Extension de la planification HTN aux problèmes temporels

Nicolas Cavrel Humber Fiorino Damien Pellier

Univ. Grenoble Alpe, LIG, France
Nom.Prenom@imag.fr

Résumé

Cet article présente TEP (Temporal Event Planning), une approche originale conçue pour les réseaux de tâches hiérarchiques (HTN) temporels. TEP se distingue par sa capacité à aborder les trois catégories de problèmes temporels, telles que catégorisées par Cushing, ce qui le différencie des approches prédominantes qui s'attaquent principalement aux deux premières catégories. TEP accomplit cela en traduisant les problèmes HTN temporels en problèmes non temporels, permettant ainsi d'utiliser des heuristiques développées dans la planification HTN non temporelle pour orienter la recherche de solutions temporelles et de surpasser les approches existantes. Cette stratégie innovante commence par affiner les tâches en actions instantanées et par relâcher les contraintes de durée, assurant ainsi une cohérence avec les heuristiques de recherche conventionnelles tout en préservant l'expressivité inhérente du problème.

1 Introduction

Le formalisme de planification des réseaux de tâches hiérarchiques (HTN) est conçu pour représenter les problèmes de planification sous forme d'ensembles de tâches complexes. Conformément à la stratégie classique de division et de conquête, ces problèmes sont résolus en décomposant de manière récursive les tâches complexes en tâches plus simples jusqu'à ce que le problème ne se compose uniquement de tâches indécomposables, appelées "actions".

Alors que la plupart des approches HTN se concentrent principalement sur la planification HTN non temporelle, e.g., [6, 12, 17], où les actions sont considérées comme instantanées, nous nous attaquons dans cet article à la résolution de problèmes de planification HTN temporelle, où les actions sont définies sur des intervalles de temps. La prise en compte de cette dimension est essentielle lors de la résolution de problèmes du monde réel, où les contraintes temporelles jouent un rôle essentiel et où la synchronisation entre agents est impérative. Les différentes approches proposées pour résoudre les problèmes de planification temporelle peuvent être catégorisées en fonction de leur capacité à aborder différentes catégories de problèmes temporels, telles que définies par Cushing [9]. La classification de Cushing distingue trois catégories de problèmes de planification temporelle. Dans la première catégorie, les problèmes temporels ont des solutions consistant en des plans séquentiels : la concurrence des actions n'est pas nécessaire (solutions non concurrentes). La deuxième catégorie englobe les problèmes temporels où les plans de solution peuvent potentiellement être concurrents, mais les solutions séquentielles sont toujours valides (solutions éventuellement concurrentes). La troisième catégorie comprend les problèmes temporels où les seuls plans de solution réalisables sont intrinsèquement concurrents, et les solutions séquentielles ne sont pas possibles (solutions concurrentes nécessaires). La plupart des approches existantes relèvent des deux premières catégories de Cushing. C'est le cas par exemple de [2, 1, 11, 22, 25], où les actions temporelles sont prétraitées en séquences de tâches non temporelles [2] ou d'actions. Un avantage important de ces approches est leur capacité à tirer parti des heuristiques de recherche et des algorithmes développés dans un cadre non temporel. D'autre part, certaines approches telles que par exemple [26, 4, 14, 7] résolvent toutes les catégories de Cushing, mais elles manquent d'efficacité en raison d'un manque d'heuristiques informatives.

Dans cet article, nous présentons TEP (Temporal Event Planning), une nouvelle approche de planification hybride regroupant des méthodes de liaison causale à ordre partiel et des méthodes HTN, adaptée aux domaines HTN temporels. Notre contribution réside dans la capacité de TEP à résoudre toutes les catégories de problèmes HTN de Cushing, tout en utilisant des heuristiques développées pour la planification HTN non temporelle. TEP commence par compiler les problèmes HTN temporels en problèmes non temporels. Il accomplit cela en affinant les tâches abstraites et les tâches duratives en actions non temporelles instantanées. Ensuite, il tente de trouver un plan de solution en relâchant les contraintes de durée, uniquement en vérifiant la cohérence des contraintes. Le problème non temporel résultant,

une fois relâché, ressemble étroitement à un problème HTN classique, ce qui le rend compatible avec les heuristiques de recherche non temporelles. En particulier, il conserve l'expressivité requise pour résoudre des problèmes dans toutes les catégories de Cushing. Si une solution au problème relâché est trouvée, TEP procède en calculant des affectations de timestamp qui satisfont les contraintes temporelles avec un solveur de CSP (Problème de Satisfaction de Contraintes).

Le document est structuré comme suit. La partie 1 introduit le formalisme de planification HTN temporelle. Dans la partie 2, l'algorithme TEP est présenté. La partie 3 démontre comment TEP résout les problèmes HTN dans la troisième catégorie de Cushing. La partie 4 examine en détail les heuristiques adoptées de la planification hybride non temporelle et utilisées dans TEP pour aborder les problèmes de planification HTN temporelle. Enfin, la partie 5 décrit l'évaluation de TEP.

2 Formalisation

Dans cette partie, nous proposons une formalisation qui intègre les caractéristiques temporelles dans la planification HTN. Les notations sont fondées sur [19] et [6].

2.1 Tâches, Réseau de Tâches, Action et Méthodes

Les concepts clés en planification HTN, et a fortiori en planification HTN temporelle, sont les *tâches* et les *réseaux de tâches*. Une *tâche* est définie par un nom et une liste de paramètres. Nous distinguons trois types de tâches : les tâches instantanées, les tâches duratives et les tâches abstraites. Contrairement aux *tâches instantanées* qui modifient l'état du monde, les *tâches duratives* et les *tâches abstraites* ne le font pas. Ce sont des noms faisant référence à d'autres tâches (soit instantanées, duratives ou abstraites) qui doivent être accomplies selon certaines contraintes. Chaque tâche a un début et une fin. Nous faisons référence respectivement au début et à la fin d'une tâche t avec des variables temporelles notées $v^s t$ et $v^e t$. Une tâche t est instantanée, i.e., que $v^s t = v^e t$, donc nous nous référerons simplement à l'instant où s'exécute la tâche instantanée t comme v_t. Un état s est défini comme un ensemble de propositions logiques.

Un *réseau de tâches* représente un ensemble partiellement ordonné de tâches. Un réseau de tâches w est un tuple $(I, \alpha, \prec)$ où I est un ensemble d'identifiants de tâches, $\alpha : I \mapsto T$ un mapping d'identifiants de tâches vers des noms de tâches [1] T, et $\prec$ un ensemble de contraintes d'ordonnancement sur les identifiants de tâches dans I. Les contraintes d'ordonnancement sont définies sur les variables de temporelles de début ou de fin des identifiants de tâches I. Les contraintes d'ordonnancement possibles sont celles de l'algèbre de points classique [8] : $<$, $\leq$, $>$, $\geq$, $=$ et $\neq$. Nous permettons également les contraintes de la forme $d = v^e t - v^s t$ pour exprimer la durée de la tâche. Par exemple, la contrainte d'ordonnancement temporel $v^s t_1 < v^e t_2$ exprime que le début de la tâche $t_1 \in I$ doit se produire strictement avant la fin de $t_2 \in I$. Notez qu'un tel ensemble de contraintes C peut être représenté sous forme d'un graphe de contraintes dont la cohérence peut être vérifiée en calculant ses composantes fortement connectées en temps polynomial $O(|C|)$. Les tâches instantanées, duratives et abstraites peuvent être réalisées respectivement en appliquant des *actions instantanées*, des *actions duratives* et des *méthodes* définies ci-dessous.

