Planification distribuée pour la coopération multi-agents

Paul Gaborit

Année 1996

Thèse

préparée au

Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS

en vue de l'obtention du

Doctorat de l'Université Paul Sabatier de Toulouse

Spécialité : Informatique et Intelligence Artificielle

par

Paul GABORIT

Planification Distribuée pour la Coopération Multi-Agents

Soutenue le 27 Septembre 1996 devant le jury:

Président	**Georges**	**GIRALT**
Directeur de thèse	**Malik**	**GHALLAB**
Rapporteurs	**Anne**	**COLLINOT**
	Jacques	**FERBER**
	Jean-Paul	**HATON**
Examinateurs	**Rachid**	**ALAMI**
	Henri	**FARRENY**
	Louis	**FERRAUD**
	Patrick	**TALLIBERT**

LAAS-CNRS
7, Avenue du Colonel Roche
31077 Toulouse Cedex 4

Avant-propos

L'ensemble des travaux que je présente dans ce mémoire de thèse ont été réalisés au Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS) du Centre National de la Recherche Scientifique (CNRS), au sein du groupe Robotique et Intelligence Artificielle (RIA).

Je remercie Alain Costes, Directeur du LAAS de m'avoir accueilli dans son laboratoire.

Un grand merci à Georges Giralt de m'avoir permis de travailler dans le groupe de recherche RIA qu'il a dirigé avec tant de passion et, bien sûr, d'avoir accepté la présidence de mon jury de thèse.

Je tiens particulièrement à exprimer ma reconnaissance à Malik Ghallab, Directeur de recherche au CNRS, qui a su, tout au long de mon travail, me guider, me conseiller et m'aider grâce à sa disponibilité, à sa sympathie et surtout à son immense compétence scientifique. J'ai pu aussi apprécier son aide et ses conseils dans les moments difficiles.

Je tiens aussi à remercier Rachid Alami, Chargé de recherche au CNRS, dont la gentillesse, l'enthousiasme, la compétence, les remarques et les idées m'ont aidé à vaincre mes doutes et à approfondir ma réflexion.

Je remercie Anne Collinot, Jean-Paul Haton et Jacques Ferber de s'être donné la peine d'être les rapporteurs de ma thèse: leurs remarques et questions me furent précieuses pour améliorer et préciser mon travail.

J'adresse aussi ma reconnaissance à Louis Feraud, Henri Farreny et Patrick Tallibert pour avoir participer à mon jury et je veux les remercier des discussions que nous avons eues à cette occasion.

Ce travail n'aurait pu se faire sans l'aide des nombreuses personnes qui ont participé au projet IxTeT. Je pense, entre autres, à Amine, Hervé, Christophe, Thierry, Philippe, Jérôme, Patrick, Brigitte, Anders avec lesquelles j'ai souvent eu de longues discussions souvent animées et heureusement constructives.

Un grand merci à Matthieu, Sara, Philippe, Pierrick, Victor, Simon, Fred, Jackie pour leur aide, leur compétence, leur amitié et leur gentillesse de tous les instants.

J'ai aussi une pensée émue pour tous les membres permanents, doctorants ou sta-

giaires du groupe RIA que j'ai côtoyés au long de ces années. C'est grâce à eux tous que l'ambiance du groupe est si chaleureuse, enflammée, passionnée ... et donc passionnante. Tous ces moments d'amitié, de travail, de réflexion, de distraction et même parfois de conflit ou de franche rigolade resteront gravés en moi comme de très bons souvenirs.

Je veux aussi remercier l'École des Mines d'Albi et tout particulièrement Christian Desmoulins, son Directeur, et Christophe Levaillant, Chef du Centre Matériaux, qui m'ont permis de mener à bien ce travail en me libérant de mon travail d'enseignant-chercheur. Je salue au passage tous mes collègues et amis de l'École et particulièrement Luc, Jean-José et Denis.

Je ne saurais oublié tous mes amis et ma famille qui, chacun, à leur manière, m'ont accompagné et soutenu par leur amitié.

À Marie et Nicolas qui ont certainement supporté le plus dur mais qui ont été mon plus fidèle soutien.

Toulouse, Octobre 1996.

Sommaire

Introduction

L'objectif principal des travaux présentés dans ce mémoire est l'étude de la possibilité pour plusieurs agents d'élaborer des plans leur permettant d'atteindre les buts qui leur sont fixés tout en gérant leurs interactions entre eux et sur le monde. Les techniques usuelles de planification ne s'adaptent pas facilement dans un contexte réparti. En effet, une planification du problème complet intégrant l'ensemble des agents est souvent trop complexe. Quant à la méthode consistant à ne prendre en compte les interactions avec les autres que lors de l'exécution de plans individuels, elle ne suffit pas à résoudre dans le cas général tous les problèmes de partage du monde. C'est donc une approche intermédiaire que nous proposons permettant l'élaboration de plans individuels tenant compte des contraintes provenant des plans des autres, des synchronisations nécessaires et du partage des mêmes ressources tout en garantissant la cohérence du plan global.

Depuis longtemps, la planification est un des thèmes principaux de recherche en Intelligence Artificielle. L'objectif est de prévoir et d'organiser, en fonction du monde, les actions que doit réaliser le système afin de satisfaire des buts fixés.

Selon le degré de connaissance que le système a du monde, on distingue deux approches différentes. Lorsque l'évolution du monde est très prévisible, il est possible de planifier de nombreuses actions jusqu'aux buts et espérer que tout se déroulera comme prévu. C'est l'approche de planification prédictive. Au contraire lorsque l'évolution du monde est peu prévisible ou très imprécise, le système doit se contenter de prévoir quelques actions qui, en fonction de sa connaissance actuelle du monde, semblent le rapprocher du buts. Cette deuxième approche s'appelle la planification réactive. Ces deux approches ne s'opposent pas et souvent même, pour un même monde et selon le degré d'abstraction auquel on se situe, elles se complètent. Nos travaux se situe dans le cadre de la planification prédictive.

Pour mener à bien cette planification à long terme, on peut modéliser le monde de plusieurs manières. Les plus courantes se basent sur une description logique de l'état du monde. La recherche d'un plan consiste alors à trouver un chemin depuis l'état courant jusqu'à un état but; la transition d'un état à un autre s'effectue par une action qui modifie le monde. Notre approche, quant à elle, effectue sa recherche dans l'espace des plans partiels. Un plan partiel contient la description de l'ensemble de l'évolution du monde sous la forme de formules logiques. Le planificateur le complète peu à peu, en y insérant, par exemple, de nouvelles tâches afin de le rendre cohérent. Ceci permet, contrairement à la représentation à base d'états, de prendre en compte explicitement le parallélisme des actions, les changements du monde contingents (c.-à-d. en dehors du contrôle du planificateur) et les contraintes temporelles numériques.

Les univers multi-agents sont un domaine d'étude récent qui intéresse de nombreuses disciplines. L'application des techniques de planification aux multi-agents est un domaine encore largement ouvert.

Pour être considéré comme un agent, un système intégrant un planificateur doit être capable d'élaborer un plan en fonction des informations qu'il possède sur le monde et sur le comportement des autres agents, puis de l'exécuter en coopérant avec les autres. Si l'on considère des agents similaires, le comportement adopté par les autres agents sera lui aussi le résultat d'une planification. Chaque agent doit alors pouvoir prendre en compte le plans des autres pour élaborer son propre plan afin de s'assurer de la cohérence globale des actions de tous, de garantir la bonne utilisation des ressources disponibles et enfin de réaliser la synchronisation correcte des agents entre eux.

Dans l'approche que nous proposons, chaque agent, dans un premier temps, élabore son propre plan. Dans un second temps, il essaie de composer ce plan avec les plans provenant des agents qui ont prévu d'agir sur les mêmes attributs ou ressources du monde. Cette composition permet à l'ensemble des agents de constituer de manière incrémentale un plan global sans qu'à aucun moment, ce plan global soit centralisé.

Afin de gérer au mieux le parallélisme entre les tâches des plans des différents agents, ces plans doivent contenir des informations numériques sur les durée prévues de ces tâches ou plus précisément sur les durées séparant les changements d'état des attributs ou déterminant l'utilisation des ressources communes. C'est l'une des raisons qui nous a poussé à choisir IxTeT comme système de planification.

Ce système que nous présentons est développé au LAAS depuis 1989. À partir de la

première version, écrite en LISP et ne gérant que des contraintes temporelles symboliques, IxTeT s'est peu à peu enrichi pour devenir un système complet intégrant autour d'un même formalisme un compilateur de plans partiels, un logiciel de suivi de chroniques et un planificateur gérant, entre autre, les ressources et le parallélisme des tâches. Il est le fruit du travail de plusieurs chercheurs. Actuellement, ce système contient plus de 80000 lignes de C++. Nous avons participé à son développement depuis le début et nous en assurons actuellement la maintenance.

Nous tentons donc de formaliser et d'intégrer cette capacité de planification distribuée dans le système IxTeT en proposant des opérateurs de composition de plans permettant de distribuer le calcul du plan global entre les différents agents.

Dans le premier chapitre, nous exposons la représentation formelle adoptée qui repose sur une logique temporelle réifiée. Nous y définissons les notions d'événements, d'assertions de persistance et de propositions d'utilisation de ressources.

Le second chapitre est consacré entièrement au fonctionnement du système de planification IxTeT. Une fois décrite la notion de cohérence des plans, nous y présentons, en y apportant quelques améliorations, les méthodes permettant de construire peu à peu un plan en supprimant les défauts qu'il contient ainsi que la stratégie utilisée par IxTeT pour réduire la complexité du problème.

C'est dans le troisième chapitre que nous exposons notre méthode permettant de réaliser des opérateurs de composition de plans. Nous justifions formellement (en nous appuyant sur le formalisme présenté au premier chapitre) la technique de rupture des chaînes de causalité et son utilisation (dans le fonctionnement du système tel qu'il est présenté dans le second chapitre) pour réaliser une union de plans ou une insertion de nouveaux buts.

La mise en œuvre de ces nouveaux opérateurs inter-plans nous permet, dans le quatrième chapitre, d'exhiber un algorithme de planification incrémentale et un autre de planification distribuée. Puis, en présentant quelques éléments permettant de réaliser de la supervision de plan à partir du système de reconnaissance de chroniques d'IxTeT, nous mettons en évidence les problèmes ouverts que posent les liens entre planification et exécution et qui sont critiques dans le cadre d'une distribution de la planification entre plusieurs agents.

Le dernier chapitre illustre, par quelques exemples, l'utilité et les performances des opérateurs de composition de plans et se termine par une comparaison entre O-Plan et

I_XT_ET.

Nous avons réuni en annexe les algorithmes de gestion des contraintes (temporelles ou entre les variables) ainsi qu'un exemple complet de description d'un domaine de planification exprimé dans le langage de programmation d'I_XT_ET.

Chapitre I

La logique temporelle utilisée dans $I_X T_E T$

Afin d'élaborer un raisonnement sur le monde, il est nécessaire de le modéliser. Ce modèle doit s'appuyer sur une représentation logique afin de prouver la validité du raisonnement. Dans ce chapitre, nous présentons formellement la logique temporelle utilisée dans $I_X T_E T$.

La logique dont nous avons besoin se doit de pouvoir représenter le temps et l'action. Au-delà des logiques classiques, nous aurions pu nous tourner vers les logiques modales qui permettent d'exprimer le temps mais aussi la nécessité ou tout autre type de modalité [Gardiès 79] [Konolige 84]. En mixant des modalités temporelles et épistémiques (portant sur le savoir), il est même possible de modéliser un ensemble d'agents agissant sur le monde [Gaborit 90]. Si ces logiques ont un grand pouvoir expressif, leur utilisation pratique se heurte à un problème de complexité. C'est pourquoi, afin de pouvoir manipuler le temps explicitement, la logique utilisée par $I_X T_E T$ est une logique réifiée par le temps. $I_X T_E T$ n'est pas pour autant un démonstrateur de théorèmes de cette logique. Elle ne lui sert qu'à des fins descriptives. Il est nécessaire, en particulier, de fournir une description complète du monde afin de garantir la cohérence et la [illegible] la logique utilisée par $I_X T_E T$

Pour des soucis de performance [Vilain 86], nous nous sommes basés sur des instants plutôt que sur des intervalles temporels [Allen 81] [Allen 83b] tout en essayant d'intégrer suffisamment d'opérateurs provenant de l'algèbre des intervalles. La représentation de l'état du monde s'effectue sur la base d'attributs multi-valués sur lesquels portent des prédicats temporels de changement et de persistance [Shoham 87].

IxTeT intégrant la notion d'utilisation de ressources, nous terminerons ce chapitre en présentant les trois prédicats liés à ces aspects et en précisant la sémantique qui leur est attachée.

I.1 Les domaines et les notations

Dans cette section sont définis tous les ensembles et les notations nécessaires pour présenter formellement la logique utilisée par IxTeT.

I.1.1 Le temps

Dates et durées

La définition de l'ensemble des dates et durées représentant le temps n'est pas un problème simple. Dans notre représentation, les seules hypothèses posées sur cet ensemble sont les suivantes :

- Que l'on puisse définir une relation d'ordre totale sur l'ensemble des dates et des durées (cette relation d'ordre totale est traditionnellement notée $<$).
- Qu'il existe deux éléments particuliers permettant de borner l'ensemble des dates et durées (ils sont traditionnellement notés $-\infty$ et $+\infty$).
- Qu'il existe une loi de composition interne définissant un groupe commutatif (elle est traditionnellement notée $+$ et son élément neutre est 0).
- Que cet ensemble soit dense (c.-à-d. que, quelque soit d_1 et d_2 deux éléments distincts de notre ensemble tel que $d_1 < d_2$ alors il existe d' tel que $d_1 < d' < d_2$).

Cette dernière hypothèse est une hypothèse forte que l'implémentation informatique d'IxTeT ne respecte pas puisque les dates et durées y sont représentées par des entiers.

C'est en fait une discrétisation de notre ensemble que nous avons implémentée. Il s'avère que cette limitation n'est pas rédhibitoire actuellement. Il suffit en fait de s'assurer que la granularité choisie est suffisante pour le problème que l'on veut traiter.

Dans IxTeT l'unité de base choisie (par défaut) représente un centième de seconde ce qui suffit amplement pour le type de problèmes que nous voulions résoudre (où les plans durent plusieurs minutes voire plusieurs heures). Il suffirait d'accorder une autre valeur sémantique à l'unité de base pour permettre des traiter des problèmes à d'autres échelles temporelles.

Cette limitation explicite de l'implémentation d'IxTeT ne nous permet pas de traiter des problèmes avec de fortes variations d'amplitude temporelle. Pour ce type de problème, il nous faudrait utiliser une technique gérant une granularité variable (qui serait une manière élégante d'implémenter la notion théorique de densité).

Dans ce mémoire, nous supposerons que les dates et durées appartiennent à $\bar{\mathbb{R}}$ (L'ensemble des réels augmentés de $+\infty$ et $-\infty$).

Les intervalles temporels

A partir de la définition de cet ensemble nous pouvons définir **I**, l'ensemble des intervalles temporels par :

$$\mathbf{I} = \left\{ [I^-, I^+] \in \bar{\mathbb{R}}^2, \text{ tel que } I^- \leqslant I^+ \right\} \cup \{\emptyset\}$$

On définit l'inverse $(-)$ d'un intervalle, la conjonction $(\cap)$ et la composition $(\oplus)$ de deux intervalles par[1] :

$$\begin{aligned} -I &= [-I^+, -I^-] \\ I \oplus J &= [I^- + J^-, I^+ + J^+] \\ I \cap J &= \begin{cases} [\max(I^-, J^-), \min(I^+, J^+)] & \text{si } \max(I^-, J^-) \leqslant \min(I^+, J^+) \\ \emptyset & \text{si } \max(I^-, J^-) > \min(I^+, J^+) \end{cases} \end{aligned}$$

Les comparaisons logiques d'inclusion $(\subseteq)$, d'inclusion stricte $(\subset)$ et d'égalité $(=)$

1. Toute opération dont l'un des opérandes est l'intervalle vide produit comme résultat l'intervalle vide. Par définition, on admet que $+\infty + (-\infty) = 0$

entre intervalles sont classiquement définies par:

$$\begin{aligned} I = J &\equiv (I^- = J^-) \text{ et } (I^+ = J^+) \\ I \subset J &\equiv \big((I^- > J^-) \text{ et } (I^+ \leqslant J^+)\big) \text{ ou } \big((I^- \geqslant J^-) \text{ et } (I^+ < J^+)\big) \\ I \subseteq J &\equiv (I^- \geqslant J^-) \text{ et } (I^+ \leqslant J^+) \end{aligned}$$

Nous pouvons remarquer que:

- Ces deux opérations binaires sont commutatives
- La composition ($\oplus$) est *presque*[2] distributive sur la conjonction ($\cap$)
- L'inclusion est conservée par composition ($\oplus$)

Les instants

Nous définissons **Tp** comme l'ensemble des instants (ce sont des variables temporelles). Nos prédicats temporels s'appuierons sur ces instants. Nous posons différents types de contraintes entre ces instants:

- Des contraintes d'ordre de type symbolique ($t_1 = t_2$, $t_1 < t_2$, *etc.*).
- Des contraintes d'ordre numérique ($Durée_{min} < (t_2 - t_1) < Durée_{max}$).

I.1.2 Les attributs du monde et les variables

Dans la logique d'IxTeT, le monde est représenté par un ensemble d'attributs paramétrés par des variables (l'arité d'un attribut donné est fixe). Ces attributs sont multi-valués, mais, à un instant donné, on ne donne à chacun qu'une et une seule valeur. Par exemple, l'attribut suivant décrit la position d'un objet donné comme argument:

$$\text{Position}\,(?objet) = \text{Piece1}$$

2. La preuve de l'algorithme de propagation complète des contraintes temporelles (Algorithme A.1 page 124) s'appuie explicitement sur cette distributivité. Mais la composition n'est distributive sur la conjonction que lorsque le résultat de celle-ci n'est pas l'intervalle vide. Heureusement, dans l'algorithme, la rencontre d'un intervalle vide signifie l'incohérence des contraintes. Cette quasi-distributivité suffit donc pour propager correctement l'ensemble des contraintes lorsqu'il est cohérent.

hold (P $(?v_1, \ldots, ?v_n) : ?v, (t, t')$)	La valeur de l'attribut P $(?v_1, \ldots, ?v_n)$ reste à $?v$ durant tout l'intervalle temporel $[t, t'[$
event (P $(?v_1, \ldots, ?v_n) : (?v, ?v'), t$)	La valeur de l'attribut P $(?v_1, \ldots, ?v_n)$ passe de la valeur $?v$ à la valeur $?v'$ à l'instant t

FIG. I.1 – *Notation et signification des deux prédicats temporels* ***hold*** *et* ***event***

Dans l'implémentation actuelle d'IxTeT (pour des raisons de décidabilité et de complexité de algorithmes utilisés), les domaines des variables et des valeurs des attributs sont finis.

Nous définissons un ensemble de constantes **Cst** qui contiendra toutes les valeurs possibles des variables et des attributs.

Nous noterons $\mathcal{P}$ **(Cst)** l'ensemble des parties de **Cst**.

Tous les noms de variables (ex : ?robot) seront précédés d'un point d'interrogation et entièrement en minuscule pour les distinguer des valeurs des constantes dont le nom commence par une majuscule (ex : Robot).

Nous noterons **Var** l'ensemble des variables et **Att** l'ensemble des attributs.

Un attribut P sera donc noté P $(?v_1, \ldots, ?v_n)$ où P représente le nom de l'attribut (P $\in$ **Att**) et où les $?v_i$ sont les paramètres de l'attribut ($?v_i \in$ **Var**).

I.1.3 Les prédicats temporels

Dans notre logique, il existe deux prédicats temporels :

- **hold** qui représente la persistance de la valeur d'un attribut sur un intervalle temporel donné.
- **event** qui représente le changement de valeur d'un attribut à un instant donné.

La notation et le sens accordé à chacun de ces prédicats sont présentés dans la figure I.1

I.2 Les expressions logiques bien formées (EBF)

L'ensemble des expressions logiques autorisées dans notre représentation (ainsi que les conditions sous lesquelles ce sont des EBF) font l'objet d'un tableau (figure I.2 page ci-contre). Il conviendrait, pour être exhaustif, d'y ajouter l'ensemble des opérateurs de la logique propositionnelle ($et, ou, non, \Rightarrow$, *etc.*)[3].

Ces expressions sont regroupées en trois groupes qui sémantiquement et algorithmiquement seront traités différemment. Nous y trouvons tout d'abord les contraintes entre les instants (contrainte de durées entre deux instants, égalité, ordonnancement strict ou non), puis les contraintes sur les variables (restriction du domaine des valeurs possibles, égalité, inégalité, dépendance du domaine d'une variable par rapport à celui d'une autre variable) et enfin les prédicats temporels (persistance et changement).

I.3 Interprétations et modèles sémantiques

Dans cette section, nous définissons les interprétations et les modèles sémantiques de notre logique.

Soit :

- V_t, une application définie de **Tp** sur $\bar{\mathbb{R}}$ qui, à chaque instant, associe une date absolue.
- V_v, une application définie de **Var** sur **Cst** qui, à chaque variable, associe une valeur constante.
- V_a, une application définie de **Att** $\times \bar{\mathbb{R}}$ sur **Cst** qui, à chaque couple (attribut, date), associe une valeur constante.

Une interprétation V se définit par un tuple $V = (V_t, V_v, V_a)$.

Un modèle sémantique $\mathcal{M}_V$ est une application définie de l'ensemble des EBF sur l'ensemble {Vrai, Faux}.

La figure I.3 page 16 décrit la définition de cette application pour une interprétation donnée. Comme pour les EBF, il conviendrait d'y ajouter, pour être exhaustif, les

3. Ces opérateurs de la logique propositionnelle font partie de la logique que nous décrivons mais I_XT_ET ne gère que la conjonction.

EBF	Conditions
Contraintes temporelles	
$(t_1 - t_2) \in I$	$t_i \in \mathbf{Tp}$ $I \in \mathbf{I}$
$t_1 = t_2$ $t_1 < t_2$ $t_1 \leqslant t_2$ $t_1 > t_2$ $t_1 \geqslant t_2$	$t_i \in \mathbf{Tp}$
Contraintes sur les variables	
$?v \in D$	$?v \in \mathbf{Var}$ $D \in \mathcal{P}(\mathbf{Cst})$
$?v_1 = ?v_2$ $?v_1 \neq ?v_2$	$?v_i \in \mathbf{Var}$
$?v_1 \in D_1 \Rightarrow ?v_2 \in D_2$	$?v_i \in \mathbf{Var}$ $D_i \in \mathcal{P}(\mathbf{Cst})$
Prédicats temporels	
$\mathbf{hold}(\mathrm{Att}(?v_1, \ldots, ?v_n) : ?v, (t_1, t_2))$	$\mathrm{Att} \in \mathbf{Att}$ $?v_i \in \mathbf{Var}$ $?v \in \mathbf{Var}$ $t_i \in \mathbf{Tp}$
$\mathbf{event}(\mathrm{Att}(?v_1, \ldots, ?v_n) : (?v, ?v'), t)$	$\mathrm{Att} \in \mathbf{Att}$ $?v_i \in \mathbf{Var}$ $?v \in \mathbf{Var}$ $?v' \in \mathbf{Var}$ $t \in \mathbf{Tp}$

FIG. I.2 – *Liste des EBF*

$\mathcal{M}_V$(*expression logique*) = Vrai	**si et seulement si**
Contraintes temporelles	
$(t_1 - t_2) \in I^*$	$I^- \leqslant (V_t(t_1) - V_t(t_2)) \leqslant I^+$
$t_1 = t_2$	$V_t(t_1) = V_t(t_2)$
$t_1 < t_2$	$V_t(t_1) < V_t(t_2)$
$t_1 \leqslant t_2$	$V_t(t_1) \leqslant V_t(t_2)$
$t_1 > t_2$	$V_t(t_1) > V_t(t_2)$
$t_1 \geqslant t_2$	$V_t(t_1) \geqslant V_t(t_2)$
Contraintes sur les variables	
$?v \in D$	$V_v(?v) \in D$
$?v_1 = ?v_2$	$V_v(?v_1) = V_v(?v_2)$
$?v_1 \neq ?v_2$	$V_v(?v_1) \neq V_v(?v_2)$
$?v_1 \in D_1 \Rightarrow ?v_2 \in D_2$	$(V_v(?v_1) \notin D_1)$ ou $(V_v(?v_2) \in D_2)$
Prédicats temporels	
hold (Att $(?v_1, \ldots, ?v_n) : ?v, (t_1, t_2)$)	$V_t(t_1) < V_t(t_2)$ et $\forall\, d \in [V_t(t_1), V_t(t_2)[$ alors $V_a(\text{Att}\,(V_v(?v_1), \ldots, V_v(?v_n)), d) = V_v(?v)$
event (Att $(?v_1, \ldots, ?v_n) : (?v, ?v'), t$)	$\exists\, d \in \bar{\mathbb{R}}, \quad \exists\, d' \in \bar{\mathbb{R}}$ tel que $d < V_t(t) < d'$ $\mathcal{M}_V$(**hold** (Att $(?v_1, \ldots, ?v_n) : ?v, (d, t)$)) = Vrai $\mathcal{M}_V$(**hold** (Att $(?v_1, \ldots, ?v_n) : ?v', (t, d')$)) = Vrai

FIG. I.3 – *Définion d'un modèle sémantique* $\mathcal{M}_V$

définitions liées aux opérateurs de la logique propositionnelle ($et, ou, non, \Rightarrow$, *etc.*).

Voici quelques définitions classiques :

- On dit qu'une interprétation V satisfait un ensemble A de formules logiques si et seulement si, pour toute formule α appartenant à A, on vérifie $\mathcal{M}_V(\alpha) = \mathsf{Vrai}$ (On interprète la conjonction de toutes les formules logique de A).
- La relation de déductibilité ($\models$) s'établit entre deux ensembles de formules logiques. Étant donné deux ensembles de formules logiques A et B, on établit $A \models B$ si et seulement si, pour toute interprétation V satisfaisant A, V satisfait B.
- Un ensemble de formules logiques A est dit valide si et seulement si $\emptyset \models A$.
- Un ensemble de formules logiques A est dit contradictoire si et seulement si $A \models \mathsf{Faux}$.

I.4 Logique et gestion des ressources

La définition formelle de la sémantique liée à l'utilisation des ressources nécessiterait l'utilisation d'aspects fonctionnels dans notre logique. Ces développements seraient longs et n'apporteraient que peu de choses à la compréhension du formalisme d'I$_X$T$_E$T.

Nous présentons donc ici très rapidement l'aspect sémantique lié à l'utilisation des ressources sans en préciser les spécifications formelles. P. Laborie, qui a intégré la gestion des ressources dans la version actuelle d'I$_X$T$_E$T, fournit dans [Laborie 95] une présentation beaucoup plus complète de ces problèmes et des solutions apportées.

I.4.1 Définition d'une ressource

On peut distinguer plusieurs types de ressources (partageables ou non, discrètes ou continues, individualisées ou banalisées).

En fait, on appelle ressource toute substance ou ensemble d'objets dont le coût ou la quantité disponible induit des contraintes sur les tâches qui l'utilisent.

Dans IxTeT, à l'instar des attributs, une ressource est représentée par un nom paramétré par des variables. Les paramètres permettent d'individualiser certaines ressources. L'ensemble des ressources sera noté **Res**.

