Extending Hierarchical Partial-Order Causal-Link Planning to Temporal Problem Solving

Nicolas Cavrel, Humbert Fiorino, Damien Pellier

Extending Hierarchical Partial-Order Causal-Link Planning to Temporal Problem Solving

1st Nicolas Cavrel
Univ. Grenoble Alpes, LIG
Grenoble, France
Nicolas.Cavrel@univ-grenoble-alpes.fr

2nd Humbert Fiorino
Univ. Grenoble Alpes, LIG
Grenoble, France
Humbert.Fiorino@imag.fr

3rd Damien Pellier
Univ. Grenoble Alpes, LIG
Grenoble, France
Damien.Pellier@imag.fr

***Abstract*—This paper introduces TEP (Temporal Event Planning), a novel POCL (Partial Ordered Causal Link) HTN approach tailored for temporal Hierarchical Task Networks (HTNs). TEP stands out for its capability to address the three categories of temporal problems as categorized by Cushing, distinguishing it from prevailing approaches that primarily tackle the initial two categories. TEP accomplishes this by translating temporal HTN problems into non-temporal counterparts and excels in performance by leveraging heuristics developed in non-temporal HTN planning to steer the search for temporal solutions. This innovative strategy initiates by refining tasks into instantaneous actions and relaxing duration constraints, ensuring alignment with conventional search heuristics while upholding the problem's innate expressiveness.**

***Index Terms*—AI Planning, HTN, Temporal Planning**

I. Introduction

The Hierarchical Task Network (HTN) planning formalism is designed to represent planning problems as sets of complex tasks. Inline with the classical divide and conquer strategy, these problems are solved by recursively breaking down complex tasks into simpler ones until the problem consists solely of indecomposable tasks, referred to as "actions."

While most HTN approaches primarily focus on non-temporal HTN planning, e.g., [1]–[4]where actions are considered instantaneous, we delve into temporal HTN planning, where actions are defined over specific time intervals. Considering temporal planning is essential when addressing real-world problems, where time constraints play a pivotal role, and synchronization among agents is imperative. On a formal level, characterizing actions as durative introduces the possibility of defining novel types of constraints on tasks, such as Allen's interval constraints. These constraints are instrumental in representing temporal-specific limitations, like the partial overlap of two actions or a set of requirements that must be true throughout the entire duration of actions. All these constraints significantly enhance the expressiveness of planning problems and enable dealing with more realistic scenarios.

The various approaches proposed for solving temporal planning problems can be categorized by their capability to address different categories of temporal problems as defined by Cushing [5]. Cushing's classification distinguishes three categories of temporal planning problems. In the first category, temporal problems have solutions consisting in sequential plans: action concurrency is not required (non-concurrent solutions). The second category encompasses temporal problems where solution plans can potentially be concurrent, but sequential solutions are still valid (possibly concurrent solutions). The third category comprises temporal problems where the only feasible solution plans are inherently concurrent, and sequential solutions are not possible (necessary concurrent solutions).

On one hand, most of the literature approaches fall into Cushing's first two categories. This is the case of [6]–[8] where temporal actions are preprocessed into sequences of non-temporal tasks [6] or actions. One important advantage of these approaches is their ability to leverage search heuristics and algorithms developed within a non-temporal framework. On the other hand, some approaches such as [9]–[11] do solve all the Cushing's categories, but they lack efficiency due to a dearth of informative heuristics.

In this paper, we introduce TEP (Temporal Event Planning), a novel POCL (Partial Ordered Causal Link) HTN planning approach encompassing Partial Order Causal Link and HTN methods, tailored for temporal HTN domains. Our contribution lies in TEP's ability to solve all of Cushing's categories for HTN problems, while utilizing heuristics developed for non-temporal HTN planning. TEP starts by compiling temporal HTN problems into non-temporal ones. It accomplishes this by refining abstract tasks and durative tasks into instantaneous non-temporal actions. Subsequently, it attempts to find a solution plan by relaxing the duration constraints, solely by checking constraint consistency. The resulting non-temporal relaxed problem closely resembles a classical HTN problem, making it compatible with non-temporal search heuristics. Importantly, it maintains the required expressiveness to tackle problems across all of Cushing's categories. If a solution to the relaxed problem is found, TEP proceeds by computing timestamp assignments that satisfy the temporal constraints with a CSP (Constraint Satisfaction Problem) solver.

The paper is structured as follows. Section 1 introduces the temporal HTN planning formalism. In Section 2, the TEP algorithm is presented. Section 3 demonstrates how TEP solves HTN problems within Cushing's third category. Section 4 examines in detail the heuristics adopted from non-temporal planning and employed in TEP to address temporal HTN planning. Finally, Section 5 evaluates TEP performance.

II. Problem Statement

In this section we propose a formalization that incorporates temporal features in HTN planning. The notations are based on [12] and [1].

A. Tasks, Task Network, Action and Methods

The key concepts in HTN planning and a fortiori in temporal HTN planning are the *tasks* and *task networks*.

A *task* is given by a name and a list of parameters. We distinguish three kinds of tasks: snap tasks, durative tasks and abstract tasks. Unlike *snap tasks* that do change the states of the world, *durative tasks* and *abstract tasks* do not. They are names referring to other tasks (either snap, durative or abstract) that must be achieved with respect to some constraints. Every task has a start and an end time point. We refer to the start and end time points of a task t with temporal variables denoted respectively v_t^s and v_t^e. A snap task t is instantaneous, i.e. $v_t^s = v_t^e$, thus we will simply refer to the time point of the snap task t as v_t. As usually in planning, a state s is defined as a set of ground atoms.

A *task network* represents a partially ordered multi-set of tasks. A *task network* w is a tuple $(I, \alpha, \prec)$ where I is a set of task identifiers, $\alpha : I \mapsto T$ a mapping from task identifiers to task names T, and $\prec$ is a set of ordering constraints over the task identifiers in I. Ordering constraints are defined over the start or the end time variables of the task identifiers I. The possible ordering constraints are those from the classical point algebra: $<, \leq, >, \geq, =$ and $\neq$. We allow also the constraints of the form $d = v_t^e - v_t^s$ to express task duration. For instance, the temporal ordering constraint $v_{t_1}^s < v_{t_2}^e$ expresses that the start of the task $t_1 \in I$ must occur strictly before the end of $t_2 \in I$. Note that such a set of constraints C can be represented as a constraint graph whose consistency can be checked by computing its strongly connected components in polynomial time $O(|C|)$. Snap, durative and abstract tasks can be achieved respectively by applying *snap actions*, *durative actions* and *methods* defined below.