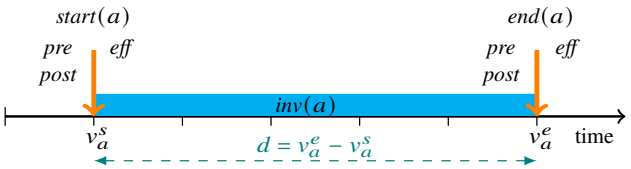

FIGURE 1 – Chronologie de l'application d'une action durative a.

Une *action instantanée* a est un tuple $(name(a), pre(a), post(a), eff(a))$, où $name(a)$ est le nom de la tâche accomplie par a et $pre(a)$, $post(a)$ et $eff(a)$ sont des ensembles de propositions. Soit v_a l'instant auquel a est exécuté. Une action instantanée a **deux** ensembles de conditions à satisfaire pour être exécutée : $pre(a)$ qui doit être vraie strictement avant v_a (cas classique en planification) et $post(a)$ qui doit être vraie strictement après v_a, i.e., dans l'état résultant de l'exécution de a. $post(a)$ **est une nouveauté pour exprimer des propriétés invariantes qui doivent être satisfaites sur un intervalle de temps**, et qui seront utilisées pour résoudre la 3ème catégorie de Cushing.

Enfin, comme en planification non temporelle, l'exécution de a produit les effets $eff(a)$ tels que $eff(a) = eff^+(a) \cup eff^-(a)$ et $eff^+(a) \cap eff^-(a) = \emptyset$ où $eff^+(a)$ et $eff^-(a)$ sont des ensembles de propositions, respectivement vraies et fausses après l'exécution de a.

Une *action durative* a est un tuple $(name(a), start(a), end(a), inv(a), d)$: $name(a)$ est la tâche accomplie par a, $start(a)$ et $end(a)$ sont des actions instantanées ; $inv(a)$ est un ensemble de propositions qui doivent être vraies après l'exécution de $start(a)$ et jusqu'au début de $end(a)$, i.e., sur l'intervalle $]v^s a, v^e a[$ et $d = v_a^e - v_a^s$ est la durée de a. Nous supposons comme en planification temporelle [10] que $v_a^s < v_a^e$ est vrai. Par conséquent, la durée de a est un nombre strictement positif. Le déroulement temporel de l'application d'une action durative est donné dans la Figure 1.

Une *méthode* m est un tuple $(name(m), w)$, où $name(m)$ est le nom de la tâche accomplie par m et w est le réseau de tâches qui définit comment $name(m)$ doit être décomposée en sous-tâches.

1. Les identifiants de tâches sont nécessaires car une tâche peut apparaître plusieurs fois dans un réseau de tâches.

2.2 Plans Temporels Partiels, Défauts et Menaces

Pour traiter des caractéristiques temporelles, nous étendons la définition d'un *plan partiel* généralement utilisé en planification hybride [6] et en planification POCL [15] afin de gérer les *postconditions* des actions instantanées comme suit. Un *plan temporel partiel* π est un tuple (w, C) où $w = (I, \alpha, \prec)$ est un réseau de tâches qui définit les tâches de π et ses contraintes d'ordonnancement, et C est un ensemble de liens de causalité de la forme $\langle t_i \xrightarrow{p} t_j \rangle$ avec t_i et t_j deux identifiants de tâches instantanées dans I, et p une proposition. Le but d'un lien de causalité est de confirmer qu'une précondition ou une postcondition p d'une tâche t_j est soutenue par une autre tâche t_i qui produit p comme effet. Deux cas doivent être considérés selon que t_i soutient une précondition ou une postcondition de t_j. Si t_i soutient une précondition de t_j, c'est-à-dire que $p \in \mathit{eff}(t_i)$ et $p \in \mathit{pre}(t_j)$ alors $(v^e_{t_i} < v^s_{t_j}) \in \prec$ (cas classique en planification POCL). Si t_i soutient une postcondition de t_j, c'est-à-dire que $p \in \mathit{eff}(t_i)$ et $p \in \mathit{post}(t_j)$ alors $(v^e_{t_i} \leq v^s_{t_j}) \in \prec$: dans ce cas, **la contrainte d'ordonnancement n'est pas stricte**. Par extension, une tâche instantanée t_k dans un plan temporel partiel π est une *menace* sur un lien de causalité $\langle t_i \xrightarrow{p} t_j \rangle$ en fonction de si t_i soutient une précondition ou une postcondition p de t_j. Si t_i soutient une précondition p de t_j, alors t_k menace $\langle t_i \xrightarrow{p} t_j \rangle$ si et seulement si t_k a $\neg p$ comme effet, et $(v^e_{t_i} < v^s_{t_k})$, $(v^e_{t_k} < v^s_{t_j})$ sont compatibles avec $\prec$ (le cas habituel en planification POCL). Si t_i soutient une postcondition p de t_j, alors t_k menace $\langle t_i \xrightarrow{p} t_j \rangle$ si et seulement si t_k a $\neg p$ comme effet, et $(v^e_{t_i} < v^s_{t_k})$, $(v^e_{t_k} \leq v^s_{t_j})$ dans $\prec$. Ici, la contrainte d'ordonnancement entre t_k et t_j n'est pas stricte. Un *défaut* dans un plan partiel $\pi = (w, C)$ avec $w = (I, \alpha, \prec)$ est soit (1) une condition ouverte, c'est-à-dire une précondition ou une postcondition d'une tâche qui n'est pas soutenue par un lien de causalité, (2) une menace, c'est-à-dire une tâche qui peut interférer avec un lien de causalité, ou (3) un défaut de décomposition, c'est-à-dire une tâche abstraite ou durative qui n'est pas décomposée en tâches instantanées.

2.3 Problème HTN temporel et solution

Un problème de planification HTN (Hierarchical Task Network) temporelle P est un tuple $(L, T, A, M, s_0, w_0, g)$, où L est un ensemble fini de propositions, T est un ensemble de tâches, A est un ensemble d'actions duratives, M est l'ensemble des méthodes, s_0 est l'état initial construit sur L, w_0 est le réseau de tâches initial à réaliser, et g est un ensemble de propositions définissant l'objectif à atteindre.

