to systematically resolve these conflicts. We propose a general procedure, outlined in Algorithm 1, which uses a graph-colouring technique to ensure all actions within each parallel timestep are non-interfering. The core idea is to maintain the original layering structure as much as possible, but to 'stretch out' any layer where actions would clash if run simultaneously. First, group plan steps by their release time t and build a conflict graph G_t whose vertices are those steps, inserting an edge where two actions have interfering effects. This isolates each independent time-slice of the original POCL schedule. Then, colouring each graph G_t assigns each action to a colour-class (i.e, to one parallel layer) such that no two conflicting actions share a layer. The number of colours χ_t determines how many layers are needed for

Algorithm 1: Convert a POCL plan into a parallel plan

Input: POCL plan $P = (PS, \prec, CL)$ with release-time schedule $r : PS \to \{0, \ldots, k-1\}$
Output: Parallel plan $(A_1, \ldots, A_m)$

1: // Step 1: Build and colour conflict graphs
2: **for** each $t = 0$ to $k-1$ **do**
3: $S_t \leftarrow \{ps \in PS \mid r(ps) = t\}$
4: Build conflict graph $G_t = (S_t, E_t)$ where $\{ps, ps'\} \in E_t$ iff ps and ps' interfere
5: Compute proper colouring of G_t into χ_t colors
6: Partition S_t into colour-classes $S_t^{(1)}, \ldots, S_t^{(\chi_t)}$
7: **end for**
8: // Step 2: Compute offsets for layering
9: $\mathsf{off}(0) \leftarrow 0$
10: **for** $t = 1$ to $k-1$ **do**
11: $\mathsf{off}(t) \leftarrow \sum_{j=0}^{t-1} \chi_j$
12: **end for**
13: // Step 3: Assign each colour-class to a parallel layer
14: **for** each $t = 0$ to $k-1$ **do**
15: **for** $c = 1$ to χ_t **do**
16: $A_{\mathsf{off}(t)+c} \leftarrow S_t^{(c)}$
17: **end for**
18: **end for**
19: **return** $(A_1, \ldots, A_{\sum_t \chi_t})$

block t. Finally, stitch together these layers in increasing order of t, using cumulative offsets. This preserves the original release-time ordering and ensures all causal and interference constraints are met.

The makespan of the resulting plan is $m = \sum_t \chi_t$. To minimize this makespan, Line 5 of Algorithm 1 requires finding an optimal colouring for each conflict graph G_t — i.e., using the minimum number of colours (the chromatic number $\chi(G_t)$). As graph coloring is NP–complete (Garey, Johnson, and Stockmeyer 1974), the choice of colouring procedure for Line 5 dictates the algorithm's overall complexity. If an optimal colouring algorithm like Lawler's (1976) is used, which runs in $\mathcal{O}(|E_t| \cdot |S_t| \cdot 2.445^{|S_t|})$ time, then the overall runtime is dominated by this exponential step. In the worst case, all n steps execute at $t = 0$, leading to the following bound:

Theorem 2. *The runtime of Algorithm 1, when using an exact colouring algorithm, is in $\mathcal{O}(C \cdot n \cdot 2.445^n)$, where C is the total number of interfering pairs.*

In practice, a polynomial-time greedy heuristic (Welsh and Powell 1967) can be used for Line 5. This makes Algorithm 1 polynomial-time, though the resulting makespan may not be minimal. We now provide makespan bounds for the parallel plan constructed by Algorithm 1, depending on the colouring strategy used. We first establish an upper bound that holds for any proper colouring.

Theorem 3. *Given a POCL plan with makespan k and C interfering pairs, algorithm 1 using any proper colouring constructs a parallel plan with makespan at most $k + C$.*

Proof. Suppose the makespan of the computed parallel plan is m. By Brooks' Theorem (Brooks 1941), $\chi(G_t) \leq \Delta(G_t) + 1$. Summing this over all k steps yields $m \leq k + \sum_t \Delta(G_t)$. Since the maximum degree $\Delta(G_t)$ cannot exceed the number of conflicts $|E_t|$ in that time step, the total makespan is bounded by $k + \sum_t |E_t| = k + C$. □

If an optimal colouring is used for each G_t in Algorithm 1, we can establish a tighter bound.