A *snap action* a is a tuple $(name(a), pre(a), post(a), eff(a))$, where $name(a)$ is the name of the task achieved by a and $pre(a)$, $post(a)$ and $eff(a)$ are sets of ground atoms. Let v_a be the time point at which a is executed. A snap action has **two** sets of conditions to satisfy to be executed: $pre(a)$ that must hold strictly before v_a (classical case in planning) and $post(a)$ that must hold strictly after v_a, i.e., in the state resulting of the execution of a. $post(a)$ **is a novelty to express invariant properties that must be satisfied over an interval of time**, and that will be used to solve Cushing's 3rd category.

Finally, as in non-temporal planning, the execution of a produces the effects $eff(a)$ such that $eff(a) = eff^+(a) \cup eff^-(a)$ and $eff^+(a) \cap eff^-(a) = \emptyset$ where $eff^+(a)$ and $eff^-(a)$ are sets of ground atoms, respectively true and false after the execution of a.

A *durative action* a is a tuple $(name(a), start(a), end(a), inv(a), d)$: $name(a)$ is the task achieved by a, $start(a)$ and $end(a)$ are snap actions ; $inv(a)$ is a set of ground atoms that must hold after the execution of $start(a)$ and until the beginning of $end(a)$, i.e., on the interval $]v_a^s, v_a^e[$ and $d = v_a^e - v_a^s$ is the duration of a. We assume as in temporal planning [13] that $v_a^s < v_a^e$ is true. Therefore the duration of a is a strictly positive number. The timeline of a durative action application is given in Figure 1.

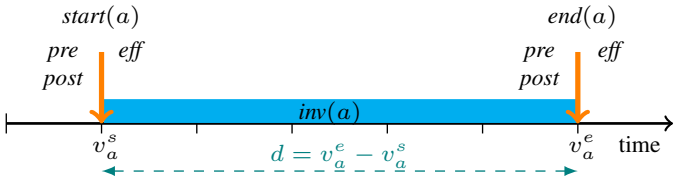

Fig. 1. Timeline of a durative action a application.

A *method* m is a tuple $(name(m), w)$, where $name(m)$ is the name of the task achieved by m and w the task network that defines how $name(m)$ must be decomposed into sub-tasks.

B. Partial Temporal Plans, Flaws and Threats

To deal with temporal features, we extend the definition of a *partial plan* usually used in POCL HTN planning [1] and in POCL planning [14] in order to handle the *postconditions* of the snap actions as follows.

A *partial temporal plan* π is a tuple (w, C) where $w = (I, \alpha, \prec)$ is a task network that defines the tasks of π and its ordering constraints, and C is a set of causal links of the form $\langle t_i \xrightarrow{p} t_j \rangle$ with t_i and t_j two identifiers of snap tasks in I, and p a ground atom.

The purpose of a causal link is to assert that a precondition or a postcondition p of a task t_j is supported by another task t_i that produces p as effect. Two cases have to be considered depending on whether t_i supports a precondition or a postcondition of t_j. If t_i supports a precondition of t_j, i.e., $p \in eff(t_i)$ and $p \in pre(t_j)$ then $(v_{t_i}^e < v_{t_j}^s) \in \prec$ (classical case in POCL planning). If t_i supports a postcondition of t_j, i.e., $p \in eff(t_i)$ and $p \in post(t_j)$ then $(v_{t_i}^e \leq v_{t_j}^s) \in \prec$: in this case, **the ordering constraint is not strict**.

By extension, a snap task t_k in a partial temporal plan π is a *threat* on a causal link $\langle t_i \xrightarrow{p} t_j \rangle$ depending on whether t_i supports a precondition or a postcondition p of t_j. If t_i supports a precondition p of t_j, then t_k threats $\langle t_i \xrightarrow{p} t_j \rangle$ if and only if t_k has $\neg p$ as effect, and $\{(v_{t_i}^e < v_{t_k}^s), (v_{t_k}^e < v_{t_j}^s)\}$ are consistent with $\prec$ (the usual case in POCL). If t_i supports a postcondition p of t_j, then t_k threats $\langle t_i \xrightarrow{p} t_j \rangle$ if and only if t_k has $\neg p$ as effect, and $\{(v_{t_i}^e < v_{t_k}^s), (v_{t_k}^e \leq v_{t_j}^s)\}$ are in $\prec$. Here the ordering constraint between t_k and t_j is not strict.

A *flaw* in a partial plan $\pi = (w, C)$ with $w = (I, \alpha, \prec)$ is either (1) an open condition, i.e., a precondition or a postcondition of a task that is not supported by a causal link, (2) a threat, i.e., a task that may interfere with a causal link, or (3) a decomposition flaw, i.e., an abstract or a durative task that is not decomposed into snap tasks.

C. Temporal HTN Problem and Solution

A ground *temporal HTN planning problem* P is a tuple $(L, T, A, M, s_0, w_0, g)$, where L is a finite set of ground atoms, T is a set of tasks, A is a set of durative actions, M is the

set of methods, s_0 is the initial state built over L, w_0 is the initial task network to carry out, and g is a set of ground atoms defining the goal to reach.

The solution of a temporal HTN planning problem is a partial temporal plan π obtained by refining an initial partial plan π_0 as in POCL planning into snap tasks by applying durative actions and methods. The initial plan π_0 is built with w_0 as task network and two special snap tasks t_0 and t_∞ ordered respectively as the first task and the last task of w_0. t_0 has no precondition and the initial state as positive effects. t_∞ has the goal as precondition and no effects. Formally, a partial temporal plan $\pi = (w, C)$ is solution of a planning problem $P = (L, T, A, M, s_0, w_0, g)$ if and only if: (1) π is a refinement of the initial partial temporal plan π_0, (2) π is executable in the initial state s_0. Thus, (2.1) all the tasks in π are snap tasks, (2.2) π has no flaws, i.e., no open precondition or postcondition, and no threats, (2.3) for all t in π, v_t is assigned and satisfies the ordering constraints in w.