On associe à chaque ressource, une capacité initiale (c.-à-d. la quantité de ressource disponible à l'origine des temps). Cette capacité initiale est une valeur prise dans $\mathbb{R}^+$. Les quantités exprimées dans les propositions d'utilisation, de production et de consommation de ressource se définissent elles-aussi par une valeur dans $\mathbb{R}^+$. Nous définissons Q_0 comme une application de **Res** sur $\mathbb{R}^+$ qui à chaque ressource associe la valeur de sa capacité initiale.

Dans l'implémentation actuelle d'IxTeT, les capacités sont obligatoirement des entiers positifs ($\mathbb{N}^+$). De plus, les ressources ne sont que de deux types :

- **Des ressources banalisées partageables :** ce sont des ressources sans paramètre avec une capacité initiale quelconque.
- **Des ressources non-partageables individualisées :** ce sont des ressources avec un seul paramètre et de capacité initiale unitaire.

Il serait possible de gérer des ressources avec plusieurs paramètres et partageables mais cela nécessiterait des algorithmes beaucoup plus complexes. On constate que ces deux seuls types de ressources suffisent dans la plupart des cas à représenter le monde dans lequel on veut planifier (la perte relative d'expressivité est très largement compensée par l'efficacité des algorithmes mis en œuvre.)

I.4.2 Utilisation des ressources

Dans notre logique, il existe trois actions possibles sur une ressource. Nous les présentons dans la figure I.4 page ci-contre.

En fait, dans IxTeT, pour faciliter l'interprétation logique et la vérification de la cohérence de l'ensemble des propositions d'utilisation de ressources, la production et la consommation de ressources sont exprimées en terme d'utilisation de ressources.

Produire une quantité q de ressource à un instant est équivalent (du moins logiquement) à dire que cette quantité de ressource existait depuis l'origine des temps mais

use (Res $(?v_1, \ldots, ?v_n) : q, (t_1, t_2)$)	**L'emprunt** de la quantité q de ressource durant l'intervalle temporel $[t_1, t_2[$.
produce (Res $(?v_1, \ldots, ?v_n) : q, t$)	**La production** d'une quantité q de ressource à l'instant t
consume (Res $(?v_1, \ldots, ?v_n) : q, t$)	**La consommation** d'une quantité q de ressource à l'instant t

FIG. I.4 – *Actions possibles sur les ressources.*

qu'elle était utilisée jusqu'à cet instant. Ce qui se traduit formellement par :

$$\textbf{produce}\,(\text{Res}\,(?v_1, \ldots, ?v_n) : q, t) \quad \text{et} \quad Q_0(\text{Res}\,(?v_1, \ldots, ?v_n)) = q_0$$
$$\equiv \quad Q_0(\text{Res}\,(?v_1, \ldots, ?v_n)) = q_0 + q \quad \text{et} \quad \textbf{use}\,(\text{Res}\,(?v_1, \ldots, ?v_n) : q, (-\infty, t))$$

Consommer une quantité q de ressource à un instant est équivalent à dire (du moins logiquement) que cette quantité de ressource est utilisée jusqu'à la fin des temps. Ce qui se traduit formellement par :

$$\textbf{consume}\,(\text{Res}\,(?v_1, \ldots, ?v_n) : q, t) \quad \equiv \quad \textbf{use}\,(\text{Res}\,(?v_1, \ldots, ?v_n) : q, (t, +\infty))$$

Les deux équivalences précédentes nous permettent donc de remplacer toutes les propositions de consommation et de production par des propositions d'utilisation.

Nous définissons la fonction Q (quantité utilisée) par :

$$Q(\textbf{use}\,(\text{Res}\,(?v_1, \ldots, ?v_n) : q, (t_1, t_2))) = q$$

I.4.3 Interprétation sémantique liée aux ressources

Comme nous l'avons expliqué précédemment, l'interprétation des prédicats d'utilisation de ressources ne se fait qu'en considérant l'ensemble complet de ces propositions. Ce que nous proposons ici ne permet donc que de vérifier la cohérence globale de l'utilisation de chaque ressource.

Soit RES, un ensemble de propositions d'utilisation de ressources (où chaque *produce* ou *consume* est remplacé par son équivalence en terme de *use*) et soit V, une interprétation de notre logique.

Pour une date d donnée, on définit l'ensemble USE_V par :

$$USE_V(\mathrm{R}\,(C_1,\ldots,C_n)\,,d) = \left\{ \mathbf{use}\,(\mathrm{Res}\,(?v_1,\ldots,?v_n) : q,(t_1,t_2)) \in RES \quad \text{tel que} \right.$$
$$\mathrm{Res} = \mathrm{R},$$
$$V_v(?v_1) = C_1,\ \ldots,\ V_v(?v_n) = C_n,$$
$$\left. d \in [V_t(t_1), V_t(t_2)[\right\}$$

Un ensemble RES des *propositions* d'utilisation de ressources (où chaque « produce » ou « consume » est remplacé par son équivalence en terme de « use ») est satisfaisable par une interprétation V donnée si et seulement si :

$$\forall\, d \in \mathbb{R}, \quad \forall\, \mathrm{R} \in \mathbf{Res}, \quad \sum_{\alpha \in USE_V(\mathrm{R},d)} Q(\alpha) \quad \leqslant \quad Q_0(\mathrm{R}) \qquad \text{(I.1)}$$

Chapitre II

Le fonctionnement d'IxTeT

II.1 IxTeT et autres planificateurs

Comme nous l'avons déjà rappelé dans notre introduction, les planificateurs utilisant des opérateurs de changement d'états (similaire à ceux introduits dans STRIPS par [Fikes 71]) se basent sur deux approches fondamentalement différentes selon qu'ils explorent un espace d'états ou un espace de plans partiels.

Les premiers recherchent un chemin dans un graphe d'état depuis l'état initial jusqu'à l'un des états buts. Chaque transition représente l'utilisation d'un opérateur de changement d'état. Cette recherche peut s'effectuer soit par progression depuis l'état initial soit par régression depuis les buts. La recherche par progression a l'avantage de construire le plan chronologiquement et permet donc (si l'on s'interdit des retours-arrière trop important temporellement) de commencer l'exécution du plan alors que son élaboration n'est pas terminée. Mais une bonne heuristique permettant de choisir correctement la prochaine tâche (ou opérateur) à effectuer est très difficile à trouver dans le cas général[1]. La recherche par régression n'utilisant que les actions nécessairement

1. Les planificateurs réactifs se heurtent à un problème similaire

liées aux buts spécifiées (directement ou non) permet de beaucoup diminuer les mauvais choix mais on perd alors la possibilité d'exécuter le plan au fur et à mesure de son calcul. Dans tous les cas, le problème principal lié à ce type d'approche est la linéarité des plans produits. Ce type de planificateurs ne peut donc pas répondre aux besoins de la planification multi-agents puisqu'ils n'intègrent pas la notion de parallélisme ni la notion de durée des tâches.

La seconde approche par recherche dans un espace de plans partiels est utilisée par NOAH [Sacerdoti 75] puis développée dans TWEAK [Chapman 87] ou dans SNLP [McAllester 91]. La principale différence entre ces deux derniers systèmes concerne la manière dont ils garantissent la validité d'un fait. Le premier utilise un critère de vérité qui est recalculé à chaque insertion d'une nouvelle tâche dans le plan. Le second, quant à lui, utilise la notion d'événement (ou d'action) établisseur et celle de lien causal entre cet événement et le fait à valider.

L'utilisation d'une représentation métrique du temps (réellement nécessaire pour une planification multi-agents) est apparue l'une des premières fois dans le système DEVISER [Vere 83] puis étendue par TRIPTIC [Hertzberg 93]. Ces approches ne permettent malheureusement pas d'utiliser toute la richesse d'une représentation temporelle explicite. TIMELOGIC [Allen 83a] [Allen 91] quant à lui utilise toute la puissance de l'algèbre d'intervalles mais de ce fait se heurte à un problème de complexité.

Les systèmes de planification permettant à la fois de gérer le temps, l'affectation des ressources et le parallélisme des tâches ne sont pas encore très nombreux. Citons SIPE [Wilkins 88] et O-PLAN [Currie 91]. Nous reviendrons sur ce dernier système dans le chapitre concernant les exemples (page 89).

I_XT_ET fait partie de cette dernière famille de planificateurs généraux. C'est un système de planification qui travaille à partir de plans partiels. Les tâches qu'il peut utiliser ainsi que le problème à traiter (situation initiale et buts à atteindre) lui sont spécifiés sous forme de plans partiels. Seul le plan final (lorsqu'il existe) n'est plus un plan partiel puisqu'il ne contient plus de buts pendants.

Pour trouver une solution I_XT_ET explore l'espace des plans partiels. Son point de départ est le plan partiel initial. Un plan solution est un plan sans défaut. Pour modifier le plan initial et le transformer peu à peu en plan solution, I_XT_ET insérera soit une nouvelle tâche, soit une nouvelle contrainte temporelle, soit une nouvelle contrainte sur les variables afin de supprimer un ou plusieurs défauts du plan partiel courant.

Les choix d'insertion ne se faisant pas de manière certaine, l'exploration doit pouvoir effectuer des retours-arrière. De plus, l'espace à explorer est potentiellement infini. Donc l'exploration s'effectue au moyen d'un algorithme A_ϵ guidé par une heuristique d'engagement minimum.

Dans ce chapitre, nous présentons le fonctionnement de base du planificateur en commençant par préciser la notion de plans partiels. Puis nous décrirons comment est vérifiée la cohérence de ces plans et comment se réalise la détection des défauts ainsi que leur suppression. Nous terminerons en présentant rapidement les mécanismes de contrôle du planificateur ainsi que le langage de description de tâches et son compilateur.

II.2 Les plans partiels

Le planificateur IxTeT ne travaille qu'avec des *plans partiels*. Un plan partiel est la conjonction d'un ensemble de formules de notre logique. Nous noterons donc:

$$\mathsf{P} = (Evt \wedge Hld \wedge Res \wedge C_V \wedge C_T)$$

Où:

- *Evt* est la conjonction de tous les prédicats temporels de type événement (**event**).
- *Hld* est la conjonction de tous les prédicats temporels de type assertion (**hold**).
- *Res* est la conjonction de toutes les propositions d'utilisation de ressources (**use**, **produce** et **consume**).
- C_V est la conjonction des contraintes sur les variables.
- C_T est la conjonction des contraintes sur les instants.

Nous utiliserons la même notation pour désigner indifféremment la conjonction ou l'ensemble de formules logiques d'un même type.

II.2.1 La cohérence des plans partiels

Cohérence temporelle

On dira que P est *temporellement cohérent* si et seulement s'il existe au moins une interprétation V_t telle que V_t satisfait C_T.

Cohérence vis-à-vis des contraintes sur les variables

On dira que P est *cohérent vis-à-vis des contraintes sur les variables* si et seulement s'il existe au moins une interprétation V_v telle que V_v satisfait C_V.

Cohérence vis-à-vis des attributs

On dira que P est *cohérent vis-à-vis des attributs* si et seulement si les deux conditions suivantes sont vérifiées :

1. P est cohérent temporellement et vis-à-vis des contraintes sur les variables.
2. Pour chaque couple (V_t, V_v) satisfaisant $(C_T \wedge C_V)$ il existe au moins une interprétation V contenant V_t et V_v telle que :

$$V \text{ satisfait } (C_T \wedge C_V \wedge Hld \wedge Evt)$$
$$\forall\, evt \notin Evt \quad V \text{ ne satisfait pas } (C_T \wedge C_V \wedge Hld \wedge Evt \wedge evt)$$

Ce qui signifie que P décrit explicitement et complètement tous les changements d'états contenus dans V (V ne peut accepter aucun autre événement).

Cohérence vis-à-vis des ressources

On dira que P est *cohérent vis-à-vis des ressources* si et seulement si les deux conditions suivantes sont vérifiées :

1. P est cohérent temporellement et vis-à-vis des contraintes sur les variables.
2. Pour toute interprétation V satisfaisant $(C_T \wedge C_V)$, Res satisfait le critère de cohérence des ressources (équation I.1 page 20).

Cohérence d'un plan partiel

On dira que P est *cohérent* si et seulement s'il est cohérent vis-à-vis des attributs et vis-à-vis des ressources.

II.2.2 Les opérateurs de planification ou tâches

Les opérateurs de planification (que nous appelons des tâches dans IxTeT) sont des plans partiels. Ces tâches sont fournies au système en utilisant un langage de description textuel (§ II.6 page 43) qui, après compilation, permettra de les stocker sous forme de plans partiels directement utilisables par le système de planification.

La compilation permet déjà de vérifier, pour chaque tâche prise séparément, la cohérence temporelle et la cohérence vis-à-vis des contraintes sur les variables. Les mécanismes utilisés par le compilateur pour valider cette cohérence sont les mêmes que ceux qu'utilise le planificateur.

Nous noterons T l'ensemble des tâches fournies pour un problème de planification donné :

$$\mathsf{T} = \left\{ \mathsf{T}^{\mathsf{i}} = (Evt^{\mathsf{i}} \wedge Hld^{\mathsf{i}} \wedge Res^{\mathsf{i}} \wedge C_V^{\mathsf{i}} \wedge C_T^{\mathsf{i}}) \text{ avec } \mathsf{i} \in [1, \ldots, \mathsf{t}] \right\}$$

II.2.3 Le plan partiel initial P_0

Le plan partiel initial P_0 constitue le point de départ de l'algorithme de planification d'IxTeT. Il est constitué d'une part de la description logique de l'ensemble du monde et d'autre part de la description logique des buts à atteindre :

$$\mathsf{P}_0 = \mathsf{P}_{\mathsf{Monde}} \wedge \mathsf{P}_{\mathsf{Buts}}$$

La compilation de ce plan initial nous garantit sa cohérence temporelle et sa cohérence vis-à-vis des contraintes sur les variables (§ II.2.1 page précédente).

L'objectif du planificateur est de trouver un ensemble de formules logiques Sol tel que :

$$(\mathsf{P}_{\mathsf{Monde}} \wedge \mathsf{Sol}) \models \mathsf{P}_{\mathsf{Buts}}$$
$$(\mathsf{P}_{\mathsf{Monde}} \wedge \mathsf{Sol}) \not\models \mathsf{Faux}$$

Dans le plan solution, tous les changements du monde (sur les attributs du monde ou sur l'utilisation des ressources) doivent obligatoirement provenir soit d'instances de tâches provenant de T soit de la description du monde P_{Monde}. Dans ce dernier cas, ils représentent des changements du monde contingents qui sont en dehors du contrôle du planificateur.

Sol n'est donc constitué que d'instances de tâches de T, de contraintes temporelles, de contraintes sur les variables ou propositions de persistance sur des attributs.

Dans son mécanisme de résolution, le planificateur va construire incrémentalement Sol par des conjonctions successives de *résolvantes*.

II.3 Vérification de la cohérence

Nous présentons ici les méthodes utilisées pour vérifier la cohérence des plans partiels. La cohérence temporelle et la cohérence vis-à-vis des variables doivent être vérifiées en permanence. Par contre la cohérence vis-à-vis des attributs et celle vis-à-vis des ressources constituent l'objectif à atteindre. En effet un plan partiel complètement cohérent est un plan solution pour I_XT_ET.

II.3.1 Cohérence des contraintes temporelles

Afin de pouvoir vérifier la cohérence temporelle, I_XT_ET utilise un algorithme de propagation complète des contraintes temporelles entre les instants. Nous supposons que les contraintes sont exprimées par une matrice K qui, lorsque les contraintes sont propagées, vérifie les propriétés suivantes :

$K(i,j)$ est un intervalle contenant toutes les durées possibles de $V_t(t_j) - V_t(t_i)$
$K(j,i) = -K(i,j)$

À l'initialisation, toutes les contraintes sont posées par défaut à $]-\infty, +\infty[$. Lors de l'ajout d'une contrainte entre deux instants, la matrice est modifiée en utilisant les

équivalences suivantes[2] :

$$\begin{aligned}
t_i = t_j \text{ est équivalent à } K(i,j) &\leftarrow K(i,j) \cap [0,0] \\
t_i \leqslant t_j \text{ est équivalent à } K(i,j) &\leftarrow K(i,j) \cap [0,+\infty[\\
t_i \geqslant t_j \text{ est équivalent à } K(i,j) &\leftarrow K(i,j) \cap]-\infty,0] \\
t_i < t_j \text{ est équivalent à } K(i,j) &\leftarrow K(i,j) \cap]0,+\infty[\\
t_i > t_j \text{ est équivalent à } K(i,j) &\leftarrow K(i,j) \cap]-\infty,0[\\
(t_j - t_i) \in I \text{ est équivalent à } K(i,j) &\leftarrow K(i,j) \cap I
\end{aligned}$$

Puis cette modification est propagée à travers les autres contraintes.

Nous décrivons l'algorithme général de propagation des contraintes entre n instants (algorithme A.1 page 124) dont la complexité est $O(n^3)$, ainsi que sa version incrémentale (algorithme A.2 page 124) dont la complexité est $O(n^2)$ et qui permet de propager l'ajout ou la modification d'une seule contrainte.

Si l'un des intervalles de la matrice des contraintes devient l'intervalle $\emptyset$, alors l'ensemble des contraintes est incohérent.

L'algorithme suivant (algorithme A.3 page 125) montre comment on ajoute une nouvelle contrainte en détectant les éventuelles incohérences avant l'ajout et en évitant de propager inutilement une contrainte redondante.

Cet algorithme interroge la base de données des contraintes temporelles (en fait la matrice K complètement propagée) grâce à un algorithme très simple (algorithme A.4 page 125) qui permet de savoir si l'ajout d'une contrainte créera une incohérence ou nécessitera une propagation.

Lorsque l'ensemble des contraintes temporelles est complètement propagées, on démontre qu'une interprétation V satisfaisait cet ensemble si et seulement si la condition

2. On constate que la précédence stricte des instants nécessite l'emploi d'intervalles ouverts qu'actuellement nous ne gérons pas mais, I_XT_ET utilisant un ensemble discret de durées, on peut obtenir un résultat similaire avec les équivalences suivantes (dans lesquelles la valeur 1 représente la plus petite durée strictement positive) :

$$\begin{aligned}
t_i < t_j \text{ est équivalent à } K(i,j) &\leftarrow K(i,j) \cap [1,+\infty[\\
t_i > t_j \text{ est équivalent à } K(i,j) &\leftarrow K(i,j) \cap]-\infty,1]
\end{aligned}$$

suivante est vérifiée :

$$\forall\, t_1 \in \mathbf{Tp}, \quad \forall\, t_2 \in \mathbf{Tp}, \quad (V_t(t_2) - V_t(t_1)) \in K(t_1, t_2)$$

II.3.2 La cohérence des contraintes sur les variables

Dans IXTET, nous gérons quatre types différents de contraintes sur les variables : l'égalité ou l'inégalité de deux variables, la restriction du domaine d'une variable et la dépendance entre deux variables.

Les questions que l'on veut poser au gestionnaire de contraintes sur les variables sont :

- L'ensemble des contraintes sur **Var** (l'ensemble des variables) est-il globalement cohérent ?
- Peut-on unifier deux variables particulières (en gardant l'ensemble cohérent) ?
- Peut-on poser une contrainte d'inégalité entre deux variables particulières (en gardant l'ensemble cohérent) ?
- Peut-on réduire le domaine d'une variable particulière (en gardant l'ensemble cohérent) ?
- Peut-on poser une contrainte de dépendance entre les domaines de deux variables particulières (en gardant l'ensemble cohérent) ?

La vérification de la cohérence globale d'un ensemble de contraintes de ce type (inégalités et relations de dépendance) et le calcul des réponses aux requêtes précédentes sont des problèmes NP-complets (le réseau de contraintes est équivalent à un CSP binaire général).

Étant donné le grand nombre de requêtes et d'ajouts effectués lors d'une planification, il est trop long de vérifier la cohérence globale à chaque étape. IXTET ne fait donc qu'une vérification locale de cohérence sur les contraintes reliant les variables. La validation globale n'est faite que pour d'éventuels plan solutions.

Nous présentons en annexe les algorithmes utilisés pour propager localement chaque type de contraintes. Les contraintes d'égalité entre les variables pouvant être représen-

tées par une relation d'équivalence, IxTeT s'appuie sur les classes de cette relation pour décrire le domaine des variables et exprimer les autres types de contraintes.

Nous noterons :

$E = \{e_1, \ldots, e_k\}$	la partition de **Var** en k classes d'équivalence;
ϵ : **Var** $\longrightarrow E$	une fonction qui à chaque variable associe sa classe d'équivalence;
$D_{e_1}, \ldots, D_{e_k}$	les domaines de valeurs associés à chaque classe.

L'ajout d'une contrainte d'égalité entre deux variables consiste à fusionner les deux classes d'équivalence et à leur associer l'intersection des deux domaines (algorithme B.1 page 128).

La restriction du domaine d'une variable consiste à restreindre le domaine de sa classe (algorithme B.2 page 128).

Les autres contraintes sont stockées dans deux listes ($K_{\neq}$ et $K_{\Rightarrow}$) qui sont mises à jour à chaque nouvel ajout (algorithme B.3 page 128 et algorithme B.4 page 128).

Dans chaque cas, on propage ces modifications via un algorithme de filtrage par consistance d'arcs (algorithme B.5 page 129 et algorithme B.6 page 129). Sa complexité est $O(k)$ où k représente le nombre de contraintes contenues dans $K_{\neq}$ et $K_{\Rightarrow}$.

En utilisant ces algorithmes, on constate que la réponse aux requêtes précédentes peut se calculer en temps constant (du fait de la vérification seulement locale des contraintes, la réponse à certaines requêtes peut être parfois faussement positive).

Par contre, pour être sûr de la validité d'un plan solution, il faut effectuer une propagation complète des contraintes, ce qui se fait par un algorithme classique de CSP NP-complet (algorithme B.7 page 130). Heureusement, les réseaux de contraintes que nous traitons sont relativement petits (de l'ordre de quelques dizaines de variables avec 3 ou 4 contraintes par variables) et en fin de planification le domaine de chaque variable est souvent réduit à un singleton ce qui réduit considérablement la complexité effective de l'algorithme.

Après propagation complète d'un ensemble de contraintes sur les variables, on vérifie qu'une interprétation V satisfait cet ensemble si et seulement si elle vérifie la condition suivante :

$$\forall\, ?v \in \textbf{Var}, \quad V_v(?v) \in D_{\epsilon(?v)}$$

II.3.3 Cohérence des attributs

Dans I_XT_ET, l'état du monde est décrit (sauf pour les ressources) par des propositions de persistance ou de changement de valeur d'attributs (**hold** et **event**).

Pour qu'un plan partiel soit cohérent vis-à-vis des attributs, il faut que chaque attribut ne puisse prendre qu'une et une seule valeur à un moment donné et que chaque changement de valeur soit effectivement décrit par *un et un seul* événement.

Au sens strictement logique, on pourrait admettre plusieurs événements identiques simultanés (c.-à-d. même attribut, même valeur précédente, même valeur suivante et même instant) mais dans I_XT_ET nous n'acceptons qu'un seul événement. Ceci afin de garantir l'unicité de l'action, puisqu'un événement représente un changement d'état contingent (c.-à-d. imposé de l'extérieur) ou un changement d'état réalisé par une tâche. Si nous autorisions deux événements identiques et simultanés dans notre représentation, nous risquerions d'accepter qu'un changement d'état soit effectué par deux tâches distinctes.

De manière informelle, la cohérence vis-à-vis des attributs suppose que l'ensemble des événements concernant un attribut donné soit totalement ordonné temporellement, que les changements de valeurs liés à ces événements soient cohérents, qu'il existe pour chacun des couples successifs formés par ces événements une ou plusieurs propositions de persistance recouvrant totalement l'intervalle temporel et cohérent avec les valeurs des deux événements de ce couple.

Soit :

- $\mathsf{E}(\mathrm{Att}\,(C_1,\ldots,C_n)\,,t)$, l'ensemble des événements ayant lieu à l'instant t et portant sur l'attribut $\mathrm{Att}\,(C_1,\ldots,C_n)$: [3]

$$\mathsf{E}(\mathrm{Att}\,(C_1,\ldots,C_n)\,,t) = \{\textbf{event}\,(\mathrm{Att}\,(?v_1,\ldots,?v_n) : (?v,?v')\,,t') \in Evt \text{ tel que } \forall\, i\ C_i \in D_{\epsilon(?v_i)} \text{ et } t = t'\}$$

- $\mathsf{H}^{\leqslant}(\mathrm{Att}\,(C_1,\ldots,C_n)\,,t)$, l'ensemble des propositions de persistance portant sur l'attribut $\mathrm{Att}\,(C_1,\ldots,C_n)$ fixant une valeur au moins jusqu'à t :[3]

$$\mathsf{H}^{\leqslant}(\mathrm{Att}\,(C_1,\ldots,C_n)\,,t) = \{\textbf{hold}\,(\mathrm{Att}\,(?v_1,\ldots,?v_n) : ?v,(t_1,t_2)) \in Hld \text{ tel que } \forall\, i\ C_i \in D_{\epsilon(?v_i)} \text{ et } t_1 < t \leqslant t_2\}$$

- $\mathsf{H}^{\geqslant}(\mathrm{Att}(C_1,\ldots,C_n),t)$, l'ensemble des propositions de persistance portant sur l'attribut $\mathrm{Att}(C_1,\ldots,C_n)$ fixant une valeur au moins à partir de t :[3]

$$\mathsf{H}^{\geqslant}(\mathrm{Att}(C_1,\ldots,C_n),t) = \{\mathbf{hold}(\mathrm{Att}(?v_1,\ldots,?v_n):?v,(t_1,t_2)) \in \mathit{Hld} \text{ tel que } \forall\, i\ C_i \in D_{\epsilon(?v_i)} \text{ et } t_1 \leqslant t < t_2\}$$

- Val, une fonction définie de Hld sur $\mathcal{P}(\mathbf{Cst})$ qui, à chaque proposition de persistance, associe le domaine de sa valeur de persistance :

$$\mathit{Val}\big(\mathbf{hold}(\mathrm{Att}(?v_1,\ldots,?v_n):?v,(t_1,t_2))\big) = D_{\epsilon(?v)}$$

- $\mathit{Val}^{\leqslant}(\mathrm{Att}(C_1,\ldots,C_n),t)$, l'ensemble des valeurs possiblement prises par l'attribut $\mathrm{Att}(C_1,\ldots,C_n)$ au moins jusqu'à t :

$$\mathit{Val}^{\leqslant}\big(\mathrm{Att}(C_1,\ldots,C_n),t\big) = \bigcup_{h \in \mathsf{H}^{\leqslant}(\mathrm{Att}(C_1,\ldots,C_n),t)} \mathit{Val}(h)$$

- $\mathit{Val}^{\geqslant}(\mathrm{Att}(C_1,\ldots,C_n),t)$, l'ensemble des valeurs possiblement prises par l'attribut $\mathrm{Att}(C_1,\ldots,C_n)$ au moins à partir de t :

$$\mathit{Val}^{\geqslant}\big(\mathrm{Att}(C_1,\ldots,C_n),t\big) = \bigcup_{h \in \mathsf{H}^{\geqslant}(\mathrm{Att}(C_1,\ldots,C_n),t)} \mathit{Val}(h)$$

À partir de ces définitions et de manière formelle, on pourra dire qu'un plan partiel est cohérent vis-à-vis des attributs si et seulement si les propositions suivantes sont vérifiées :

$$\forall\, t \in \mathbf{Tp},\ \forall\ \mathrm{Att}(C_1,\ldots,C_n),\quad \mathtt{Card}(\mathsf{E}(\mathrm{Att}(C_1,\ldots,C_n),t)) \leqslant 1 \qquad \text{(II.1)}$$

$$\forall\, t \in \mathbf{Tp},\ \forall\ \mathrm{Att}(C_1,\ldots,C_n),\quad \mathtt{Card}(\mathsf{Val}^{\leqslant}(\mathrm{Att}(C_1,\ldots,C_n),t)) = 1 \qquad \text{(II.2)}$$

$$\forall\, t \in \mathbf{Tp},\ \forall\ \mathrm{Att}(C_1,\ldots,C_n),\quad \mathtt{Card}(\mathsf{Val}^{\geqslant}(\mathrm{Att}(C_1,\ldots,C_n),t)) = 1 \qquad \text{(II.3)}$$

$$\begin{aligned}&\forall\, t \in \mathbf{Tp},\ \forall\ \mathrm{Att}(C_1,\ldots,C_n),\\ &\mathsf{E}(\mathrm{Att}(C_1,\ldots,C_n),t) = \left\{\mathbf{event}\left(\mathrm{Att}(C_1,\ldots,C_n):(C,C'),t\right)\right\}\\ &\Rightarrow \left(\mathsf{Val}^{\leqslant}(\mathrm{Att}(C_1,\ldots,C_n),t) = \{C\}\right) \text{ et } \left(\mathsf{Val}^{\geqslant}(\mathrm{Att}(C_1,\ldots,C_n),t) = \{C'\}\right)\end{aligned} \qquad \text{(II.4)}$$

3. Lorsque l'on exprime des conditions sur t, cela signifie que ces conditions ne sont pas incompatibles avec les contraintes temporelles du plan partiel considéré.

$$\forall\, t \in \mathbf{Tp},\ \forall\, \mathrm{Att}\,(C_1,\ldots,C_n)\,,$$
$$\mathsf{Card}(\mathsf{E}(\mathrm{Att}\,(C_1,\ldots,C_n)\,,t)) = 0$$
$$\Rightarrow \left(\mathsf{Val}^{\leqslant}(\mathrm{Att}\,(C_1,\ldots,C_n)\,,t) = \mathsf{Val}^{\geqslant}(\mathrm{Att}\,(C_1,\ldots,C_n)\,,t)\right) \quad \text{(II.5)}$$

La première proposition (II.1) garantit l'unicité d'un événement lorsqu'il a lieu. Les deux suivantes ((II.2) et (II.3)) garantissent que l'attribut prend bien une et une seule valeur à chaque instant. Quant aux deux dernières propositions ((II.4) et (II.5)), elles garantissent la cohérence et l'ordonnancement temporel correct des changements de valeur de cet attribut.

