On Guiding Search in HTN Temporal Planning with non Temporal Heuristics

Nicolas Cavrel, Damien Pellier, Humbert Fiorino
Univ. Grenoble Alpes - LIG
Grenoble, France
{nicolas.cavrel, damien.pellier, humber.fiorino}@univ-grenoble-alpes.fr

Abstract

The Hierarchical Task Network (HTN) formalism is used to express a wide variety of planning problems as task decompositions, and many techniques have been proposed to solve them. However, few works have been done on temporal HTN. This is partly due to the lack of a formal and consensual definition of what a temporal hierarchical planning problem is as well as the difficulty to develop heuristics in this context. In response to these inconveniences, we propose in this paper a new general POCL (Partial Order Causal Link) approach to represent and solve a temporal HTN problem by using existing heuristics developed to solve non temporal problems. We show experimentally that this approach is performant and can outperform the existing ones.

Introduction

Among planning formalisms, Hierarchical Task Network (HTN) planning is one of the most expressive. In addition to classical STRIPS actions, HTN allows to express complex abstract tasks in the form of decompositions into subtasks and their ordering constraints. HTN has been used in a wide variety of applications (Barreiro et al. 2012; Lallement, de Silva, and Alami 2018; Milot et al. 2021). However, despite the pioneering work of (Lemai 2004; Au et al. 2003; Goldman 2006), few approaches have been proposed to deal with time in HTN planning.

Temporal HTN approaches can be classified according to the expressiveness of the solution plans they can generate in the sense of Cushing's classification of temporal problems (Cushing 2007). This classification defines three categories of temporal planning problems. The first category contains the temporal problems whose solutions are solely sequential solution plans (non-concurrent solution plans). The second one contains the temporal problems whose solution plans can possibly be concurrent, but for which there exists a sequential solution plan (possibly concurrent solution). Finally, the third one includes the temporal problems for which the only existing solution plans are necessarily concurrent (necessary concurrent solution).

Most of the approaches are able to solve only temporal problems of the first two Cushing's categories. Among these approaches, we distinguish state-space approaches (Au et al. 2003; Asuncion et al. 2005). These approaches deal with temporal actions as classical ones by either preprocessing the temporal actions into a sequence of classical tasks (Au et al. 2003), or by compiling temporal actions into classical actions. The latter approach is also widely used in STRIPS planning (Fox and Long 2003; Celorrio, Jonsson, and Palacios 2015). Approaches converting temporal problems into classical ones can benefit from the classical search heuristics and algorithms. At the same time, several plan-space approaches have been proposed (Younes and Simmons 2003; Bechon et al. 2014; Bit-Monnot et al. 2020). They aim at refining an initial HTN into a solution by refining tasks and building causal relations between them, and use *STN* (Simple Temporal Networks) (Dechter, Meiri, and Pearl 1991) to deal with temporal constraints. These approaches are relatively efficient but suffer from their lack of flexibility, especially when dealing with real problems, where tasks are necessarily concurrent.

To our best knowledge, the only approach capable of producing expressive plans of the third Cushing's category is an algorithm in plan-spaces that handles constraints using *Chronicles* (Bit-Monnot et al. 2020). Chronicle approaches keep tracks of the value of each proposition with regards to time into a *timeline*, then try to find non conflicting timelines for every proposition. These approaches are very expressive but suffer from their lack of efficiency due to the lack of informative heuristics.

In this paper, we propose a new planning approach for hierarchical temporal planning able to solve planning problems for the two first Cushing's categories, called HTEP (*Hierarchical Temporal Event Planner*). HTEP is based on POCL (*Partial Ordered Causal Link*) (Bercher, Keen, and Biundo 2014). The particularity of our approach is to be both flexible (HTP produces partial temporal solution plans) and to be able to exploit the heuristics developed for non temporal HTN planning for plan and flaw selection. To do so, HTEP starts by compiling a temporal problem into a non temporal one by refining abstract tasks and durative tasks into instantaneous non temporal actions. Then, it tries to find a solution plan by relaxing the duration constraints on tasks and by checking only their consistency. The relaxed non-temporal problem is expressed as a partial plan in the POCL semantics. Hence classical POCL search algorithms and heuristics can be used to solve it. Finally, if a solution to the relaxed problem is found, HTEP tries to find timestamp assignments matching the temporal constraints by using a simple CSP solver.

The paper is organized as follows. Section 1 introduces the temporal planning formalism. In the second section we present our planner, from the relaxed problem to the search algorithm solving it, with the heuristic used to guide it. Finally, the third section shows the performance results of HTEP.

Problem Statement

In this section we propose a formalization of a temporal HTN planning problem and its solution. The notations are based on (Höller et al. 2020) and (Abdulaziz and Koller 2022) to deal with time.