It remains to define how to refine a partial temporal plan into a plan containing only snap tasks by using durative actions and methods. First, consider the case of the refinement of a durative task. To carry out this refinement it is first necessary to slightly modify the definition of the two snap actions *start*(a) and *end*(a) to translate the invariant properties *inv*(a) into the POCL logic. This modification consists in, on the one hand, adding *inv*(a) to the postconditions and the effects of *start*(a), and, in the other hand, to the preconditions of *end*(a). It is now possible to express the invariant properties of a durative task by classical causal links between the effects of *start*(a) and the preconditions of *end*(a) in accordance with PDDL 2.1 semantics, which constrain *inv*(a) to be checked on the interval $]v_a^s, v_a^e[$.

Formally, let $a = (\textit{name}(a), \textit{start}(a), \textit{end}(a), \textit{inv}(a), d)$ be a durative action that refines a task identifier i of a plan $\pi_1 = (w_1, C_1)$ with $w_1 = (I_1, \alpha_1, \prec_1)$ such that $i \in I_1$. Then, a refines π_1 into a plan $\pi_2 = (w_2, C_2)$ with $w_2 = (I_2, \alpha_2, \prec_2)$ and

$$
\begin{aligned}
I_2 &= (I_1 - \{i\}) \cup \{i^s, i^e\} \\
\alpha_2 &= (\alpha_1 - (i, \textit{name}(a))) \cup (i^s, \textit{start}(a)) \cup (i^e, \textit{end}(a)) \\
\prec_2 &= \{v_i^s < v_i^e\} \cup \{v_i^e - v_i^s = d\} \\
&\quad \cup \{(v_i' < v_i^s) \mid \forall (v_i' < v_i) \in \prec_1\} \\
&\quad \cup \{(v_i' \leq v_i^s) \mid \forall (v_i' \leq v_i) \in \prec_1\} \\
&\quad \cup \{(v_i^e < v_i') \mid \forall (v_i < v_i') \in \prec_1\} \\
&\quad \cup \{(v_i^e \leq v_i') \mid \forall (v_i \leq v_i') \in \prec_1\}
\end{aligned}
\tag{1}
$$

Finally, the last case is method refinement. Let $m = (t_m, w_m)$ be a method with a task network $w_m = (I_m, \alpha_m, \prec_m)$ that refines a task identifier i of a plan $\pi_1 = (w_1, C_1)$ with $w_1 = (I_1, \alpha_1, \prec_1)$ such that $i \in I_1$. Then, m refines π_1 into a plan $\pi_2 = (w_2, C_2)$ with $w_2 = (I_2, \alpha_2, \prec_2)$ and

$$
\begin{aligned}
I_2 &= (I_1 - \{i\}) \cup I_m \\
\alpha_2 &= \alpha_1 \cup \alpha_m - \{(i, t_m)\} \\
\prec_2 &= \prec_m \ \cup \{(v_i' < v_i^s) \mid \forall (v_i' < v_i) \in \prec_1\} \\
&\quad \cup \{(v_i' \leq v_i^s) \mid \forall (v_i' \leq v_i) \in \prec_1\} \\
&\quad \cup \{(v_i^e > v_i') \mid \forall (v_i > v_i') \in \prec_1\} \\
&\quad \cup \{(v_i^e \geq v_i') \mid \forall (v_i \geq v_i') \in \prec_1\} \\
&\quad \cup \{(v_i' \geq v_i^s) \mid v_i' \in I_m)\} \ \cup \{(v_i' \leq v_i^e) \mid v_i' \in I_m)\} \\
&\quad \cup \{(v_i^s \leq v_i^e)\} \\
C_2 &= C_1
\end{aligned}
\tag{2}
$$

III. Temporal Event Planner

In this section, we define the search procedure of TEP (*Temporal Planning Event*). This procedure is built upon the POCL planning procedure proposed by [1]. It deals with the two sets of conditions that must hold to execute an action: preconditions and postconditions. We show how TEP is able to use the heuristics developed for non-temporal HTN planning and how it can solve all the Cushing's categories [5] of temporal problems. It is important to emphasize that the problems of Cushing's third category are problems whose solution plans necessarily require concurrency. To our best knowledge, TEP is the only HTN planner capable of solving this Cushing's category.

A. Search Procedure

The key idea of TEP is to carry out the search for a solution plan by interleaving two steps. The first step refines an initial partial plan into a plan containing only snap tasks as in POCL HTN planning while checking the consistency of the task ordering constraints. Unlike POCL HTN planning, TEP checks both the consistency of $<$ and $\leq$ constraints. This first step boils down to solve a non-temporal problem, allowing the search to be guided by classical heuristics. We will detail this point in the next section. The aim of TPE first step is to verify the criteria 1, 2.1 and 2.2 of a solution plan. The second step assigns a value to all timestamps in the snap tasks of the refined plan with respect to the ordering constraints as well as the duration constraints $d = v_a^e - v_a^s$[1]

The whole TPE search procedure is given in Algo. 1. It takes as input a problem $P = (L, T, A, M, s_0, w_0, g)$ and returns a solution plan π as well as a set V of timestamp assignments for each snap task if the procedure succeeds, and failure otherwise. TPE starts (line 1) by expressing the initial state s_0, the initial task network w_0 and the goal g as a partial plan to refine π_0. π_0 has w_0 as task network and two special snap tasks t_0 and t_∞ ordered respectively as the first task and the last task of w_0. t_0 has no precondition and the initial state as positive effect. t_∞ has the goal as precondition and no effects. Then (line 2) π_0 is pushed to the pending list of partial plans to be explored

[1]For reasons of simplicity, we limit ourselves here to this type of constraints, bearing in mind that the CSP approach can be used to express more complex constraints.

Algorithm 1: TEP$(L, T, A, M, s_0, w_0, g)$

```
1  π_0 ← the inital plan built from s_0, w_0 and g
2  open ← {π_0}
3  while open ≠ ∅ do
4      π ← non-deterministically select plan in open
5      flaws ← the set of flaws of π
6      if flaws = ∅ then
7          V ← search timestamp assignments for π
8          if V ≠ ∅ then return (π, V)
9          else continue
10     φ ← arbitrarily select a flaw in flaws
11     open ← open ∪ solveFlaw(π, φ)
12 return Failure;
```

and the first step of refinements starts. At each iteration a new partial plan π is selected (line 4) and its flaws are computed (line 5). If π contains flaws, one of them is selected (line 10). Solving this flaw generates a new set of partial temporal plans that are added to the pending list of partial plans to explore (line 11). If a flawless plan is selected (line 7), TPE gets into the second step, and uses a CSP solver to find the set V of timestamp assignments for all the snap tasks of π. If the CSP solver succeeds, π and V are returned, otherwise the procedure keeps iterating in step one, and tries to refine a new partial plan.