La solution d'un problème de planification HTN temporelle est un plan temporel partiel π obtenu en affinant un plan partiel initial π_0 comme dans la planification POCL en tâches instantanées en appliquant des actions duratives et des méthodes. Le plan initial π_0 est construit avec w_0 comme réseau de tâches et deux tâches instantanées spéciales t_0 et t_∞ ordonnées respectivement comme la première tâche et la dernière tâche de w_0. t_0 n'a pas de précondition et l'état initial comme effets positifs. t_∞ a l'objectif comme précondition et pas d'effets. Formellement, un plan temporel partiel $\pi = (w, C)$ est une solution d'un problème de planification $P = (L, T, A, M, s_0, w_0, g)$ si et seulement si : (1) π est un affinement du plan temporel partiel initial π_0, (2) π est exécutable dans l'état initial s_0. Ainsi, (2.1) toutes les tâches dans π sont des tâches instantanées, (2.2) π n'a pas de défauts, c'est-à-dire pas de précondition ou postcondition ouverte, et pas de menaces, (2.3) pour toutes les tâches t dans π, v_t est assigné et satisfait les contraintes d'ordonnancement dans w.

Il reste à définir comment affiner un plan temporel partiel en un plan ne contenant que des tâches instantanées en utilisant des actions duratives et des méthodes. Tout d'abord, considérons le cas de l'affinement d'une tâche durative. Pour effectuer cet affinement, il est d'abord nécessaire de modifier légèrement la définition des deux actions instantanées $\mathit{start}(a)$ et $\mathit{end}(a)$ pour traduire les propriétés invariantes $\mathit{inv}(a)$ dans la logique POCL. Cette modification consiste, d'une part, à ajouter $\mathit{inv}(a)$ aux postconditions et aux effets de $\mathit{start}(a)$, et, d'autre part, aux préconditions de $\mathit{end}(a)$. Il est maintenant possible d'exprimer les propriétés invariantes d'une tâche durative par des liens de causalité classiques entre les effets de $\mathit{start}(a)$ et les préconditions de $\mathit{end}(a)$ conformément à la sémantique PDDL 2.1, ce qui contraint $\mathit{inv}(a)$ à être vérifié sur l'intervalle $]v^s_a, v^e_a[$.

Formellement, soit $a = (\mathit{name}(a), \mathit{start}(a), \mathit{end}(a), \mathit{inv}(a), d)$ une action durative qui affine un identifiant de tâche i d'un plan $\pi_1 = (w_1, C_1)$ avec $w_1 = (I_1, \alpha_1, \prec_1)$ tel que $i \in I_1$. Alors, a affine π_1 en un plan $\pi_2 = (w_2, C_2)$ avec $w_2 = (I_2, \alpha_2, \prec_2)$ et

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

Enfin, le dernier cas est celui de l'affinement de méthode. Soit $m = (t_m, w_m)$ une méthode avec un réseau de tâches $w_m = (I_m, \alpha_m, \prec_m)$ qui affine un identifiant de tâche i d'un plan $\pi_1 = (w_1, C_1)$ avec $w_1 = (I_1, \alpha_1, \prec_1)$ tel que $i \in I_1$. Alors, m affine π_1 en un plan $\pi_2 = (w_2, C_2)$ avec $w_2 = (I_2, \alpha_2, \prec_2)$ et

$$
\begin{aligned}
I_2 &= (I_1 - \{i\}) \cup I_m \\
\alpha_2 &= \alpha_1 \cup \alpha_m - \{(i, t_m)\} \\
\prec_2 &= \prec_m \ \cup \{(v'_i < v^s_i) \mid \forall (v'_i < v_i) \in \prec_1\} \\
&\quad \cup \{(v'_i \leq v^s_i) \mid \forall (v'_i \leq v_i) \in \prec_1\}
\end{aligned}
$$

$$\cup \{(v_i^e > v_i') \mid \forall (v_i > v_i') \in \prec_1\}$$
$$\cup \{(v_i^e \geq v_i') \mid \forall (v_i \geq v_i') \in \prec_1\}$$
$$\cup \{(v_i' \geq v_i^s) \mid v_i' \in I_m)\} \cup \{(v_i' \leq v_i^e) \mid v_i' \in I_m)\}$$
$$\cup \{(v_i^s \leq v_i^e)\}$$
$$C_2 = C_1 \tag{2}$$

3 Temporal Event Planner

Dans cette partie, nous définissons la procédure de recherche de TEP (*Temporal Planning Event*). Cette procédure est fondée sur la procédure de planification hybride POCL proposée par [6]. Elle traite des deux ensembles de conditions qui doivent être remplies pour exécuter une action : les préconditions et les postconditions. Nous montrons comment TEP est capable d'utiliser les heuristiques développées pour la planification hybride non temporelle et comment elle peut résoudre toutes les catégories de problèmes temporels de Cushing [9]. Il est important de souligner que les problèmes de la troisième catégorie de Cushing sont des problèmes dont les plans de solution nécessitent nécessairement la concurrence. À notre connaissance, TEP est le seul planificateur hybride capable de résoudre cette catégorie de Cushing.

3.1 Procédure de recherche

La principale idée de TEP est de mener la recherche d'un plan de solution en intercalant deux étapes. La première étape affine un plan partiel initial en un plan ne contenant que des tâches instantanées, comme dans la planification hybride, tout en vérifiant la cohérence des contraintes d'ordonnancement des tâches. Contrairement à la planification hybride, TEP vérifie à la fois la cohérence des contraintes $<$ et $\leq$. Cette première étape se résume à résoudre un problème non temporel, ce qui permet à la recherche d'être guidée par des heuristiques classiques. Nous détaillerons ce point dans la prochaine section. L'objectif de la première étape de TEP est de vérifier les critères 1, 2 (a) et 2 (b) d'un plan de solution. La deuxième étape attribue une valeur à toutes les variables temporelles associées aux tâches instantanées du plan affiné, en respectant les contraintes d'ordonnancement ainsi que les contraintes de durée $d = v_a^e - v_a^s$ [2].

La procédure de recherche TEP est donné dans l'algorithme 1. Il prend en entrée un problème $P = (L, T, A, M, s_0, w_0, g)$ et renvoie un plan de solution π ainsi qu'un ensemble V représente les valeurs affectées aux variables temporelles pour chaque tâche instantanée si la procédure réussit, et un échec sinon. TEP commence (ligne 1) par exprimer l'état initial s_0, le réseau de tâches initial w_0 et l'objectif g sous la forme d'un plan partiel à raffiner π_0.

2. Pour des raisons de simplicité, nous nous limitons ici à ce type de contraintes, en gardant à l'esprit que l'approche CSP peut être utilisée pour exprimer des contraintes plus complexes.