Remarquons qu'avec les définitions que nous avons données, pour garantir la cohérence vis-à-vis des attributs, nous imposons que le domaine de chaque variable soit un singleton. D'autre part, la notion de cohérence que nous donnons ici est trop forte par rapport à la sémantique que nous donnons à notre logique.

D'un point de vue pratique, la définition de cohérence que nous donnons n'est pas utilisable telle quelle. C'est pourquoi, dans IxTeT, nous utilisons une notion de cohérence locale basée sur les notions d'explications et de menaces. Cette cohérence locale est une variante du critère décrit dans [McAllester 91]. Quant à la preuve de son équivalence avec le critère de cohérence globale que nous venons de présenter et son adaptation à IxTeT, ils sont complètement décrits dans [Laruelle 94] et [Laborie 95]. Nous allons en faire ici une présentation synthétique rapide.

Définition d'une proposition expliquée

Soit une proposition concernant l'attribut $\mathrm{Att}\,(?x_1,\ldots,?x_n)$. Elle peut être de deux formes :

$$\textbf{event}\,(\mathrm{Att}\,(?x_1,\ldots,?x_n) : (?v_1,?v_2)\,,t_1)$$
$$\text{ou}\quad \textbf{hold}\,(\mathrm{Att}\,(?x_1,\ldots,?x_n) : ?v_1,(t_1,t_2))$$

On dira que cette proposition est expliquée si et seulement si les deux propositions suivantes sont vraies :

- Il existe un événement (que nous appelons *événement établisseur*) :

$$\textbf{event}\,\left(\mathrm{Att}\,\left(?x'_1,\ldots,?x'_n\right) : \left(?v'_1,?v'_2\right),t'_1\right)$$
$$\text{tel que}\quad ?v'_2 = ?v_1,\quad \forall i\ ?x_i = ?x'_i,\quad t'_1 \leqslant t_1$$

- Si $(t'_1 < t_1)$ alors il existe une assertion (que nous appelons *lien causal*) qui maintient la valeur de l'attribut entre l'événement établisseur et la proposition à expliquer :

$$\textbf{hold}\left(\text{Att}\left(?x''_1,\ldots,?x''_n\right) : ?v''_1, \left(t''_1, t''_2\right)\right)$$
$$\text{tel que} \quad ?v''_1 = ?v_1, \quad \forall i\ ?x_i = ?x''_i, \quad t''_1 = t'_1, \quad t''_2 = t_1$$

Définition d'une menace

Soit une proposition de persistance α concernant l'attribut $\text{Att}\left(?x_1,\ldots,?x_n\right)$ et s'exprimant par :

$$\alpha \equiv \textbf{hold}\left(\text{Att}\left(?x_1,\ldots,?x_n\right) : ?v, \left(t_1, t_2\right)\right)$$

Une menace sur α peut être :

- soit un événement **event** $\left(\text{Att}\left(?x'_1,\ldots,?x'_n\right) : \left(?v'_1, ?v'_2\right), t'\right)$ tel que la proposition suivante ne puisse être déduite du plan partiel considéré :

$$(\exists i\ ?x'_i \neq ?x_i) \text{ ou } (t' < t_1) \text{ ou } (t_2 < t')$$
$$\text{ou } \Big((t' = t_1) \text{ et } (?v'_2 = ?v)\Big) \text{ ou } \Big((t' = t_2) \text{ et } (?v'_1 = ?v)\Big)$$

- soit une proposition **hold** $\left(\text{Att}\left(?x'_1,\ldots,?x'_n\right) : ?v', \left(t'_1, t'_2\right)\right)$ telle que la proposition suivante ne puisse être déduite du plan partiel considéré :

$$(\exists i\ ?x'_i \neq ?x_i) \text{ ou } (t'_2 < t_1) \text{ ou } (t_2 < t'_1) \text{ ou } \Big((\forall i\ ?x'_i = ?x_i) \text{ et } (?v' = ?v)\Big)$$

Critère de cohérence vis-à-vis des attributs

On démontre qu'un plan partiel dont toutes les propositions sont expliquées et qui ne contient aucune menace est nécessairement cohérent vis-à-vis des attributs. Il suffit donc de montrer qu'un plan partiel ne contient aucune proposition inexpliquée ni aucune menace pour qu'on le considère comme cohérent vis-à-vis des attributs.

Définition d'une assertion primordiale

Pour pouvoir expliquer une proposition, il faut trouver dans le plan partiel un événement établisseur qui précède nécessairement cette proposition. Or, pour garantir la

cohérence du plan partiel vis-à-vis des attributs, il faut aussi expliquer cet événement établisseur... Cette définition récursive de l'explication pose donc un problème puisqu'elle nécessite un événement initial expliqué par un autre moyen.

L'implémentation actuelle d'IxTeT laisse au programmeur le soin d'introduire explicitement des propositions expliquées dans son plan initial. À sa charge de vérifier la cohérence des propositions qu'il a ainsi expliquées (et donc du plan solution produit). D'autre part, si le programmeur oublie, pour un attribut donné, de spécifier au moins une proposition expliquée, le processus de planification ne pourra pas se terminer...

Afin de supprimer cette opération d'explication explicite (et toutes les erreurs potentielles que cela autorise), nous pouvons ajouter une nouvelle manière d'expliquer une proposition par l'intermédiaire d'une proposition de persistance que nous appellerons *assertion primordiale.*

Une proposition peut donc être expliquée soit par la définition que nous avons donnée précédemment soit si les propositions suivantes sont vérifiées :

- Il existe une proposition de persistance (*assertion primordiale*) :

$$\textbf{hold}\left(\text{Att}\left(?x'_1,\ldots,?x'_n\right) : ?v'_1, \left(-\infty, t'_2\right)\right)$$
$$\text{tel que} \quad ?v'_1 = ?v_1, \quad \forall i\ ?x_i = ?x'_i, \quad t'_2 \leqslant t_1$$

- Si $t'_2 < t_1$ alors il existe un lien causal qui maintient la valeur de l'attribut entre l'assertion primordiale et la proposition à expliquer :

$$\textbf{hold}\left(\text{Att}\left(?x''_1,\ldots,?x''_n\right) : ?v''_1, \left(t''_1, t''_2\right)\right)$$
$$\text{tel que} \quad ?v''_1 = ?v_1, \quad \forall i\ ?x_i = ?x''_i, \quad t''_1 = t'_2, \quad t''_2 = t_1$$

En ajoutant que toute proposition de persistance dont l'intervalle temporel est de la forme $]-\infty, t]$ est nécessairement expliquée, nous supprimons alors le problème de récursion sans obliger le programmeur à expliquer lui-même certaines propositions.

On pourrait même autoriser le planificateur à introduire lui-même ces assertions primordiales lorsqu'elles n'existent pas puisque ce n'est qu'une manière comme une autre d'expliquer l'état d'un monde. Mais cela laisse trop de liberté au système et nous préférons continuer à exiger que le plan initial contienne une description complète du monde.

Cette nouvelle méthode d'explication n'est pas encore intégrée à IxTeT.

II.3.4 Cohérence des ressources

Le critère utilisé pour vérifier la cohérence des propositions d'utilisation des ressources d'un plan partiel correspond exactement à l'interprétation sémantique que nous avons donnée (voir § I.4.3 page 19).

Encore une fois, ce critère global n'est pas utilisable pratiquement d'un point de vue algorithmique. Nous allons donc présenter ici le critère de cohérence local qui est utilisé par IxTeT pour vérifié la cohérence de l'ensemble *Res* des propositions d'utilisation de ressources d'un plan partiel.

Ce critère est basé sur la notion d'*ensemble critique* (EC). Un EC est un sous-ensemble de *Res* qui vérifie les propriétés suivantes :

- L'ensemble des contraintes temporelles autorise le recouvrement des intervalles temporels des propositions concernées. (équation II.6)
- L'ensemble des contraintes sur les variables autorise l'unification des attributs de ressource des propositions concernées sur une seule ressource. (équation II.7)
- La somme des quantités utilisées par les propositions concernées dépasse la quantité initiale disponible pour la ressource. (équation II.7)

De manière formelle, on appelle *ensemble critique*, tout sous-ensemble EC de Res :

$$EC = \left\{ \mathbf{use}\left(\mathrm{Res}\left(?v_1{}^k, \ldots, ?v_n{}^k\right) : q^k, \left(t_1{}^k, t_2{}^k\right)\right) \in Res \text{ avec } k \in [1, \ldots, m] \right\}$$

tel que :

$$\forall k \in [1, \ldots, m],\ \forall l \in [1, \ldots, m], \quad \left(C_T \wedge (t_1{}^k < t_2{}^l)\right) \not\models \mathsf{Faux} \tag{II.6}$$

$$\exists R = \mathrm{Res}\,(C_1, \ldots, C_n) \in \mathbf{Res} \text{ tel que}$$
$$\left(\forall i \in [1, \ldots, n],\ \forall k \in [1, \ldots, m]\ C_i \in D_{\epsilon(?v_i{}^k)}\right) \quad \text{et} \quad \left(Q_0(R) < \sum_{k \in [1,\ldots,m]} q^k\right) \tag{II.7}$$

On appelle *ensemble critique minimal* (ECM) tout ensemble critique tel que :

$$\forall E \subset ECM \text{ tel que } E \neq ECM, \quad E \text{ n'est pas un ensemble critique}$$

Un ensemble critique minimal représente en fait un conflit entre plusieurs propositions d'utilisation d'une même ressource qui, si elles avaient lieu simultanément, dépasseraient la capacité de cette ressource. Ces ensembles sont minimaux au sens où, quelle que soit la proposition que l'on réussit à supprimer de l'ensemble, le conflit disparaît.

On démontre que tout plan partiel est cohérent vis-à-vis des propositions d'utilisation de ressources si et seulement s'il ne contient aucun ensemble critique minimal.

II.4 Recherche des défauts et calcul des résolvantes associées

Nous avons déjà expliqué que IxTeT utilisait un algorithme de type A_ϵ pour explorer l'ensemble des plans partiels. Dans cette exploration, la transition d'un plan partiel à un autre se fait par la détection d'un défaut et par l'application d'une résolvante pour supprimer ce défaut. À chaque défaut correspondent plusieurs résolvantes.

II.4.1 Définition d'un défaut et d'une résolvante

Un *défaut* est soit :

1. une proposition inexpliquée ;
2. une menace ;
3. un ensemble critique minimal.

Une résolvante peut être composée :

- de nouvelles contraintes temporelles ($(t_i - t_j) \in I$) ;
- de nouvelles contraintes entre les variables ;
- de liens causaux entre des propositions à expliquer et leurs événements établisseurs ;
- de nouveaux opérateurs de planification (des tâches) afin d'utiliser l'un de leurs effets (événement ou production de ressource).

Donc, d'une manière générale, une résolvante est une conjonction de formules logiques ρ. L'application d'une résolvante ρ à un plan partiel $\mathsf{P_i}$ pour produire le plan partiel

P_{i+1} sera notée λ :

$$P_{i+1} = P_i \lambda \rho$$

Cette opération consiste à réaliser la conjonction des deux ensembles de formules logiques (en prenant garde lorsque l'on insère une nouvelle tâche de renommer ses différents instants et variables pour qu'ils ne correspondent pas à ceux du plan partiel).

L'ensemble des défauts liés à un plan P sera noté :

$$\mathit{Défauts}(P)$$

L'ensemble des résolvantes permettant de supprimer le défaut *def* du plan P sera noté :

$$\mathit{Résolvantes}(P, \mathit{def})$$

Pour chaque type de défaut, il existe plusieurs résolvantes possibles. Nous allons les détailler dans les trois sections suivantes.

II.4.2 Les résolvantes d'un défaut de type « proposition inexpliquée »

Soit une proposition inexpliquée α de la forme :

$$\begin{aligned} \alpha \equiv \quad & \textbf{event}\,(\mathrm{Att}\,(?x_1, \ldots, ?x_n) : (?v_1, ?v_2), t_1) \\ \text{ou } \alpha \equiv \quad & \textbf{hold}\,(\mathrm{Att}\,(?x_1, \ldots, ?x_n) : ?v_1, (t_1, t_2)) \end{aligned}$$

Soit un événement *evt* :

$$\mathit{evt} \equiv \textbf{event}\left(\mathrm{Att}\left(?x'_1, \ldots, ?x'_n\right) : \left(?v'_1, ?v'_2\right), t'_1\right)$$

Pour que l'événement *evt* puisse être l'événement établisseur de α, il faut et il suffit que la contrainte suivante ne soit pas incompatible avec le plan partiel courant :

$$\left(\forall i \in [1, \ldots, n]\ ?x'_i = ?x_i\right) \text{ et } \left(?v'_2 = ?v_1\right) \text{ et } \left(t'_1 \leqslant t_1\right) \qquad \text{(II.8)}$$

Il faut de plus ajouter le lien causal entre l'événement établisseur et la proposition à expliquer :

$$\textbf{hold}\left(\mathrm{Att}\,(?x_1, \ldots, ?x_n) : ?v_1, \left(t'_1, t_1\right)\right)$$

L'événement *evt* peut provenir soit du plan partiel courant P soit d'une tâche T^i.

Dans le premier cas, la résolvante ρ est de la forme :

$$\rho \equiv \left(\forall i \in [1,\ldots,n]\ ?x_i' = ?x_i\right) \wedge (?v_2' = ?v_1) \wedge (t_1' \leqslant t_1) \wedge \mathbf{hold}\left(\mathrm{Att}\,(?x_1,\ldots,?x_n) : ?v_1, (t_1', t_1)\right)$$

Lorsque l'événement établisseur *evt* provient d'une tâche T^i, la résolvante ρ est alors :

$$\rho \equiv T^i \wedge \left(\forall i \in [1,\ldots,n]\ ?x_i' = ?x_i\right) \wedge (?v_2' = ?v_1) \wedge (t_1' \leqslant t_1) \wedge \mathbf{hold}\left(\mathrm{Att}\,(?x_1,\ldots,?x_n) : ?v_1, (t_1', t_1)\right)$$

Notons que pour chaque proposition inexpliquée, il existe autant de résolvantes que d'événements provenant de P ou des tâches de T et répondant au critère (II.8). Nous noterons $\rho(\alpha)$ l'ensemble des résolvantes possibles d'une proposition inexpliquée α.

Si l'on intègre la notion d'assertion primordiale, il faut alors augmenter l'ensemble des résolvantes d'une proposition inexpliquée par la résolvante consistant à ajouter un lien causal entre la proposition à expliquer et une assertion primordiale. Cette résolvante est pratiquement identique à celle permettant de lier une proposition à un événement établisseur.

II.4.3 Les résolvantes d'un défaut de type « menace »

Soit une proposition de persistance α concernant l'attribut $\mathrm{Att}\,(?x_1,\ldots,?x_n)$ et s'exprimant par :

$$\alpha \equiv \mathbf{hold}\,(\mathrm{Att}\,(?x_1,\ldots,?x_n) : ?v, (t_1, t_2))$$

S'il existe une menace sur α sous la forme d'un événement *evt* :

$$evt \equiv \mathbf{event}\left(\mathrm{Att}\,(?x_1',\ldots,?x_n') : (?v_1', ?v_2'), t_1'\right)$$

l'ensemble des résolvantes est alors :

$$\rho(\alpha, evt) = \Big\{\rho \in \big\{(?x_1' \neq ?x_1),\ \ldots,\ (?x_n' \neq ?x_n),\ (t_1' < t_1),\ (t_2 < t_1'),\ \big((t_1' = t_1) \wedge (?v_2' = ?v)\big),\ \big((t_1' = t_2) \wedge (?v_1' = ?v)\big)\big\} \text{ tel que } (C_V \wedge C_T \wedge \rho) \not\models \mathsf{Faux}\Big\}$$

S'il existe une menace sur α sous la forme d'une proposition de persistance hld:

$$hld \equiv \textbf{hold}\left(\text{Att}\left(?x_1', \ldots, ?x_n'\right) : ?v_1', \left(t_1', t_2'\right)\right)$$

l'ensemble des résolvantes est alors:

$$\rho(\alpha, hld) = \Big\{\rho \in \big\{(?x_1' \neq ?x_1),\ \ldots,\ (?x_n' \neq ?x_n),\ (t_2' < t_1),\ (t_2 < t_1'),\ \big((?x_1' = ?x_1) \wedge \cdots \wedge (?x_n' = ?x_n) \wedge (?v_1' = ?v)\big)\big\}$$
$$\text{tel que } (C_V \wedge C_T \wedge \rho) \not\models \text{Faux} \Big\}$$

II.4.4 Les résolvantes d'un défaut de type « ensemble critique minimal »

Soit ECM un ensemble critique minimal, il existe trois types de résolvantes possibles pour supprimer cet ensemble critique:

- Ajouter une contrainte temporelle entre deux propositions quelconques appartenant à ECM afin qu'elles n'utilisent plus la ressource durant le même intervalle temporel;
- Ajouter une contrainte d'inégalité entre des variables provenant de deux propositions quelconques appartenant à ECM afin qu'elles n'utilisent plus la même ressource;
- Insérer une (ou plusieurs) instance d'une (ou plusieurs) tâche T^{i} capable de produire une ressource utilisée par ECM en imposant que cette production ait lieu avant l'une quelconque des propositions appartenant à ECM.

Nous ne détaillerons ici ni la méthode ni les algorithmes utilisés par $\mathrm{I_XT_ET}$ pour calculer l'ensemble des résolvantes liées à un ECM. Ils sont présentés complètement dans [Laborie 95]. Nous dirons simplement qu'il existe un algorithme polynômial permettant de trouver un ensemble minimal (au sens non-redondant) de résolvantes à partir du moment où on se limite à des types de ressources d'arité inférieur ou égal à 1. De plus, ce calcul n'est pas refait à chaque modification du plan afin d'augmenter l'efficacité du système. Une fois calculé, cet ensemble de résolvantes est juste remis à jour jusqu'au moment où il devient vide (c'est à dire déjà valide) ou incohérent.

En pratique, les conflits (représentés par des ECM) sont souvent limités à trois ou quatre propositions, ce qui n'entraine pas une complexité excessive du calcul des résolvantes liées à un ECM.

II.5 La planification et son contrôle

Pour poser un problème à IxTeT, il faut lui spécifier :

$$\mathsf{P}_0 = \mathsf{P}_{\mathsf{Monde}} \wedge \mathsf{P}_{\mathsf{Buts}} = (Evt_0 \wedge Hld_0 \wedge Res_0 \wedge C_{V0} \wedge C_{T0})$$
$$\mathsf{T} = \{\mathsf{T}^{\mathsf{i}} = (Evt^{\mathsf{i}} \wedge Hld^{\mathsf{i}} \wedge Res^{\mathsf{i}} \wedge C_V^{\mathsf{i}} \wedge C_T^{\mathsf{i}}) \text{ avec } \mathsf{i} \in [1, \ldots, \mathsf{t}]\}$$

Où :

P_0 est la plan partiel initial tel que $(C_{T0} \wedge C_{V0}) \not\models \mathsf{Faux}$

T est l'ensemble des t tâches insérables tel que $\forall i\ (C_T^{\mathsf{i}} \wedge C_V^{\mathsf{i}}) \not\models \mathsf{Faux}$

Ces deux conditions sont vérifiées par le compilateur. La cohérence temporelle et celle vis-à-vis des contraintes sur les variables sont donc garanties tant pour le plan initial que pour les tâches.

L'objectif du planificateur est de trouver un chemin entre P_0 et un plan $\mathsf{P}_{\mathsf{Solution}}$ (c.-à-d. sans défauts) avec :

$$(\mathsf{P}_0 \curlywedge \rho_0 \curlywedge \cdots \curlywedge \rho_n) = \mathsf{P}_{\mathsf{Solution}}$$

tel que :

$$(\mathsf{P}_{\mathsf{Monde}} \curlywedge \rho_0 \curlywedge \cdots \curlywedge \rho_n) \models \mathsf{P}_{\mathsf{Buts}}$$

II.5.1 Algorithme de base

La boucle (algorithme II.1 page suivante) principale de planification est basée sur un algorithme d'exploration en profondeur guidé par une heuristique du type A_ϵ pour choisir, en cas d'échec, un nouveau nœud à développer (algorithme dit de « backtrack search »). Cet algorithme recherche un chemin à partir de P_0 jusqu'à un plan solution (un plan sans défaut) dans un graphe où chaque nœud est un plan partiel (cohérent temporellement et vis-à-vis des contraintes sur les variables) et où les arêtes sont les résolvantes des défauts de ces plans.

À chaque étape de planification, IxTeT sélectionne un défaut du plan courant puis il choisit une résolvante particulière parmi celles supprimant ce défaut. On montre que, pour garantir la complétude du processus de planification, les seuls retours-arrière

Alg. II.1 PLANIFIER(P, T)

```
- ∴ - P est le plan partiel à rendre cohérent en insérant éventuellement des instances
     des tâches contenues dans l'ensemble T
- ∴ - Les retours-arrières ne s'effectuent que sur les choix et non pas sur les sélections
Début
    Si Défauts(P) = ∅ Alors
        Retour (P)    - ∴ - P est cohérent
    Sinon
        Sélection d'un défaut def pris dans Défauts(P)
        Si Résolvantes(P, def) = ∅ Alors
            - ∴ - P contient un défaut qui ne peut pas être supprimé
            - ∴ - Le planificateur effectue un retour-arrière
            Retour (Échec)
        Sinon
            Choix d'une résolvante ρ prise dans Résolvantes(P, def)
            - ∴ - Appel récursif avec retour-arrière en cas d'échec
            Retour (PLANIFIER(P ⅄ ρ, T))
        Fin Si
    Fin Si
Fin
```

nécessaires sur les choix effectués portent sur le choix de la résolvante et non pas du défaut. Intuitivement, on comprend que l'objectif du planificateur consiste à supprimer tous les défauts. Un choix judicieux de l'ordre dans lequel sont traités ces défauts, s'il n'influence en rien la complétude du processus, permet de limiter la taille du graphe à explorer. Dans I_XT_ET, ce choix se base sur une hiérarchie d'abstraction des attributs qui évolue dynamiquement et qui favorise le choix des défauts concernant des attributs principaux.

II.5.2 La stratégie de moindre engagement et l'heuristique mise en œuvre

Pour effectuer le choix d'une résolvante ou le choix du prochain plan partiel à développer lors d'un retour-arrière, I_XT_ET se base sur une stratégie de moindre engagement.

On définit la fonction **S** qui à chaque plan partiel associe le nombre de plan solutions (c.-à-d. cohérents) accessibles à partir de ce plan par application d'une succession de résolvantes (donc accessibles par le graphe de plans partiels que nous avons défini précédemment). En faisant abstraction de l'aspect temporel, le fait que les domaines

des variables sont finis implique que cette fonction a toujours une valeur fini. En tenant compte de l'aspect temporel, dans certain cas, cette valeur est infinie. Mais, dans le cadre d'IXTET, l'implémentation informatique discrète de l'ensemble des durées et des dates permet, là encore, de donner une valeur finie à cette fonction.

L'application d'une résolvante ρ à un plan partiel P produit un nouveau plan $(P \curlywedge \rho)$ pour lequel le nombre de plans solutions accessibles est obligatoirement inférieur ou égal :

$$\mathbf{S}(P \curlywedge \rho) \leqslant \mathbf{S}(P)$$

On définit la fonction **Engagement** qui, à chaque couple (P, ρ), associe une valeur dans $[0, 1]$ correspondant à la mesure de l'engagement de la résolvante ρ pour le plan partiel P :[4]

$$\mathbf{Engagement}(P, \rho) = 1 - \frac{\mathbf{S}(P \curlywedge \rho)}{\mathbf{S}(P)}$$

La stratégie de moindre engagement consiste simplement à choisir en priorité la résolvante dont l'engagement est minimal de manière à augmenter la probabilité de trouver un plan solution.

Pour guider l'algorithme A_ϵ lors de ses retours-arrière, IXTET utilise aussi la stratégie de moindre engagement. L'estimation f d'un plan partiel P_i lié à un nœud du graphe est la somme de deux fonctions g et h. La fonction g représente la somme de tous les engagements réalisés depuis le plan initial. La fonction h est l'estimation de l'engagement minimal pour aboutir à un plan solution (c.-à-d. la somme des engagements minimaux pour résoudre chacun des défauts encore présents dans P_i) :

$$f(P_i) = g(P_i) + h(P_i)$$

Où :

$$g(P_i) = \sum_{p \in [0,\ldots,i-1]} \mathbf{Engagement}(P_p, \rho_p)$$

4. On remarque deux cas particuliers :

Engagement$(P, \rho) = 0$ qui signifie que ρ ne change rien à P $(P \models \rho)$
Engagement$(P, \rho) = 1$ qui signifie que $(P \curlywedge \rho)$ est incohérent $((P \curlywedge \rho) \models \text{Faux})$

Pratiquement, nous ne rencontrerons jamais ces deux cas dans IXTET puisque toutes les résolvantes calculées sont utiles et cohérentes.

$$h(\mathsf{P_i}) = \sum_{def \in \mathit{Défauts}(\mathsf{P_i})} \min\Big(\{e = \textbf{Engagement}(\mathsf{P_i}, \rho) \text{ avec } \rho \in \mathit{Résolvantes}(\mathsf{P_i}, def)\}\Big)$$

Lors de ses retours-arrière, IxTeT privilégie le choix du plan partiel P pour lequel $f(\mathsf{P})$ est minimal.