B. Repairing Flaws

The flaws in partial plans are very similar to the flaws in non-temporal Task Networks. We have 3 different types of flaws: (1) open conditions, i.e., preconditions or postconditions of tasks that are not supported by a causal link, (2) threats, i.e. tasks that may interfere with a causal link, or (3) decomposition flaws, i.e., abstract or durative tasks that are not still decomposed into snap tasks. Flaws are repaired as follows.

a) Open conditions: We distinguish two repair mechanisms for open conditions depending on whether an open condition p of a task $t_j \in \pi$ is a precondition or a postcondition (see Figure 2). If p is a *precondition*, then p is repaired by adding to π a *causal link* $\langle t_i \xrightarrow{p} t_j \rangle$ and an ordering constraint $(t_i < t_j)$. If p is a *postcondition*, then p is repaired by adding to π a *causal link* $\langle t_i \xrightarrow{p} t_j \rangle$ and an ordering constraint $(t_i \leq t_j)$. In both cases, the added constraints must be consistent with current ordering constraints in π. Note that the repair of open preconditions is done as in non-temporal POCL HTN planning whereas the repair of open postconditions introduce $\leq$ constraints instead of $<$.

b) Threats: Suppose that a task t_k threatens a causal link $\langle t_i \xrightarrow{p} t_j \rangle$. This threat can be solved in two different ways (see Figure 3). The first one is to constrain t_k strictly before the task t_i that produces p by adding $(t_k < t_i)$ in the ordering constraints. The second one is to constrain t_k after or at the same time of task t_j by adding the constraint $(t_j \leq t_k)$ if p is a precondition of t_j, or strictly after if p is a postcondition of t_j. As always, adding ordering constraints to π must ensure

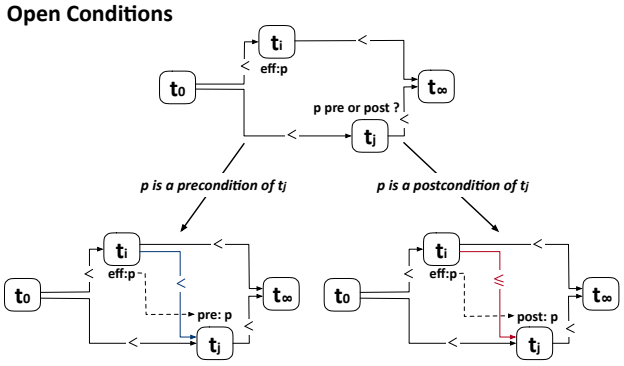

Fig. 2. Open condition repair: A causal link $\langle t_i \xrightarrow{p} t_j \rangle$ is added to the plan to support the open condition, and a precedence constraint $(t_i < t_j)$ or $(t_i \leq t_j)$ depending on whether the open condition is a *precondition* or a *postcondition*.

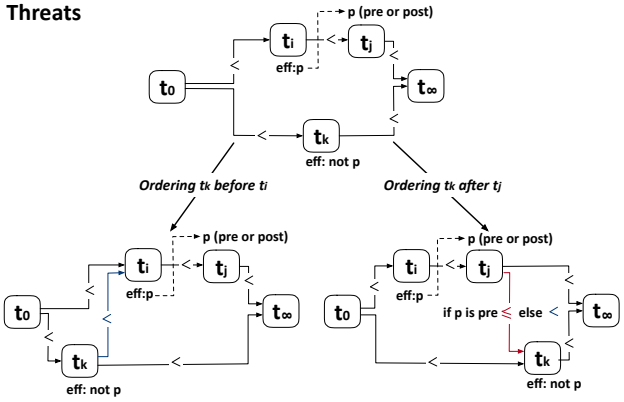

Fig. 3. There are two ways of resolving a *threat* t_k on a causal link $\langle t_i \xrightarrow{p} t_j \rangle$, one can either order the threatening task strictly before t_i or at the same time (or after) the task t_j depending on whether the supported open condition is a precondition or a postcondition.

constraint consistency. The repair of a threat is very similar to threat repair in POCL HTN planning except we allow non-strict ordering constraints.

c) Decomposition Flaws: We distinguish two different repair mechanisms depending on whether $\pi = (w, C)$ contains a durative or an abstract task t to decompose by applying Equation 1 and 2 respectively. Figure 4 shows an example of durative task decomposition (right) and abstract task decomposition (left). For the durative task decomposition, we assume that t is decomposed by a durative action a such that $t_s = \mathit{start}(a)$ and $t_e = \mathit{end}(a)$. We show in red the modifications of the two snap tasks t_s and t_e needed to translate the invariant properties $inv(a)$ into the POCL logic. For the abstract task decomposition, we assume that t is decomposed by a method m into three subtasks t_i, t_j, t_k with $t_i < t_k$. No other constraints bind t_i, t_j and t_k.

C. An Example of Cushing's third category

In this section, we illustrate how TPE deals with the third category of Cushing's classification where solution plans are necessary concurrent.

Suppose that two durative tasks t_a and t_b can be decomposed by two durative actions a and b. The timelines of a and

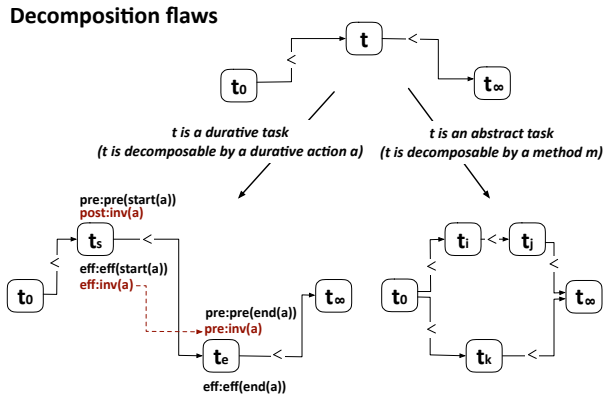

Fig. 4. The two possible ways of decomposing a task depending on whether the task is durative (right) or abstract (left).