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

```
1  π0 ← the inital plan built from s0, w0 and g
2  open ← {π0}
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 return Failure
10     φ ← arbitrarily select a flaw in flaws
11     open ← open ∪ solveFlaw(π, φ)
12 return Failure;
```

π_0 a w_0 comme réseau de tâches et deux tâches instantanées spéciales t_0 et t_∞ ordonnées respectivement comme la première tâche et la dernière tâche de w_0. t_0 n'a pas de précondition et l'état initial comme effet positif. t_∞ a l'objectif comme précondition et aucun effet. Ensuite (ligne 2), π_0 est ajouté à la liste en attente des plans partiels à explorer et la première étape de raffinement démarre. À chaque itération, un nouveau plan partiel π est sélectionné (ligne 4) et ses défauts sont calculés (ligne 5). Si π contient des défauts, l'un d'eux est sélectionné (ligne 10). La résolution de ce défaut génère un nouvel ensemble de plans temporels partiels qui sont ajoutés à la liste en attente des plans partiels à explorer (ligne 11). Si un plan sans défaut est sélectionné (ligne 7), TEP passe à la deuxième étape et utilise un solveur CSP pour trouver l'ensemble V des affectations des variables temporelles pour toutes les tâches instantanées de π. Si le solveur CSP réussit, π et V sont renvoyés, sinon la procédure continue d'itérer, retourne à la première étape et essaie de raffiner un nouveau plan partiel.

3.2 Réparation des défauts

Les défauts dans les plans partiels sont très similaires aux défauts dans les réseaux de tâches non temporels. Nous avons 3 types différents de défauts : (1) des conditions ouvertes, i.e., des préconditions ou des postconditions de tâches qui ne sont pas supportées par un lien de causalité, (2) des menaces, i.e., des tâches qui peuvent interférer avec un lien de causalité, ou (3) des défauts de décomposition, i.e., des tâches abstraites ou duratives qui ne sont pas encore décomposées en tâches instantanées. Les défauts sont réparés comme suit.

Conditions ouvertes. Nous distinguons deux mécanismes de réparation pour les conditions ouvertes en fonction de savoir si une condition ouverte p d'une tâche $t_j \in \pi$ est une précondition ou une postcondition (voir Figure 2). Si p est une *précondition*, alors p est réparé en ajoutant à π un *lien de causalité* $\langle t_i \xrightarrow{p} t_j \rangle$ et une contrainte d'ordonnancement $(t_i < t_j)$. Si p est une *postcondition*, alors p

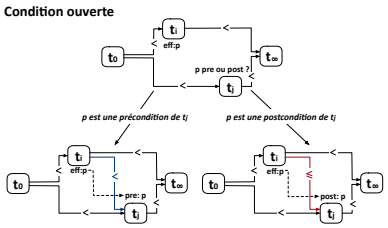

FIGURE 2 – Réparation d'une condition ouverte : Un lien de causalité $\langle t_i \xrightarrow{p} t_j \rangle$ est ajouté au plan pour soutenir la condition ouverte, et une contrainte de précédence ($t_i < t_j$) ou ($t_i \leq t_j$) selon que la condition ouverte est une *précondition* ou une *postcondition*.

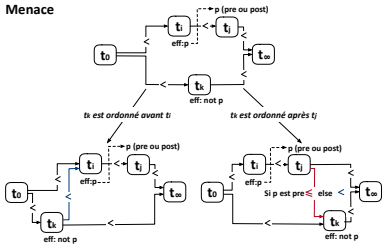

FIGURE 3 – Il existe deux façons de résoudre une *menace* t_k sur un lien de causalité $\langle t_i \xrightarrow{p} t_j \rangle$, on peut soit ordonner la tâche menaçante strictement avant t_i, soit en même temps (ou après) la tâche t_j selon que la condition ouverte soutenue est une précondition ou une postcondition.

est réparé en ajoutant à π un *lien de causalité* $\langle t_i \xrightarrow{p} t_j \rangle$ et une contrainte d'ordonnancement ($t_i \leq t_j$). Dans les deux cas, les contraintes ajoutées doivent être cohérentes avec les contraintes d'ordonnancement actuelles dans π. Notez que la réparation des préconditions ouvertes est effectuée comme dans la planification hybride non temporelle, tandis que la réparation des postconditions ouvertes introduit des contraintes $\leq$ au lieu de $<$.

Menaces. Supposons qu'une tâche t_k menace un lien de causalité $\langle t_i \xrightarrow{p} t_j \rangle$. Cette menace peut être résolue de deux manières différentes (voir Figure 3). La première consiste à contraindre t_k strictement avant la tâche t_i qui produit p en ajoutant ($t_k < t_i$) dans les contraintes d'ordonnancement. La deuxième consiste à contraindre t_k après ou en même temps que la tâche t_j en ajoutant la contrainte ($t_j \leq t_k$) si p est une précondition de t_j, ou strictement après si p est une postcondition de t_j. Comme toujours, l'ajout de contraintes d'ordonnancement à π doit garantir la cohérence des contraintes. La réparation d'une menace est très similaire à la réparation de menace dans la planification hybride, sauf que nous autorisons des contraintes d'ordonnancement non strictes.

Défauts de décomposition. Nous distinguons deux mécanismes de réparation différents selon que $\pi = (w, C)$

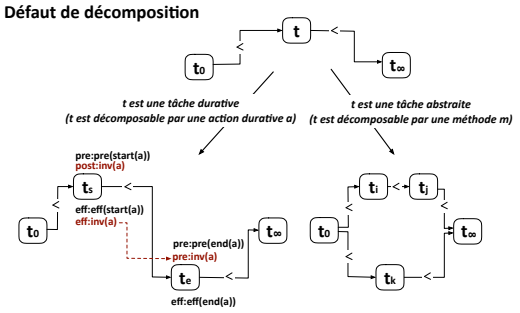

FIGURE 4 – Les deux façons possibles de décomposer une tâche dépendent de savoir si la tâche est durative (à droite) ou abstraite (à gauche).

contient une tâche durative ou abstraite t à décomposer en appliquant respectivement les Équations 1 et 2. La Figure 4 montre un exemple de décomposition de tâche durative (à droite) et de décomposition de tâche abstraite (à gauche). Pour la décomposition de tâche durative, nous supposons que t est décomposée par une action durative a telle que $t_s = start(a)$ et $t_e = end(a)$. Nous montrons en rouge les modifications des deux tâches instantanées t_s et t_e nécessaires pour traduire les propriétés invariantes $inv(a)$ dans la logique POCL. Pour la décomposition de tâche abstraite, nous supposons que t est décomposée par une méthode m en trois sous-tâches t_i, t_j, t_k avec $t_i < t_k$. Aucune autre contrainte ne lie t_i, t_j et t_k.

3.3 Un exemple de la troisième catégorie de Cushing

Dans cette partie, nous illustrons comment TEP gère la troisième catégorie de la classification de Cushing où les plans de solution nécessitent une concurrence nécessaire.

Supposons que deux tâches duratives t_a et t_b peuvent être décomposées par deux actions duratives a et b. La chronologie de l'exécution de a et b est donnée par la Figure 5 : a et b ont la même durée d. L'action a produit l'effet *(doing a)* et a *(doing b)* comme invariant, et symétriquement, b produit l'effet *(doing b)* et a *(doing a)* comme invariant. Le problème n'a pas de méthodes, et l'état initial et l'état final du problème sont vides. Le réseau de tâches initial du problème contient les tâches t_a et t_b sans contraintes d'ordonnancement. Le plan partiel obtenu par TEP après la décomposition de t_a et t_b en tâches instantanées en appliquant les actions duratives a et b, est représenté dans la Figure 6. La seule solution à ce problème est un plan concurrent où t_a et t_b sont exécutées en même temps.