Action, Methods, Tasks and Plan

A key concept in HTN planning and a fortiori in temporal HTN planning is the concept of task. Each task is given by a name and a list of parameters. We distinguish three kinds of tasks: the snap actions, the durative actions and the abstract tasks. Unlike snap actions that do change the state 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. We consider every task has a start and an end time point. We refer to the start and the end time points of a task t with temporal variables denoted respectively v_t^s and v_t^e. Since a snap action t is instantaneous, $v_t^s = v_t^e$, thus we will simply refer to the time point of the snap action t as v_t.

The durative actions and the abstract tasks can be refined respectively by applying *snap actions* and *methods* defined below.

A *snap action* a is nearly an action in the sense of classical planning, i.e., a tuple $(\textit{name}(a), \textit{precond}(a), \textit{effect}(a))$. $\textit{name}(a)$ is the name of a. The preconditions $\textit{precond}(a)$ and effects $\textit{effect}(a)$ are sets of ground predicates. Let v_a be the time point at which a is supposed to be executed. a is executable if $\textit{precond}(a)$ hold strictly before v_a. As in classical planning, the execution of a produces the effects $\textit{effect}(a)$ such that $\textit{effect}(a) = \textit{effect}^+(a) \cup \textit{effect}^-(a)$ and $\textit{effect}^+(a) \cap \textit{effect}^-(a) = \emptyset$ where $\textit{effect}^+(a)$ and $\textit{effect}^-(a)$ are conjunctions of predicates, respectively true and false after the execution of a. Finally, we say that a snap action a refines a task t if $t = \textit{name}(a)$.

A *durative action* a is a tuple $(\textit{name}(a), \textit{start}(a), \textit{end}(a), \textit{inv}(a), d)$: $\textit{name}(a)$ is the name of a, $\textit{start}(a)$ and $\textit{end}(a)$ are snap actions ; $\textit{inv}(a)$ is a set of ground predicates 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 PDDL 2.1 (Fox and Long 2003) that $v_a^s < v_a^e$ is true. Therefore the duration of a is a strictly positive number. Similarly to a snap action, a durative action refines a task t if $t = \textit{name}(a)$.

An *abstract task* must be decomposed into durative actions in order to be performed. The several ways of decomposing an abstract task are described through *methods*. Even though we cannot set a duration for an abstract task in the general case, the start and end point of an abstract can still be subject to temporal constraints.

A *method* m is a tuple $(\textit{name}(m), \textit{task}(m), \textit{subtasks}(m), \alpha, \textit{constraints}(m))$, where $\textit{name}(m)$ is the name of the method, $\textit{task}(m)$ is the task refined by the method, $\textit{subtasks}(m)$ the set of tasks symbols (possibly empty) which refines $\textit{task}(m)$, $\alpha : \textit{subtasks}(m) \mapsto \mathcal{T}$ maps the task symbols to a set of task names and $\textit{constraints}(m)$ is a set of temporal ordering constraints over $\textit{subtasks}(m)$. Temporal ordering constraints are defined over the time variable start or the end of the subtasks $\textit{subtasks}(m)$ of m. The possible qualitative temporal ordering constraints are those from the classical point algebra (Broxvall and Jonsson 2003): $<$, $\leq$, $>$, $\geq$, $=$ and $\neq$. For instance, the temporal ordering constraint $v_{t_1}^s < v_{t_2}^e$ expresses that the start of the task t_1 must occur strictly before the end of t_2. A method m refines a task t if $t = task(m)$. Note, that consistency checking of such a set of constraints C can be refined by computing strongly connected components of the constraint graph associated in polynomial time $O(|C|)$.

A *partial temporal plan* π is a tuple $(T, \alpha, \mathcal{C}, \mathcal{L})$ where:

- T is a set of task symbols.
- $\alpha : T \mapsto \mathcal{T}$ a mapping from task symbols to task names.
- $\mathcal{C}$ is a set temporal ordering constraints over the tasks symbols in T. The constraints are like those used in methods.
- $\mathcal{L}$ is a set of causal links of the form $\langle t_i \xrightarrow{p} t_j \rangle$ with t_i and t_j two snap actions in T such as $(t_i < t_j) \in \mathcal{C}$ and $p \in \textit{effect}(\alpha(t_i))$ and $p \in \textit{precond}(\alpha(t_j))$ (classical causal link definition in POCL).

A snap task t_k in a partial temporal plan π is a *threat* on a causal link $\langle t_i \xrightarrow{p} t_j \rangle$ if and only if (1) t_k has an effect $\neg p$ and (2) the ordering constraints $(t_i < t_k)$ and $(t_k < t_j)$ are consistent with $\mathcal{C}$ if $t_i < t_j$.

A *flaw* in a partial temporal plan $\pi = (\mathcal{T}, \mathcal{C}, \mathcal{L})$ is either (1) an open precondition, i.e., a precondition or a postcondition of task $t \in \mathcal{T}$ not supported by a causal link or (2) a threat, i.e., a task that may interfere with a causal link or (3) a task $t \in \mathcal{T}$ that is not a snap task.