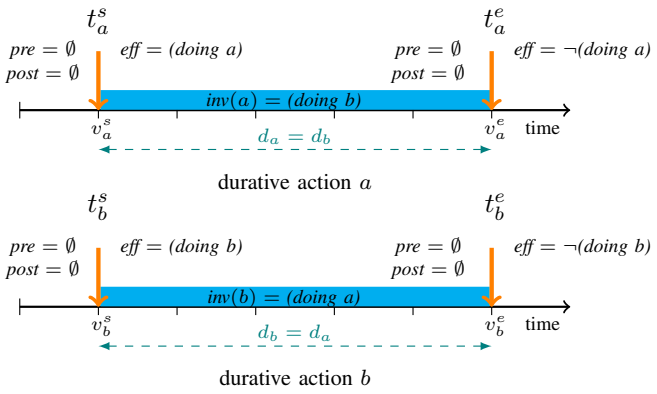

Fig. 5. The two durative actions of the Cushing's problem

b are given in Figure 5: a and b have the same duration d. Action a produces the ground atom *(doing a)* and has *(doing b)* as invariant, and symmetrically, b produces the ground atom *(doing b)* and has *(doing a)* as invariant. The problem has no methods, and the initial state and the goal state of the problem are empty. The initial task network of the problem contains the tasks t_a and t_b with no ordering constraints. The partial plan obtained by TEP after decomposing t_a and t_b into snap tasks by applying the durative action a and b, is depicted in Figure 6. The only solution to this problem is a concurrent plan where t_a and t_b are executed at the same time.

Figure 6 depicts the partial plan obtained after decomposing t_a and t_b with a and b. It has four flaws. Two flaws are open conditions that must be supported by causal links: the postcondition *(doing a)* of the task t_a^s and the postcondition *(doing b)* of the task t_b^s. The two other flaws are threats: the task t_a^e threatens the causal link $\langle t_b^s \overset{(doing\ a)}{\longrightarrow} t_b^e \rangle$ and the task t_b^e threatens the causal link $\langle t_a^s \overset{(doing\ b)}{\longrightarrow} t_a^e \rangle$.

Assume that the open conditions are selected and resolved first. To fix these two flaws, TEP must add to the partial plan two causal links, the first one $\langle t_b^s \overset{(doing\ b)}{\longrightarrow} t_a^s \rangle$ to support the postcondition *(doing b)* of t_a^s and the second one $\langle t_a^s \overset{(doing\ a)}{\longrightarrow} t_b^s \rangle$ to support the postcondition *(doing a)* of t_b^s. In addition to these two causal links, TPE must add two ordering

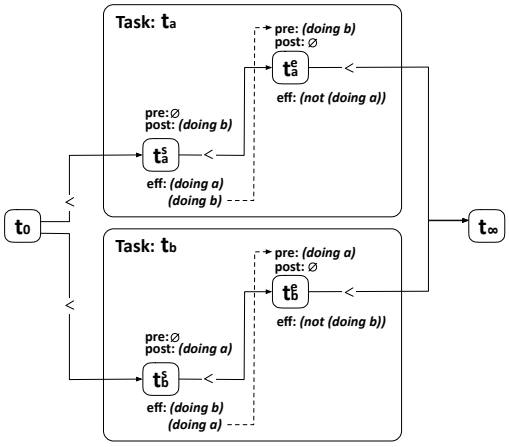

Fig. 6. Partial plan obtained in the Cushing's problem after decomposing the two durative tasks t_a and t_b by applying the durative actions a and b. The dashed arrows represent causal links and the plain black arrows the ordering constraints

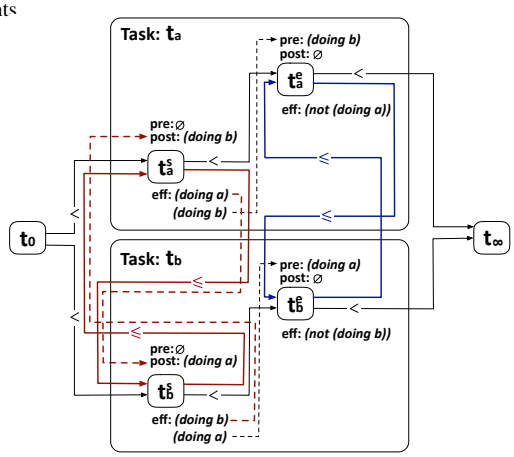

Fig. 7. The partial plan corresponding to the Cushing's problem after the resolution of the two open conditions and the two treats. Causal links and ordering constraints in red depict the modification of the partial plan needed to fix the open conditions, and in blue to fix the threats.

constraints indicating that t_a^s must be executed before or at the same time as t_b^s since t_a^s now supports a postcondition of t_b^s, and conversely, that t_b^s must be executed before or at the same time as t_a^s since t_b^s now supports a postcondition of t_a^s. The resulting partial plan is depicted in Figure 7. The causal links and the ordering constraints added are shown in red.

Finally, the threats have to be fixed. In the general case, a threat is resolved by constraining the threatening snap task either *strictly* before the threatened causal link, or *strictly or at the same time* after the causal link depending on whether the causal link support a precondition or a postcondition. In our example, the only valid resolution is to constrain t_b^e (resp. t_a^e) after or at the same time as t_a^e (resp. t_b^e). The resulting partial plan is displayed on Figure 7. The ordering constraints added are shown in blue. All the flaws are now resolved. TEP returns the partial plan in Figure 7 as solution.

IV. TEP Heuristics

The main benefit of TEP is to exploit (with some adaptations presented below) the heuristics developed for non-temporal

POCL HTN planning to solve temporal problems. TEP search procedure (See Algo. 1) relies on two non deterministic choices, which are in practice achieved by using two heuristic functions. The first choice performs a non deterministic choice over the set of pending partial temporal plans to explore, and decides which one to explore first (line 4). The heuristic functions guiding this choice are called *plan selection heuristics* and greatly impact both the search performance (i.e., the time required to find a solution plan) and the quality of the returned plan (i.e., the number of actions in the solution plan).

The second choice (line 9) performs a non deterministic choice over the flaws to be solved in the current partial plan. The heuristic functions guiding this choice are called *flaw selection heuristics*. Note that every flaw in the partial plan has eventually to be solved in order to find a solution plan. Hence, the *admissibility* of the TEP procedure only depends on plan selection heuristics. However, the *order* in which the flaws are resolved with respect to the flaw selection heuristics greatly impacts the search performance of the procedure. In the following, we present plan and flaw selection heuristics as implemented in TPE.