Deux remarques s'imposent tout de même. Tout d'abord, il est pratiquement impossible de calculer la fonction **Engagement**. IxTeT utilise donc des heuristiques liées à chaque type de résolvantes pour calculer une estimation « réaliste » de cette fonction. Ces heuristiques sont complètement décrites dans [Laruelle 94] et [Laborie 95]. D'autre part la fonction h n'est pas obligatoirement une heuristique minorante. Le plan solution généré par IxTeT n'est donc pas obligatoirement optimal vis-à-vis du critère d'engagement minimum. Heureusement l'ensemble des expérimentations faites avec IxTeT montrent très clairement que ces estimations sont suffisamment efficaces pour produire *rapidement* des plans de bonne qualité (c.-à-d. peu contraints et donc flexibles).

II.6 Spécification et résolution de problèmes de planification

Pour alimenter le système de planification (plan initial ou tâches), nous utilisons un langage de description de plans partiels. Ces fichiers de description sont compilés.

Tous les attributs et toutes les ressources doivent être déclarés avant d'être utilisés afin d'éviter les fautes de frappe mais aussi pour garantir que toutes les variables utilisées ont un domaine de valuation fini (pour garantir la complétude du processus de planification).

Chaque tâche déclarée dans le fichier source sera compilée en un plan partiel dans le fichier résultat. Le compilateur, outre la vérification syntaxique, garantit :

- la cohérence temporelle des tâches
- la cohérence des tâches vis-à-vis des contraintes sur les variables
- la non-redondance des instants et des variables (tous les instants ou variables unifiés sont réunis sur une seule instance).

Pour planifier, on spécifie à IxTeT quelle est la tâche initiale (le plan partiel $\mathsf{P_0}$) et quel est l'ensemble des tâches qu'il a le droit d'insérer (l'ensemble T).

II.6.1 Exemple de description de tâches

La figure II.1 page ci-contre présente un exemple de tâche exprimé dans le langage de description de plan partiel d'IxTeT (extrait d'un problème de planification multi-robots).

La tâche R1SortDe (ligne 20) décrit le déplacement du robot R1 à partir de l'une des trois pièces Bureau1, Bureau2 ou Bureau3 jusque dans la pièce Couloir.

Le seul attribut d'état modifié par cette tâche est l'attribut Position(?robot) (déclaré à la ligne 8) qui prend ses valeurs dans le domaine Positions (l'union des domaines Bureaux et {Couloir} déclarée à la ligne 6) et qui indique la position du robot.

Entre l'instant (start) de début de la tâche et l'instant (sortDe) où le robot sort de la pièce, on impose par l'intermédiaire d'un **hold** (ligne 25) que cet attribut reste à la valeur de la variable ?de. Puis cet attribut change de valeur (ligne 26) et doit rester à la valeur Couloir jusqu'à l'instant de fin de la tâche (ligne 27).

Bien qu'elle utilise l'attribut générique Position, cette tâche est spécifique au robot R1. C'est pourquoi nous utilisons la constante R1 comme paramètre de l'attribut Position.

Chaque tâche est bornée temporellement par un instant de début et un instant de fin contraints par (start < end). Tous les autres instants présents dans la tâche doivent avoir lieu entre ces deux instants. Nous avons ajouté explicitement deux contraintes temporelles supplémentaires qui indiquent que l'instant de sortie de la pièce aura lieu entre 1 et 3 minutes après le début de la tâche (ligne 32) et que cette tâche sera terminée entre 1 et 3 minutes après l'entrée dans le couloir (ligne 33).

Le déplacement du robot utilisant les moteurs, nous spécifions que cette tâche consomme 10 unités de la ressource ChargeBatterieR1 (ligne 29). Cette consommation est continue entre le début et la fin de la tâche mais, IxTeT ne sachant gérer que les consommations (ou les productions) instantanées, nous avons arbitrairement fixé la date de consommation d'énergie à l'instant où le robot sort de la pièce.

La gestion de la batterie du robot illustre une autre limitation de la représentation des ressources utilisées dans IxTeT. En effet la ressource ChargeBatterieR1 contraint le planificateur à recharger la batterie du robot dès lors que son niveau de charge ne suffit plus à répondre au besoin de consommation des tâches. IxTeT garantit donc qu'en aucun cas notre robot ne tombera en panne de batterie (le niveau de la ressource ne peut descendre en dessous de zéro). Par contre, rien n'empêche le niveau de cette ressource

```
1   constant ConsommationDeplacement = 10;
2   constant DureeDeplacementMin = 1:00;
3   constant DureeDeplacementMax = 3:00;
4   constant Robots = {R1, R2, R3, R4};
5   constant Bureaux = {Bureau1, Bureau2, Bureau3}
6   constant Positions = Bureaux | {Couloir};
7
8   attribute Position(?robot) {
9       ?robot in Robots;
10      ?value in Positions;
11  }
12
13  resource ChargeBatterieR1() {
14      capacity = 50;
15  }
16  resource DechargeBatterieR1() {
17      capacity = 0;
18  }
19
20  task R1SortDe(?de) (start, end) {
21      ?de in Bureaux;
22
23      timepoint sortDe;
24
25      hold (Position(R1):?de , (start , sortDe));
26      event (Position(R1): (?de, Couloir), sortDe);
27      hold (Position(R1): Couloir, (sortDe, end));
28
29      consume (ChargeBatterieR1, ConsommationDeplacement, sortDe);
30      produce (DechargeBatterieR1, ConsommationDeplacement, sortDe);
31
32      (end − sortDe) in [DureeDeplacementMin, DureeDeplacementMax];
33      (sortDe − start) in [DureeDeplacementMin, DureeDeplacementMax];
34  }
```

FIG. II.1 – *Exemple de description de tâche*

	Nombre de nœuds	Retours-arrière	Temps (s)
Paramètres par défaut	396	51	45.6
Paramètres optimaux	101	7	6.3

FIG. II.2 – *Influence des paramètres de contrôles sur les performances d'I_XT_ET*

de dépasser son niveau initial (par exemple via l'insertion de plusieurs tâches de rechargement de la batterie dès le début du plan). Pour empêcher cela, nous sommes obligés d'introduire une seconde ressource (DechargeBatterieR1) qui, utilisée en opposition de la ressource ChargeBatterieR1, garantit que nous ne dépasserons jamais la charge maximale.

Cette technique de codage par double ressources, consommées et produites en opposition, s'est révélée absolument indispensable pour spécifier les problèmes mettant en jeu des ressources de capacité limitée (batterie, parking, *etc.*). Il devrait être possible, sans trop de modifications, d'intégrer directement ce type de ressources dans I_XT_ET afin de pouvoir lui appliquer des algorithmes de gestion spécifique plus efficace que la gestion générale de deux ressources que le système considère actuellement de manière totalement découplée.

II.6.2 Exemple de résolution de problème de planification

Nous avons mis en annexe (annexe C page 131) un exemple de problème complet. Nous ne rentrerons pas dans le détail des techniques de modélisation d'un problème.

Nous n'avons pas encore parlé des paramètres de contrôles d'I_XT_ET. L'heuristique utilisée est contrôlée par un certain nombre de paramètres. Selon les valeurs choisies, un même problème peut se révéler soluble très rapidement ou au contraire amener le planificateur à explorer de nombreuses branches inutiles.

Parmi les principaux paramètres, on peut en remarquer deux qui influencent très fortement la qualité des résultats. Le premier paramètre permet de contrôler le moment où la résolution des défauts de type faisabilité est déclenchée (la faisabilité concerne la résolution des défauts d'explication d'assertions). Le second est similaire mais concerne cette fois la satisfiabilité (la résolution des défauts de type menace).

Le tableau II.2 présente les mesures de performances du planificateur sur l'exemple que nous présentons en annexe (annexe C page 131). La détermination des paramètres

optimaux a été effectuée par essais et erreurs. Cette détermination reste encore du domaine de l'expertise puisque il faut nécessairement bien connaître le fonctionnement interne d'I_XT_ET pour aboutir à des valeurs acceptables.

Nous ne nous étendrons pas plus longtemps sur ces problèmes qui sont pourtant fondamentaux pour le bon fonctionnement du planificateur et donc pour celui des améliorations que nous lui apportons.

II.6.3 Visualisation des solutions

Les plans solutions produits par I_XT_ET recouvrent en fait un ensemble de plans dans lesquels les tâches à effectuer (ou plus finement les événements) sont contraintes temporellement. Mais, il est encore possible de modifier la plupart des dates relatives de début et de fin tâche (dans la mesure où on respecte les contraintes temporelles incluses dans le plan).

Afin de visualiser ces solutions, nous avons réalisé un petit utilitaire qui, dans sa fenêtre d'affichage du plan, permet à l'utilisateur de déplacer les tâches dans le temps en respectant les contraintes. Les deux copies d'écrans (figure II.3 page suivante) montrent un même plan solution avec deux instanciations temporelles différentes.

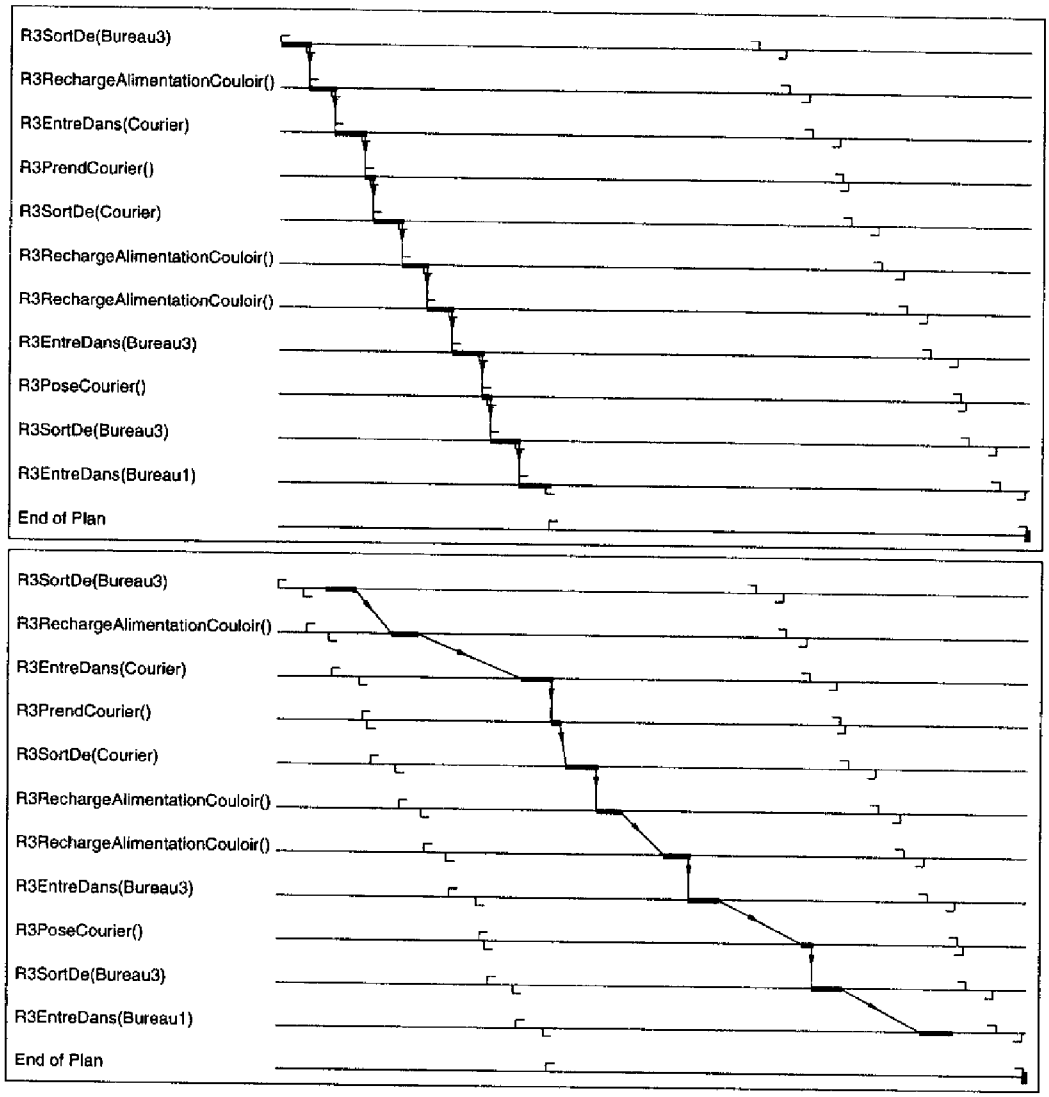


FIG. II.3 – *Visualisation du même plan solution avec instanciation temporelle au plus tôt puis instanciation temporelle quelconque*

Chapitre III

Opérateurs sur les plans partiels

La planification classique prend comme données initiales un plan vide et un ensemble de buts pour produire un plan solution (s'il existe) en insérant peu à peu des tâches afin de satisfaire les buts. I_XT_ET, quant à lui, prend comme données initiales un plan partiel quelconque qu'il va essayer de rendre cohérent en insérant éventuellement de nouvelles tâches.

Nous avons dit précédemment que ce plan initial contenait la description du monde et les buts à atteindre. Mais rien n'empêche de prendre, comme plan initial, un plan partiel construit à partir d'autres informations. En fait, I_XT_ET n'a besoin d'aucune information a priori sur la composition de son plan partiel initial.

Nous pouvons utiliser cette propriété pour réaliser des nouveaux opérateurs combinant des plans partiels afin d'essayer de réaliser une planification incrémentale ou distribuée. Nous allons décrire dans ce chapitre ces nouveaux opérateurs.

III.1 Définitions

Nous allons tout d'abord définir quelques termes afin de préciser notre propos. Nous utiliserons les expressions :

union de plans lorsque nous réunirons plusieurs plans afin de les rendre cohérents entre eux mais sans modifier aucunement les tâches et événements qui les composent ;

fusion de plans lorsque nous réunirons plusieurs plans afin de les synchroniser en autorisant l'ajout ou la suppression de tâches pour rétablir la cohérence;

insertion de buts lorsque nous insérerons dans un plan existant un ensemble de buts additionnels afin d'élaborer un nouveau plan permettant d'atteindre ces buts en tenant compte du plan précédent mais sans modifier les tâches et événements qui le composaient ;

planification globale lorsque nous réunirons plusieurs ensembles de buts (exprimés sous la forme de plans partiels initiaux) afin de produire un plan cohérent permettant d'atteindre ces buts.

La figure III.1 page suivante illustre ces quatre opérations sur la base d'un exemple volontairement très simple où deux robots doivent chacun passer une même porte (suffisamment large pour qu'ils puissent passer ensemble) qu'ils devront refermer après leur passage[1].

L'opération d'union de plans ordonne simplement les tâches des deux plans. L'opération de fusion de plans supprime en plus les deux tâches inutiles de fermeture et d'ouverture de porte. Grâce à l'insertion des buts de R2 dans le plan de R1, R2 profite de l'ouverture de la porte par R1 pour passer (le plan de R1 n'est en rien modifié). Avec l'opération de planification globale, n'importe lequel des deux robots pourraient ouvrir ou refermer la porte, par contre ils profitent tous les deux de ces opérations pour passer la porte.

1. Dans cette figure le symbole || signifie que deux actions peuvent être réalisées en parallèle.

Données R1	Données R2	Résultats
Union de plans		
Plan R1 R1: Ouvrir Porte R1: Passer Porte R1: Fermer Porte	**Plan R2** R2: Ouvrir Porte R2: Passer Porte R2: Fermer Porte	**Plan Global** R1: Ouvrir Porte R1: Passer Porte R1: Fermer Porte R2: Ouvrir Porte R2: Passer Porte R2: Fermer Porte
Fusion de plans		
Plan R1 R1: Ouvrir Porte R1: Passer Porte R1: Fermer Porte	**Plan R2** R2: Ouvrir Porte R2: Passer Porte R2: Fermer Porte	**Plan Global** R1: Ouvrir Porte R1: Passer Porte || R2: Passer Porte R2: Fermer Porte
Insertion de buts		
Plan R1 R1: Ouvrir Porte R1: Passer Porte R1: Fermer Porte	**Buts de R2** Passer la Porte	**Plan Global** R1: Ouvrir Porte R1: Passer Porte || R2: Passer Porte R1: Fermer Porte
Planification globale		
Buts de R1 Passer la Porte	**Buts de R2** Passer la Porte	**Plan Global** R2: Ouvrir Porte R1: Passer Porte || R2: Passer Porte R1: Fermer Porte

FIG. III.1 – *Illustration des quatre méthodes de combinaison de plans*

III.2 Suppression des liens causaux

La méthode que nous utilisons dans les opérations d'union de plans et d'insertion de buts se base sur *la suppression des liens causaux.*

Nous avons vu que I_XT_ET, lors de son processus d'élaboration de plans solutions, insérait soit des nouvelles contraintes (sur les variables ou sur les instants), soit des nouvelles tâches, soit des liens causaux. Les contraintes et les tâches insérées sont absolument nécessaires pour garantir la bonne exécution du plan et la satisfaction des buts. Par contre, lors de l'ajout d'un lien causal pour expliquer une proposition, seule la contrainte temporelle est nécessaire pour maintenir explicitement l'ordre des tâches. Le lien causal ne sert qu'à garantir la relation de causalité entre les tâches en proposant une manière possible (proposée par le planificateur) d'expliquer les propositions imposées. Si l'on veut s'autoriser à produire une autre explication pour une proposition tout en conservant l'ordre des tâches, il nous faut pouvoir rompre ces liens causaux. Cette remise en cause est nécessaire lors de la composition de plans par l'un ou l'autre des opérateurs de composition de plans partiels que nous allons décrire.

Par exemple, en reprenant notre exemple du passage d'une porte, l'élaboration du plan de R2 explique l'événement d'ouverture de la porte par le fait qu'elle était fermée depuis le début des temps. Lors de l'union, cet événement peut s'expliquer par le fait que R1 a refermé la porte après son passage. Le plan (au sens de ses actions sur le monde) de R2 reste valide mais sa faisabilité s'explique différemment.

Nous allons maintenant présenter formellement le mécanisme de suppression des liens causaux.

III.2.1 Les chaînes de causalité

Soit un plan solution P produit par I_XT_ET. Ce plan P est cohérent vis-à-vis des attributs: toutes ses propositions sont expliquées et il n'existe aucune menace. On démontre alors que, pour un attribut donné Att $(C_1, \ldots, C_n)$, l'ensemble des événements le concernant sont totalement ordonnés (ils forment une chaîne cohérente de causalité).

Définition d'une chaîne de causalité

Soit deux événements distincts du plan P et portant sur le même attribut :

$$e_1 = \textbf{event}\,(\text{Att}\,(C_1,\ldots,C_n) : (\star,\star)\,, t_1)$$
$$e_2 = \textbf{event}\,(\text{Att}\,(C_1,\ldots,C_n) : (\star,\star)\,, t_2)$$

e_1 peut être expliqué de trois manières différentes :

1. Si e_1 est expliqué par e_2, e_2 fait alors partie de la chaîne de causalité de e_1. La définition même de l'explication nous donne alors $t_2 < t_1$.
2. Si e_1 est expliqué par une assertion primordiale, cette assertion est de la forme :

   $$\textbf{hold}\,(\text{Att}\,(C_1,\ldots,C_n) : \star, (-\infty, t_1))$$

   Comme il n'existe aucune menace dans P, on sait qu'il n'existe aucun événement ayant possiblement lieu durant l'intervalle temporel couvert par cette assertion primordiale. On peut donc conclure que $t_2 > t_1$.
3. Si e_1 est expliqué par un autre événement e_3 distincts de e_2, on a alors :

   $$e_3 = \textbf{event}\,(\text{Att}\,(C_1,\ldots,C_n) : (\star,\star)\,, t_3)$$
   avec le lien causal $\textbf{hold}\,(\text{Att}\,(C_1,\ldots,C_n) : \star, (t_3, t_1))$ et $t_3 < t_1$

   Comme il n'existe aucune menace dans P, l'événement e_2 a obligatoirement lieu en dehors de l'intervalle temporel couvert par le lien causal reliant e_1 et e_3. On a donc $t_2 < t_3$ ou $t_2 > t_1$. Une récurrence sur e_3 nous permet alors de nous ramener soit au cas 2 pour conclure que $t_2 > t_1$ soit au cas 1 pour conclure que e_2 est un événement de la chaîne de causalité de e_1 (construite par récurrence) et donc que $t_2 < t_1$.

Dans tous les cas, les deux événements e_1 et e_2 sont ordonnés. Il existe donc un ordre total sur l'ensemble des événements portant sur un même attribut dans un plan P cohérent vis-à-vis des attributs. Cet ordre total définit ce nous appelons *une chaîne de causalité.*

Définition d'un faisceau d'assertions

Comme il n'existe aucune menace dans le plan solution P, toute proposition de persistance est obligatoirement contrainte temporellement par la proposition qui l'explique

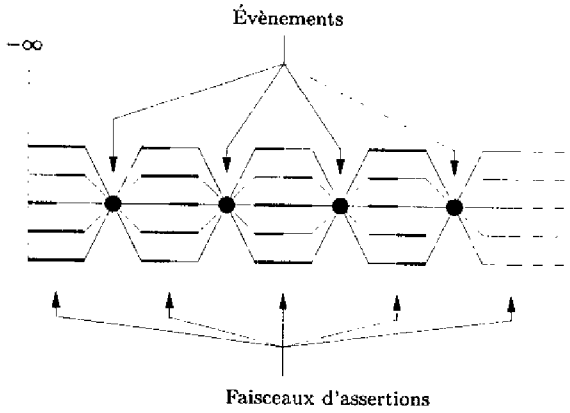


FIG. III.2 – *Chaîne de causalité d'un attribut dans un plan solution*

(événement ou assertion primordiale) et son successeur (événement). Donc l'ordre total sur les événements d'un attribut permet de définir une relation d'équivalence sur les propositions de persistance de cet attribut. Nous appellerons *faisceau d'assertions* (Fsc) une classe d'équivalence de cette relation.

Soit p une proposition portant sur l'attribut $\mathrm{Att}(C_1, \ldots, C_n)$:

$$\mathsf{Fsc}(p) = \left\{ hld = \mathbf{hold}\left(\mathrm{Att}(C_1, \ldots, C_n) : \star, (\star, \star)\right) \text{ tel que } p \text{ explique } hld \right\}$$

Nous pouvons partitionner cet ensemble en deux: l'ensemble Fsc_{hld} contenant toutes les propositions de persistance imposées (soit par le plan partiel initial soit par une tâche insérée) et l'ensemble Fsc_{lc} contenant tous les liens causaux que le planificateur a introduits.

$$\mathsf{Fsc}_{hld}(p) = \left\{ hld \in \mathsf{Fsc}(\mathrm{Att}(C_1, \ldots, C_n), p) \text{ tel que } hld \text{ n'est pas un lien causal} \right\}$$
$$\mathsf{Fsc}_{lc}(p) = \left\{ hld \in \mathsf{Fsc}(\mathrm{Att}(C_1, \ldots, C_n), p) \text{ tel que } hld \text{ est un lien causal} \right\}$$

La figure III.2 montre la chaîne de causalité pour un attribut donné : les événements sont totalement ordonnés et, entre chaque couple d'événements successifs, on trouve un faisceau de propositions de persistance (non nécessairement ordonnées mais obligatoirement contraintes par rapport aux deux événements qui les encadrent) constitué de liens causaux introduits par le planificateur (au minimum celui qui permet d'expliquer les deux événements) et d'assertions provenant soit des tâches, soit de la description initiale du monde, soit encore des buts.

III.2.2 Rupture des chaînes de causalité

Comme nous l'avons déjà montré, la suppression des liens causaux permettra (lors de l'ajout d'un plan, d'un but ou de nouvelles tâches) d'expliquer différemment les événements et assertions existantes en autorisant l'insertion d'un ou plusieurs événements entre les événements d'une chaîne de causalité existante.

On peut distinguer trois niveaux d'analyse pour supprimer les liens causaux :

1. Dans un premier temps, la suppression brutale de tous les liens causaux est la solution de facilité. Mais le planificateur doit alors refaire l'analyse de l'ensemble des attributs sur toute la durée du plan pour reconstruire toutes les chaînes de causalité.
2. Dans un second temps, une analyse plus détaillée permet d'identifier les attributs dont la chaîne de causalité peut être modifiée par l'insertion de nouveaux événements provenant d'un autre plan. Ces attributs que nous nommons attributs communs sont ceux qui sont présents à la fois dans le plan courant et dans l'ensemble des nouvelles propositions qui vont être ajoutées à ce plan. Seules les chaînes de causalité des attributs communs peuvent être remises en cause et doivent donc être rompues.
3. Un troisième niveau d'analyse permet de ne supprimer, dans une chaîne de causalité d'un attribut commun, que les liens causaux potentiellement modifiables. En effet, dans une chaîne de causalité, certains faisceaux de propositions de persistances ne peuvent être rompus : si, par exemple, entre deux événements successifs, il existe une proposition de persistance imposée (qui n'est pas un lien causal), qui a pour borne les instants des deux événements, il est impossible d'insérer de nouveaux événements durant cette proposition et il est donc inutile de rompre la chaîne de causalité entre ces deux instants.

Les deux premiers niveaux d'analyse sont très simples à mettre en œuvre. Nous allons détailler formellement le troisième niveau.

Définition de l'extension nécessaire d'une proposition

Nous définissons l'extension d'une proposition comme étant l'ensemble Ext(p) des propositions de persistance prolongeant nécessairement la valeur d'un attribut à partir

d'une proposition p donnée. Durant l'intervalle temporel couvert par l'ensemble de ces propositions, la valeur de l'attribut reste obligatoirement à la valeur imposée par p.

Soit p un événement ou une proposition primordiale:

$$p = \textbf{event}\,(\text{Att}\,(C_1,\ldots,C_n) : (\star,\star)\,, t)$$
$$\text{ou} \quad p = \textbf{hold}\,(\text{Att}\,(C_1,\ldots,C_n) : \star, (-\infty, t))$$

Soit h, une proposition de la forme:

$$h = \textbf{hold}\,(\text{Att}\,(C_1,\ldots,C_n) : \star, (t_1, t_2))$$

Les deux équations suivantes donnent une définition récursive de $\mathsf{Ext}(p)$:

$$h \in \mathsf{Ext}(p) \quad \Leftarrow \quad \Big(h \in \mathsf{Fsc}_{hld}(p) \text{ et } C_T \models (t_1 \leqslant t)\Big) \tag{III.1}$$

$$h \in \mathsf{Ext}(p) \quad \Leftarrow \quad \Big(h \in \mathsf{Fsc}_{hld}(p) \text{ et } \exists\, h' = \textbf{hold}\,\big(\text{Att}\,(C_1,\ldots,C_n) : \star, (t_1', t_2')\big) \text{ tel que } h' \in \mathsf{Ext}(p) \text{ et } C_T \models (t_1 \leqslant t_2')\Big) \tag{III.2}$$

L'équation (III.1) permet d'inclure dans $\mathsf{Ext}(p)$ toutes les propositions de persistance imposées recouvrant nécessairement la proposition p. L'équation (III.2) y ajoute toutes les propositions de persistance imposées qui recouvrent nécessairement une proposition déjà présente dans $\mathsf{Ext}(p)$.

Critère de suppression d'un lien causal

À partir des définitions précédentes nous pouvons établir un critère de suppression d'un lien causal.