La Figure 6 représente le plan partiel obtenu après la décomposition de t_a et t_b avec a et b. Il présente quatre défauts. Deux défauts sont des conditions ouvertes qui doivent être soutenues par des liens causaux : la postcondition *(doing a)* de la tâche t_a^s et la postcondition *(doing b)* de la tâche t_b^s. Les deux autres défauts sont des menaces : la tâche t_a^e menace le lien causal $\langle t_b^s \xrightarrow{(doing\ a)} t_b^e \rangle$ et la tâche t_b^e menace le

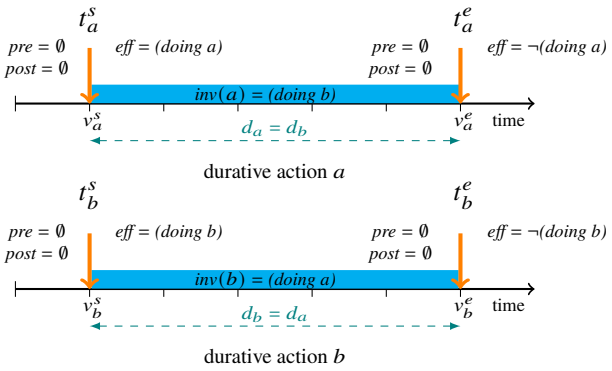

FIGURE 5 – Les deux actions duratives du problème de Cushing

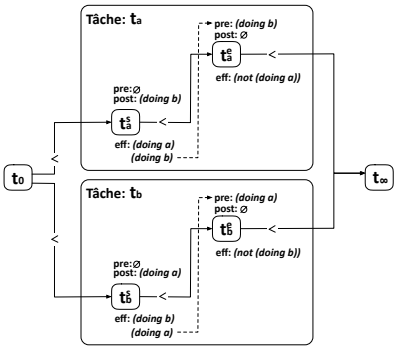

FIGURE 6 – Plan partiel obtenu dans le problème de Cushing après avoir décomposé les deux tâches duratives t_a et t_b en appliquant les actions duratives a et b. Les flèches en pointillés représentent les liens causaux et les flèches pleines en noir représentent les contraintes d'ordonnancement.

lien causal $\langle t_a^s \overset{(doing\ b)}{\longrightarrow} t_a^e \rangle$.

Supposons que les conditions ouvertes soient sélectionnées et résolues en premier. Pour corriger ces deux défauts, TEP doit ajouter au plan partiel deux liens causaux, le premier $\langle t_b^s \overset{(doing\ b)}{\longrightarrow} t_a^s \rangle$ pour soutenir la postcondition *(doing b)* de t_a^s et le deuxième $\langle t_a^s \overset{(doing\ a)}{\longrightarrow} t_b^s \rangle$ pour soutenir la postcondition *(doing a)* de t_b^s. En plus de ces deux liens causaux, TEP doit ajouter deux contraintes d'ordonnancement indiquant que t_a^s doit être exécuté avant ou en même temps que t_b^s puisque t_a^s supporte maintenant une postcondition de t_b^s, et inversement, que t_b^s doit être exécuté avant ou en même temps que t_a^s puisque t_b^s supporte maintenant une postcondition de t_a^s. Le plan partiel résultant est représenté dans la Figure 7. Les liens causaux et les contraintes d'ordonnancement ajoutés sont indiqués en rouge.

Finalement, les menaces doivent être corrigées. En général, une menace est résolue en contraignant la tâche instantanée menaçante soit *strictement* avant le lien causal menacé, soit *strictement ou en même temps* après le lien causal, selon

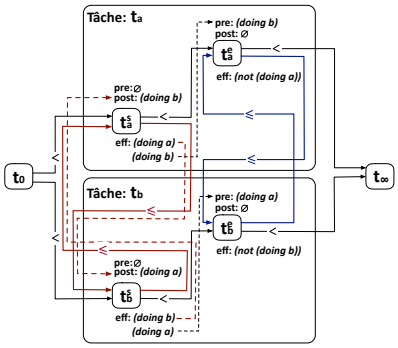

FIGURE 7 – Le plan partiel correspondant au problème de Cushing après la résolution des deux conditions ouvertes et des deux menaces. Les liens causaux et les contraintes d'ordonnancement en rouge représentent la modification du plan partiel nécessaire pour corriger les conditions ouvertes, et en bleu pour corriger les menaces.

que le lien causal supporte une précondition ou une postcondition. Dans notre exemple, la seule résolution valide est de contraindre t_b^e (resp. t_a^e) après ou en même temps que t_a^e (resp. t_b^e). Le plan partiel résultant est affiché sur la figure 7. Les contraintes d'ordonnancement ajoutées sont indiquées en bleu. TEP a maintenant résolu tous les défauts. TEP renvoie le plan partiel de la figure 7 comme solution.

4 Les heuristiques de TEP

Les heuristiques de TEP ont pour principal avantage d'exploiter (avec certaines adaptations présentées ci-dessous) les heuristiques développées pour la planification hybride non temporelle afin de résoudre des problèmes temporels. La procédure de recherche de TEP (voir Algo. 1) repose sur deux choix non déterministes, qui sont en pratique réalisés en utilisant deux fonctions heuristiques. Le premier choix effectue une sélection non déterministe parmi l'ensemble des plans partiels temporels en attente à explorer, et décide lequel explorer en premier (ligne 4). Les fonctions heuristiques guidant ce choix sont appelées *heuristiques de sélection de plan* et impactent considérablement à la fois les performances de la recherche (le temps nécessaire pour trouver un plan de solution) et la qualité du plan retourné (le nombre d'actions dans le plan de solution).

Le deuxième choix (ligne 9) effectue une sélection non déterministe parmi les défauts à résoudre dans le plan partiel actuel. Les fonctions heuristiques guidant ce choix sont appelées *heuristiques de sélection de défaut*. Notez que chaque défaut dans le plan partiel doit éventuellement être résolu afin de trouver un plan de solution. Ainsi, la *faisabilité* de la procédure TEP ne dépend que des heuristiques de sélection de plan. Cependant, l'*ordre* dans lequel les défauts sont résolus par rapport aux heuristiques de sélection de défaut impacte considérablement les performances de recherche de la procédure. Dans ce qui suit, nous présen-

tons les heuristiques de sélection de plan et de défaut telles qu'implémentées dans TEP.