Theorem 4. *Given a POCL plan with makespan k and C interfering pairs, Algorithm 1 with optimal colouring constructs a parallel plan with makespan in $\mathcal{O}\left(k\sqrt{1+\frac{C}{k}}\right)$.*

Proof. For each conflict graph G, there must be at least one edge between every pair of colour classes, so $\chi(G)(\chi(G)-1) \leq 2|E|$, which gives $\chi(G) \leq \frac{1+\sqrt{1+8|E|}}{2}$. Thus, $m \leq \frac{k}{2} + \frac{1}{2}\sum_{t=0}^{k-1}\sqrt{1+8|E_t|}$. By Cauchy-Schwarz, $\sum_{t=0}^{k-1}\sqrt{1+8|E_t|} \leq \sqrt{k}\sqrt{k+8C} = k\sqrt{1+\frac{8C}{k}}$, where $C = \sum_t |E_t|$. Therefore

$$m \leq \frac{k}{2}\left(1+\sqrt{1+\frac{8C}{k}}\right) = \mathcal{O}\left(k\sqrt{1+\frac{C}{k}}\right). \quad \square$$

It is straightforward to see that the makespan depends on the structure of the conflict graphs G_t. In favourable cases, many steps can be executed in parallel; however, in the worst case, no parallelism is possible when each G_t is a clique.

Corollary 3. *In the worst case, a valid parallel plan computed by Algorithm 1 may require $|PS|$ layers; that is, the plan must be executed sequentially.*

While we provide one constructive method of finding a minimal-makespan parallel plan, we now show that the general problem of transforming a POCL plan is intractable.

Definition 7. *Let* POCL–PARALLELISATION *be the language consisting of all tuples (P, k) such that $P = (PS, \prec, CL)$ is a POCL plan, $k \in \mathbb{N}$ is a makespan bound, and there exists a partition $(A_1, \ldots, A_m)$ of PS, with $m \leq k$, satisfying: (1) for all $ps \in A_i$, every $ps' \prec ps$ appears in some A_j with $j < i$; (2) for all $ps \in A_i, i \geq r(ps)$ where r is the unique optimal parallel execution; and (3) for all $ps, ps' \in A_i$, ps and ps' are non-interfering.*

Theorem 5. POCL–PARALLELISATION *is NP–complete.*

Proof. Membership: For an instance (P, k), guess a partition $(A_1, \ldots, A_m)$. Verification can be done in polynomial time by checking that (1) all actions in P are assigned to exactly one A_i, (2) that $m \leq k$, (3) that no two actions in any A_i interfere, and (4) that all ordering constraints and release time constraints are respected.

Hardness: We reduce from GRAPH K–COLOURING (Karp 1972). Given $(G = (V, E), k)$, we construct an instance (P, k'). The plan P contains a step ps_v for each vertex $v \in V$. We introduce interference for each edge $(u, v) \in E$ by defining a unique fluent f_{uv} such that ps_u adds it and ps_v deletes it. Initially, all other preconditions and effects are empty, the plan has no ordering constraints, and all steps have an optimal release time of 0. Finally, we set the target makespan bound to $k' = k$. This transformation is polynomial in the size of (G, k). We now show that G is k-colourable iff (P, k') is in POCL–PARALLELISATION.

$\Longrightarrow$ Suppose G is k-colourable. Then there exists a function $c : V \to \{1, \ldots, k\}$ such that $c(u) \neq c(v)$ whenever $(u, v) \in E$. Define the layers $A_i = \{ps_v \mid c(v) = i\}$. Since c is a proper colouring, if $ps_u, ps_v \in A_i$, then u and v have the same colour, implying $(u, v) \notin E$. By our construction, this means ps_u and ps_v do not interfere. All actions $ps_v \in PS$ are assigned to one of these k timesteps. Thus, we have a valid parallel plan with makespan $k = k'$.