Soit lc, un lien causal appartenant à $\mathsf{Fsc}_{lc}(p)$:

$$lc = \textbf{hold}\,(\text{Att}\,(C_1,\ldots,C_n) : \star, (t_1, t_2))$$

Ce lien causal est supprimable si et seulement si la proposition suivante est vérifiée:

$$\nexists h, \text{ avec } h = \textbf{hold}\,\big(\text{Att}\,(C_1,\ldots,C_n) : \star, (t_1', t_2')\big) \text{ tel que } \quad h \in \mathsf{Ext}(p) \text{ et } C_T \models (t_2 \leqslant t_2') \tag{III.3}$$

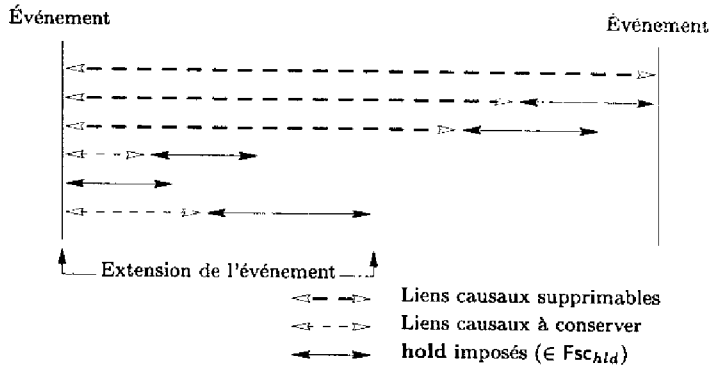


FIG. III.3 – *Application du critère de suppression des liens causaux*

Ce qui signifie que le lien causal n'est pas nécessairement entièrement recouvert par une ou plusieurs propositions de persistances imposées et appartenant à l'extension de p. La figure III.3 explicite cette notion.

III.3 L'union de plans

L'opération d'union de deux ou plusieurs plans sera notée :

$$\underset{ct}{\bowtie}(\mathsf{P}_1, \ldots, \mathsf{P}_n) = \begin{cases} \mathsf{P}_{\bowtie} & \text{si l'union est faisable} \\ \emptyset & \text{s'il n'existe pas de solution} \end{cases}$$

L'indice ct de l'opérateur d'union de plans représente les contraintes temporelles entre les instants des différents plans.

Cette opération d'union est commutative mais pas associative puisque, à chaque fois que l'on réalise une union de plans, le planificateur choisit un ordonnancement particulier entre les tâches des plans à réunir. Une fois choisi, cet ordre ne peut plus être remis en cause.

L'algorithme III.1 page suivante montre les étapes successives d'une union de plans. Nous allons maintenant en détailler le contenu.

Alg. III.1 UNION DE PLANS($P_1, \ldots, P_n, ct$)

– ∴ – *Chaque plan est de la forme :* $P_i = C_{T_i} \wedge C_{V_i} \wedge Hld_i \wedge Evt_i \wedge Use_i$
– ∴ – *Les contraintes temporelles inter-plans sont contenues dans ct*
Début
  – ∴ – *On réalise l'union des contraintes temporelles*
  $C_T \leftarrow ct \wedge C_{T_1} \wedge \cdots \wedge C_{T_n}$
  Si C_T est incohérent **Alors**
    Retour (∅)
  Fin Si

  – ∴ – *On réalise l'union des ensembles de variables*
  RENOMMER LES VARIABLES DE CHAQUE PLAN
  $C_V \leftarrow C_{V_1} \wedge \cdots \wedge C_{V_n}$

  – ∴ – *On suppose que la description du monde est identique*
  SUPPRIMER DANS CHAQUE PLAN LES ÉVÉNEMENTS INITIAUX REDONDANTS
  – ∴ – *On réalise l'union de tous les prédicats*
  $P \leftarrow P_{Monde} \wedge C_T \wedge C_V \wedge (Hld_1 \wedge Evt_1 \wedge Use_1) \wedge \cdots \wedge (Hld_n \wedge Evt_n \wedge Use_n)$
  SUPPRIMER DANS P LES LIENS CAUSAUX CASSABLES DES ATTRIBUTS COMMUNS
  Si MISE À JOUR DES CAPACITÉS DES RESSOURCES échoue **Alors**
    Retour (∅)
  Fin Si

  – ∴ – *Reconstruire la cohérence de l'union des plans*
  PLANIFIER(P, ∅)
  Retour (P)
Fin

III.3.1 Union des contraintes temporelles et des contraintes sur les variables

Les variables de chaque plan sont éventuellement renommées pour qu'elles restent distinctes (normalement, dans un plan solution, toutes les variables devraient avoir un domaine réduit à un singleton et donc elles devraient pouvoir être supprimées). Puis on réalise la conjonction des contraintes sur les variables.

Il nous faut aussi synchroniser les plans entre eux. Pour cela, l'ajout d'au moins une contrainte temporelle inter-plans pour chaque plan est absolument nécessaire pour réunir chaque réseau de contraintes à celui des autres plans. Elle permettra de situer temporellement un plan par rapport aux autres. Cette première contrainte est toujours insérable. Si on ajoute d'autres contraintes, elles peuvent être incohérentes. C'est une

cause possible de l'échec de l'union de plans. L'indice *ct* de l'opérateur d'union de plan $\bowtie$ correspond aux contraintes temporelles inter-plans que l'on veut ajouter (en général, on identifie, par exemple, les instants de départ des plans).

III.3.2 Suppression des événements initiaux redondants

Les plans que nous réunissons doivent avoir été planifiés avec la même description du monde. Cette hypothèse qui peut sembler forte ne l'est pas puisque déjà le mécanisme de planification (pour des raisons de complétude et de cohérence) exige une description exacte et complète du monde. Il serait donc étonnant que plusieurs descriptions exactes et complètes du même monde puissent être différentes.

Cette description du monde peut contenir des événements. Ce sont des événements contingents (c.-à-d. en dehors du contrôle du planificateur). Pour rétablir la cohérence de l'union des plans, il faut supprimer les occurrences multiples de ces événements pour n'en garder qu'une (nous avons imposé l'unicité des événements – cf § II.3.3 page 30). Le même mécanisme peut être appliqué aux assertions contingentes mais il n'est nécessaire que pour des raisons d'efficacité puisque une occurrence multiple d'une proposition de persistance n'est pas une incohérence.

Remarque : un plan est généralement contraint temporellement par rapport à la description du monde qu'on lui a donné. Dans le cas de l'union des plans, ces contraintes temporelles constituent des contraintes temporelles inter-plans puisque les événements initiaux sont identiques. L'ensemble *ct* peut donc être vide si il existe au moins une contrainte temporelle entre un instant décrivant le monde et un instant de chacun des plans.

III.3.3 Identification des attributs communs et suppression des liens causaux

Pour chaque attribut, on examine l'ensemble des événements et des liens causaux le concernant. S'il existe au moins un événement dans un plan et un lien causal supprimable dans un autre plan concernant tous les deux cet attribut, alors on supprime, dans tous les plans, les liens causaux supprimables le concernant.

On peut ajouter un quatrième niveau d'analyse de « supprimabilité » d'un lien causal: pour qu'un lien causal d'un plan soit supprimable, il faut qu'il réponde au

FIG. III.4 – *Capacités d'une ressource commune lors de l'union de deux plans*

	P_1	P_2	Union
Capacité initiale	10	10	10
Capacité initiale virtuelle	18	14	22 $(= 10 + (18 - 10) + (14 - 10))$

critère de l'équation III.3 page 56 mais aussi qu'il existe un événement d'un autre plan qui constitue une menace pour lui (dans le sens où d'une part les contraintes temporelles inter-plans autorisent cet événement à avoir lieu durant le lien causal et d'autre part cet événement porte sur le même attribut commun).

Toutes les propositions qui étaient précédemment expliquées grâce à l'un des liens causaux supprimés ne sont plus expliquées. Toutes les autres le demeurent.

III.3.4 Mise à jour des capacités des ressources communes

Une ressource commune est une ressource pour laquelle il existe, dans plusieurs plans, des propositions d'utilisation.

Pour chaque ressource commune, il faut vérifier que sa capacité initiale a bien été déclarée de manière similaire dans tous les plans. D'autre part, toutes les propositions de production ayant été remplacées par leur équivalent en terme de propositions d'emprunt (avec une augmentation virtuelle de la capacité initiale), il nous faut additionner ces augmentations. La figure III.4 donne un exemple de la modification de la capacité virtuelle d'une ressource commune lors de l'union de deux plans.

Contrairement aux propositions portant sur les attributs d'état, les propositions d'utilisation de ressources ne sont pas nécessairement ordonnées dans un plan solution. En cas de conflit entre deux plans pour l'utilisation d'une ressource, le mécanisme de rétablissement de la cohérence du planificateur fonctionnera parfaitement. Il n'y a donc aucun autre traitement particulier à appliquer à ces propositions.

III.3.5 Synchronisation de l'union des plans

Une fois les contraintes réunies et vérifiées, les liens causaux des attributs communs supprimés et les capacités des ressources communes mise à jour, il nous faut maintenant rétablir la cohérence du plan commun. Pour cela nous utilisons le mécanisme de base du planificateur qui va supprimer peu à peu les défauts du plan. Mais comme on ne veut pas

ajouter de nouvelles tâches au plan, nous lui fournissons, comme ensemble T de tâches insérables, l'ensemble vide. Ainsi les défauts du plan n'ont comme seules résolvantes possibles que l'insertion de nouvelles contraintes ou l'ajout des liens causaux. De plus, comme toutes les variables ont en principe un domaine réduit à un singleton, seules des contraintes temporelles sont insérables.

III.3.6 Résultat d'une union de plans

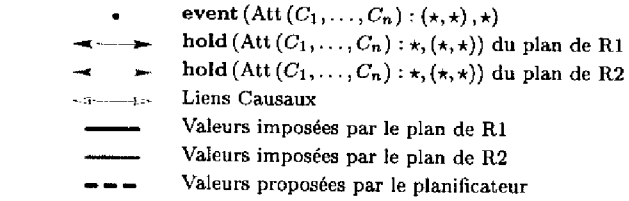
Le résultat (s'il existe) est un plan cohérent qui vérifie toutes les contraintes de chacun des plans mais qui en plus synchronise les actions de manière à garantir la cohérence des changements d'état du monde et l'utilisation des ressources en deçà de leur capacité maximale.

Pour un attribut Att $(C_1, \ldots, C_n)$ qui peut prendre trois valeurs (A, B ou C), la figure III.5 page suivante illustre ce qui se passe lors de l'union des plans hypothétiques de deux robots R1 et R2 (nous avons considéré pour des raisons de simplification que tous les instants étaient reliés par des contraintes temporelles fixes). On notera la présence de liens causaux souvent fort différents entre les deux plans de R1 et R2 et le résultat de l'union. Par contre, comme il se doit, les événements et propositions de persistance reflètent strictement l'union des deux plans.

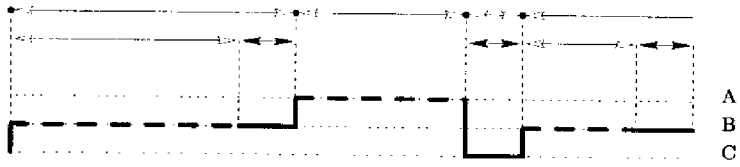
III.4 La fusion de plans

L'opération de fusion de plans se base en partie sur les mêmes opérations que l'union de plans. Il faut réunir les deux plans en leur ajoutant des contraintes de synchronisation. Mais dans ce cas, lors de la phase de reconstruction de la cohérence de l'union des plans, on autorise le planificateur aussi *à insérer* et *à supprimer* des tâches dans les deux plans. Cette opération se rapproche des mécanismes de révision de plan.

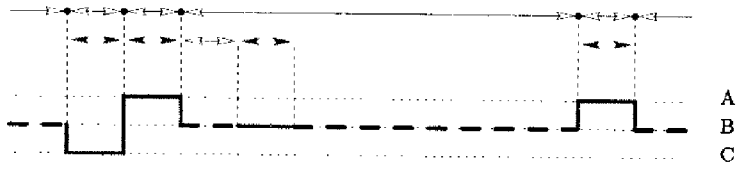
La suppression de tâches dans un plan est un problème qui a déjà été étudié dans le cadre d'I_XT_ET [Chater 95]. Voici, en quelques mots, un rapide aperçu des difficultés rencontrées. En effet il faut savoir si l'on supprime seulement la tâche elle-même ou si l'on supprime aussi toutes les tâches qui ont été insérées pour satisfaire les sous-buts inclus dans la tâche que l'on supprime. Si on choisit de ne supprimer que le tâche elle-même, on risque d'exécuter des tâches inutiles voir même incohérentes avec le plan



Attribut Att $(C_1, \ldots, C_n)$ dans le plan de R1



Attribut Att $(C_1, \ldots, C_n)$ dans le plan de R2



Attribut Att $(C_1, \ldots, C_n)$ dans l'union des plans de R1 et R2

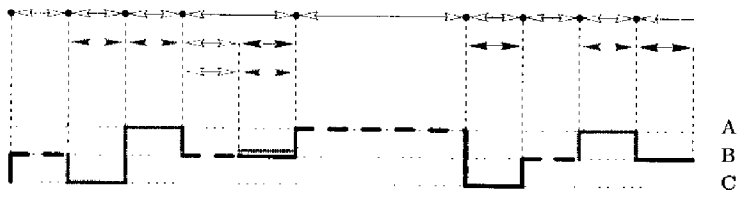


FIG. III.5 – *Résultat d'une opération d'union de plans*

produit. Si on décide de supprimer aussi les tâches insérées pour satisfaire les sous-buts, il se pose un problème de complexité dû à la récursivité de cette suppression mais aussi un problème de complétude car une tâche insérée peut satisfaire deux sous-buts différents dont un seul est contenu dans une tâche à supprimer.

En supposant même qu'on sache supprimer correctement les tâches, encore faut-il déterminer celles qu'il serait pertinent de supprimer. La détermination d'un critère efficace de choix de tâches à supprimer n'est pas simple.

L'exemple (très simple) présenté au début de ce chapitre illustre déjà les problèmes liés à la fusion. En effet comment détecter qu'une tâche de fermeture de porte suivie d'une tâche d'ouverture de porte peuvent être supprimées sans risque pour le reste du plan? Comment garantir que la suppression de l'exécution de ces tâches ne supprimera pas des effets de bord absolument indispensable? Quelle représentation des tâches doit-on fournir au planificateur pour qu'il puisse répondre à ces questions?

La représentation actuelle utilisée dans $I_X T_E T$ ne permet pas de réaliser efficacement la suppression de tâches.

Même sans ce problème de suppression de tâches, le simple fait de pouvoir ajouter de nouvelles tâches nous oblige à supprimer (ou à marquer) un plus grand nombre de liens causaux des plans à fusionner (nous détaillerons ce point en expliquant l'opération d'insertion de buts).

Pour toutes ces raisons, nous n'avons pas mis en œuvre l'opération de fusion de plans.

III.5 L'insertion de buts

Les premières étapes d'une opération d'insertion de buts sont très similaires à celles de l'union de plan. Là encore, il faut supprimer les événements initiaux redondants, les liens causaux portant sur des attributs communs et reconstruire la cohérence du plan ainsi obtenu.

Nous noterons l'opération d'insertion de buts par :

$$P \xleftarrow[ct]{T} B = \begin{cases} P_{\leftarrow} & \text{s'il existe une solution} \\ \emptyset & \text{s'il n'existe pas de solution} \end{cases}$$

Alg. III.2 INSERTION DE BUTS(P, B, T, ct)

– ∴ – *Le plan solution* P *est de la forme :* $\mathsf{P} = \mathsf{P}_{\mathsf{Monde}} \wedge C_{T\mathsf{P}} \wedge C_{V\mathsf{P}} \wedge Hld_{\mathsf{P}} \wedge Evt_{\mathsf{P}} \wedge Use_{\mathsf{P}}$
– ∴ – *Le plan partiel* B *est de la forme :* $\mathsf{B} = \mathsf{B}_{\mathsf{Monde}} \wedge \mathsf{B}_{\mathsf{Buts}}$
– ∴ – *Avec* $\mathsf{B}_{\mathsf{Buts}} = C_{T\mathsf{B}} \wedge C_{V\mathsf{B}} \wedge Hld_{\mathsf{B}} \wedge Evt_{\mathsf{B}} \wedge Use_{\mathsf{B}}$
– ∴ – *Les contraintes temporelles inter-plans sont contenues dans* ct
– ∴ – T *est un ensemble de modèles de tâches*
Début
  – ∴ – *On réalise l'union des contraintes temporelles*
  $C_T \leftarrow ct \wedge C_{T\mathsf{P}} \wedge C_{T\mathsf{B}}$
  Si C_T est incohérent **Alors**
    Retour ($\emptyset$)
  Fin Si

  – ∴ – *On réalise l'union des ensembles des variables*
  RENOMMER LES VARIABLES DE CHAQUE PLAN
  $C_V \leftarrow C_{V\mathsf{P}} \wedge C_{V\mathsf{B}}$

  – ∴ – *On insère les buts dans le plan*
  $\mathsf{P}_{nv} \leftarrow \mathsf{P}_{\mathsf{Monde}} \wedge C_T \wedge C_V \wedge (Hld_{\mathsf{P}} \wedge Evt_{\mathsf{P}} \wedge Use_{\mathsf{P}}) \wedge (Hld_{\mathsf{B}} \wedge Evt_{\mathsf{B}} \wedge Use_{\mathsf{B}})$
  SUPPRIMER DANS P_{nv} LES LIENS CAUSAUX CASSABLES DES ATTRIBUTS COMMUNS
  Si MISE À JOUR DES CAPACITÉS DES RESSOURCES échoue **Alors**
    Retour ($\emptyset$)
  Fin Si

  – ∴ – *Planifier les nouveaux buts insérés dans le plan précédent*
  PLANIFIER(P_{nv}, T)
  Retour (P_{nv})
Fin

Où P représente le plan solution dans lequel on veut insérer le plan partiel B qui contient les nouveaux buts, ces deux plans étant synchronisés entre eux par ct. T est l'ensemble des tâches dont des instances pourront être insérées pour rendre cohérent le plan résultat.

L'algorithme III.2 montre les différentes étapes d'une insertion de buts. L'opération de mise à jour des variables et des contraintes temporelles et celle de suppression des événements initiaux redondants sont identiques à celles de l'union de plans.

III.5.1 Identification des attributs communs et des liens causaux supprimables

La principale différence par rapport à l'union de plans provient surtout du fait que l'on autorise maintenant l'insertion de tâches supplémentaires qui contiennent des propositions concernant des attributs génériques (paramétrés par des variables).

On dira qu'un attribut est commun s'il apparaît dans une proposition du plan P et s'il est unifiable avec un attribut d'une proposition provenant de B ou d'une tâche appartenant à T.

Il faut alors identifier les liens causaux supprimables dans le plan P. Il doivent pour cela concerner un attribut commun tel que nous venons de le définir et répondre au critère de l'équation (III.3) page 56.

Notons tout de suite que le nombre d'attributs communs (et donc le nombre de liens causaux supprimables) est très nettement plus grand lors d'une insertion de buts que lors d'une union de plans :

- Pour l'union de plans les seuls attributs communs sont ceux qui sont modifiés par les deux plans (l'instanciation des variables des attributs est déjà réalisée).
- Pour l'insertion de buts, les attributs communs sont ceux qui sont modifiés par le plan P et que n'importe quelle tâche peut potentiellement modifier indépendamment de ceux qui seront réellement modifiés par le plan produit (le choix des attributs à utiliser pour atteindre le but fixé reste à faire).

Nous pouvons donc envisager non pas la suppression pure et simple des liens causaux supprimables mais plutôt leur marquage pour un traitement ultérieur. On insère alors un nouveau type de résolvante pour les défauts de type menace : si un lien causal marqué comme supprimable est menacé, outre les résolvantes habituelles (contraintes temporelles, contraintes sur les variables), le planificateur peut envisager de supprimer ce lien (rendant ainsi inexpliquée la proposition à laquelle était relié ce lien causal). Ceci semble très simple mais il faut encore estimer l'engagement de ce type de résolvante de manière cohérente par rapport aux autres résolvantes possibles (une estimation réelle mais inutilisable nous amènerait à lui donner une valeur d'engagement négatif puisque la suppression d'un lien causal ne peut qu'augmenter le nombre de plans solutions accessibles).

Cette amélioration pourrait aussi servir dans le cas de l'union de plans. Nous n'avons

pas encore intégré cette amélioration dans $I_X T_E T$ et pour l'instant, comme dans l'union de plans, nous supprimons simplement tous les liens causaux supprimables.

III.6 La planification globale

La planification globale ne pose aucun problème particulier quant à sa mise en œuvre par rapport à une planification simple.

Supposons W problèmes de planification distincts. Chacun de ces problèmes est de la forme :

$$\mathsf{P}_{0w} = \mathsf{P}_{\mathsf{Monde}w} \wedge \mathsf{P}_{\mathsf{Buts}w}$$
$$\text{et } \mathsf{T}_w = \text{l'ensemble des tâches insérables pour réaliser ce plan}$$

Pour que ces plans puissent être planifiés globalement, il faut et il suffit qu'il partage la même description du monde :

$$\forall\, w \in [1, \ldots, W],\ \forall\, w' \in [1, \ldots, W] \quad \mathsf{P}_{\mathsf{Monde}w} = \mathsf{P}_{\mathsf{Monde}w'}$$

Le problème de planification globale que nous poserons s'exprimera alors par :

$$\mathsf{P}_0 = ct \wedge \mathsf{P}_{\mathsf{Monde}} \wedge \bigwedge_{w \in [1,\ldots,W]} \mathsf{P}_{\mathsf{Buts}w}$$
$$\text{et } \mathsf{T} = \bigcup_{w \in [1,\ldots,W]} \mathsf{T}_w$$

Où ct représente l'ensemble des contraintes temporelles permettant de synchroniser les buts entre eux. ct peut être vide puisque les buts sont synchronisés au moins par rapport à $\mathsf{P}_{\mathsf{Monde}}$.

Chapitre IV

La planification incrémentale et distribuée

Dans le chapitre précédent, nous avons décrits les nouveaux opérateurs que nous avons réalisés qui permettent de combiner entre eux des plans. Nous allons maintenant exposer comment ces opérateurs (planification normale, union de plans, insertion de buts et planification globale) peuvent être mis en œuvre afin de réaliser une planification incrémentale et une planification distribuée multi-agents. Nous exposerons ensuite les liens qui existent entre planification et exécution de plans.

IV.1 La planification incrémentale

Dans une application réelle, il est rare de connaître tous les buts à l'avance ou il peut être avantageux de traiter les buts incrémentalement. La plupart du temps, les buts sont présentés successivement au système de planification. Celui-ci est alors face à l'alternative suivante:

- Soit replanifier pour l'ensemble des buts;
- Soit modifier le plan actuel pour atteindre ces nouveaux buts.

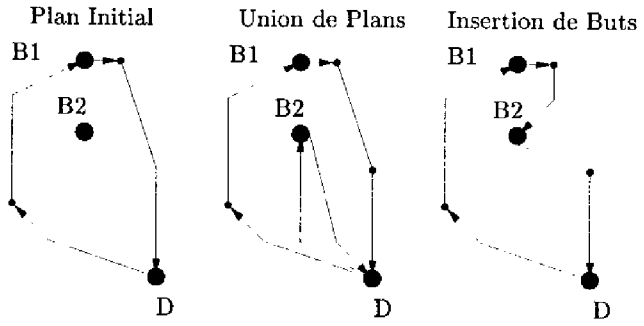


FIG. IV.1 – *Inconvénients de l'union de plans par rapport à l'insertion de buts*

Le premier terme de l'alternative ne peut donner satisfaction pour des raisons de complexité (il est dommage de jeter tout le travail de planification déjà réalisé pour, à chaque fois, replanifier un plan de plus en plus gros) mais aussi pour des raisons liées à l'exécution (on ne peut pas arrêter brutalement le plan précédent qui est déjà en cours d'exécution).

Essayons de mettre en œuvre le second terme de l'alternative grâce aux opérateurs que nous avons décrits précédemment sans nous préoccuper des problèmes liés à l'exécution que nous aborderons plus tard (§ IV.3 page 77).

L'idée initiale consiste à essayer successivement les différentes méthodes permettant d'aboutir à une solution en privilégiant celles dont le coût est le plus faible. On calcule tout d'abord un plan satisfaisant le nouveau but indépendamment du plan précédent et on essaie de réunir ces deux plans. En cas d'échec de l'union de plans, on tente alors une insertion du nouveau but dans le plan courant. Si les deux méthodes précédentes échouent, il faut alors faire une planification globale pour l'ensemble des buts.

Cette idée semble séduisante. Elle nous garantie de trouver une solution (si elle existe). Elle évite de replanifier entièrement à chaque ajout de but. Par contre, la solution produite ne sera pas obligatoirement satisfaisante. En effet la première méthode utilisée (celle qui coûte le moins cher) est l'union de plans. Cette méthode nécessite une description initiale du monde. Quelle sera-t-elle? Dans un premier temps, on peut se contenter d'une description unique qui sera fournie à chaque planification. On s'expose alors à une redondance des tâches effectuées. La solution sera cohérente au sens logique mais ne sera pas satisfaisante. Illustrons cela sur un exemple de type planification de trajectoire.

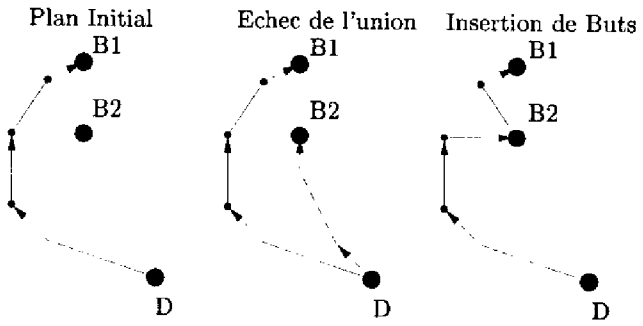


FIG. IV.2 – *Exemple encore plus critique où l'union de plans échoue*

La figure IV.1 page précédente montre le plan initial puis le résultat d'une opération d'union de plans et enfin celui d'une insertion de buts. Un plan permettait de se déplacer jusqu'au point B1 puis de revenir. Le nouveau but consiste à aller jusqu'au point B2 et à revenir. On voit bien dans ce cas que l'insertion de buts produit un plan beaucoup plus intéressant.

La figure IV.2 montre un second exemple dans le cas où le plan initial ne précisait pas de revenir au point de départ. L'union de plan échoue car il est impossible de réunir deux plans dont aucun ne se termine par la situation initiale de l'autre.

On peut penser fournir alors comme plan initial une description du monde reflétant la fin du plan courant. Mais le plan produit aura de grandes chances de ne pouvoir que s'enchaîner à la suite du plan précédent. On ne peut alors bénéficier ni des éventuelles capacités de parallélisme du système pour lequel on planifie ni des actions du plan précédent.

L'union de plans ne peut donc pas donner de bons résultats en général pour réaliser de la planification incrémentale. La raison principale étant que les buts successifs que l'on envoie à un système sont rarement découplés (temporellement, géométriquement et logiquement). La seconde raison est algorithmique : l'avantage de l'union sur l'insertion est dû au plus grand nombre de liens causaux supprimés par l'union mais dans le cas de la planification incrémentale cette différence est beaucoup moins flagrante puisque le système utilise le même ensemble de tâches qui modifient donc les mêmes attributs.

Il est donc clair qu'on peut se passer de l'opération d'union de plans pour effectuer de la planification incrémentale.