Temporal HTN Planning Problem and Solution

A *temporal HTN planning problem* $\mathcal{P}$ is a tuple $(L, \mathcal{T}, \mathcal{A}, \mathcal{M}, s_0, \pi_0, g)$, where L is a finite set of logical propositions, $\mathcal{T}$ is a set of tasks, $\mathcal{A}$ is a set of durative actions and $\mathcal{M}$ is the set of methods, $s_0 \subseteq L$ is the initial state in the set of states $\mathcal{S}$, π_0 is the initial partial temporal plan, and $g \subseteq L$ is a of ground predicates describing the goal.

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 built into snap tasks by applying methods and durative actions. Formally, a partial temporal plan π is solution of a planning problem $\mathcal{P} = (L, \mathcal{T}, \mathcal{A}, \mathcal{M}, s_0, \pi_0, g)$ if and only if:

1. π is a refinement of the initial partial temporal plan π_0: π_0 contains two special snap task: t_0 with no precondition but with s_0 as effects and t_∞ with the goal g as precondition but no effects and $v_{t_0}^e < v_{t_\infty}^s$ in $\mathcal{C}$.
2. π needs to be executable in the initial state s_0. Thus,

(a) all tasks in π are snap tasks,

(b) π has no flaws, i.e., no open precondition and no causal threats,

(c) for all t in π, v_t is assigned and satisfies the temporal constraints of π.

It remains to define how to refine a partial temporal plan into a plan containing only snap actions by using methods and durative actions.

First, consider the case of the method refinement. Let $m = (\mathit{name}(m), \mathit{task}(m), \mathit{subtasks}(m), \alpha, \mathit{constraints}(m))$ be a method that refines a task t and plan $\pi_1 = (T_1, \alpha_1, \mathcal{C}_1, \mathcal{L}_1)$ a plan such that $t \in \mathcal{T}_1$. Then, m refines π_1 into a plan $\pi_2 = (T_2, \alpha_2, \mathcal{C}_2, \mathcal{L}_2)$ and

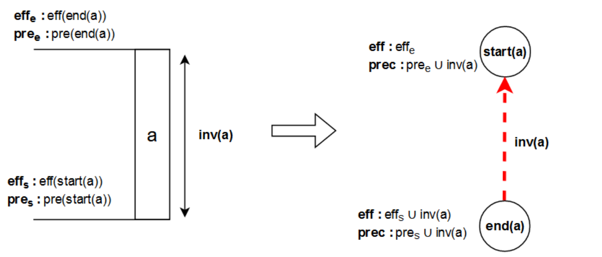
$$\begin{aligned}
T_2 &= (T_1 - \{t\}) \cup \mathit{subtasks}(m)\\
\alpha_2 &= \{(t', \alpha_1(t')) \mid t' \in T_1 \backslash \{t\}\} \cup \alpha\\
\mathcal{C}_2 &= \mathcal{C}_1 \cup \mathit{constraints}(m)\\
&\quad \cup \{c \mid \forall u \in \mathit{subtasks}(m)\ v_t^s \leq v_u^s\}\\
&\quad \cup \{c \mid \forall u \in \mathit{subtasks}(m)\ v_t^e \geq v_u^e\}\\
&\quad \cup \{v_t^s \leq v_t^e\}\\
\mathcal{L}_2 &= \mathcal{L}_1
\end{aligned}$$

Consider now the case of the durative action refinement. Let $a = (\mathit{name}(a), \mathit{start}(a), \mathit{end}(a), \mathit{inv}(a), d)$ a durative action. To realize this refinement it is first necessary to slightly modify the definition of the two snap actions $\mathit{start}(a)$ and $\mathit{end}(a)$ to translate the invariant properties $\mathit{inv}(a)$ into the POCL logic. This modification consists in adding $\mathit{inv}(a)$ to the effects of $\mathit{start}(a)$ and to the preconditions of $\mathit{start}(a)$ and $\mathit{end}(a)$ (see Figure 1). It is now possible to express the invariant properties of a durative task by classical causal links between the effects of $\mathit{start}(a)$ and the preconditions of $\mathit{end}(a)$ in accordance with PDDL 2.1 semantics that constrains $\mathit{inv}(a)$ to be checked on the interval $]\mathit{start}(a), \mathit{end}(a)[$. More formally, suppose a refines a task t of a plan $\pi_1 = (T_1, \alpha_1, \mathcal{C}_1, \mathcal{L}_1)$ such that $t \in T_1$. Then, a refines π_1 into a plan $\pi_2 = (T_2, \alpha_2, \mathcal{C}_2, \mathcal{L}_2)$ and

$$\begin{aligned}
T_2 &= (T_1 - \{t\}) \cup \{v_t^s, v_t^e\}\\
\alpha_2 &= \alpha_1 \cup \{(v_t^s, \mathit{start}(a)); (v_t^e, \mathit{end}(a))\}\\
\mathcal{C}_2 &= \mathcal{C}_1 \cup \{v_t^s < v_t^e, v_t^e - v_t^s = d\}\\
\mathcal{L}_2 &= \mathcal{L}_1 \cup \{l \mid \forall p \in \mathit{inv}(a)\ l = \langle \mathit{start}(a) \xrightarrow{p} \mathit{end}(a)\rangle\}
\end{aligned}$$