$\Longleftarrow$ Suppose $(P, k') \in$ POCL–PARALLELISATION. Then the partition $(A_1, \ldots, A_k)$ forms a proper k–colouring of G, by assigning each vertex v the colour i such that $ps_v \in A_i$. The conflict constraint ensures that no adjacent vertices share a colour. □

The foregoing analysis has primarily addressed conversions among non-sequential plan types. To provide a complete picture, we now explicitly consider their relationship with sequential plans. A sequential plan $\bar{a} = a_1, \ldots, a_n$ has makespan n, and can be trivially converted into a parallel plan $\bar{A} = \{a_1\}, \ldots, \{a_n\}$, where each action occupies its own timestamp. By Theorem 1 and Proposition 1, PO and POCL plans can be derived also with a makespan of n. Conversely, converting from a non-sequential plan to a sequential plan generally does not preserve makespan if the original plan exploited parallelism. See Figure 2 for an example.

Coming back to the makespan relationships between plans, we have established that makespan-preserving conversions 'upwards' require the addition of polynomially many ordering constraints. Unfortunately, conversion down the hierarchy while preserving makespan is only possible in the trivial POCL to PO case.

Complexity Results

The most basic decision problem is deciding whether a plan exists for a given planning problem $\Pi = (\mathcal{F}, \mathcal{A}, \mathcal{I}, \mathcal{G})$, without any restrictions on its makespan. As demonstrated in the previous section, conversion between each plan type can occur in polynomial time (although without necessarily preserving makespan). Therefore, the existence of one type of plan implies the existence of all of the others. For sequential plans, plan existence is PSPACE–complete (Bylander 1994), and so it is straightforward to see the following:

Proposition 5 (Equivalence of Unbounded Plan Existence)**.** *For any given classical planning problem $\Pi = (\mathcal{F}, \mathcal{A}, \mathcal{I}, \mathcal{G})$, the existence of a sequential, parallel, PO, or POCL plan is equivalent, and deciding any of them is PSPACE–complete.*

Makespan-bounded Plan Existence

The language $\{\langle \Pi, k\rangle \mid \Pi \text{ has a plan of makespan} \leq k\}$, where Π is a classical planning problem, defines the MAKESPAN-BOUNDED PLAN EXISTENCE (MBPE) problem. We assume a standard encoding function $\langle\cdot\rangle$ which maps objects to strings over $\{0, 1\}^*$. We first consider binary-encoded k. MBPE is only interesting for

non-sequential plans, as the sequential case is precisely length-bounded plan existence, which has already been shown to be PSPACE–complete (Bylander 1994). We leverage this result to show:

Theorem 6. MBPE *for parallel plans is PSPACE–complete.*

Proof. Membership: Let $\Pi = (\mathcal{F}, \mathcal{A}, \mathcal{I}, \mathcal{G})$. We adapt the Graphplan procedure (Blum and Furst 1997) to use only polynomial space. To avoid an exponential blowup from materializing the full planning graph, we only store two pieces of information at any time: the current time step t (requiring $O(\log k)$ bits, initialized to 0) and the set S of true atoms at the end of each time step t (initialized to $\mathcal{I}$).

The procedure performs the following steps iteratively.

1. **Check goal**: If $S \supseteq \mathcal{G}$, accept.
2. **Check bound**: If $t \geq k$, halt and reject, as the goal cannot be achieved in makespan k.
3. **Guess actions**: Nondeterministically guess a subset of actions $A_t \subseteq \mathcal{A}$ to execute in time t.
4. **Verify guess**: Check if the guessed set A_t is valid:
   (a) **Preconditions**: For each $a \in A_t$, $pre(a) \subseteq S$.
   (b) **Non-interfering**: Check that all pairs $a_i, a_j \in A_t$ are non-interfering.
   (c) If either check fails, reject this branch.
5. **Update state**: if A_t is valid:
   (a) Compute the effects: $Add_t = \bigcup_{a \in A_t} add(a)$ and $Del_t = \bigcup_{a \in A_t} del(a)$.
   (b) Update the set of true atoms: $S \leftarrow (S \setminus Del_t) \cup Add_t$.
   (c) Increment time step $t \leftarrow t + 1$.
   (d) Go back to step 1.