Alg. IV.1 PLANIFICATION INCRÉMENTALE(P, B, T, *ct*)

```
- ∴ - P est le plan à mettre à jour
- ∴ - B contient la description des nouveaux buts
- ∴ - T est l'ensemble des tâches insérables
- ∴ - ct représente des contraintes temporelles entre P et B
Début
    - ∴ - On essaie par insertion de buts
    P_nv ← (P ←(T, ct)— B)
    Si P_nv ≠ ∅ Alors
        Retour (P_nv)
    Fin Si
    - ∴ - On essaie une planification globale
    - ∴ - (On suppose que Buts est l'ensemble des plans partiels initiaux prédédents)
    P_nv ← PLANIFIER(B ∧ Buts ∧ ct, T)
    Si P_nv ≠ ∅ Alors
        Retour (P_nv)
    Fin Si
    Retour (échec)
Fin
```

L'algorithme IV.1 montre la technique utilisée pour mettre à jour un plan afin de satisfaire de nouveaux buts. On essaie en premier une insertion de buts. En cas d'échec, on se replie sur une planification globale de l'ensemble des buts.

Cet algorithme nous fournit à coup sûr une solution (si elle existe) puisque en dernier recours une replanification globale de tous les buts sera envisagée. $I_X T_E T$ garantissant dans ce cas la complétude de la planification, nous aurons donc notre solution.

IV.2 La planification distribuée

L'objectif de la planification distribuée est de produire un plan global cohérent pour un ensemble d'agents.

Le qualificatif « *distribué* » peut se comprendre à différents niveaux. Dans tous les cas, on exige qu'il n'existe pas d'entité fixe qui gère l'ensemble du plan (une station centrale par exemple). Un premier niveau de distribution consiste à dire que, au besoin, chaque agent peut se charger de la mise à jour du plan global. C'est le calcul du plan qui est distribué. Un second niveau de distribution est atteint si, à aucun instant, un

agent ne connaît la totalité du plan. En plus du calcul, c'est l'information décrivant le plan global qui est elle-même distribuée.

Pour réaliser ce premier niveau de distribution, il suffit d'appliquer la méthode de planification incrémentale de manière distribuée. Les arguments que nous développions concernant l'inutilité de l'union de plans dans le cas d'un seul agent n'ont plus de sens dans le cadre d'une planification distribuée multi-agents. En effet, les plans de différents agents ne partagent que peu d'attributs ou ressources en comparaison de tous les attributs et ressources qui leur sont propres.

La première tentative d'un agent consistera donc à élaborer de manière indépendante un plan lui permettant d'atteindre ses buts puis d'essayer de le réunir avec le plan global. Cette union pouvant amener l'agent à surcontraindre le plan des autres, elle doit s'effectuer en section critique. Comme pour la planification incrémentale, en cas d'échec, l'agent tentera alors une insertion de buts et même, éventuellement une planification globale. L'algorithme IV.2 page suivante montre le cheminement d'un agent qui vient de recevoir de nouveaux buts.

Cette méthode distribue effectivement le calcul du plan global sur tous les agents mais présente l'inconvénient majeur de centraliser l'ensemble du plan à chaque modification. Nous allons montrer maintenant comment mettre en œuvre un critère de localité pour d'une part réduire la quantité d'information à traiter pour chaque ajout de plan et pour d'autre part éviter la centralisation du plan global.

IV.2.1 La propagation des contraintes temporelles par blocs

La gestion des contraintes temporelles telle que nous l'avons décrite nécessite de connaître l'ensemble des contraintes entre tous les instants. Mais il est possible de décomposer cette gestion lorsqu'on gère des blocs d'instants indépendants ou contraints uniquement par transitivité.

Propagation globale par blocs

Nous appellerons $Pr(M)$ le résultat de la propagation des contraintes temporelles d'une matrice de contraintes M (rappelons que le résultat de cette propagation est

Alg. IV.2 PLANIFICATION DISTRIBUÉE(B, T, *ct*)

```
- ∴ - B contient la description des nouveaux buts
- ∴ - T est l'ensemble des tâches insérables
- ∴ - ct représente les contraintes temporelles entre les buts et le plan global
Début
    - ∴ - On calcule un plan indépendant
    P ← PLANIFIER(B ∧ ct, T)

    - ∴ - Entrée en section critique
    ATTENDRE LE DROIT DE PLANIFIER GLOBALEMENT
    - ∴ - On tente de le réunir avec le plan global courant
    P_G ← RÉCUPÉRER LE PLAN GLOBAL
    P_nv ← ⋈_ct(P_G, P)
    Si P_nv = ∅ Alors
        - ∴ - On essaie par insertion de buts
        P_nv ← (P_G ←_ct^T B)
        Si P_nv = ∅ Alors
            - ∴ - On essaie une planification globale
            Buts ← RÉCUPÉRER L'ENSEMBLE DES PLANS PARTIELS INITIAUX
            P_nv ← PLANIFIER(B ∧ Buts ∧ ct, T)
        Fin Si
    Fin Si
    Si P_nv ≠ ∅ Alors
        MISE À JOUR DU PLAN DES AUTRES AGENTS
    Fin Si
    LIBÉRER LE DROIT DE PLANIFIER GLOBALEMENT
    - ∴ - Sortie de section critique

    Retour (P_nv)
Fin
```

obligatoirement une matrice antisymétrique)[1].

Soit deux matrices de contraintes K et LM complètement propagées et cohérentes, de tailles respectives k et $(l+m)$ et telles que LM soit de la forme :

$$LM = \begin{pmatrix} L & C_{L,M} \\ -C_{L,M}{}^t & M \end{pmatrix}$$

Ajoutons les contraintes $C_{K,L}$ entre les instants de K et les l premiers instants de

1. Nous noterons $-M^t$ la matrice dans laquelle toutes les contraintes de la transposée de M ont été remplacées par leur opposée.

LM et propageons les dans une matrice KL :

$$KL = \begin{pmatrix} KL_1 & KL_{1,2} \\ -{KL_{1,2}}^t & KL_2 \end{pmatrix} = Pr \begin{pmatrix} K & C_{K,L} \\ -{C_{K,L}}^t & L \end{pmatrix}$$

Nous sommes sûrs que $KL_2 \subseteq L$ (2).

Mettons à jour la matrice LM dans une matrice LM' :

$$LM' = \begin{pmatrix} LM'_1 & LM'_{1,2} \\ -{LM'_{1,2}}^t & LM'_2 \end{pmatrix} = Pr \begin{pmatrix} KL_2 & C_{L,M} \\ -{C_{L,M}}^t & M \end{pmatrix}$$

Propageons les mêmes contraintes dans une matrice globale G :

$$G = \begin{pmatrix} G_1 & G_{1,2} & G_{1,3} \\ -{G_{1,2}}^t & G_2 & G_{2,3} \\ -{G_{1,3}}^t & -{G_{2,3}}^t & G_3 \end{pmatrix} = Pr \begin{pmatrix} K & C_{K,L} &]-\infty,+\infty[\\ -{C_{K,L}}^t & L & C_{L,M} \\ ]-\infty,+\infty[& -{C_{L,M}}^t & M \end{pmatrix}$$

On peut démontrer alors les trois propriétés suivantes :

G est cohérente si et seulement si KL est cohérente.

$$KL = \begin{pmatrix} G_1 & G_{1,2} \\ -{G_{1,2}}^t & G_2 \end{pmatrix}$$

$$LM' = \begin{pmatrix} G_2 & G_{2,3} \\ -{G_{2,3}}^t & G_3 \end{pmatrix}$$

Une démonstration formelle de ces résultats pourrait s'appuyer sur les propriétés des fermetures transitives des graphes étant donné que nos deux opérations ($\oplus$ et $\cap$) sont distributives [Aho 74]. Nous ne la développerons pas ici.

En fait le calcul de la matrice G est équivalent à celui de KL et LM', le premier se faisant de manière globale et le second étant un calcul par blocs. La propriété de complétude de la propagation est conservée tant que l'on n'introduit pas de contraintes directes entre K et M. Dans le cas contraire, seul le calcul global de G assure la complétude.

On démontre aussi que le calcul par blocs est plus intéressant, en terme du nombre d'opérations, que le calcul global pour des valeurs de k, l et m proches (en comparant $O((k+l)^3 + (l+m)^3)$ et $O((k+l+m)^3)$).

2. au sens où chaque contrainte de KL_2 est au moins aussi forte que celle de L

Alg. IV.3 MISE À JOUR INCRÉMENTALE PAR BLOCS(LM, KL_2)

```
- ∴ - LM est la matrice à mettre à jour
- ∴ - KL2 est la mise à jour du bloc L de LM
- ∴ - l est la taille du bloc L (et de KL2)
Début
    LM' ← LM
    Pour i ← 1 à l Faire
        Pour j ← 1 à l Faire
            Si KL2(i,j) ⊂ LM'(i,j) Alors
                AJOUTER INCRÉMENTALEMENT LA CONTRAINTE KL2(i,j) DANS LM'
            Fin Si
        Fin Pour
    Fin Pour
    Retour (LM')
Fin
```

Ajout incrémental de contraintes par blocs

Dans le cadre de la planification, la propagation incrémentale est beaucoup plus nécessaire que la propagation globale. Réaliser le calcul par blocs à chaque ajout de contrainte dans la matrice KL est pénalisant puisqu'il nécessite une propagation globale pour mettre à jour la matrice LM'. Mais on peut retarder cette propagation jusqu'à l'ajout de toutes les contraintes. Ce qui se traduit par une complexité de $O(n(k+l)^2 + (l+m)^3)$ à comparer avec la complexité de n ajouts incrémentaux dans la matrice globale G dont la complexité est $O(n(k+l+m)^2)$.

On peut même éviter cette mise à jour par propagation globale en ajoutant incrémentalement dans LM les différences entre KL_2 et L pour obtenir LM'. L'algorithme IV.3 montre alors la propagation des modifications de KL_2 dans LM'.

Au pire cas, la complexité de cet algorithme est $O(l^2(l+m)^2)$ à comparer à celle d'une propagation globale qui est $O((l+m)^3)$. Cette mise à jour par propagation incrémentale ne semble donc pas très intéressante. Mais la complexité moyenne (que nous avons obtenue par simulation sur plusieurs milliers de cas aléatoires) nous montre qu'elle amène en fait un gain non négligeable (de l'ordre de $l/3$ environ) par rapport à la propagation globale[3]. Ceci est dû au fait qu'une matrice de contraintes temporelles

3. Pour obtenir ce résultat, nous avons essayé plusieurs types de parcours de la matrice KL_2 afin d'améliorer les performances moyennes. Un parcours en diagonale n'utilisant que la partie inférieure de

totalement propagée contient beaucoup d'informations redondantes.

En utilisant cet algorithme pour la mise à jour, l'ajout de n contraintes dans KL puis la mise à jour de LM a pour complexité moyenne $O(n(k+l)^2 + 3l(l+m)^2)$ qui reste tout de même supérieure à celle de l'ajout incrémental dans la matrice global G (de complexité $O(n(k+l+m)^2)$).

Utilisation dans le cadre multi-agents

Les trois matrices K, L et M dont nous parlions peuvent décrire le cas ou deux agents veulent réaliser un plan global. K contient tous les instants liés aux prédicats propres au premier agent, M contient tous les instants liés aux prédicats propres au second agent et L contient tous les instants liés aux prédicats communs au deux agents. $C_{K,L}$ et $C_{L,M}$ représentent les contraintes temporelles entre plans.

Dans une coopération entre trois agents ou plus, le nombre de blocs augmente et par transitivité, on récupère de plus en plus d'actions coûteuses de mise à jour par propagation globale (mais sur des matrices plus petites). De plus lorsque l'on obtient un réseau de contraintes directes contenant des cycles entre blocs, nous sommes obligés de revenir à une gestion globale.

$$\text{Matrice gérée par l'agent A} = \begin{pmatrix} A_1 & A_{1,2} & \\ -A_{1,2}{}^t & A_2 & (AB) \\ & -(AB)^t & B_1 \end{pmatrix} \tag{IV.1}$$

$$\text{Matrice gérée par l'agent B} = \begin{pmatrix} B_1 & B_{1,2} & B_{1,3} & \\ -B_{1,2}{}^t & B_2 & B_{2,3} & \\ -B_{1,3}{}^t & -B_{2,3}{}^t & B_3 & (BC) \\ & & -(BC)^t & C_1 \end{pmatrix} \tag{IV.2}$$

$$\text{Matrice gérée par l'agent C} = \begin{pmatrix} C_1 & C_{1,2} \\ -C_{1,2}{}^t & C_2 \end{pmatrix} \tag{IV.3}$$

Nous pouvons illustrer ceci par l'exemple suivant: si un agent A se synchronise avec un agent B qui est lui-même synchronisé avec un troisième agent C, l'agent A et l'agent C se retrouve synchronisés par transitivité et cette synchronisation peut être assurée par une propagation par blocs (l'agent B servant d'intermédiaire). L'agent A

la matrice semble être celle qui fournit les meilleurs résultats.

travaille sur les blocs A_1, A_2 et B_1 (équation IV.1). Lorsque l'agent A a propagé ses contraintes entre plans dans le bloc AB, l'agent B récupère la mise à jour de B_1 et propage ces modifications dans la matrice composée des blocs B_1 (mis à jour), B_2, B_3 et C_1 (équation IV.2). La modification est alors transmise à l'agent C qui récupère la mise à jour de son bloc C_1 pour le propager dans sa propre matrice (équation IV.3).

$$\begin{pmatrix} A_1 & A_{1,2} & & & & & (AC) \\ -{A_{1,2}}^t & A_2 & (AB) & & & & \\ & -(AB)^t & B_1 & B_{1,2} & B_{1,3} & & \\ & & -{B_{1,2}}^t & B_2 & B_{2,3} & & \\ & & -{B_{1,3}}^t & -{B_{2,3}}^t & B_3 & (BC) & \\ & & & & -(BC)^t & C_1 & C_{1,2} \\ -(AC)^t & & & & & -{C_{1,2}}^t & C_2 \end{pmatrix} \quad \text{(IV.4)}$$

Dans un second temps (équation (IV.4)), les deux agents A et C doivent se synchroniser explicitement via les nouvelles contraintes contenues dans (AC). Nous sommes alors contraints (par l'apparition d'un cycle) de synchroniser globalement les trois agents ensemble en considérant la matrice globale.

IV.2.2 La localité des plans

Lors d'une union de plans ou d'une insertion de buts, les seuls prédicats qu'un agent doit synchroniser entre son propre plan (ou ses propres tâches dans le cas de l'insertion de buts) et le plan global sont les prédicats portant sur des attributs ou des ressources communes. Il est donc possible, lors d'une union de plans, de se limiter à ces seuls prédicats (et aux instants qui s'y rapportent) pour limiter tant la quantité d'information à traiter que le nombre d'instants à gérer. Les propriétés de la propagation des contraintes temporelles par blocs nous garantissent la complétude de cette union.

On peut donc imaginer un système de planification distribuée utilisant cette technique pour éviter que chaque agent qui veut ajouter un plan au plan global soit obligé d'utiliser toute l'information contenue dans le plan global. Ce dernier reste distribué entre les agents et n'existe donc pas en tant que tel.

Par contre, nous n'aboutirons pas à une mise à jour en parallèle puisque deux agents qui planifient simultanément risquent d'imposer séparément des contraintes qui pour

chacun semblent cohérentes mais qui, une fois propagées, ne le sont plus. Chaque action de planification, même si elle n'agit pas directement sur l'ensemble de plan global, doit encore s'effectuer en exclusion mutuelle (de tous les agents ayant un plan synchronisé avec une partie de son plan directement ou par transitivité).

Tout cela nous montre que, bien qu'il soit possible d'éviter une centralisation de l'information sur un agent à chaque modification du plan global, cela se fait au détriment des performances et sans amener de gain réel en terme de quantité d'informations échangées (le gain obtenu par la non transmission du plan global ne compensant pas la perte engendrée par le nombre de mises à jour des contraintes temporelles qu'implique la propagation par blocs). En envisageant une hiérarchisation temporelle (c.-à-d. un regroupement les instants par groupes sur lesquels porteraient les contraintes temporelles), on pourrait peut-être amélioré les performances mais c'est une piste que nous n'avons pas explorée.

Nous ne pouvons donc pas atteindre le second niveau de distributivité décrit au début de cette section sans perdre en performance par rapport au premier niveau. Le problème de la planification en présence de contraintes temporelles numériques n'est donc pas facilement distribuable.

IV.3 Les problèmes liés à l'exécution

Il faut maintenant envisager l'exécution des plans. Il est évident que les spécifications d'un système de planification sont intimement liées à la manière dont les plans produits vont être exécutés. D'autre part, dès lors que l'on parle d'exécution, on se retrouve obligé d'envisager les cas d'échec et donc les répercussions que cela peut avoir sur le système de planification. S'il est possible dans un premier temps d'ignorer tous ces problèmes dans le cadre de la planification mono-agent, le fait de planifier à plusieurs ou même de planifier incrémentalement nous contraint à nous intéresser aux interactions entre le planificateur et le système qui exécute les plans.

Si l'on reprend le modèle classique qui représente un agent agissant sur le monde par une boucle perception – décision – action, nous voyons que nous ne pouvons pas nous abstraire d'étudier de près les interactions entre le système de planification (qui fait évidemment partie de l'aspect décision) et le reste du système.

Nous allons tout d'abord présenter une architecture d'un système de contrôle d'un

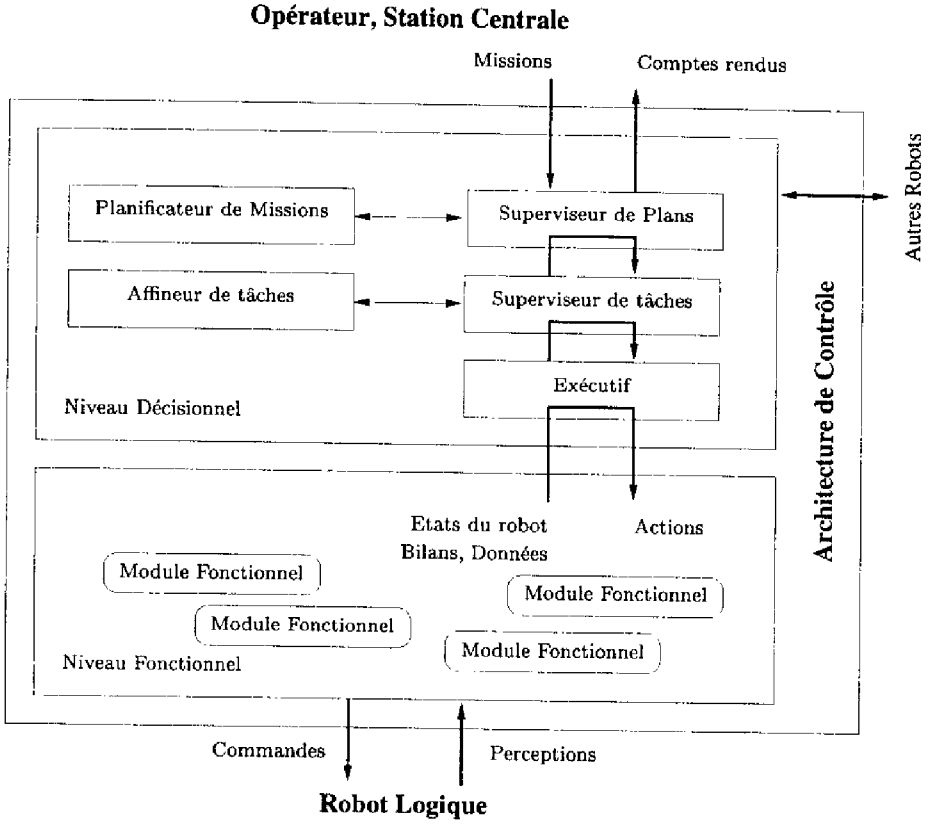


FIG. IV.3 – *Architecture de contrôle d'un robot mobile autonome*

robot mobile autonome puis nous étudierons les interactions entre planification et exécution.

IV.3.1 L'architecture du système de contrôle

La figure IV.3 présente l'architecture du système de contrôle d'un robot mobile autonome.

La spécification et la mise en œuvre de cette architecture auxquelles nous avons participées [Chatila 91] ont été, entre autres, étudiées tout d'abord par [Perebaskine 92] puis complètement développées et implémentées par [Fleury 96].

Cette architecture interagit au plus bas niveau avec le robot logique en lui envoyant des commandes et en récupérant les données de perception. Le robot logique permet de s'abstraire des contraintes spécifiques à chaque type d'actionneurs et de capteurs. La couche *robot logique* est spécifique à chaque type de système mais présente à l'architecture de contrôle une interface similaire quel que soit le robot sur lequel elle est installée.

À l'opposé, le système de contrôle communique avec l'opérateur (dans le cas d'un robot isolé) ou avec une station centrale (dans le cas multi-robots). Il reçoit donc des missions qu'il doit exécuter et renvoie des compte-rendus d'exécution.

Le système de contrôle se décompose en interne en deux parties bien distinctes:

- un niveau décisionnel « temps réel » chargé d'élaborer les plans correspondant aux missions puis d'en superviser l'exécution et d'assurer la reprise en cas d'erreur. C'est aussi à ce niveau que, dans un contexte multi-robots, sont gérées les interactions entre les robots.
- un niveau fonctionnel constitué d'un ensemble de modules. Chaque module regroupe un sous-ensemble de fonctions élémentaires du robot. Ce sont ces modules qui assurent, par exemple, les asservissements, le pilotage du robot, les actions de perceptions ou les traitements de bas niveau.

Dans le cadre du présent document, seul le niveau décisionnel nous intéresse. Ce niveau est organisé en empilant plusieurs couples (deux dans le cas de notre figure) planificateur/superviseur. Chaque couple assure la planification et l'exécution d'un plan à son propre niveau d'abstraction. Dans l'exemple de la figure IV.3 page précédente le premier couple (planificateur – superviseur de plans) transforme une mission en un plan et contrôle le bon déroulement de ce plan. Le deuxième couple (affineur de tâches – superviseur de tâches) transforme les tâches d'un plan en actions élémentaires et en vérifie la bonne exécution.

L'exécutif transforme les demandes d'exécution d'actions élémentaires en requêtes directement transmises aux modules fonctionnels et analyse l'ensemble des bilans et comptes-rendus d'exécution pour en faire une synthèse en terme d'événements qui remonte vers le superviseur.

L'aspect *temps réel* au niveau décisionnel est un facteur important du bon fonctionnement de cette architecture. Le temps de prise en compte d'un événement provenant

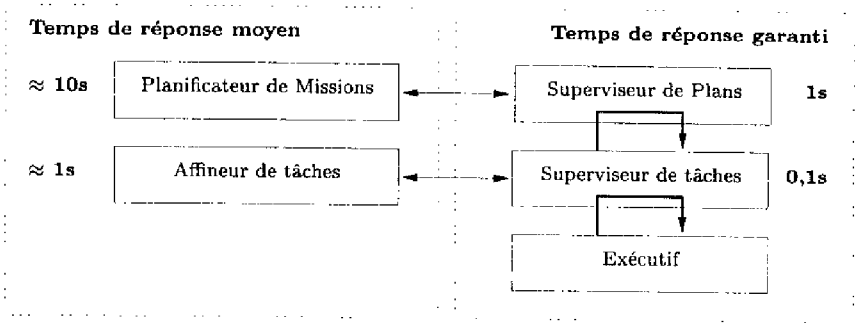


FIG. IV.4 – *Aspect temps réel du niveau décisionnel*

du niveau fonctionnel doit être borné, connu et compatible avec l'application afin de garantir une bonne réactivité du robot à son environnement (dans certaines application, c'est la survie même de l'engin qui est en jeu). Or, il est rarement possible de garantir un temps de réponse lors d'une planification ou de l'affinement d'un plan. C'est cela qui justifie la mise en œuvre d'un couple planificateur/superviseur à chaque couche du niveau décisionnel.

Le superviseur fonctionne avec un temps de réponse garanti et compatible avec le niveau d'abstraction auquel il est lié ce qui permet au planificateur de travailler sans contrainte de temps réel. La figure IV.4 montre les temps de réponse attendus dans le cas d'un robot mobile autonome.

Cette architecture a été spécifiée pour un robot mobile autonome mais la pertinence de l'approche utilisée pour structurer le niveau décisionnel reste tout aussi valable si l'on envisage un agent au sens large indépendamment de son mode d'action réel sur le monde.

IV.3.2 La supervision de plans

I_XT_ET n'intègre pas seulement un système de planification. Il contient aussi un système de suivi de chroniques que nous connaissons bien puisque nous avons participé à ses spécifications et à son développement. Ce système analyse un flot d'événements décrivant les changements du monde pour y reconnaître certaines situations ou chroniques particulières. Cette reconnaissance permet de déclencher des actions en retour. Chaque

modèle de chroniques est décrit par un ensemble d'événements génériques (paramétrés par des variables) contraints temporellement entre eux.

La toute première version de ce système a été spécifiée par A. Aloui et se basait sur des contraintes temporelles purement symboliques [Alaoui 90]. L'intégration de cette première version que nous avons réalisée dans le cadre du projet européen SKIDS [Ghallab 91], nous amena à suggérer plusieurs améliorations. Avec C. Dousson, nous avons alors formalisé et implémenté une nouvelle version qui répond aux spécifications suivantes [Dousson 93] :

- les contraintes temporelles sont numériques et les attributs sont paramètrables par des variables sur lesquelles on peut poser des contraintes;
- le formalisme logique est celui que nous avons décrit et qui est utilisé dans le système de planification (basé sur les prédicats **hold** et **event**), les modèles de chroniques sont décrits dans un langage identique à celui de description des modèles de tâches ;
- les délais entre l'occurrence des événements et leur reconnaissance par le système (dûs principalement aux temps de traitement variables des différentes méthodes de perception ou d'interprétation) sont pris en compte explicitement ;
- le système est capable d'extraire, en cours de reconnaissance, l'ensemble des événements attendus afin de déclencher automatiquement des actions de focalisation du système de perception ;
- la reconnaissance des instances de modèles de chroniques est faite au plus tôt ;
- pour un jeu de modèles de chroniques donné, on peut calculer un temps de réponse maximum du système.

C. Dousson a continué à développer ce système de suivi d'évolutions, en particulier dans le cadre du projet européen TIGER, en y ajoutant, par exemple, la possibilité de traduire sous forme d'événements l'observation de phénomènes continus et en étudiant la possibilité de générer automatiquement les modèles de chroniques par apprentissage [Marois 95]. L'ensemble de ces travaux est présenté dans sa thèse [Dousson 94].

La figure IV.5 page suivante présente un graphe de contraintes temporelles entre les trois événements d'une chronique fictive (la contrainte en pointillé représente le résultat de la propagation des deux autres contraintes). Au fur et à mesure que les événements ont lieu, leurs dates d'occurrence sont intégrées dans le système et, par mise à jour du

FIG. IV.5 – *Modèle de chronique : contraintes temporelles*

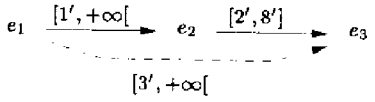
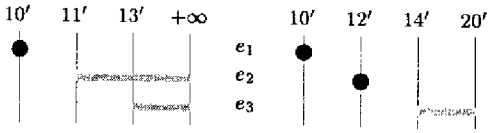


FIG. IV.6 – *Modèle de chronique : dates possibles d'occurrence (avant et après réception de e_2)*



graphe, on peut en déduire l'intervalle temporel possible d'occurrence des événements à venir. La figure IV.6 présente ces intervalles alors que l'événement e_1 a eu lieu à la date 10′ puis après intégration du fait que l'événement e_2 a eu lieu à la date 12′.