Hierarchical Temporal Planning Event

The particularity of our approach is that it works by interleaving two steps to take advantage of the heuristics developed for non-temporal hierarchical planning. The first step is to compile a temporal problem in a non temporal one. To that end, we refine abstract tasks and durative actions into instantaneous non temporal actions called *snap actions*, and we search for a solution plan by guiding this search with non temporal POCL HTN heuristics, and by *checking* constraint consistency. The second step is to *find* time assignments matching the temporal constraints by using a simple CSP solver.



$\mathbf{eff_e}$: eff(end(a))
$\mathbf{pre_e}$: pre(end(a))
a
inv(a)
$\mathbf{eff_s}$: eff(start(a))
$\mathbf{pre_s}$: pre(start(a))
eff : $\mathrm{eff_e}$
prec : $\mathrm{pre_e} \cup$ inv(a)
start(a)
inv(a)
eff : $\mathrm{eff_S} \cup$ inv(a)
prec : $\mathrm{pre_S} \cup$ inv(a)
end(a)

Figure 1: Our compilation of durative actions into snap actions. The invariant conditions $\mathit{inv}(a)$ are added to the preconditions of $\mathit{start}(a)$ and $\mathit{end}(a)$ and a causal link protecting $\mathit{inv}(a)$ (represented as a red dashed line) is added between the two snap actions.

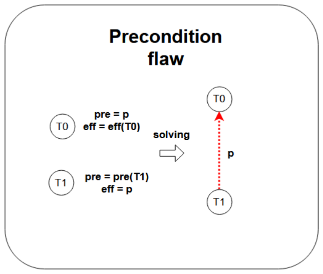
Algorithm 1: HTEP($\mathcal{S}, \mathcal{T}, \mathcal{A}, \mathcal{M}, s_0, \pi_0, g$)

```
1  open ← $\{\pi_0\}$
2  while open $\neq \emptyset$ do
3      $\pi$ ← non-deterministically select plan in open
4      flaws ← the set of flaws of $\pi$
5      if flaws $= \emptyset$ then
6          $V$ ← search of time variable assignment of $\pi$
7          if $V \neq \emptyset$ then return $\pi$ and $V$
8      $\phi$ ← deterministically select a flaw in flaws
9      open ← open $\cup$ $solveFlaw(\omega, \phi)$
10 return Failure;
```

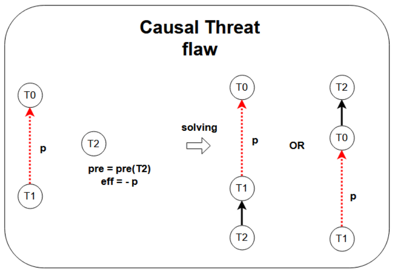
In this section, we present the search procedure implemented in our approach, called HTEP (Hierarchical Temporal Event Planner). We first give an overview of its search procedure based on hybrid planning (Bercher, Keen, and Biundo 2014), then we detail the particular way of handling specific temporal flaws of our approach, and we terminate by presenting the implemented heuristics.

Search Procedure

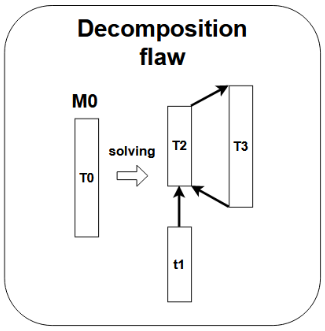
The HTEP search procedure is given in Algorithm 1. It takes as input a hierarchical temporal planning problem and returns a temporal partial plan. The procedure starts by compiling the tasks of the initial plan π_0 into snap actions (line 1). Recall that π_0 contains two special snap actions: t_0 with no precondition but with s_0 as effects and t_∞ with the goal g as precondition but no effects and $v_{t_0}^e < v_{t_\infty}^s$ in $\mathcal{C}$. Then, π_0 is added to the pending list of partial plans to explore *open* (line 2) and the main loop starts (line 3). At each iteration a plan π is non-deterministically selected in *open* (line 4). Then, the flaws of π are computed (line 5). If the plan has no flaws (line 6), HTEP searches an assignment of the time variables of π by using a CSP that matches its constraints. If such assignment exists (line 8), π is solution. Otherwise, one flaw is deterministically selected (line 9). Solving this flaw, generates a new set of partial temporal plans, which are added to the *open* list (line 10).