We now analyse the space requirements of the procedure. We store the current time step $t \leq k$, which requires $\mathcal{O}(\log t)$ space. Since t is incremented up to k, this is $\mathcal{O}(\log k)$ overall. Although k is exponential in $|\langle k \rangle|$, $\log k$ is polynomial in $|\langle k \rangle|$. Similarly, S is bounded by the number of ground atoms $|\mathcal{F}|$, so takes $\mathcal{O}(|\mathcal{F}|)$ space. Verification in step 4 and updating the state in step 5 can both be done in polynomial space and time relative to the input size. We additionally had to guess at most $\mathcal{O}(|\mathcal{A}|)$ actions at each iteration. The total space required by the procedure is $\mathcal{O}(\log k + |\mathcal{F}| + |\mathcal{A}|)$, which is polynomial in the size of the input. Therefore, the problem is in NPSPACE. By Savitch's Theorem (Savitch 1970), NPSPACE = PSPACE.

Hardness: We reduce from BOUNDED SEQUENTIAL PLAN EXISTENCE, which is known to be PSPACE–complete (Bylander 1994). Formally, the language is $\text{BPE} = \{\langle \Pi, k \rangle \mid \Pi \text{ has a sequential plan of length } \leq k\}$. The core idea is to modify the planning problem Π into a new problem Π' such that any valid parallel solution plan in Π' must effectively execute only one 'original' action per time step, forcing the makespan of the parallel plan P' in Π' to be equal to the length of the sequential plan P in Π.

Let $\langle \Pi, k \rangle$ be an instance of BPE, where $\Pi = (\mathcal{F}, \mathcal{A}, \mathcal{I}, \mathcal{G})$ and k is the length bound. We construct an instance $\langle \Pi', k \rangle$ of MBPE as follows: let $\Pi' = (\mathcal{F}', \mathcal{A}', \mathcal{I}', \mathcal{G}')$, where $\mathcal{F}' = \mathcal{F} \cup \{odd, even\}$, $\mathcal{I}' = \mathcal{I} \cup \{odd\}$ and $\mathcal{G}' = \mathcal{G}$. For each $a = (pre, add, del) \in \mathcal{A}$, create two new actions a_{odd} and a_{even} in $\mathcal{A}'$:

$$a_{odd} = (pre \cup \{odd\}, add \cup \{even\}, del \cup \{odd\})$$
$$a_{even} = (pre \cup \{even\}, add \cup \{odd\}, del \cup \{even\})$$

The makespan bound for Π' is the same k as in the sequential case. This transformation is polynomial in the size of Π.

We now show that Π has a sequential solution of length $\leq k$ iff Π' has a parallel solution of makespan $\leq k$.

$\implies$ Assume that there exists a sequential plan $\bar{a} = a_1, \ldots, a_n$ for Π with $n \leq k$ that is applicable to $\mathcal{I}$ and generates the goal description $\mathcal{G}$. This is trivially a parallel plan with makespan $t \leq k$.

$\impliedby$ Assume there exists a parallel plan $\bar{A} = A_1, \ldots, A_t$ for Π' with $t \leq k$, applicable to $\mathcal{I}'$ resulting in a superset of $\mathcal{G}'$. We claim that each step A'_i must contain exactly one action. To prove this, consider any two distinct actions $a'_i, a'_j \in \mathcal{A}'$ proposed for inclusion in the same timestep A'_j. There are three cases to check:

1. If $x = (a_x)_{odd}$ and $y = (a_y)_{odd}$: Both require odd as a precondition. However, $odd \in \text{del}((a_x)_{odd})$. Thus, $\text{pre}((a_y)_{odd}) \cap \text{del}((a_x)_{odd}) = \{odd\} \neq \emptyset$. The actions interfere.
2. If $x = (a_x)_{even}$ and $y = (a_y)_{even}$: Similar to case 1, both require $even$, and $even \in \text{del}((a_x)_{even})$, so they interfere.
3. If $x = (a_x)_{odd}$ and $y = (a_y)_{even}$: $\text{pre}((a_x)_{odd})$ includes odd, and $\text{pre}((a_y)_{even})$ includes $even$. By construction of actions in $\mathcal{A}'$, any state reachable from $\mathcal{I}'$ contains either odd or $even$, but never both (as one is deleted when the other is added). Therefore, $(a_x)_{odd}$ and $(a_y)_{even}$ cannot have their preconditions simultaneously satisfied in any valid state, so they cannot be co-applicable in A'_j.