4.1 Les heuristiques de sélection de plan

Parmi les heuristiques de sélection de plan pour la planification hybride non temporelle, e.g., [24, 23, 6], les plus efficaces sont celles développées par [6]. Leur principe est d'estimer le nombre de changements nécessaires pour atteindre une solution de plan en comptant le nombre de défauts dans le plan partiel actuel. L'idée de Bercher est d'étendre ce principe à la planification hybride en estimant plus précisément le nombre de défauts dans un plan en tenant compte des défauts qui seront introduits par les tâches qui n'ont pas encore été décomposées. Cette estimation est basée sur une structure appelée Graphes de Décomposition de Tâches (TDG) qui encode la décomposition du problème hiérarchique. Dans la section suivante, nous présentons (1) comment TEP étend le concept de TDG pour encoder non seulement les décompositions de tâches abstraites mais aussi les décompositions de tâches duratives dans une nouvelle structure appelée Graphes de Décomposition de Tâches Temporelles (TTDG), et (2) comment les heuristiques classiques pour la planification hybride développées dans [6, 5] peuvent être dérivées à partir d'un TTDG pour résoudre les problèmes de planification hybride temporelle.

Graphes de Décomposition de Tâches Temporelles. Un TTDG d'un problème de planification $P = (L, T, A, M, s_0, w_0, g)$ est un graphe orienté AND/OR $G = (V_T, V_M, V_A, E)$, où V_T est un ensemble de sommets comprenant des tâches instantanées, duratives et abstraites qui peuvent être obtenues en décomposant le plan partiel initial π_0 construit à partir de s_0, w_0, et g. V_M et V_A sont respectivement des ensembles de sommets de méthode ou de durative qui décomposent une tâche abstraite ou une tâche durative dans V_T. Enfin, E est un ensemble d'arêtes qui relient les nœuds de V_T à V_M ou V_A. Pour des raisons de brièveté, les nœuds enfants d'un nœud v sont notés $child(v) = v_i | (v, v_i) \in E$. Un TTDG est illustré dans la Figure 8. Dans le cas général, un TTDG comme un TDG peut être un graphe cyclique.

Le calcul des heuristiques. Pour généraliser les différentes heuristiques de sélection de plan développées en planification hybride à la planification temporelle, nous devons d'abord définir deux estimations cruciales qui peuvent être extraites d'un TTDG : (1) une estimation du nombre de décompositions nécessaires pour décomposer le plan en tâches instantanées, et (2) une estimation des conditions ouvertes à prendre en charge.

Ces deux estimations reposent sur les concepts de tâches obligatoires et de cardinalité des tâches. Les tâches obligatoires, notées $M(t)$, sont des tâches qui apparaissent dans toutes les méthodes de décomposition ou actions duratives

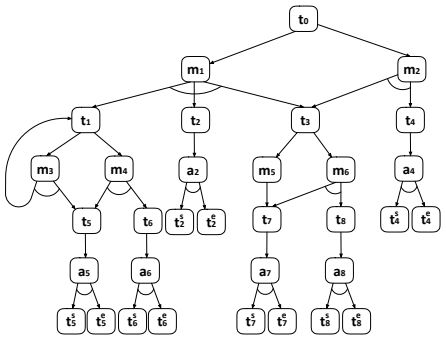

FIGURE 8 – Un exemple simple de TTDG représenté sous forme de graphe AND/OR. Les symboles t_0, t_1, t_3 représentent des tâches abstraites. $t_4, t_5 \ldots, t_8$ représentent des tâches duratives et $t_4^s, t_4^e, \ldots, t_8^s, t_8^e$ représentent des tâches instantanées. $m_1, \ldots m_6$ décrivent des sommets de méthode et $a_5, \ldots a_8$ des sommets d'actions duratives.

associées à la même tâche t. En termes plus simples, l'ensemble des tâches obligatoires $M(t)$ inclut les tâches qui seront obligatoirement incluses dans un plan partiel lorsque la tâche t est décomposée. Par exemple (voir Figure 8 pour le contexte), nous avons $M(t_o) = t_3, t_7, t_7^s, t_7^e$, $M(t_1) = t_5, t_5^s, t_5^e$, $M(t_3) = t_7, t_7^s, t_7^e$, $M(t_4) = t_4^s, t_4^e$, $M(t_5) = t_5^s, t_5^e$, $M(t_6) = t_6^s, t_6^e$, $M(t_7) = t_7^s, t_7^e$ et $M(t_8) = t_8^s, t_8^e$. La cardinalité d'une tâche t noté TC peut être définie de manière simple comme le nombre de tâches obligatoires, i.e., $TC = |M(t)|$. La cardinalité d'une tâche t sert de borne inférieure pour estimer l'effort requis pour la décomposer.

Le calcul de la première estimation revient à calculer TC. En pratique, cela est déterminé à partir d'un TTDG de la manière suivante :

$$TC(v) = \begin{cases} 1 & \text{si } v \in V_T \text{ et } v \text{ est instantanée} \\ \min\limits_{v_i \in child(v)} TC(v_i) & \text{si } v \in V_T \text{ et } v \text{ est abstraite} \\ \sum\limits_{v_i \in child(v)} TC(v_i) & \text{sinon} \end{cases}$$

La deuxième estimation, connue sous le nom de *MME* (Effort de Modification Minimal), représente le nombre de conditions ouvertes, comprenant à la fois les préconditions et les postconditions, qui doivent être satisfaites pour chaque nœud. Elle peut être déterminée à partir d'un TTDG de la manière suivante :

$$MME(v) = \begin{cases} 1 + |pre(v) \cup post(v)| & \text{si } v \in V_T \text{ et } v \text{ est inst.} \\ \sum\limits_{d \in Child} TC(d) & \text{si } v \in V_M. \\ 1 + \min\limits_{d \in Child} TC(d) & \text{si } v \in V_T \text{ et } v \text{ est abst.} \end{cases}$$

où $pre(v)$ désigne les préconditions d'un nœud primitif et $post(v)$ ses postconditions.

Il est finalement possible d'étendre le calcul des heuristiques à un plan temporel $\pi = (w, C)$ aux quatre heuristiques classiques proposées par [6, 5] comme suit. Ici,

$w = (I, \alpha, <)$ définit les tâches et leurs contraintes associées, tandis que C représente les liens de causalité entre les tâches au sein de π, et F désigne ses défauts.

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

nombre de modifications qui seront introduites dans le plan, comme mentionné dans [5]. Cependant, elles ne garantissent pas un plan de solution optimal en ce qui concerne le makespan. La preuve est omise par manque de place.

4.2 Heuristiques de sélection des défauts

La stratégie la plus courante pour sélectionner l'ordre des défauts est de choisir en premier les défauts ayant le moins de moyens de résolution possibles, ce qui signifie que les défauts les plus contraignants sont résolus en premier. Cette stratégie est connue sous le nom de LCFR (Least Cost Flaw Repair) [13]. Bien que cette heuristique de sélection de défauts ait été initialement développée pour des problèmes non hiérarchiques, elle a été utilisée en planification hybride par le planificateur hybride de pointe PANDA [6]. Lors de son adaptation à la planification hiérarchique, la priorité est donnée à la décomposition des tâches abstraites par rapport aux autres défauts pour maintenir la complétude de la procédure. Cela est dû au fait que le nombre de moyens de résolution pour un défaut dépend du choix de la décomposition.