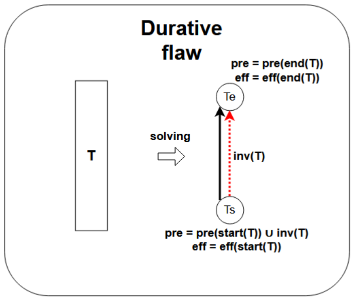
(a) A Precondition flaw is solved by adding a *causal link* from a provider event and the needer event.



(b) A Causal Threat flaw is solved by ordering the threatening event either before or after the causally linked events.



(c) Decomposition flaws are solved by applying a method, temporal constraints are applied over start and end point of tasks.



(d) Durative flaws are solved by compiling a durative task into snap actions.

Figure 2: The four types of flaws encountered by our planner and how to solve them. On every figure, black arrows represent ordering constraints and red dashed arrows the causal links.

Temporal Flaws

In addition to the classic flaws in POCL (precondition and causal threat flaws), HTEP requires the management of two specific types of flaws: the decomposition flaws that are repaired by applying a method and by decomposing an abstract temporal task in primitive temporal tasks, and the durative flaws that can be repaired by decomposing a durative action into snap actions. We detail each flaw encountered in the following:

- **Precondition flaw:** This flaw occurs for each precondition introduced in the partial plan. It is resolved by adding a *causal link* between a preceding snap action having as effect the required precondition. We denote a causal link c by : $t_1 \xrightarrow{p} t_2$. The resolution of this flaw is represented on Figure 2a.
- **Causal threat flaw:** There is a causal threat flaw for each causal link $c : t_1 \xrightarrow{p} t_2$ and each snap action a (with $p \in del(a)$) that is neither ordered before t_1 nor ordered after t_2. A causal threat is resolved either by adding a constraint: $a < t_1$ or $t_2 < a$ as represented on Figure 2b.
- **Decomposition flaw:** As in hybrid planning, decomposition flaw occurs for each abstract task still in the partial plan. They are solved by decomposing the abstract task with a method. Note that the decomposed tasks are still considered as durative and still need to be compiled into snap actions. This flaw is represented on Figure 2c.
- **Durative flaw:** A durative flaw occurs for each durative action still in the plan. In order to solve this flaw, the durative action is decomposed into two snap actions representing the start and the end points of the task. Finally, a causal link protecting the invariant preconditions is added between the two snap actions. These invariant preconditions are then added as preconditions of the start snap action.

Partial Order Planning heuristics

The HTEP search procedure relies on two selection functions. The first one performs a non deterministic choice over the set of partial temporal plans in the *open* list, and decides which partial plan to explore first (line 4). This function is called *plan selection heuristic* and greatly impact both the search performances (e.g. the time required to find a solution plan) but also the quality of the returned plan (e.g. the cost of the plan according to some optimization function). The second selection function, called *flaw selection heuristic*, selects (line 9) the flaw to be solved in the current partial plan to explore. Note that every flaw in the partial plan will eventually have to be solved in order to find a solution network. Hence, the *admissibility* of a POCL procedure only depends on the *plan selection* heuristic. However, the *order* in which the flaws are resolved, defined by the flaw selection heuristic, greatly impacts the search performances of the procedure. In the following we will present the plan selection and flaw heuristics implemented in HTEP.

Plan selection heuristics In the literature, there are two main categories of plan selection heuristics. The first type of heuristics has a POCL related approach which analyzes the flaws of a partial plan to infer a heuristic value. A well-known heuristics has been proposed in (Nguyen and Kambhampati 2001) and simply counts the number of open conditions to satisfy in the partial plan. This idea has been further refined in PANDA (Bercher, Keen, and Biundo 2014) where the use of TDG (Task Decomposition Graphs) allows to also estimate the number of open conditions that will be introduced by refining the plan. This led to two plan selection heuristics. The first one denoted h_{TC} computes the cardinality of the mandatory tasks that will appear in the partial plan decomposition. It is usually summed with the number of flaws remaining in the plan forming a heuristics denoted h_{F+TC}. The second TDG heuristic is denoted h_{MME} and estimates the number of modifications required to refine the partial plan into a solution one. The h_{MME} has been further refined to take into account the number of causal links already introduced into the plan by a heuristic denoted h_{TDGm}. Finally, FAPE (Bit-Monnot et al. 2020) has proposed a plan selection heuristics in their chronicle planner. This heuristics is also an estimation of the remaining effort required to obtain a solution from a partial plan. In the following we will denote this heuristic h_{FAPE}.