Therefore, for A'_j to be a valid non-interfering and co-applicable set of actions, it must contain at most one action. Since a plan should make progress, each A'_j must contain exactly one action $a'_j \in \mathcal{A}'$. So, $\bar{A} = \{a'_1\}, \{a'_2\}, \ldots, \{a'_t\}$. The initial state $\mathcal{I}'$ contains odd, so a'_1 must be of the form $(a_1)_{odd}$. After its execution, the state will contain $even$, so a'_2 must be of the form $(a_2)_{even}$, and so on. The turn-taking fluents odd, $even$ ensure that the "type" of action (odd or even) alternates correctly.

We can construct a sequential plan $\bar{a} = a_1, a_2, \ldots, a_t$ for Π by taking a_j to be the original action from $\mathcal{A}$ from which a'_j was derived (i.e., strip off the $_{odd}$ or $_{even}$ suffix and the turn-taking fluents from its definition).

- **Length**: The length of $\bar{a}$ is t. Since the makespan of $\bar{A}$ was $t \leq k$, the length bound for $\bar{a}$ is satisfied.
- **Applicability**: Let $S_0 = \mathcal{I}$ and $S'_0 = \mathcal{I}'$. Let S_j be the state after a_j in $\bar{a}$, and S'_j be the state after $A'_j = \{a'_j\}$ in $\bar{A}$. We now show that if a'_j is applicable in S'_{j-1} and $\mathcal{F}$-fluents in S'_{j-1} match S_{j-1}, then a_j is applicable in S_{j-1}. The preconditions of a'_j are $\text{pre}(a_j)$ plus either odd or $even$. Since a'_j is applicable in S'_{j-1}, $\text{pre}(a_j) \subseteq S'_{j-1}$.

The $\mathcal{F}$-fluents in S'_{j-1} (i.e., $S'_{j-1} \cap \mathcal{F}$) are identical to the fluents in S_{j-1} (the state in Π before a_j). Thus, $\text{pre}(a_j) \subseteq S_{j-1}$, so a_j is applicable. The effects on $\mathcal{F}$-fluents are also preserved.

- **Goal**: The final state S'_t of $\bar{A}$ satisfies $\mathcal{G}' = \mathcal{G}$. Since the operations on $\mathcal{F}$-fluents in Π' mirror those in Π, the final state S_t of $\bar{a}$ satisfies $\mathcal{G}$.

Therefore, $\bar{a}$ is a valid sequential plan for Π of length $t \leq k$. Since the reduction is polynomial, and BPE is PSPACE–complete, MBPE for parallel plans is PSPACE–hard.

In conclusion, the problem is PSPACE–complete. □

While this result in conjunction with Theorem 1 and Corollary 1 gives us PSPACE–hardness for PO and POCL plans, Proposition 4 prevents us from claiming membership. However, PSPACE membership for PO (and thus POCL, by Proposition 1) plans can be established by adapting the membership proof of Theorem 6. The core modification restricts non-interference checks to *relevant effects* only. Let $E = \bigcup_{a \in \mathcal{A}} (add(a) \cup del(a))$ be the set of all effects. The set of relevant effects $R \subseteq E$ is the smallest set containing all $e \in \mathcal{G}$, and closed under: if $e \in add(a) \cup del(a)$ for some $a \in \mathcal{A}$ with $pre(a) \cap R \neq \emptyset$, then $e \in R$. The key insight is that conflicts arising from irrelevant effects (effects not in R) can be safely ignored, as they do not create causal dependencies that increase makespan. Consequently, our decision procedure need only enforce non-interference among effects in R. Since $|R| \leq |\mathcal{F}|$, computing R requires only polynomial space. The adapted decision procedure thus runs in space $\mathcal{O}(\log k + 2|\mathcal{F}| + |\mathcal{A}|)$, establishing PSPACE membership. We therefore obtain the following corollary:

Corollary 4. MBPE *is PSPACE–complete for classical planning problems regardless of whether plans are represented as parallel plans, PO plans, or POCL plans.*

Makespan-bounded Plan Existence with Unary-encoded Makespan

Our complexity analysis has so far assumed the makespan bound k is encoded in binary. The preceding PSPACE–hardness result is a direct consequence of this choice, as it allows the value of k to be exponential relative to the overall input size. It is therefore interesting to determine if this encoding is the sole source of hardness, or if the problem remains difficult even for polynomially bounded makespans. We investigate this by analyzing the complexity when k is encoded in unary. This analysis is also practically motivated, reflecting the incremental search strategy of systems like SAT-based planners. It is known that BPE is NP–complete when k is encoded in unary (Bäckström and Jonsson 2011). We extend this result to MBPE for parallel plans by adapting the reduction used in the binary setting.