Il est possible d'utiliser ce système pour faire du suivi d'exécution. Le formalisme logique étant le même, on peut transformer automatiquement un plan produit par le système de planification en une chronique utilisable par le système de reconnaissance de chroniques. La reconnaissance de cette chronique permettra alors de valider la bonne exécution du plan. Par contre il faut aussi décrire des chroniques permettant d'identifier les situations non nominales afin de détecter au plus tôt l'échec éventuel du plan. Cette seconde partie est loin d'être triviale et ne peut pas découler directement d'une simple analyse du plan à réaliser. Il semble nécessaire d'enrichir le langage de description de tâches afin que le programmeur puisse spécifier lui-même les conditions de détection des anomalies. Des travaux, actuellement en cours d'études au LAAS, devraient bientôt faire l'objet d'une thèse à ce sujet.

Dans tous les cas, le système de suivi d'exécution intégré à un superviseur doit pouvoir fournir, à chaque instant, l'état du monde et du système (au niveau d'abstraction qui est le sien) ainsi que l'avancement du plan courant en terme de dates possibles d'occurrence des événements à venir ou non encore perçus.

IV.3.3 Liens entre exécution et planification

Le superviseur reçoit les missions (ou buts) envoyées par l'opérateur ou une station centrale. Il envoie alors au planificateur outre ces buts, la description actuelle de l'état du monde (et du système) ainsi que le plan courant. Nous avons expliqué comment

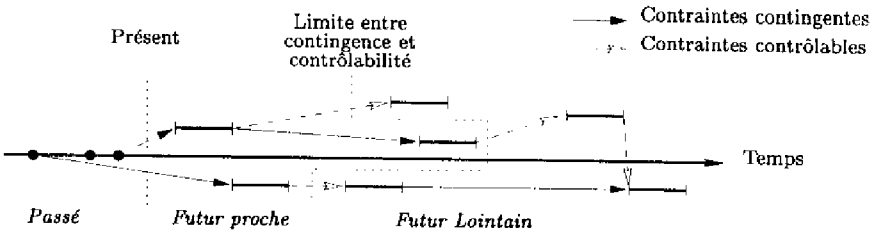


FIG. IV.7 – *Les trois phases d'un plan en cours d'exécution*

le planificateur pouvait alors élaborer incrémentalement un nouveau plan intégrant la mission reçue (§ IV.1 page 67). Mais il se pose un problème de temps réel lié au fait que durant la phase de planification, l'exécution du plan courant se poursuit. Or, lors de la mise à jour incrémentale du plan, il se peut que l'ajout de contraintes temporelles modifie les conditions d'exécution du plan actuel.

F.Robert propose, dans sa thèse [Robert 96], une solution très élégante dans le cadre de la planification de déplacement multi-robots (projet européen MARTHA). Il s'abstrait du problème de sur-contrainte en forçant l'insertion des tâches d'un nouveau plan *après* celles du plan global existant (ou éventuellement en parallèle lorsqu'il n'y pas d'interactions directes). Cet ajout se faisant sur un horizon temporel court, il obtient malgré tout un entrelacement raisonnable des tâches. Mais cette approche ne se généralise que dans le cadre d'application où chaque interaction entre les agents est limitée dans le temps (dans le cas du déplacement multi-robots, elles se limitent à la durée de traversée d'un carrefour ou d'une aire) et dans la mesure où on ne tient compte ni de contraintes temporelles numériques ni de contraintes de précédences sur des événements attendus ou à dates fixes. Par contre, le mécanisme de délibération qu'il propose reste valide indépendamment de ces limitations.

Dans le cadre général d'une planification multi-agents, nous aimerions donc nous autoriser à surcontraindre le plan global préexistant. En reprenant notre exemple des deux robots qui veulent passer une porte, on aimerait autoriser R2, lorsqu'il planifie par insertion de son but dans le plan de R1, à contraindre R1 à ne refermer la porte qu'après le passage des deux robots.

Pour cela, il nous faut distinguer trois phases dans un plan en cours d'exécution

(illustrées par la figure IV.7 page précédente) :

1. la phase « *passé* » qui contient en quelque sorte le compte-rendu d'exécution des tâches déjà réalisées et dans laquelle tous les instants sont instanciés à des dates fixes.
2. la phase « *présent et futur proche* » qui contient l'ensemble des tâches en cours d'exécution c.-à-d. celles dont certains instants sont contraints de manière contingente par rapport à des instants déjà instanciés. Ces tâches constituent en fait la partie non modifiable du plan en cours.
3. la phase « *futur lointain* » qui contient l'ensemble des tâches pour lesquelles il est encore possible de surcontraindre temporellement tous les instants qu'elles contiennent sans perturber l'exécution en cours.

Chaque agent, lorsqu'il envoie son propre plan à un agent qui veut planifier, spécifie alors (pour chaque attribut ou ressource) quelle est la limite entre son futur proche et son futur lointain sachant qu'il lui est interdit d'exécuter une action au-delà de cette limite.

La planification puis l'exécution de nouveaux buts s'effectuent alors en deux temps :

1. l'agent récupère tout d'abord le plan global dans lequel il va planifier pour ces nouveaux buts (par union de plans, par insertion de buts ou même par une planification globale de l'ensembles des buts). Il ne peut que modifier (dans le sens surcontraindre) la partie du plan global qui est dans la phase « *futur lointain* » ;
2. une fois effectuée cette modification du « *futur lointain* », il devra alors valider cette phase en déplaçant peu à peu vers le futur la limite qui sépare le « *futur proche* » du « *futur lointain* ».

On peut voir le premier temps comme une phase de négociation puisque, à ce stade, les tâches qui ont été planifiées par un agent peuvent encore être sur-contraintes ou même remises en cause par un autre agent. Le second temps valide définitivement le plan choisi qui sera exécuté tel quel : les seules modifications temporelles de cette partie du plan ne proviendront que du superviseur de plan qui instanciera peu à peu à des dates réelles les instants présents dans les tâches.

I_XT_ET, en l'état actuel, ne peut pas gérer ces différentes phases car son système de

gestion des contraintes temporelles n'intègre pas les contraintes contingentes que nous allons présenter tout de suite.

IV.3.4 Contrôlabilité ou contingence des contraintes temporelles

Jusqu'à présent, nous n'avions pas parlé de contraintes contingentes. Ce type de contraintes temporelles exprime le fait que le planificateur ne peut pas contrôler l'intervalle de temps qui sépare deux instants. À l'inverse, toutes les contraintes que nous exprimions jusqu'à maintenant étaient des contraintes contrôlables. Il est évident que dans de nombreux cas, les événements et assertions qui décrivent une tâche sont contraints entre eux de manière contingente. Par exemple, dans le cas d'une tâche qui permet de fermer une porte, il est pratiquement impossible de contrôler la durée exacte qui sépare le déclenchement de l'action de l'événement réel de fermeture de la porte. La contrainte entre ces deux instants est donc une contrainte temporelle contingente. D'autre part, les synchronisations entre les différents agents ne peuvent pas s'effectuer entre deux instants dont les dates d'occurrence sont incontrôlables. Là encore, on voit apparaître cette notion fondamentale de contingence temporelle.

Dans le cas où la contrainte temporelle entre deux instants impose une durée fixe, la notion de contingence ou de contrôlabilité n'a plus de sens. C'est par cet artifice que tous les exemples de planification qui ont été traités jusqu'à présent par I_XT_ET simulaient la notion de contingence temporelle (par exemple en fixant la date de fermeture d'une porte non pas à sa date réelle mais à la date de fin de la tâche de fermeture). Au niveau d'abstraction auquel nous travaillions, cette approximation n'avait que peu d'importance.

Mais, dans le cas de la planification incrémentale ou distribuée, nous venons de voir que la contingence ne provient plus de la description des tâches mais de l'exécution elle-même. Cette méthode empirique de prise en compte de la contingence temporelle ne fonctionne plus puisque la durée exacte des contraintes contingentes est fixée par le superviseur au fur et à mesure de l'exécution et ne peut donc pas être choisi de manière arbitraire. Pour assurer la phase de négociation, il faut pouvoir intégrer dans le planificateur la gestion explicite des deux types de contraintes : contrôlables et contingentes.

L'intégration de la contingence dans le gestionnaire de contraintes temporelles d'I_XT_ET a été étudiée par T.Vidal [Vidal 95] [Vidal 96]. Il propose une méthode qui

permet une prise en compte de la contingence dans un réseau de contraintes temporelles par des algorithmes polynômiaux. Malheureusement, pour obtenir ces résultats, il pose un certain nombre d'hypothèses restrictives sur le type de contraintes que l'on peut traiter, la plus forte étant l'indépendance au sens probabiliste des contraintes contingentes (en effet ce type de contraintes est formalisé par des variables aléatoires). Or, dans le cadre de la planification distribuée ou incrémentale, les synchronisations entre les agents impliquent nécessairement des interdépendances entre les durées des contraintes contingentes. D'autres travaux sur les CSP mixtes semblent pouvoir amener des résultats intéressants dans ce domaine [Fargier 95] mais, à notre connaissance, ils ne sont pas parvenus à une maîtrise algorithmique suffisante pour permettre leur utilisation pratique dans le cadre d'un planificateur.

Supposons que notre système sache gérer ce type de contraintes, on peut alors attendre de sa part deux types de solution :

1. Il existe un plan valide quelles que soit les durées réelles prises par les contraintes contingentes.
2. Pour chaque combinaison de durées réelles prises par les contraintes contingentes, il existe un plan valide.

Dans le premier cas, nous sommes sûrs du plan produit mais notre système planifie dans un contexte très contraint. Il doit considérer chaque contrainte contingente comme une variable aléatoire et valider son plan dans tous les cas possibles. Le plus souvent, il n'existera pas de plan unique valide dans tous les cas de figures.

Dans le second cas, le planificateur doit être capable de produire un plan différent pour chacune des différentes combinaisons des contraintes contingentes. On se rapproche d'une forme de planification conditionnelle. En supposant même que l'on sache produire ces plans, encore faut-il pouvoir effectuer les choix au fur et à mesure de l'exécution. En effet, le choix du plan à exécuter doit pouvoir se faire sur des événements déjà observés. On voit apparaître ici, pour la première fois, une dissymétrie entre passé et futur que le planificateur doit prendre en compte lors de l'élaboration des différents plans.

IV.3.5 Synchronisation des plans

Une autre contrainte imposée par le lien entre exécution et planification concerne l'analyse du plan global produit pour en déduire les synchronisations à mettre en œuvre. En effet, une fois effectuée l'intégration d'un nouveau plan au plan global, il faut traduire les contraintes temporelles entre les instants des plans des différents agents en terme de synchronisation. Pour garantir la sûreté du fonctionnement d'une application multi-agents, il n'est pas possible de s'appuyer simplement sur l'horloge interne de chaque agent (il est très coûteux de garantir qu'elles soient synchronisées et de plus certains déclenchements d'actions d'un agent sont conditionnés par des événements provenant d'un autre agent dont on ne connaît pas la date d'occurrence a priori). Les synchronisations doivent se faire sur la base d'échanges de messages entre agents. Il faut aussi garantir que les synchronisations planifiées sont réalisables au sens où elles ne contraignent pas deux instants dont on ne peut pas contrôler l'occurrence.

En tenant compte de l'exécution, cette garantie de contrôlabilité du plan est une notion liée directement aux contraintes temporelles contingentes [Vidal 95] et pour laquelle nous n'avons pas encore de méthode de gestion.

Si l'on s'abstrait des problèmes liés à l'exécution, il faut encore analyser le graphe de contraintes temporelles produit pour le traduire en terme de messages de synchronisation. Pour cela, il nous faut trouver un réseau de contraintes temporelles cohérent compatible avec le réseau produit et tel que toutes les contraintes posées entre des instants des deux plans soient des contraintes de précédences portant sur des événements contrôlables. Mais il n'est pas sûr qu'une solution existe puisqu'encore une fois cela revient à contraindre un peu plus les instants. Pour le garantir, il faudrait que le gestionnaire de contraintes en tienne compte durant la phase de planification en reportant, par exemple, une contrainte sur un instant d'une tâche sur l'instant de début ou de fin de la tâche (selon le sens de la contrainte). Une piste d'étude consisterait à définir une typologie des instants permettant à l'utilisateur de préciser, dans la description des tâches, les instants dont l'occurrence est contrôlable.

On s'aperçoit donc que la gestion des contraintes dues à l'exécution et à la mise en œuvre de l'exécution doit être directement intégrée au gestionnaire de contraintes temporelles afin que le planificateur puisse en tenir compte. Cette prise en compte est encore un problème largement ouvert.

Chapitre V

Exemples de planification distribuée

Les exemples réunis ici illustrent l'utilisation des opérateurs entre plans partiels que nous avons développés.

Les tableaux de mesures indiquent le nombre de nœuds (le nombre de plans partiels intermédiaires) développés par le planificateur ainsi que le nombre de retours-arrière nécessaires à l'élaboration de chaque plan. Ces valeurs sont caractéristiques de chaque exemple et ne dépendent en rien du type de machine sur lequel le calcul est effectué. Par contre les temps d'exécution (exprimés en secondes) sont fortement liés au choix de la machine mais aussi à l'activité courante de celle-ci. Afin de pouvoir comparer ces mesures, toutes les mesures dans un même tableau ont été calculés sur une même machine (une SUN Ultra SPARC modèle 1 avec 1 processeur en général). Par contre, il ne faut pas comparer les temps entre plusieurs tableaux puisque ceux-ci n'ont pas obligatoirement été calculés dans les mêmes conditions.

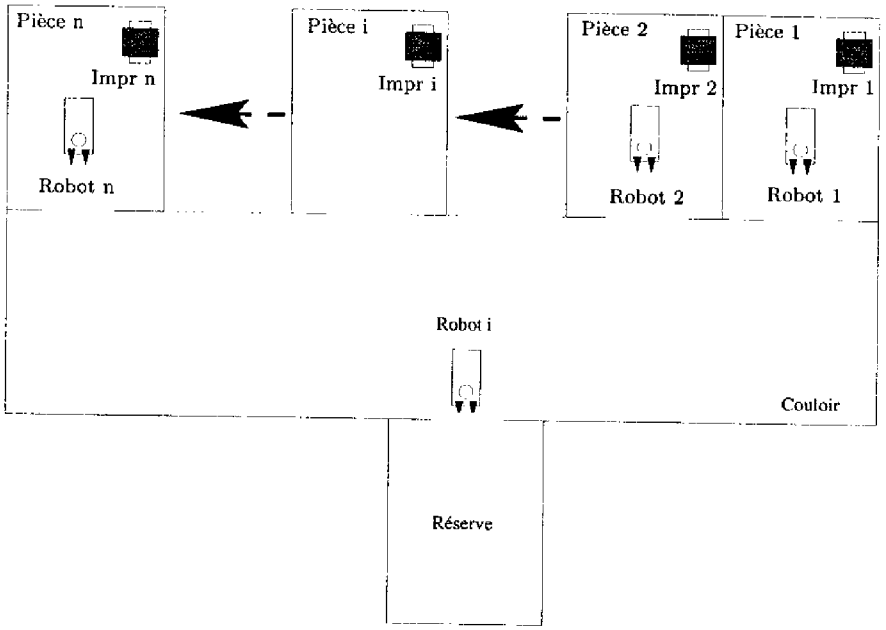


FIG. V.1 – *1er exemple : topologie des lieux pour n robots*

Nb de robots	Insertion des buts (en s)	Temps (en s)	Nb de nœuds	Retours-arrière	Temps cumulé (en s)
1 robot	—	0,2	17	1	0,2
2 robots	0,070	0,9	29	1	1,2
3 robots	0,110	2,4	41	1	3,7
4 robots	0,180	5,4	53	1	9,3
5 robots	0,230	9,8	65	1	19,3
6 robots	0,340	16,5	77	1	36,1
7 robots	0,540	25,9	89	1	62,6
8 robots	0,590	39,1	100	1	102,3
9 robots	0,740	56,9	111	1	159,9
10 robots	0,950	85	123	1	245,9

FIG. V.2 – *1er exemple : mesures de performance de l'insertion de buts avec sélection des liens causaux à supprimer*

Nb de robots	Temps (en s)	Nb de nœuds	Retours-arrière
5 robots	16,5	121	1
7 robots	47,8	173	1
10 robots	121,6	249	1

FIG. V.3 – *1er exemple : mesures de performance de l'insertion de buts avec suppression de tous les liens causaux*

Calcul	Temps (en s)	Nb de nœuds	Retours-arrière
Plan du Robot 10	0,2	17	1
Union	8,1	30	0

FIG. V.4 – *1er exemple : mesures des performance de l'union de plans pour le dixième robot*

V.1 1er exemple : Un cas d'école

Voici un exemple conçu uniquement pour comparer les performances de l'union de plans, de l'insertion de buts et de la planification globale (en pratique, ce type de problèmes se traite d'ailleurs beaucoup plus efficacement par une approche où la coopération a lieu au moment de l'exécution plutôt que durant la phase de planification [Robert 96]). Il s'agit de n robots qui ont pour mission d'alimenter leur imprimante respective en allant chercher du papier à la réserve (figure V.1 page précédente). La seule contrainte concerne le couloir. Il ne peut contenir qu'un seul robot. Ce problème peut être résolu à la fois par insertion de buts ou par union de plans.

Le tableau de la figure V.2 page ci-contre montre les performances obtenues en utilisant l'insertion de buts pour n robots (jusqu'à $n = 10$). Chaque opération d'insertion est découpée en deux phases. La première consiste à insérer les buts. La seconde rétablit la cohérence. Nous donnons les temps pour chacune des deux phases et le temps cumulé pour planifier l'ensemble du plan global. On peut constater que le temps d'exécution de la première phase ne prend que 1% du temps global.

Le tableau de la figure V.3 donne les performances de l'insertion de buts lorsqu'on supprime brutalement tous les liens causaux présents dans le plan. En comparant ces mesures à celles du tableau précédent, on mesure le gain qu'apporte une analyse des liens causaux à supprimer. Concernant le nombre de nœuds développés, il est pratiquement divisé par deux dans cet exemple. Le critère de filtrage est donc performant. Par contre

le gain en temps est nettement moins important (de l'ordre du tiers dans cet exemple). En effet, les nœuds qui ne sont plus explorés concernaient des défauts dont la résolvante était déjà imposée par le plan initial. Ce n'est malheureusement pas ce type de nœuds qui est très coûteux durant la phase de planification.

Le tableau de la figure V.4 page précédente indique le temps nécessaire au calcul du plan du dixième robot puis à l'union de ce plan avec l'union des 9 plans précédents. Le temps de calcul des 9 plans précédents est exactement identique. Le temps nécessaire à leurs réunions successives est pratiquement linéaire en fonction du nombre de plans déjà réunis.

Nous ne montrons pas le résultat de l'expérimentation de la planification globale car, au-delà de quatre ou cinq robots, le planificateur ne parvient plus à gérer l'explosion combinatoire en un temps raisonnable.

V.2 Intrépide: un exemple de planification distribuée

Dans cet exemple, nous avons considéré des robots personnels. Chacun des robots a son propre opérateur à qui il doit livrer le courrier. Le courrier arrive dans un bac placé dans la salle de réception du courrier. Ce bac est accessible par un seul robot à la fois. Le couloir reliant la salle de réception du courrier et le bureau de chaque opérateur ne peut accueillir que deux robots simultanément. Une prise unique est disponible dans ce couloir afin que les robots puissent y recharger leur batterie. La topologie des lieux est très similaire à celle de l'exemple précédent.

Le robot R1 (respectivement R2 et R3) qui est dans le bureau 1 (respectivement le bureau 2 et le bureau 3) a pour but d'aller chercher le courrier et de le ramener puis de ressortir dans le couloir. On demande en plus au robot R3 de terminer son plan dans le bureau 1.

Ce domaine de planification est décrit par 4 attributs:

- **EtatRobot** (*?robot*) précise l'état de chaque robot (En Charge, Oisif, En Mouvement, Actif) ;
- **PosCourrier** (*?personne*) donne la position du courrier de chacun des opérateurs ;
- **Position** (*?robot*) indique la pièce dans laquelle est chaque robot ;
- **ChargeRobot** (*?robot*) indique la charge de chaque robot ;

et par quatre ressources :

- **BacACourrier** de capacité 1 (il n'est utilisable que par un seul robot) ;
- **PriseCouloir** de capacité 1 (elle n'est utilisable que par un seul robot) ;
- **Couloir** de capacité 2 et traduite par deux ressources (Couloir et CouloirALiberer) produites et consommées en opposition ;
- **Batterie** de capacité 50 et traduite par deux ressources (Batterie et BatterieACharger) produites et consommées en opposition (cette ressource existe pour chacun des trois robots);

Chaque robot possède son propre jeu de tâches: deux tâches lui permettant de prendre et poser le courrier, deux tâches lui permettant d'entrer et de sortir d'un pièce (depuis ou vers le couloir) et une dernière tâche lui permettant de se recharger. Les figures V.10 page 98, V.11 page 98 et V.9 page 97 montrent le codage de la tâche de rechargement, de la tâche de prise du courrier et de la tâche de sortie d'une salle. Les deux autres tâches sont décrites de manière similaire.

Tout d'abord, demandons à I_XT_ET de planifier un plan pour chacun des robots indépendamment des autres. Les trois premières lignes du tableau de la figure V.6 page suivante présentent les mesures de l'exécution de ces trois planifications (aux valeurs des variables près, les plans de R1 et R2 sont identiques).

On voit que entre R1 et R2 d'une part et R3 d'autre part, il y a une grosse différence tant en durée qu'en nombre de nœuds explorés alors que, mis à part le fait de terminer dans le bureau 1 qui nécessite une recharge supplémentaire des batteries, le but de R3 est strictement identique à celui des deux autres robots. Le nombre de retours-arrière d'I_XT_ET pour satisfaire les deux buts simultanément expliquent cette différence.

Mais en fait les deux buts donnés à R3 (aller chercher le courrier et aller dans le bureau1) peuvent être planifier incrémentalement. Le tableau suivant donne le résultat pour la planification du premier but puis pour l'insertion du second.

	Plan 1	Insertion de But 2
Nombre de nœuds	55	16
Nombre de retours-arrière	2	0
Temps global (en secondes)	1.4	1.2

FIG. V.5 – *Planification incrémentale pour R3*

On voit très bien ici, le gain obtenu par la planification incrémentale.

La figure V.12 page 99 montre les trois plans ainsi obtenus [1].

Les mesures faites pour réaliser un plan global par union de plans sont réunies dans le tableau de la figure V.6. Celles utilisant l'insertion de buts sont présentées dans le tableau de la figure V.7 page suivante. Les mesures de temps entre ces deux tableaux sont homogènes.

Plans	Nb de nœuds	Retours-arrière	Temps (en s)
P_{R1}	57	2	1.4
P_{R2}	57	2	1.4
P_{R3}	101	7	6.3
$\bowtie(P_{R1}, P_{R2})$	14	0	0.5
$\bowtie(P_{R1}, P_{R3})$	19	0	0.8
$\bowtie(P_{R2}, P_{R1})$	14	0	0.5
$\bowtie(P_{R2}, P_{R3})$	19	0	0.8
$\bowtie(P_{R3}, P_{R1})$	19	0	0.8
$\bowtie(P_{R3}, P_{R2})$	19	0	0.8
$\bowtie((\bowtie(P_{R1}, P_{R2})), P_{R3})$	18	0	3.3
$\bowtie((\bowtie(P_{R1}, P_{R3})), P_{R2})$	18	0	2.9
$\bowtie((\bowtie(P_{R2}, P_{R1})), P_{R3})$	18	0	3.2
$\bowtie((\bowtie(P_{R2}, P_{R3})), P_{R1})$	18	0	2.9
$\bowtie((\bowtie(P_{R3}, P_{R1})), P_{R2})$	18	0	3.0
$\bowtie((\bowtie(P_{R3}, P_{R2})), P_{R1})$	18	0	2.9

FIG. V.6 – *Intrépide : Mesures de performances (calcul par union)*

On remarque (comme on devait s'y attendre) que l'insertion est plus coûteuse que l'union en nombre de nœuds développés mais surtout en temps de calcul (le calcul de chaque nœud est plus coûteux lors de l'insertion car le nombre de formules logiques est plus importants). La figure V.8 page ci-contre montre les valeurs cumulées pour élaborer un plan commun pour les trois robots. Par contre le plan global produit par insertion a une durée minimum inférieure à celui produit uniquement par union de plans. Ceci est dû au meilleur entrelacement que le planificateur a réussi à calculer. Mais ce résultat n'est qu'un effet de bord et ne peut être garanti dans le cas général d'autant qu'IxTeT, rappelons-le, ne cherche pas à minimiser la durée du plan.

1. Les instanciations des plans présentés ne sont pas celles au plus tôt afin de faire ressortir les contraintes entre tâches.

Plan	Nb de nœuds	Retours-arrière	Temps (en s)
P_{R1}	57	2	1.4
P_{R2}	57	2	1.4
P_{R3}	101	7	6.3
$P_{R1} \overleftarrow{T_{R2}} Buts_{R2}$	86	5	9.2
$P_{R1} \overleftarrow{T_{R3}} Buts_{R3}$	117	9	20.7
$P_{R2} \overleftarrow{T_{R1}} Buts_{R1}$	86	5	9.4
$P_{R2} \overleftarrow{T_{R3}} Buts_{R3}$	117	9	20
$P_{R3} \overleftarrow{T_{R1}} Buts_{R1}$	144	14	42.8
$P_{R3} \overleftarrow{T_{R2}} Buts_{R2}$	144	14	42.4
$(P_{R1} \overleftarrow{T_{R2}} Buts_{R2}) \overleftarrow{T_{R3}} Buts_{R3}$	159	14	77.3
$(P_{R1} \overleftarrow{T_{R3}} Buts_{R3}) \overleftarrow{T_{R2}} Buts_{R2}$	117	8	43.1
$(P_{R2} \overleftarrow{T_{R1}} Buts_{R1}) \overleftarrow{T_{R3}} Buts_{R3}$	159	14	76.7
$(P_{R2} \overleftarrow{T_{R3}} Buts_{R3}) \overleftarrow{T_{R1}} Buts_{R1}$	117	8	42.5
$(P_{R3} \overleftarrow{T_{R1}} Buts_{R1}) \overleftarrow{T_{R2}} Buts_{R2}$	167	14	89.2
$(P_{R3} \overleftarrow{T_{R2}} Buts_{R2}) \overleftarrow{T_{R1}} Buts_{R1}$	167	14	88.9

FIG. V.7 – *Intrépide : Mesures de performances (calcul par insertion)*

Plans	Nb de nœuds	Retours-arrière	Temps (en s)
$\bowtie((\bowtie(P_{R1}, P_{R2})), P_{R3})$	247	11	12.9
$(P_{R1} \overleftarrow{T_{R2}} Buts_{R2}) \overleftarrow{T_{R3}} Buts_{R3}$	302	21	87.9

FIG. V.8 – *Intrépide : union vs insertion (valeurs cumulées)*

L'union de deux plans de R1 et R2 (figure V.13 page 100) ne pose aucun problème. Les deux robots se synchronisent pour ne pas utiliser ensemble ni le bac à courrier ni la prise d'alimentation du couloir. L'intégration des deux plans en un seul ne demande que 0,05 secondes et le rétablissement de la cohérence 0,45 secondes (14 nœuds développés sans retour-arrière).

La figure V.14 page 101 contient l'union du plan de R3 avec l'union des deux précédents. L'ordre des tâches de R1 et R2 est déjà imposé par l'union de plans précédente. Dans ce plan apparaissent pour la première fois des contraintes temporelles de synchronisation qui garantissent que le couloir n'est occupé que par deux robots maximum. On constate, par exemple, que, pour effectuer sa dernière tâche, R2 est obligé d'attendre

l'entrée de R3 dans le bureau 1.