The second type of heuristics adapt heuristics from non hierarchical planning. The two most notable approaches have been proposed in (Nguyen and Kambhampati 2001) where an adaptation of the Fast Forward heuristic (Hoffmann and Nebel 2011) has been proposed. In addition, the ADD heuristics (Baier, Bacchus, and Mcilraith 2009) has been adapted in the VHPOP planner (Younes and Simmons 2003). These heuristics are very well suited for non-hierachical POCL planning with task insertion, which is not considered here.

In that regard, we decided to use the two TDG plan selection heuristics h_{F+TC} and h_{TDGm} used in PANDA and the plan selection heuristic presented in FAPE for our experimentation.

Flaw selection heuristics When it comes to flaw selection heuristics, the known strategies aim at reducing the branching factor of the search space by resolving the flaws with the fewest number of resolvers first. It is usually expressed as a priority list to follow. To our knowledge, there is no recent comparison between the current flaw selection heuristics. In that regard, we have chosen to implement in HTEP the flaw selection heuristics of PANDA (Bercher, Keen, and Biundo 2014) (which is the LCFR heuristics presented in (Joslin and Pollack 1994)) and the priority list presented in FAPE (Bit-Monnot et al. 2020). Note that the heuristic used in FAPE is a refinement of the LCFR heuristic: while LCFR prioritizes the flaws with the fewest resolvers, the FAPE flaw selection heuristics also prioritizes unrefined tasks and preconditions first.

Experimentation

In this section, we will compare our *Temporal Event* approach with a *Chronicle* approach. Both approaches have been been coded and tested on the same device. We have used as reference the chronicle planner described in (Bit-Monnot et al. 2020) as it is the current state-of-the-art of hierarchical temporal planning. As both algorithms use a CSP solver in their procedures, we will use the same CSP solver as well. All the benchmarks and code used to produce these results will be freely available for result reproduction.

Experimental Setup

We will compare the results on three indicators:

- **Solving time**: it represents the time spent to solve the problem (from instantiation to solution)
- **Makespan of the solution**: it represents the overall length of the plan, meaning the time between the initial state and the final task end point.

The results will be presented by scoring these two indicators with the IPC scoring metric.

In this paper we consider four configurations, which are presented below:

- $HTEP + h_{TDGm}$**:** in this first configuration, HTEP is used with the h_{TDGm} heuristics described in (Bercher et al. 2017) adapted to our temporal event representation.
- $HTEP + h_{F+PC}$**:** in this configuration, the Temporal Event Planner is used with the first TDG heuristics proposed in (Bercher, Keen, and Biundo 2014) associated with h_{F+PC} heuristics.
- $TE + h_{FAPE}$**:** in this configuration, HTEP is used with the plan selection heuristics used by the FAPE planner (Bit-Monnot et al. 2020).
- $Chr + h_{FAPE}$**:** this is the chronicle planner encoded to mimic the behavior of FAPE (Bit-Monnot et al. 2020). It uses both the plan selection and flaw selection heuristics described in (Bit-Monnot et al. 2020) and uses a chronicle representation.

Note that all configurations use the same Flaw Selection heuristics described in (Bit-Monnot et al. 2020).

We chose four temporal planning domains for our tests. These domains require different levels of concurrency to be solved. We categorized these problems according to the three Cushing concurrency categories (Cushing 2007):

- **Gripper:** this domain is a sequential one, it is the simplest domain with no temporal concurrency required. These problems are members of the first Cushing (Cushing 2007) category where all solutions are sequential.
- **Satellite:** this domain is also a sequential one but optimization over the *makespan* of the solution is possible (e.g. there are several sequential plans possible with different makespans). These problems are also members of the first Cushing category.
- **Rover:** in this domain, concurrency is possible although not required. The planner can either choose to find a simple non concurrent plan or a more efficient concurrent one. These problems are members of the second Cushing category where solutions can either be sequential or concurrent.