Theorem 7. MBPE *for parallel plans is NP–complete if the makespan bound k is encoded in unary.*

Proof. Membership: We refer to the nondeterministic decision procedure in Theorem 6 for establishing membership in PSPACE. The algorithm guesses an action set A for each timestep t, verifies its validity, and updates the state. Each iteration takes time polynomial in the size of the planning problem Π, and there are at most k such iterations. As k is encoded in unary, we have $|\langle k \rangle| = k$, so its value is bounded by the input size. Specifically the input size is at least $|\Pi| + k$. Therefore, the total runtime of the nondeterministic algorithm is $k \cdot \text{poly}(|\Pi|)$, which is polynomial in the total input size. Therefore the problem is in NP.

Hardness: We reduce from BPE with unary length bound, which is known to be NP–complete (Bäckström and Jonsson 2011). Given an instance $\langle \Pi, k \rangle$, where $\Pi = (\mathcal{F}, \mathcal{A}, \mathcal{I}, \mathcal{G})$ and k is a unary encoded length bound, the problem is to decide if a sequential plan of length $\leq k$ exists for Π.

We use the same polynomial-time reduction construction as in the PSPACE–hardness proof of Theorem 6. This reduction transforms Π into a new planning problem $\Pi' = (\mathcal{F}', \mathcal{A}', \mathcal{I}', \mathcal{G}')$ designed to force parallel plans to be sequential. The makespan bound k' for Π' is set equal to k. If k is given in unary, then k' (when considered as a value) is also represented by this unary length.

As shown in Theorem 6, Π has a sequential solution of length $\leq k$ if and only if Π' has a parallel solution with makespan $\leq k' = k$. Since BPE with unary length bound is NP–hard and the reduction is polynomial, MBPE for parallel plans with a unary-encoded makespan bound k is also NP–hard. Hence, the problem is NP–complete. □

For PO and POCL plans, we obtain NP–hardness from Theorems 1 and 7, and Corollary 1. NP membership follows directly by guessing and verifying a POCL plan of at most $k \cdot |\mathcal{A}|$ plan steps. As this plan is polynomial in the size of the input, both guessing and verification require only polynomial time. We hence obtain the following corollary:

Corollary 5. MBPE *is NP–complete for classical planning problems when the makespan bound k is encoded in unary, regardless of whether plans are represented as parallel plans, PO plans, or POCL plans.*

Conclusion

This paper proves that common plan representations are not interchangeable for makespan optimization. We establish a one-way relationship in their expressive power and show that the cost of bridging this gap — finding an optimal parallel equivalent for a partial-order plan — is NP-complete. These results refute the claim by Pecora, Rasconi, and Cesta (2004) that planning graph-based planners maximize concurrency, and demonstrate that heuristics admissible for parallel planning, such as h^m_p (Haslum and Geffner 2000), are inadmissible for PO/POCL search. We also establish tight complexity bounds for makespan-bounded plan existence: PSPACE-complete for binary encoding and NP-complete for unary encoding. Since unit-duration actions are a special case of durative actions (Fox and Long 2003), our nonequivalence result—that POCL plans can achieve shorter makespans than parallel plans—applies directly to temporal planning. Ultimately, our work provides a formal characterization of these trade-offs, demonstrating that the choice of representation has significant, now-quantified consequences for plan quality and computational effort.

Acknowledgments

Pascal Bercher is the recipient of an Australian Research Council (ARC) Discovery Early Career Researcher Award (DECRA), project number DE240101245, funded by the Australian Government.