TEP met en œuvre la même stratégie de sélection des défauts que PANDA, en donnant la priorité aux défauts de décomposition, puis en priorisant les autres défauts à partir de ceux ayant le moins de moyens de résolution. La seule différence ici est que les défauts de décomposition comprennent non seulement les tâches abstraites, mais aussi les tâches duratives qui ne sont pas décomposées en tâches instantanées.

5 Expérimentations

Dans cette partie, nous comparerons TEP et ses heuristiques avec l'approche *timeline* proposée par FAPE, retenue comme la meilleure configuration [7]. L'approche *timeline* est actuellement la seule approche capable de résoudre des problèmes temporels hiérarchiques dans la troisième catégorie de Cushing. Ainsi, à notre connaissance, seule l'approche *timeline* et TEP partagent la même expressivité.

5.1 Configuration Expérimentale

Les deux planificateurs, TEP et FAPE, ont été testés sur un seul cœur d'un processeur Intel Core i7-9850H, avec une limite de 8 Go de RAM et une limite de temps de 600 secondes. Pour garantir une comparaison équitable, les deux planificateurs ont été mis en œuvre en utilisant la même bibliothèque PDDL4J [20], et ont utilisé le même solveur CSP, celui fourni par OR-Tools [21]. Les critères d'évaluation sont (1) le *temps de résolution*, qui représente le temps nécessaire pour résoudre un problème, (2) le *makespan* d'un plan de solution, qui représente la longueur totale du plan, i.e., le temps entre le début et la fin du plan de solution, et enfin, (3) la *couverture* de chaque planificateur, i.e., le nombre de problèmes résolus de chaque domaine. Notez que le temps de résolution et le makespan sont présentés sous forme de scores IPC [3] (International Planning Competition). La meilleure configuration atteint un score IPC de 1, tandis que les autres configurations reçoivent un pourcentage basé sur leurs performances par rapport à la meilleure. Pour calculer le score d'un domaine, nous additionnons les scores IPC obtenus pour chaque problème. La couverture fait référence au nombre de problèmes résolus dans un domaine.

5.2 Benchmarks de planification

Il n'existe actuellement aucun benchmark standard disponible pour la planification hiérarchique et temporelle. Par conséquent, nous proposons un ensemble de benchmarks basés sur le langage HDDL2.1 [19], qui vise à faciliter la comparaison des planificateurs hiérarchiques et temporels. Parmi ces benchmarks, certains sont des adaptations de domaines hiérarchiques non temporels des compétitions de planification IPC. Ils ont été modifiés en ajoutant des durées aux actions : *Gripper*, *Satellite* et *Rover*. Dans le domaine *Gripper*, toutes les solutions sont séquentielles et aucune concurrence temporelle n'est requise (première catégorie de Cushing). Dans les domaines *Satellite* et *Rover*, des problèmes avec plusieurs satellites ou rovers permettent la concurrence, mais elle n'est pas obligatoire. Le planificateur peut choisir de trouver soit un plan simple et non concurrent, soit un plan concurrent utilisant plusieurs rovers. Ces problèmes appartiennent à la deuxième catégorie de Cushing. De plus, certains domaines ont été spécialement conçus pour correspondre à la troisième catégorie de Cushing :

— *Area Scan* modélise un ensemble de dispositifs hétérogènes qui coopèrent pour scanner des zones. Area Scan a deux versions : une version totalement ordonnée, où les tâches abstraites des problèmes sont totalement ordonnées, et une version désordonnée où la concurrence peut également être réalisée grâce à des tâches abstraites concurrentes.
— *Table Carriers* est inspiré du domaine *Gripper*. Dans ce domaine, les agents doivent coopérer pour transporter des tables, les tables nécessitant une, deux ou trois personnes pour les transporter simultanément avec succès.

3. https://ipc2023-htn.github.io/

	Gripper			Satellite			Rover			AreaScan PO			AreaScan TO			TableCarriers			Total		
	Temps	Span	Couv.	Temps	Span	Couv.	Temps	Span	Couv.	S. Temps	Span	Couv.	Temps	Span	Couv.	Temps	Span	Couv.	S. Temps	Span	Couv.
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 1 – Résultats expérimentaux pour les différentes configurations sous forme de scores IPC (International Planning Competition) pour le temps, le makespan et la couverture.

5.3 Résultats

Les résultats sont présentés dans le Tableau 1. En termes de critère de *temps de résolution*, TEP a constamment surpassé FAPE dans tous les domaines. Cependant, le choix des heuristiques les plus performantes variait selon le domaine. Dans les domaines Gripper et Satellite, TEP combiné avec les heuristiques MME, appelé $TEP(MME)$, a obtenu les meilleurs scores. En revanche, dans le domaine Rover, la configuration $TEP(TDGm)$ a atteint le meilleur score pour la métrique du temps de résolution. Cette disparité peut être attribuée à la nature distinctive de MME et $TDGm$. Contrairement à MME, qui se concentre uniquement sur l'accomplissement de propositions au sein d'un plan partiel, les heuristiques $TDGm$ tiennent compte du nombre de liens causaux déjà intégrés dans le plan partiel. Par conséquent, tandis que $TEP(MME)$ affine continuellement un plan partiel particulier en ajoutant des liens causaux jusqu'à ce qu'il forme une solution ou soit élagué, $TDGm$ a tendance à présenter un comportement de type exploration en profondeur. L'approche contrastée en profondeur de $TDGm$ et la nature de type exploration en largeur de MME éclairent les différences dans l'efficacité de leurs heuristiques. Des différences similaires peuvent être observées entre TC et $TC+F$. Alors que la configuration $TEP(TC)$ obtient des scores supérieurs dans le domaine *AreaScan PO*, $TEP(TC + F)$ se distingue comme la configuration la plus performante dans le domaine *AreaScan TO*. Une fois de plus, l'incorporation du nombre de défauts restants dans $TC + F$ incite une stratégie de recherche en profondeur, ce qui s'avère avantageux pour la résolution de problème ou les plans solution sont assez longs. En revanche, le comportement de type exploration en largeur manifesté par les heuristiques TC conduit à de meilleures performances dans d'autres domaines.

Les différentes configurations de TEP donnent pratiquement toujours les meilleurs scores en termes de makespane. Dans les domaines *Gripper* et *AreaScan*, $TEP(TC)$ se distingue comme la configuration la plus performante. En revanche, dans le domaine *Satellite*, $TEP(TC + F)$ présente la meilleure performance, tandis que dans le domaine *TableCarriers*, $TEP(TDGm)$ surpasse les autres. De façon notable, FAPE obtient le meilleur score dans le domaine Rover. Dans l'ensemble, les configurations affichent des résultats comparables à travers les domaines testés. Les scores IPC semblent davantage influencés par la couverture des instances de domaine que par la qualité du plan, indiquant que les configurations comparées fournissent au moins des résultats comparables en termes de durée totale.

Finalement, en termes de couverture, TEP surpasse FAPE sur tous les domaines.