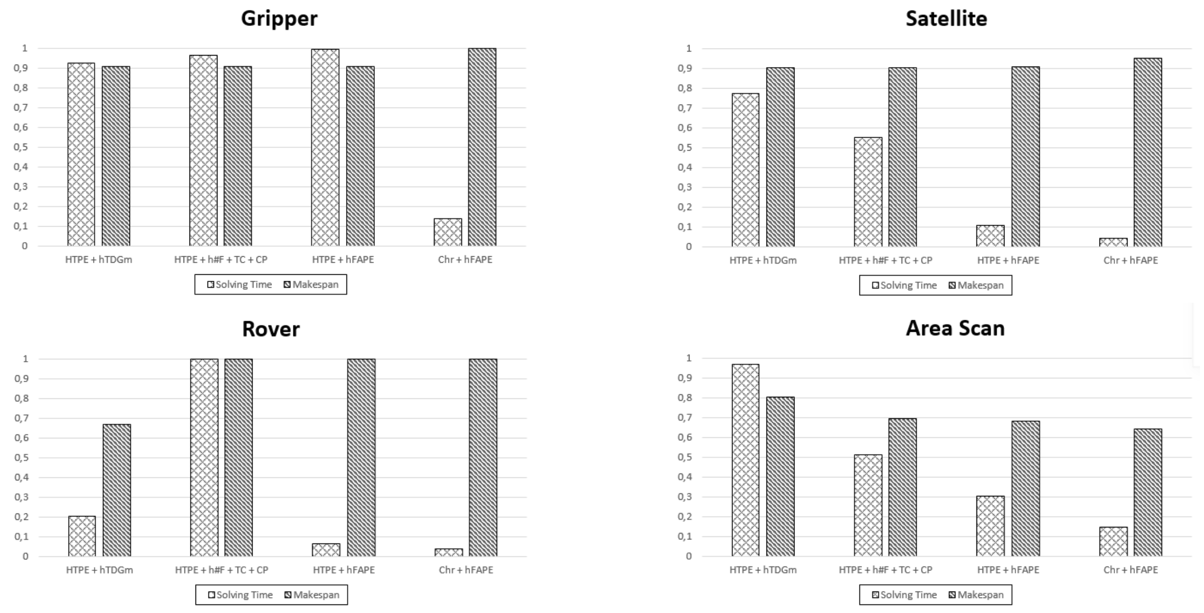


Figure 3: Results for the five configurations, with the IPC metric. Each diagram represents the results on one domain.

- **Area scan:** this domain models a set of heterogeneous devices aiming at cooperating in order to scan areas. The devices can either cooperate to reach their destination faster or act by their own, resulting in less effective plans. It presents both sequential and concurrent solution plans, with numerous makespan optimization possible.

We expressed all domains and problems in HDDL 2.1 language proposed by (D. Pellier and Bailon-Ruiz. 2023). All experiments were run on a single core of a Intel Core i7-9850H CPU, with a limit of 8GB of RAM over 600 seconds. The code and benchmarks will be made available if this paper is accepted.

Results

The results are presented on Figure 3.

We can see that there are some tendencies over all domains. On all domains and for both heuristics, the Temporal Event planner outperforms its chronicle counterpart with regards to the *Solving time* metric. HTEP handles non temporal constraints and flaws which leads to simpler plan refinements and flaws compared to a Chronicle planner. In our opinion, this is what explains the discrepancies between the $HTEP + h_{FAPE}$ and $Chr + h_{FAPE}$ configurations. In addition, the HTEP approach benefits from the use of classical HTN heuristics: the $HTEP + h_{TDGm}$ and $HTEP + h_{F+PC}$ configurations are the best performing ones on all domains. We can notice that the h_{F+PC} seems to be the best performing one on the Rover domain while h_{TDGm} is more efficient on the other domains. As explained in (Bercher, Keen, and Biundo 2014), the performance of each heuristics depends on the domain definition of the problem and a heuristics can be more or less informative depending on the domain. As a Chronicle approach should handles a larger set of constraints it should be able to eliminate partially refined plan sooner in the refinement process than HTEP. However this case does not happen often as many HTN domains describe the necessary temporal ordering through methods and task decomposition. The main sources of partial plan elimination are unsolvable non temporal flaws. Overall, it seems that the $HTEP+h_{TDGm}$ configuration is the most efficient one when it comes to the *Solving Time* metric.

Concerning the *Makespan*, we can see that the $Chr + h_{FAPE}$ configuration is the most efficient one. As this configuration handles the full temporal constraints, it can prioritize the best one in terms of makespan when two have equal heuristic value. This leads to higher quality plans in most domains. On the other hands, the HTEP configurations remain competitive with the Chronicle approach. The exception to this is displayed on the Rover domain where the $HTEP + h_{TDGm}$ configuration is the lowest one makespan wise. This heuristics aims at finding the solution with the fewest number of refinement required, regardless of the solution plan makespan. Note that if h_{TDG} is the best performing one on the Area Scan domain is due to the fact that some instances have not been solved by others configurations, thus increasing its score.

Overall, the configuration combining HTEP and the classical HTN heuristics seems to outperform the *Chronicle* approaches. This formalism allows for simpler constraints representation and managements. It also benefits from the hybrid planning heuristics and the HTN formalism which often provides the necessary ordering constraints to the planner.

Discussion

All along this paper, we used a compilation of temporal actions into snap actions by representing the invariant condition of a temporal action through a POCL causal link. The invariant are treated as precondition of the start snap action and protected until the end snap task through this causal link (see Figure 1). This compilation implies a that HTEP can not solve temporal in the third Cushing's category: in temporal planning, invariant condition can not be seen as precondition for the whole temporal action. Instead, they should be seen as *postconditions* of the start snap action, meaning that invariant condition should be verified right *after* the execution of the start. This subtlety actually adds a new layer of complexity in planning as it allows to define *necessary concurrent actions*. This has been demonstrated and explained by Cushing in his three temporal hierarchical problem classes definition (Cushing 2007).