References

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

Bercher, P.; Haslum, P.; and Muise, C. 2024. A Survey on Plan Optimization. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence (IJCAI 2024)*, 7941–7950. IJCAI Organization.

Bercher, P.; Lin, S.; and Alford, R. 2022. Tight Bounds for Hybrid Planning. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI 2022)*, 4597–4605. IJCAI Organization.

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

Bit-Monnot, A. 2023. Experimenting with Lifted Plan-Space Planning as Scheduling: Aries in the 2023 IPC. In *IPC 2023 – Proceedings of the Hierarchical Task Network (HTN) Track of the 11th International Planning Competition: Planner and Domain Abstracts*, 7–9.

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

Blum, A. L.; and Furst, M. L. 1997. Fast Planning through Planning Graph Analysis. *Artificial Intelligence*, 90(1): 281–300.

Brooks, R. L. 1941. On Colouring the Nodes of a Network. *Mathematical Proceedings of the Cambridge Philosophical Society*, 37(2): 194–197.

Bylander, T. 1994. The Computational Complexity of Propositional STRIPS Planning. *Artificial Intelligence*, 69(1): 165–204.

Bäckström, C.; and Jonsson, P. 2011. All PSPACE-Complete Planning Problems Are Equal but Some Are More Equal than Others. In *Proceedings of the 4th Annual Symposium on Combinatorial Search (SoCS 2011)*, 10–17. AAAI Press.

Cavrel, N.; Pellier, D.; and Fiorino, H. 2023. Efficient HTN to STRIPS Encodings for Concurrent Planning. In *2023 IEEE 35th International Conference on Tools with Artificial Intelligence (ICTAI 2023)*, 962–969. IEEE.

Firsov, O.; Fiorino, H.; and Pellier, D. 2023. OptiPlan – a CSP-based partial order HTN planner. In *IPC 2023 – Proceedings of the Hierarchical Task Network (HTN) Track of the 11th International Planning Competition: Planner and Domain Abstracts*, 10–11.

Fox, M.; and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research*, 20: 61–124.

Garey, M. R.; Johnson, D. S.; and Stockmeyer, L. 1974. Some Simplified NP-complete Problems. In *Proceedings of the 6th Annual ACM Symposium on Theory of Computing (STOC 1974)*, 47–63. Association for Computing Machinery.

Haslum, P.; and Geffner, H. 2000. Admissible Heuristics for Optimal Planning. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems (AIPS 2000)*, 140–149. AAAI Press.

Karp, R. M. 1972. Reducibility among Combinatorial Problems. In *Complexity of Computer Computations*, 85–103. Springer US.

Lawler, E. L. 1976. A Note on the Complexity of the Chromatic Number Problem. *Information Processing Letters*, 5(3): 66–67.

Lotem, A.; Nau, D. S.; and Hendler, J. A. 1999. Using Planning Graphs for Solving HTN Planning Problems. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI 1999)*, 534–540. AAAI Press.

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.

Pecora, F.; Rasconi, R.; and Cesta, A. 2004. Assessing the Bias of Classical Planning Strategies on Makespan-Optimizing Scheduling. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)*, 677–681. IOS Press.

Rintanen, J. 2012. Planning as Satisfiability: Heuristics. *Artificial Intelligence*, 193: 45–86.

Sapena, O.; Onaindia, E.; and Torreño, A. 2015. FLAP: Applying Least-Commitment in Forward-Chaining Planning. *AI Communications*, 28(1): 5–20.

Savitch, W. J. 1970. Relationships between Nondeterministic and Deterministic Tape Complexities. *Journal of Computer and System Sciences*, 4(2): 177–192.

Vidal, V.; and Geffner, H. 2006. Branching and Pruning: An Optimal Temporal POCL Planner Based on Constraint Programming. *Artificial Intelligence*, 170(3): 298–335.

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

Welsh, D. J. A.; and Powell, M. B. 1967. An Upper Bound for the Chromatic Number of a Graph and Its Application to Timetabling Problems. *The Computer Journal*, 10(1): 85–86.

Younes, H. L.; and Simmons, R. G. 2003. VHPOP: Versatile Heuristic Partial Order Planner. *Journal of Artificial Intelligence Research*, 20: 405–430.