A. Plan Selection Heuristics

Among plan selection heuristics in non-temporal POCL HTN planning, e.g., [1], [15], the most effective are those developed by [1]. Their principle is to estimate the number of changes required to achieve a plan solution by counting the number of flaws in the current partial plan. Bercher's idea was to extend this principle to POCL HTN planning by estimating the number of flaws in a plan, more precisely by taking into account the flaws that will be introduced by tasks that have not yet been decomposed. This estimation is based on a structure called Task Decomposition Graphs (TDG) that encodes the decomposition of the hierarchical problem. In the following section, we present (1) how TEP extends the concept of TDG to encode not only abstract task decompositions but also durative task decompositions in a new structure called Temporal Task Decomposition Graphs (TTDG), and (2) how classical heuristics for POCL HTN planning developed in [1], [16] can be derived from a TTDG to solve temporal planning problems.

a) Temporal Task Decomposition Graphs: A TTDG of a planning problem $P = (L, T, A, M, s_0, w_0, g)$ is a directed AND/OR graph $G = (V_T, V_M, V_A, E)$, where V_T is a set of vertices consisting of snap, durative, and abstract tasks that can be obtained by decomposing the initial partial plan π_0 built from s_0, w_0, and g. V_M and V_A are respectively sets of method vertices or durative action vertices that decompose an abstract task or a durative task within V_T. Finally, E is a set of edges that link nodes from V_T to V_M or V_A. For brevity, the child nodes of a node v is $child(v) = \{v_i | (v, v_i) \in E\}$. A TTDG is illustrated in Figure 8. In the general case, a TTDG like a TDG can be a cyclic graph.

b) Heuristics: To generalize the different plan selection heuristics developed in POCL HTN planning to temporal planning, we first need to define two crucial estimates that

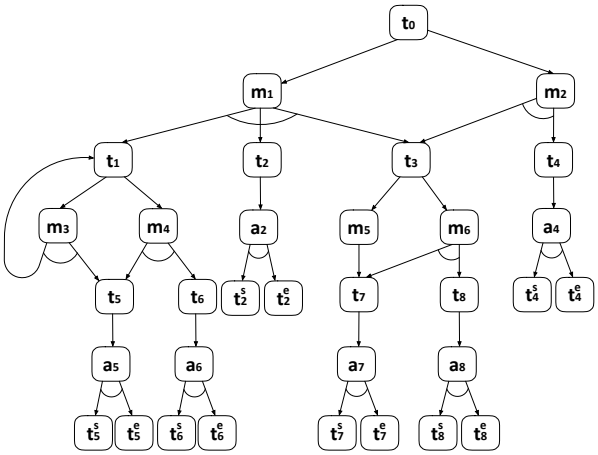

Fig. 8. A simple TTDG example depicted as a AND/OR graph. The symbols t_0, t_1, t_3 represent abstract tasks. $t_2, t_4, t_5, t_6, t_7, t_8$ represent durative tasks and $t_2^s, t_2^e, t_4^s, t_4^e, \ldots$ snap tasks. $m_1, \ldots, m_6$ depict method vertices and $a_5, \ldots, a_8$ durative action vertices.

can be extracted from a TTDG: (1) an estimate of the number of decompositions required to break down the plan into snap tasks, and (2) an estimate of the open conditions to be supported.

Both of these estimates rely on the concepts of mandatory tasks and task cardinality. Mandatory tasks, denoted as $M(t)$, are ground tasks that appear in all decomposition methods or durative actions associated with the same task t. In simpler terms, the set of mandatory tasks $M(t)$ includes tasks that will unquestionably be included in a partial plan when task t is decomposed. For example (refer to Figure 8 for context), we have $M(t_o) = M(t_3) = M(t_7) = \{t_7^s, t_7^e\}$, $M(t_1) = M(t_5) = \{t_5^s, t_5^e\}$, $M(t_4) = \{t_4^s, t_4^e\}$, $M(t_5) = \{t_5^s, t_5^e\}$, $M(t_6) = \{t_6^s, t_6^e\}$, and $M(t_8) = \{t_8^s, t_8^e\}$. The task cardinality ($TC$) of a task t can be straightforwardly defined as the number of tasks in its mandatory set, i.e., $TC = |M(t)|$. This task cardinality serves as a lower bound for estimating the effort required to decompose task t.Computing the first estimate comes down to calculating TC. In practice, it is determined from a TTDG as follows:

$$\textit{TC}(v) = \begin{cases} 1 & \text{if } v \in V_T \text{ and } v \text{ is snap} \\ \min\limits_{v_i \in child(v)} TC(v_i) & \text{if } v \in V_T \text{ and } v \text{ is abst.} \\ \sum\limits_{v_i \in child(v)} TC(v_i) & \text{otherwise} \end{cases}$$

The second estimate, known as *MME* (Minimal Modification Effort), represents the number of open conditions, including both preconditions and postconditions, that must be satisfied for each node. It can be determined from a TTDG as follows:

$$\textit{MME}(v) = \begin{cases} 1 + |pre(v) \cup post(v)| & \text{if } v \in V_T \text{ and } v \text{ is snap} \\ \sum\limits_{d \in Child} TC(d) & \text{if } v \in V_M. \\ 1 + \min\limits_{d \in Child} TC(d) & \text{if } v \in V_T \text{ and } v \text{ is abst.} \end{cases}$$

where $pre(v)$ denotes the preconditions of a primitive node and $post(v)$ its postconditions.

Finally, we can expand the application of the four classical heuristics proposed in [1], [16] to temporal plans $\pi = (w, C)$ as follows. Here, $w = (I, \alpha, \prec)$ defines the tasks and their associated constraints, while C represents the causal links between the tasks within π, and F denotes its flaws.

$$h_{TC}(\pi) = \Sigma_{i \in I | \alpha(i) \ is \ abtract} TC(\alpha(i))$$
$$h_{MME}(\pi) = \Sigma_{i \in I} MME(\alpha(i))$$
$$h_{TDGm}(\pi) = h_{MME}(\pi) - |C|$$
$$h_{TC \ + \ \#F}(\pi) = h_{TC}(\pi) + |F|$$

Note that all the heuristics presented are admissible when considering the number of modifications that will be introduced into the plan, as mentioned in [16]. However, they do not guarantee an optimal solution plan with respect to the makespan.