Conclusion

In this paper, we have presented an approach to represent and solve Temporal HTN problems by using Temporal Events. This approach relaxes the temporal problem in a simpler one, which allows to apply classical HTN search heuristics to it. We have compared this approach with the Chronicle one, which is the current state of the art in hierarchical temporal planning. We have shown that the Temporal Event approach outperforms it in terms of time spent to find a solution and is comparable to it when it comes to the quality of the solution plans. HTEP can still be improved by applying other classical HTN search techniques. In addition, we want to improve the compilation made in HTEP in order to allow it to solve temporal problems in the third Cushing category.

References

Abdulaziz, M.; and Koller, L. 2022. Formal Semantics and Formally Verified Validation for Temporal Planning. In *AAAI Conference on Artificial Intelligence*, 9635–9643.

Asuncion, M.; Castillo, L.; Fdez-Olivares, J.; Garcia-Perez, O.; Gonzalez-Munoz, A.; and Palao, F. 2005. SIADEX: An interactive knowledge-based planner for decision support in forest fire fighting. *AI Commun.*, 18: 257–268.

Au, T.-C.; Ilghami, O.; Kuter, U.; Murdock, J. W.; Nau, D. S.; Wu, D.; and Yaman, F. 2003. SHOP2: An HTN Planning System. *J. of Artif. Intell. Res.*, 20: 379–404.

Baier, J.; Bacchus, F.; and Mcilraith, S. 2009. A Heuristic Search Approach to Planning with Temporally Extended Preferences. *Artif. Intell.*, 173: 593–618.

Barreiro, J.; Boyce, M.; Do, M.; Frank, J.; Iatauro, M.; Kichkaylo, T.; Morris, P.; Ong, J.; Remolina, E.; Smith, T.; et al. 2012. EUROPA: A platform for AI planning, scheduling, constraint programming, and optimization. *4th International Competition on Knowledge Engineering for Planning and Scheduling (ICKEPS)*.

Bechon, P.; Barbier, M.; Infantes, G.; Lesire, C.; and Vidal, V. 2014. HiPOP: Hierarchical Partial-Order Planning. In *Starting AI Researchers' Symposium*.

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

Bercher, P.; Keen, S.; and Biundo, S. 2014. Hybrid Planning Heuristics Based on Task Decomposition Graphs. *Proceedings of the International Symposium on Combinatorial Search*.

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

Broxvall, M.; and Jonsson, P. 2003. Point algebras for temporal reasoning: Algorithms and complexity. *Artif. Intell.*, 149(2): 179–220.

Celorrio, S. J.; Jonsson, A.; and Palacios, H. 2015. Temporal Planning With Required Concurrency Using Classical Planning. In *ICAPS*, 129–137.

Cushing, W. 2007. Evaluating Temporal Planning Domains. *ICAPS*, 105–112.

D. Pellier, H. F., A. Albore; and Bailon-Ruiz., R. 2023. HDDL 2.1: Towards Defining an HTN Formalism with Time. In *6th ICAPS Workshop on Hierarchical Planning (HPlan 2023)*.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal Constraint Networks. *Artif. Intell.*, 49(1-3): 61–95.

Fox, M.; and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *J. of Artif. Intell. Res.*, 20: 61–124.

Goldman, R. 2006. Durative Planning in HTNs. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 382–385.

Hoffmann, J.; and Nebel, B. 2011. The FF Planning System: Fast Plan Generation Through Heuristic Search. *J. of Artif. Intell. Res.*, 14.

Höller, D.; Behnke, G.; Bercher, P.; Biundo, S.; Fiorino, H.; Pellier, D.; and Alford, R. 2020. HDDL: An Extension to PDDL for Expressing Hierarchical Planning Problems. In *AAAI Conference on Artificial Intelligence*, 9883–9891.

Joslin, D.; and Pollack, M. E. 1994. Least-Cost Flaw Repair: A Plan Refinement Strategy for Partial-Order Planning. In Hayes-Roth, B.; and Korf, R. E., eds., *NCAI*, 1004–1009.

Lallement, R.; de Silva, L.; and Alami, R. 2018. HATP: Hierarchical Agent-Based Task Planner. In *AAMAS*, 1823–1825.

Lemai, S. 2004. IXTET-EXEC: planning, plan repair and execution control with time and resource management.

Milot, A.; Chauveau, E.; Lacroix, S.; and Lesire, C. 2021. Solving Hierarchical Auctions with HTN Planning. In *ICAPS workshop on Hierarchical Planning*.

Nguyen, X.; and Kambhampati, S. 2001. Reviving Partial Order Planning. In *IJCAI*, 459–464.

Younes, H. L. S.; and Simmons, R. G. 2003. VHPOP: Versatile Heuristic Partial Order Planner. *J. of Artif. Intell. Res.*, 20: 405–430.