6 État de l'art

Les approches pour la résolution de problèmes de planification hiérarchique et temporelle peuvent être classées en fonction de leur efficacité à traiter les catégories de problèmes définies par Cushing [9].

La plus part des approches proposées peuvent résoudre des problèmes relevant des deux premières catégories grâce à différentes techniques. Une approche populaire consiste à adapter les planificateurs HTN non temporels pour gérer des formalismes temporels. Par exemple, SHOP (Simple Hierarchical Ordered Planner) [18] a été étendu pour devenir SHOP2 [2, 11] afin d'incorporer des actions avec des durées. D'autres avancées, comme GSCCB-SHOP2 [22], gèrent des contraintes temporelles et de ressources complexes. Un autre exemple est le planificateur SIADEX [1], qui applique une méthode de chaînage avant similaire à la planification classique, adaptée au contexte temporel. Il établit des méta-actions temporelles qui se lient séquentiellement pour élaborer un plan de solution. De plus, parmi les planificateurs s'attaquant aux deux premières catégories, on trouve le planificateur CHIMP [25]. CHIMP traduit les problèmes temporels et hiérarchiques en problèmes de satisfaction de contraintes pour dériver des plans de solution temporelle. Bien que ces planificateurs bénéficient de l'efficacité de la planification non temporelle, leurs méthodologies ont du mal à traiter pleinement la complexité de la planification temporelle, les rendant inadéquats pour les problèmes impliquant de la concurrence.

Les planificateurs capables de traiter la troisième catégorie de Cushing utilisent généralement des méthodes basées sur les timelines, surveillant le statut de chaque proposition au fil du temps. Parmi les exemples clés, on peut citer HSTS [16], plus tard affiné en EUROPA [3], un planificateur basé sur CSP. De plus, IxTeT [14] est indépendante du domaine et efficace pour gérer des timelines complexes. Sa dernière itération, FAPE [7], gère habilement l'ensemble des expressions de concurrence nécessaires.

7 Conclusion

Dans cet article, nous avons présenté TEP, une approche pour résoudre à la fois des problèmes hiérarchiques et temporels. Elle consiste à relaxer les problèmes temporels en problèmes non temporels afin d'appliquer des heuristiques de recherche développées pour résoudre des problèmes non-temporels. Nous avons comparé notre approche avec une approche fondée sur les timelines partageant la même expressivité en termes de catégories de Cushing. Nous avons montré que notre approche surpassait cette dernière en termes de temps d'exécution pour trouver une solution, et que les deux approches étaient comparables en ce qui concerne la qualité des plans solution (makespan).

Références

[1] Marc Asunción, Luis Castillo, Juan Fdez-Olivares, Óscar García-Pérez, Antonio González-Muñoz, and Francisco Palao. Siadex : An interactive knowledge-based planner for decision support in forest fire fighting. *AI Commun.*, 18 :257–268, 01 2005.

[2] Tsz-Chiu Au, Okhtay Ilghami, Ugur Kuter, J. William Murdock, Dana S. Nau, Dan Wu, and Fusun Yaman. Shop2 : An htn planning system. *J. Artif. Intell. Res.*, 20 :379–404, 2003.

[3] Javier Barreiro, Matthew Boyce, Minh Do, Jeremy Frank, Michael Iatauro, Tatiana Kichkaylo, Paul Morris, James Ong, Emilio Remolina, Tristan Smith, et al. Europa : A platform for ai planning, scheduling, constraint programming, and optimization, 2012.

[4] Patrick Bechon, Magali Barbier, Guillaume Infantes, Charles Lesire, and Vincent Vidal. Hipop : Hierarchical partial-order planning. In *STAIRS*, pages 51–60, 2014.

[5] Pascal Bercher, Gregor Behnke, Daniel Höller, and Susanne Biundo. An Admissible HTN Planning Heuristic. In *IJCAI*, pages 480–488, 2017.

[6] Pascal Bercher, Shawn Keen, and Susanne Biundo. Hybrid Planning Heuristics Based on Task Decomposition Graphs. In *SOCS*, pages 35–43, 2014.

[7] Arthur Bit-Monnot, Malik Ghallab, Félix Ingrand, and David E. Smith. FAPE : a Constraint-based Planner for Generative and Hierarchical Temporal Planning. *ArXiv, abs/2010.13121.*, 2020.

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

[9] William Cushing. Evaluating Temporal Planning Domains. In *ICAPS*, pages 105–112, 2007.

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

[11] R. Goldman. Durative planning in htns. In *ICAPS*, pages 382–385, 2006.

[12] Daniel Höller and Pascal Bercher. Landmark generation in htn planning. In *AAAI*, pages 11826–11834, 2021.

[13] David Joslin and Martha E. Pollack. Least-cost flaw repair : A plan refinement strategy for partial-order planning. In *AAAI*, pages 1004–1009, 1994.

[14] Solange Lemai. *IXTET-EXEC : planning, plan repair and execution control with time and resource management*. PhD thesis, Institut National Polytechnique de Toulouse, 2004.

[15] David A. McAllester and David Rosenblitt. Systematic nonlinear planning. In *AAAI*, pages 634–639, 1991.

[16] Nicola Muscettola. HSTS : Integrating Planning and Scheduling. Technical report, Robotics Institute, Camegie Mellon University, 2013.

[17] Dana Nau, Yash Bansod, Sunandita Patra, Mark Roberts, and Ruoxi Li. Gtpyhop : A hierarchical goal+task planner implemented in python. In *HPlan Workshop (ICAPS)*, 2021.

[18] Dana S. Nau, Tsz-Chiu Au, Okhtay Ilghami, Ugur Kuter, J. William Murdock, Dan Wu, and Fusun Yaman. SHOP2 : an HTN planning system. *J. Artif. Intell. Res.*, 20 :379–404, 2003.

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

[20] Damien Pellier and Humbert Fiorino. PDDL4J : a planning domain description library for Java. *J. Exp. Theor. Artif. Intell.*, 30(1) :143–176, 2018.

[21] Laurent Perron, Frédéric Didier, and Steven Gay. The cp-sat-lp solver. In *CP*, pages 3 :1–3 :2, 2023.

[22] Chao Qi, Dan Wang, Héctor Muñoz-Avila, and Peng Zhao. Hierarchical task network planning with resources and temporal constraints. *Knowledge-Based Systems*, 133 :17–32, 2017.

[23] Bernd Schattenberg. *Hybrid planning & scheduling*. PhD thesis, University of Ulm, Germany, 2009.

[24] Bernd Schattenberg, Julien Bidot, and Susanne Biundo. On the construction and evaluation of flexible plan-refinement strategies. In *German Conference on AI*, pages 367–381, 2007.

[25] Sebastian Stock, Masoumeh Mansouri, Federico Pecora, and Joachim Hertzberg. Online task merging with a hierarchical hybrid task planner for mobile service robots. In *IROS*, pages 6459–6464, 2015.

[26] Håkan L. S. Younes and Reid G. Simmons. VHPOP : versatile heuristic partial order planner. *J. Artif. Intell. Res.*, 20 :405–430, 2003.