B. Flaw selection heuristics

The most common strategy for selecting the order of flaws is to choose the flaws with the fewest possible resolvers first, meaning that the most constrained flaws are resolved first. This strategy is known as LCFR (Least Cost Flaw Repair) [17]. While this flaw selection heuristics was originally developed for non-hierarchical problems, it has been used in POCL HTN planning by the state-of-the-art planner PANDA [1]. When adapting it to hierarchical planning, priority is given to abstract task decomposition over other flaws to maintain the completeness of the procedure. This is because the number of resolvers for a flaw depends on the choice of decomposition.

TEP implements the same flaw selection strategy as PANDA, giving priority to decomposition flaws and then prioritizing the remaining flaws starting from the ones with the fewest resolvers. The only difference here is that decomposition flaws include not only abstract tasks but also durative tasks that are not decomposed into snap tasks.

V. Experimentation

In this section we will compare TEP and its heuristics with the best-known and efficient *timeline* approach proposed by FAPE. We choose the best FAPE configuration [11]. The timeline approach is currently the sole approach for solving hierarchical temporal problems in the third category of Cushing. Thus, to our best knowledge, only the timeline approach and TPE share the same expressiveness.

A. Experimental Setup

Both planners, TEP and FAPE, were tested on a single core of an Intel Core i7-9850H CPU, with a 8GB RAM limit and a time limit of 600 seconds. To ensure a fair comparison, both planners were implemented by using the same library PDDL4J [18], and utilized the same CSP solver, which is the one provided by OR-Tools [19]. The assessment criteria are (1) the *solving time*, which represents the time spent to solve a problem, (2) the *makespan* of a solution plan, which represents the overall length of the plan, meaning the time between the initial task and the final task endpoints of a solution plan, and, finally, (3) the *coverage* of each planner, meaning the number of solved problems of each domain. Note that solving time and makespan are presented as IPC (International Planning Competition) scores[2].

B. Planning benchmarks

There are currently no standard benchmarks available for hierarchical and temporal planning. Hence, we propose a set of benchmarks based on the HDDL2.1 language [12], which aims at facilitating the comparison of both hierarchical and temporal planners. Among these benchmarks, some are adaptations of hierarchical non-temporal domains of the IPC planning competitions. They have been modified by adding durations to actions: *Gripper*, *Satellite*, and *Rover*. In the *Gripper* domain, all solutions are sequential, and no temporal concurrency is required (Cushing's first category). In *Satellite* and *Rover* domains, problems with multiple satellites or rovers allow concurrency, but it is not mandatory. The planner can choose to find either a simple non-concurrent plan, or a concurrent one using multiple rovers. These problems belong to Cushing's second category. Additionally, some domains[3] have been devised specifically to correspond to the Cushing's third category:

- *Area Scan* models a set of heterogeneous devices that cooperate to scan areas. Area Scan has two versions: a totally ordered version, where the abstract tasks of the problems are totally ordered, and an unordered version where concurrency can also be achieved through concurrent abstract tasks.
- *Table Carriers* is inspired by the *Gripper* domain. In this domain, agents must cooperate to carry tables, with tables requiring either one, two, or three persons to simultaneously carry them successfully.

VI. Related Work

The various approaches to temporal planning can be classified according to their ability to deal with Cushing's categories [5]. This classification can be applied to classical planning, e.g., [20], [21], or to non-hierarchical planning. These categories outline three types of temporal problems: those exclusively allowing sequential solutions, problems permitting both sequential and concurrent solutions, and those demanding only concurrent solutions without any option for sequential planning. Many planners can solve problems falling into the first two categories through diverse techniques. One popular approach involves adapting non-temporal HTN planners to manage temporal formalisms. For instance, PYHIPOP [22] or SHOP (Simple Hierarchical Ordered Planner). [23] was expanded to SHOP2 [6], [8] to incorporate actions with durations. Further advancements, like GSCCB-SHOP2 [24], handle intricate time and resource constraints. Another case is SIADEX [7], which applies a forward-chaining method akin to classical planning, tailored to a temporal context.

[2]https://ipc2023-htn.github.io/.

[3]For each domain, a set of problems was generated using a problem generator and ranked in terms of difficulty using TEP with the MME heuristics.

	Gripper			Satellite			Rover			AreaScan PO			AreaScan TO			TableCarriers			Total		
	Time	Span	Cov.	Time	Span	Cov.	Time	Span	Cov.	S. Time	Span	Cov.	Time	Span	Cov.	Time	Span	Cov.	S. Time	Span	Cov.
TEP(TC)	6.72	**7.00**	**7/10**	6.21	6.50	**8/9**	4.96	5.43	**10/10**	19.99	**21.83**	**22/22**	19.93	**21.97**	**22/22**	7.13	8.33	9/10	64.94	71.05	**78/83**
TEP(MME)	**6.81**	**7.00**	**7/10**	**7.09**	6.71	**8/9**	6.62	7.41	9/10	17.64	21.62	**22/22**	20.25	21.74	**22/22**	**8.57**	9.34	**10/10**	**66.99**	73.82	**78/83**
TEP(TDGm)	4.09	6.00	6/10	5.24	7.38	**8/9**	**8.56**	7.63	9/10	18.67	21.62	**22/22**	**21.71**	21.74	**22/22**	4.64	**9.77**	**10/10**	62.90	**74.14**	77/83
TEP(TC + #F)	4.79	6.00	6/10	6.71	**7.63**	**8/9**	6.67	7.26	**10/10**	**20.39**	21.62	**22/22**	21.48	21.77	**22/22**	3.67	8.61	9/10	63.70	72.88	77/83
FAPE	1.52	6.00	6/10	2.74	5.88	6/9	3.64	**7.99**	8/10	10.00	18.92	19/22	13.69	21.72	**22/22**	2.73	7.00	7/10	34.31	67.51	68/83

TABLE I

Experimentation results for the different configurations in the form of IPC scores for both time, makespan, and coverage. The best configuration achieves an IPC score of 1, with other configurations receiving a percentage based on their performance relative to the best one. To compute the score of a domain, we sum the IPC scores obtained for each problem. Coverage refers to the number of problems solved within a domain.

It establishes meta-temporal actions that link sequentially to craft a solution plan. Additionally, among planners addressing the first two categories is the CHIMP planner [25]. CHIMP translates temporal and hierarchical problems into constraint satisfaction problems to derive temporal solution plans. While these planners benefit from the efficiency of non-temporal planning, their methodologies struggle to fully address the complexity of temporal planning, making them inadequate for problems involving concurrency and particularly solving third category temporal problems as defined by Cushing. Planners adept at addressing Cushing's third category typically utilize timeline-based methods, monitoring the status of each proposition over time. Key examples include HSTS [26], later refined into EUROPA [27], a CSP-based planner. Additionally, IxTeT [28], a significant approach, is domain-independent and proficient in managing intricate timelines. Its more recent iteration, FAPE [11] and ARIES [29], adeptly manages the complete range of necessary concurrency expressions.

VII. Conclusion

In this paper, we have presented TEP, an approach to represent and solve both hierarchical and temporal problems. It consists in relaxing temporal problems in non-temporal problems so as to apply HTN search heuristics to them. We have compared our approach with a timeline approach sharing the same expressiveness in terms of Cushing's categories. We have shown that our approach outperforms it in terms of runtime to find a solution, and that both approaches are comparable with respect to makespan. One way to improve the TEP approach would be to implement makespan-aware heurisitics into TEP, in order to improve the quality of the solution plans.

References

[1] P. Bercher, S. Keen, and S. Biundo, "Hybrid Planning Heuristics Based on Task Decomposition Graphs," in *SOCS*, 2014, pp. 35–43.

[2] D. Nau, Y. Bansod, S. Patra, M. Roberts, and R. Li, "Gtpyhop: A hierarchical goal+task planner implemented in python," in *HPlan Workshop (ICAPS)*, 2021.

[3] N. Cavrel, D. Pellier, and H. Fiorino, "Efficient HTN to STRIPS encodings for concurrent planning," in *ICTAI*, 2023, pp. 962–969.

[4] A. Ramoul, D. Pellier, H. Fiorino, and S. Pesty, "Grounding of HTN planning domain," *Int. J. Artif. Intell. Tools*, vol. 26, no. 5, 2017.

[5] W. Cushing, "Evaluating Temporal Planning Domains," in *ICAPS*, 2007, pp. 105–112.

[6] T.-C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. S. Nau, D. Wu, and F. Yaman, "Shop2: An htn planning system," *J. Artif. Intell. Res.*, vol. 20, pp. 379–404, 2003.

[7] M. Asunción, L. Castillo, J. Fdez-Olivares, García-Pérez, A. González-Muñoz, and F. Palao, "Siadex: An interactive knowledge-based planner for decision support in forest fire fighting," *AI Commun.*, vol. 18, pp. 257–268, 01 2005.

[8] R. Goldman, "Durative planning in htns," in *ICAPS*, 2006, pp. 382–385.

[9] H. L. S. Younes and R. G. Simmons, "VHPOP: versatile heuristic partial order planner," *J. Artif. Intell. Res.*, vol. 20, pp. 405–430, 2003.

[10] P. Bechon, M. Barbier, G. Infantes, C. Lesire, and V. Vidal, "Hipop: Hierarchical partial-order planning," in *STAIRS*, 2014, pp. 51–60.

[11] A. Bit-Monnot, M. Ghallab, F. Ingrand, and D. E. Smith, "FAPE: a Constraint-based Planner for Generative and Hierarchical Temporal Planning," *ArXiv, abs/2010.13121.*, 2020.

[12] D. Pellier, A. Albore, H. Fiorino, and R. Bailon-Ruiz, "HDDL 2.1: Towards Defining an HTN Formalism and Semantics with Time," 2023, HPLlan Workshop (ICAPS).

[13] M. Fox and D. Long, "PDDL2.1: an extension to PDDL for expressing temporal planning domains," *J. Artif. Intell. Res.*, vol. 20, pp. 61–124, 2003.

[14] D. A. McAllester and D. Rosenblitt, "Systematic nonlinear planning," in *AAAI*, 1991, pp. 634–639.

[15] B. Schattenberg, J. Bidot, and S. Biundo, "On the construction and evaluation of flexible plan-refinement strategies," in *German Conference on AI*, 2007, pp. 367–381.

[16] P. Bercher, G. Behnke, D. Höller, and S. Biundo, "An Admissible HTN Planning Heuristic," in *IJCAI*, 2017, pp. 480–488.

[17] D. Joslin and M. E. Pollack, "Least-cost flaw repair: A plan refinement strategy for partial-order planning," in *AAAI*, 1994, pp. 1004–1009.

[18] D. Pellier and H. Fiorino, "PDDL4J: a planning domain description library for Java," *J. Exp. Theor. Artif. Intell.*, vol. 30, no. 1, pp. 143–176, 2018.

[19] L. Perron, F. Didier, and S. Gay, "The cp-sat-lp solver," in *CP*, 2023, pp. 3:1–3:2.

[20] J. Benton, A. J. Coles, and A. Coles, "Temporal planning with preferences and time-dependent continuous costs," in *ICAPS*, 2012.

[21] S. J. Celorrio, A. Jonsson, and H. Palacios, "Temporal planning with required concurrency using classical planning," in *ICAPS*, 2015, pp. 129–137.

[22] C. Lesire and A. Albore, "PYHIPOP-Hierarchical Partial-Order Planner," in *International Planning Competition*, 2021.

[23] D. S. Nau, T. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman, "SHOP2: an HTN planning system," *J. Artif. Intell. Res.*, vol. 20, pp. 379–404, 2003.

[24] C. Qi, D. Wang, H. Muñoz-Avila, and P. Zhao, "Hierarchical task network planning with resources and temporal constraints," *Knowledge-Based Systems*, vol. 133, pp. 17–32, 2017.

[25] S. Stock, M. Mansouri, F. Pecora, and J. Hertzberg, "Online task merging with a hierarchical hybrid task planner for mobile service robots," in *IROS*, 2015, pp. 6459–6464.

[26] N. Muscettola, "HSTS: Integrating Planning and Scheduling," Robotics Institute, Camegie Mellon University, Tech. Rep., 2013.

[27] J. Barreiro, M. Boyce, M. Do, J. Frank, M. Iatauro, T. Kichkaylo, P. Morris, J. Ong, E. Remolina, T. Smith *et al.*, "Europa: A platform for ai planning, scheduling, constraint programming, and optimization," 2012.

[28] S. Lemai, "IXTET-EXEC: planning, plan repair and execution control with time and resource management," Ph.D. dissertation, INP Toulouse, 2004.

[29] A. Bit-Monnot, "Experimenting with Lifted Plan-Space Planning as Scheduling: Aries in the 2023 IPC," in *International Planning Competition*, 2023.