La figure V.15 page 102 montre que l'opération d'union de plans n'est pas associative. Ce plan est l'union du plan de R2 avec le plan résultat de l'union des plans de R1 et R3. Là encore, le planification a dû imposer de nouvelles contraintes temporelles pour que le nombre de robots dans le couloir ne dépasse pas la capacité admise. Mais, l'union n'ayant pas eu lieu dans le même ordre, nous n'obtenons évidemment pas le même plan que précédemment.

Les figures V.16 page 103 et V.17 page 104 montre les mêmes plans globaux mais calculés cette fois en utilisant l'insertion de buts.

En prenant la même description du monde, les mêmes jeux de tâches et les mêmes buts, nous avons tenté une planification globale. Là encore, nous avons été contraint d'abandonner alors que le planificateur développait son $10000^{\text{ème}}$ nœud au bout de 6 heures de calcul!

```
task R1SortDe(?de) (start, end) {

    variable ?porte ;
    ?de  in SALLES;
    ?porte  in PORTES;

    timepoint SortDeSalle;
    timepoint EntreDansCouloir;

    ?de  inBureau1 ⇒ ?porte  inPB1;
    ?de  inBureau2 ⇒ ?porte  inPB2;
    ?de  inBureau3 ⇒ ?porte  inPB3;
    ?de  inCourrier ⇒ ?porte  inPC;

    event (EtatRobot(R1): (Oisif, EnMouvement), start);
    hold (EtatRobot(R1): EnMouvement, (start, end));
    event (EtatRobot(R1): (EnMouvement, Oisif), end);

    hold (Position(R1): ?de  , (start , EntreDansCouloir));
    event (Position(R1): (?de, ?porte), EntreDansCouloir);
    hold (Position(R1): ?porte, (EntreDansCouloir, SortDeSalle));
    event (Position(R1): (?porte, Couloir), SortDeSalle);
    hold (Position(R1): Couloir, (SortDeSalle, end));

    consume (Couloir: 1, EntreDansCouloir);
    produce (CouloirALiberer: 1 , EntreDansCouloir);

    consume (BatterieR1: CONSO_DEPLACEMENT, end);
    produce (BatterieR1ACharger: CONSO_DEPLACEMENT, end);

    (end - start) in [DEPLAC_DUREE * 2, DEPLAC_DUREE * 2];
    (SortDeSalle - EntreDansCouloir) in [PASSAGE_PORTE, PASSAGE_PORTE];
}
```

FIG. V 9 – *Intrépide : tâche de sortie d'une pièce de R1*

```
task R1RechargeAlimentationCouloir() (start, end) {
    event (EtatRobot(R1): (Oisif, EnCharge), start);
    hold (EtatRobot(R1): EnCharge, (start, end));
    event (EtatRobot(R1): (EnCharge, Oisif), end);

    hold (Position(R1): Couloir, (start, end));

    event (AlimentationCouloir():(Libre, Utilisée), start);
    hold (AlimentationCouloir(): Utilisée, (start, end));
    event (AlimentationCouloir(): (Utilisée, Libre), end);

    produce (BatterieR1: PROD_RECHARGE, end);
    consume (BatterieR1ACharger: PROD_RECHARGE, end);

    (end - start) in [CHARGE_DUREE, CHARGE_DUREE];
}
```

FIG. V.10 – *Intrépide : tâche de rechargement de R1*

```
task R1PrendCourrie()(start, end) {
    timepoint PriseCourrier;

    hold (Position(R1): Courrier, (start, end));

    event (EtatRobot(R1): (Oisif, Actif), start);
    hold (EtatRobot(R1): Actif, (start, end));
    event (EtatRobot(R1): (Actif, Oisif), end);

    hold (ChargeRobot(R1): Aucune, (start, PriseCourrier));
    event (ChargeRobot(R1): (Aucune, CourrierPersonne1), PriseCourrier);
    hold (ChargeRobot(R1): CourrierPersonne1, (PriseCourrier, end));

    hold (PosCourrier(Personne1): Bac, (start, PriseCourrier));
    event (PosCourrier(Personne1): (Bac, R1), PriseCourrier);
    hold (PosCourrier(Personne1): R1, (PriseCourrier, end));

    consume (BatterieR1: CONSO_OP_COURRIER, end);
    produce (BatterieR1ACharger: CONSO_OP_COURRIER, end);

    use (BacACourrier:1, (start, end));

    (end - start) in [ACTION_DUREE, ACTION_DUREE];
}
```

FIG. V.11 – *Intrépide : tâche de prise du courrier par R1*

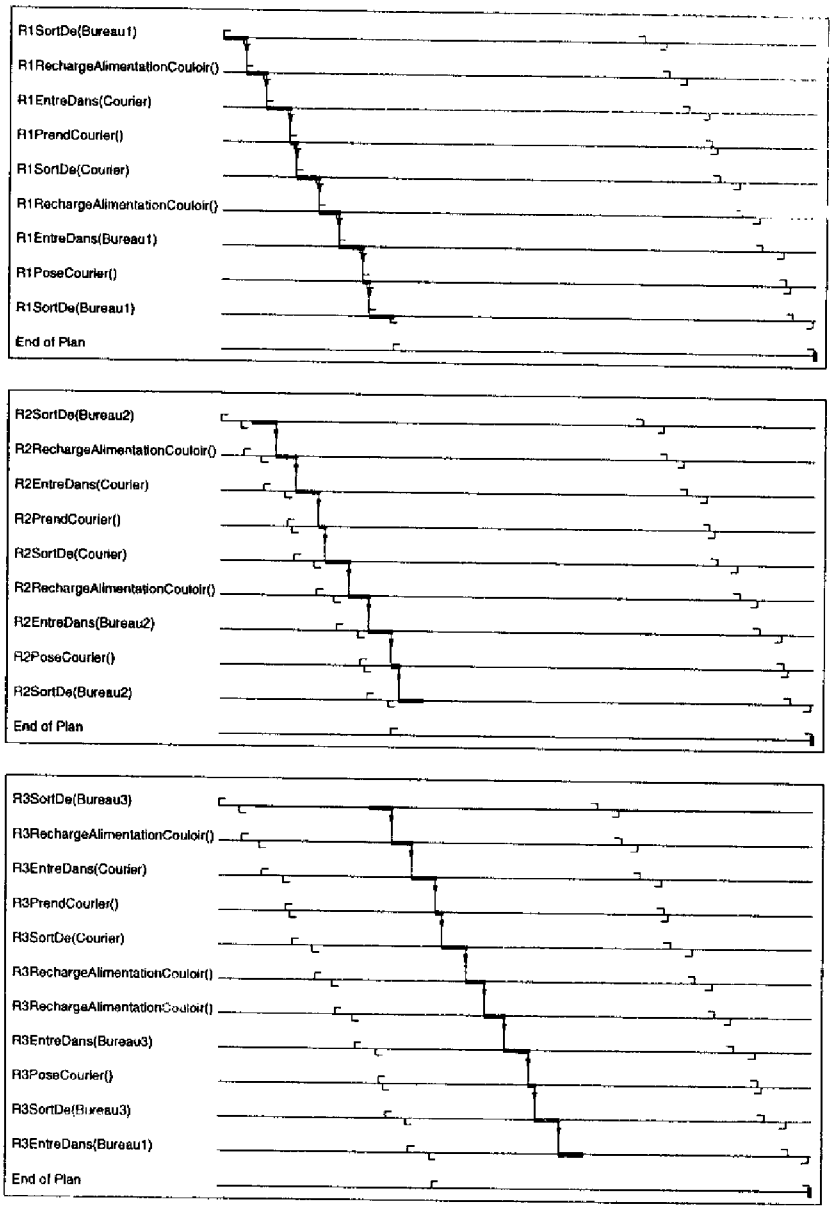


FIG. V.12 – *Intrépide: plans de R1, R2 et R3*

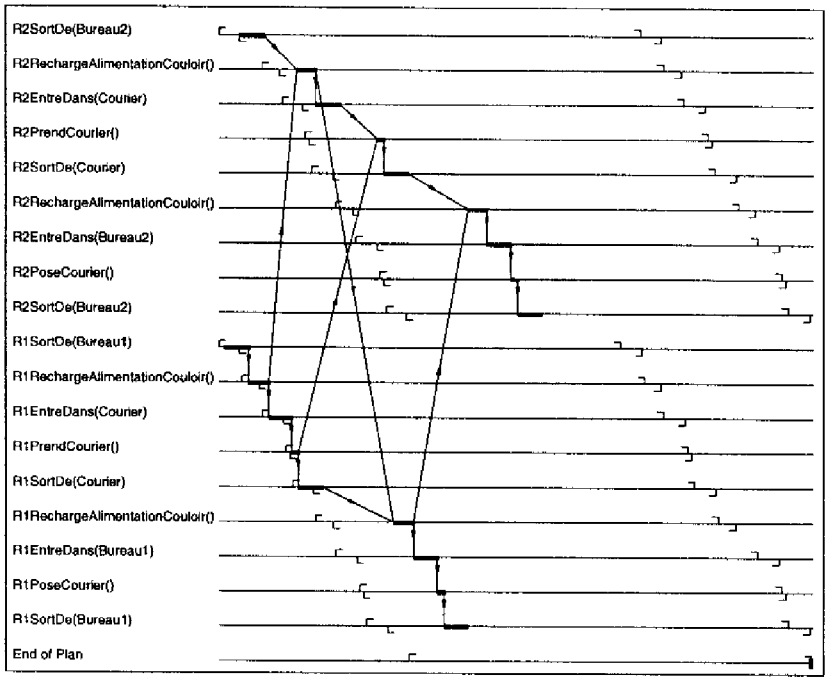


FIG. V.13 – *Intrépide: union des plans de R1 et R2*

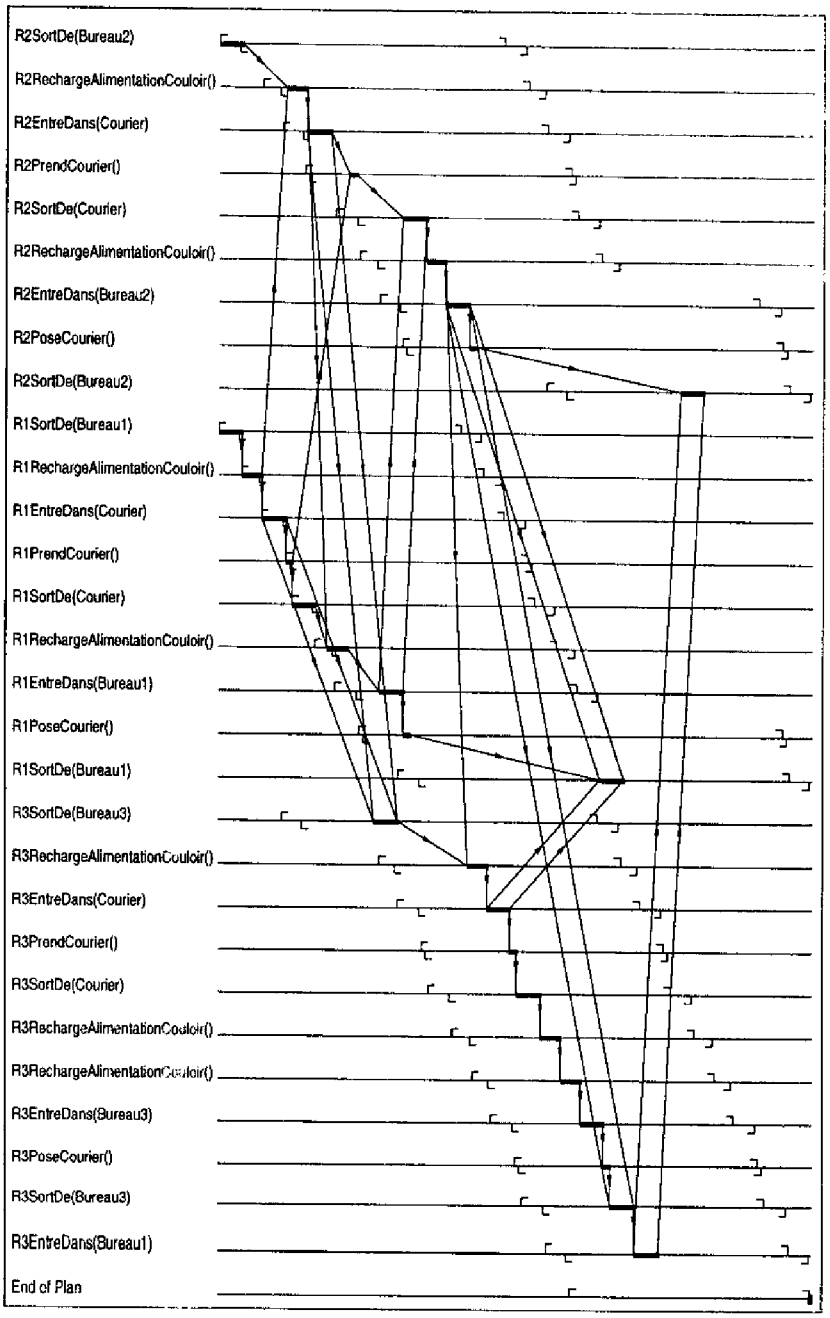


FIG. V.14 – *Intrépide: union du plan de R3 et de l'union des plans de R1 et R2*

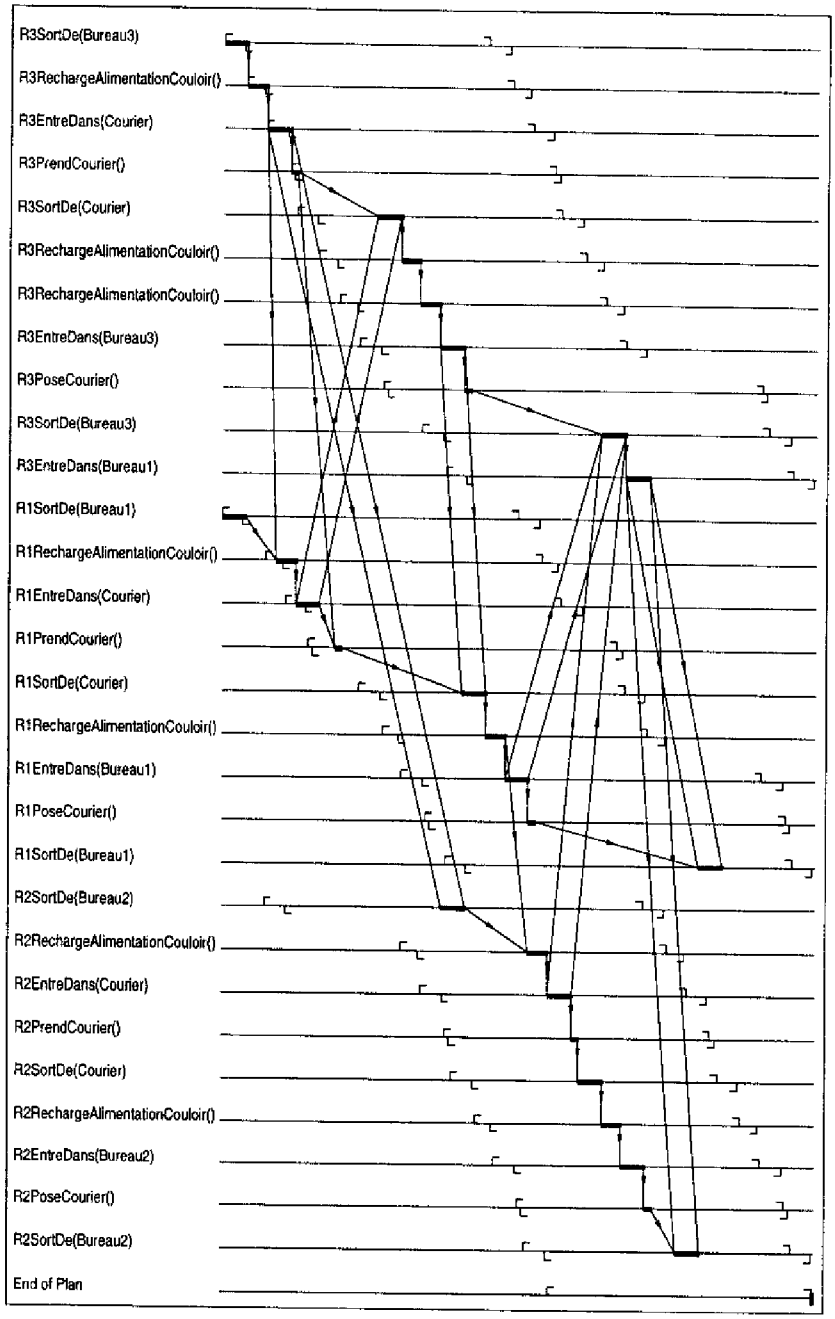


FIG. V.15 – *Intrépide: union du plan de R2 et de l'union des plans de R3 et R1*

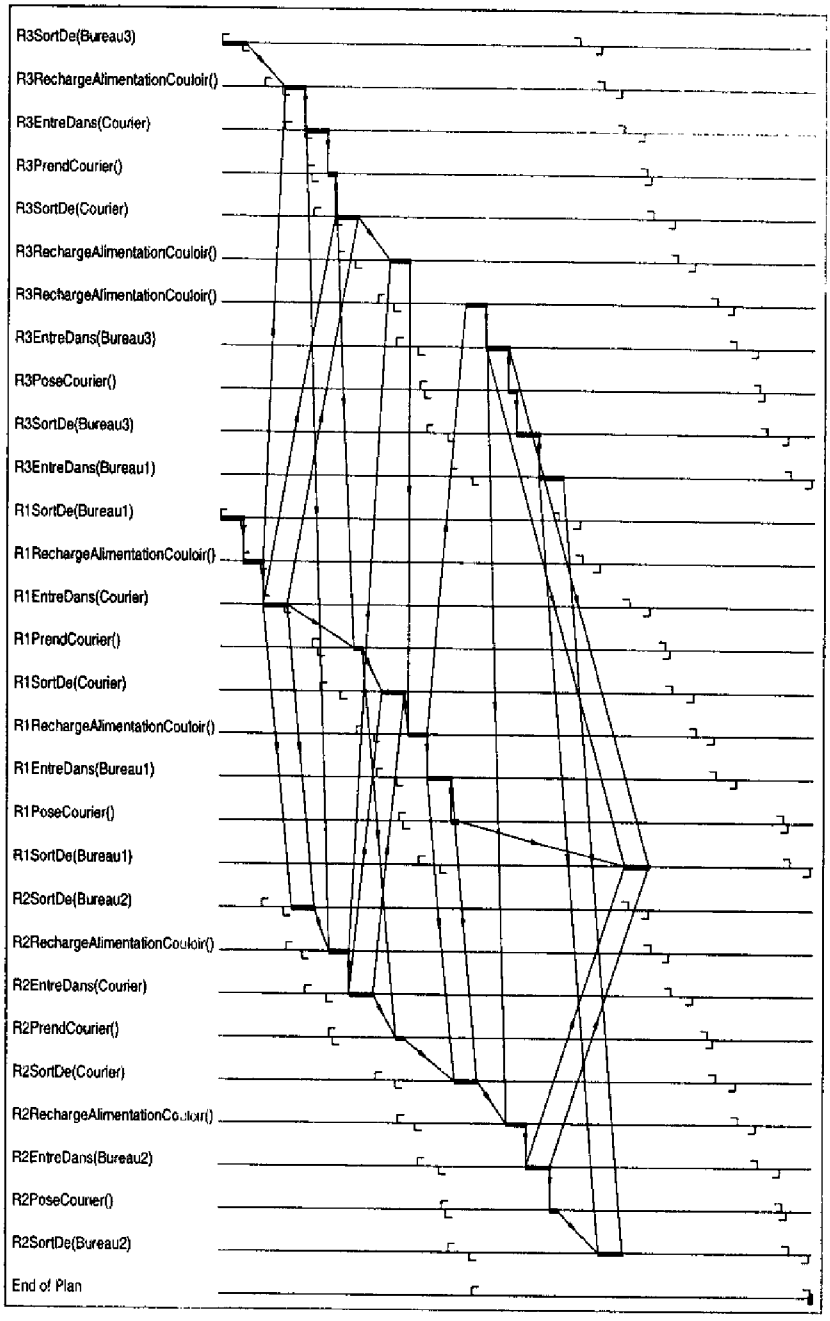


FIG. V.16 – *Intrépide: Insertion des buts de R3 dans le plan résultat de l'insertion des buts de R2 dans le plan de R1*

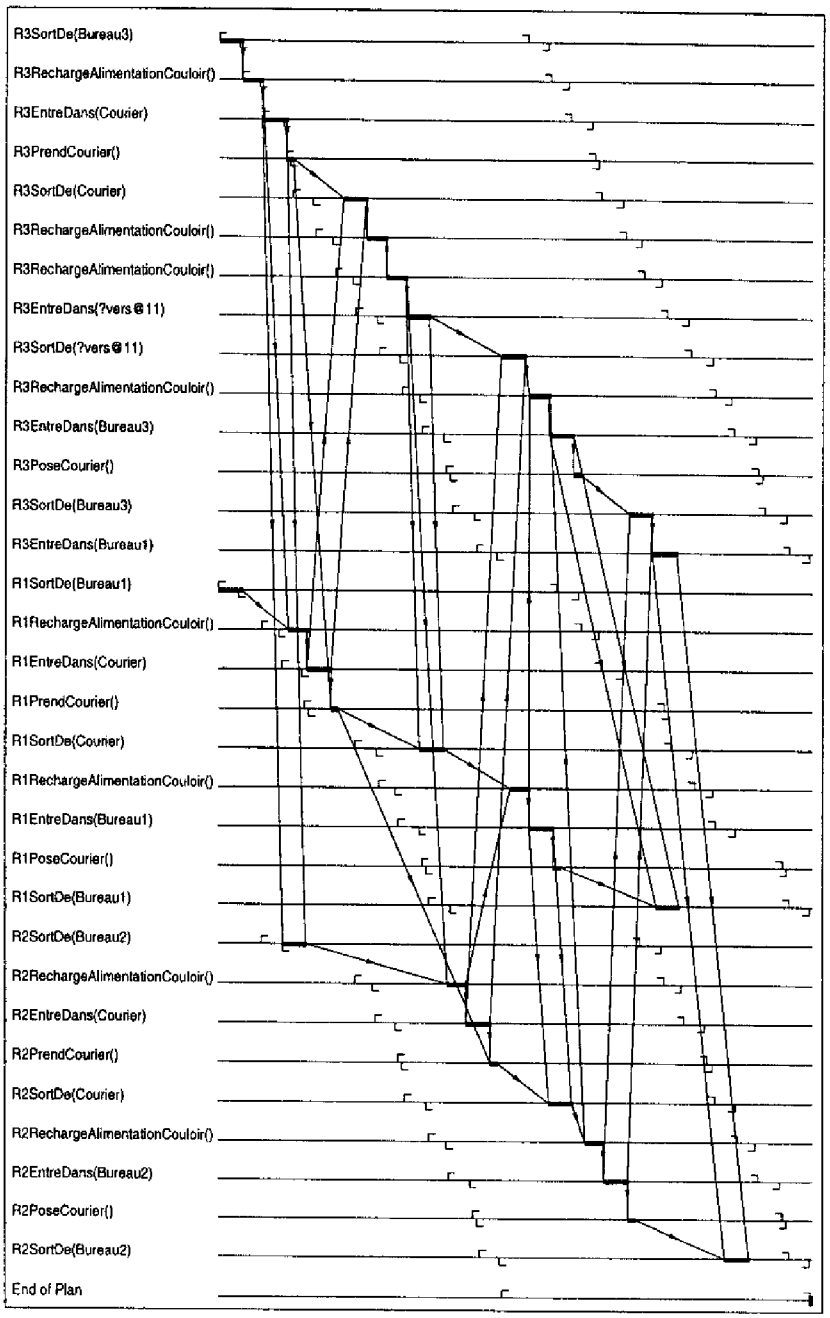


FIG. V.17 – *Intrépide: Insertion des buts de R2 dans le plan résultat de l'insertion des buts de R1 dans le plan de R3*

V.3 Comparaison entre O-Plan et IxTeT

Nous avons mené une comparaison entre O-Plan et IxTeT. O-Plan (Open Planning Architecture) est développé à l'AIAI (Artificial Intelligence Applications Institute - University of Edinburgh) depuis 1984. Depuis 1989, ce projet est en partie financé par l'armée américaine dans le cadre d'un programme de recherche ambitieux en planification. Un ensemble de références ainsi qu'une présentation du système sont disponibles sur Internet (URL : `http://www.aiai.ed.ac.uk/~oplan`).

O-Plan est un système de planification qui, comme IxTeT, intègre une représentation métrique du temps avec gestion des contraintes et gère les ressources partagées. Les tâches sont décrites par des pré-conditions et des post-conditions portant sur des attributs paramétrables. Leur éventuel parallélisme est pris en compte. Son mécanisme de recherche utilise une stratégie de moindre engagement et les plans produits le sont sous la forme d'un ordre partiel de tâches instanciées.

Par contre, alors qu'IxTeT procède par raffinement de plans partiels, O-Plan, lui, utilise une méthode de décomposition de tâches. Il prend en entrée un plan global contenant des tâches de haut niveau qu'il développe peu à peu en allant chercher leur description de plus bas niveau dans une bibliothèque de tâches (figure V.18 page suivante). Ce processus continue jusqu'à l'obtention d'un plan ne contenant que des tâches dites primitives. Le réseau de tâches ainsi obtenu constitue le plan solution. La notion de tâches diffère donc beaucoup entre les deux systèmes. De plus, IxTeT gère le non déterminisme de la recherche alors que l'utilisateur spécifie à O-Plan pratiquement toutes les tâches de l'arbre développé.

Il est intéressant de comparer ces deux systèmes bien sûr au niveau des plans résultats produits et des temps de calcul mais aussi au niveau du pouvoir descriptif des deux langages et vis-à-vis de la quantité d'informations fournie pour résoudre le problème.

Malheureusement nous ne disposons pas du système O-Plan. Nous avons donc choisi l'un des exemples disponibles pour le recoder avec IxTeT afin de mener nos comparaisons. Ce travail a été réalisé avec l'aide de A. Henriksson [Henriksson 96].

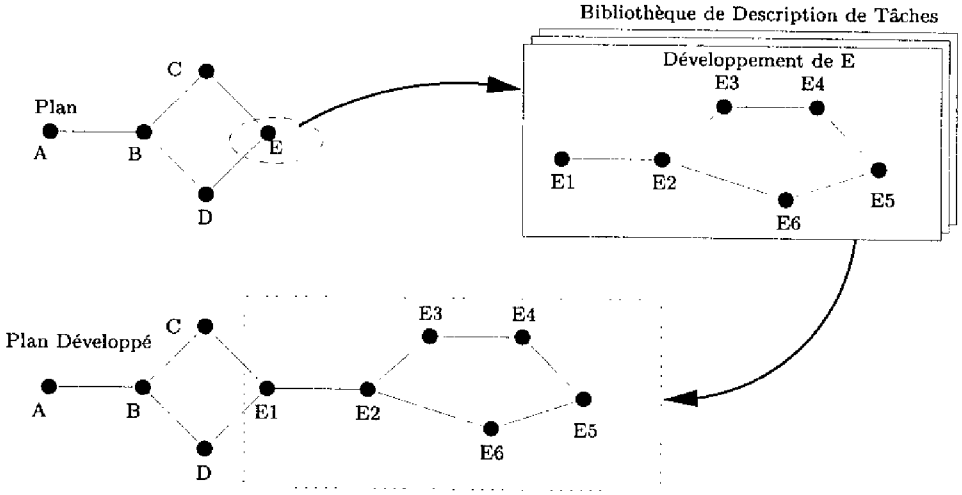


FIG. V.18 – *Développement d'une tâche par O-Plan*

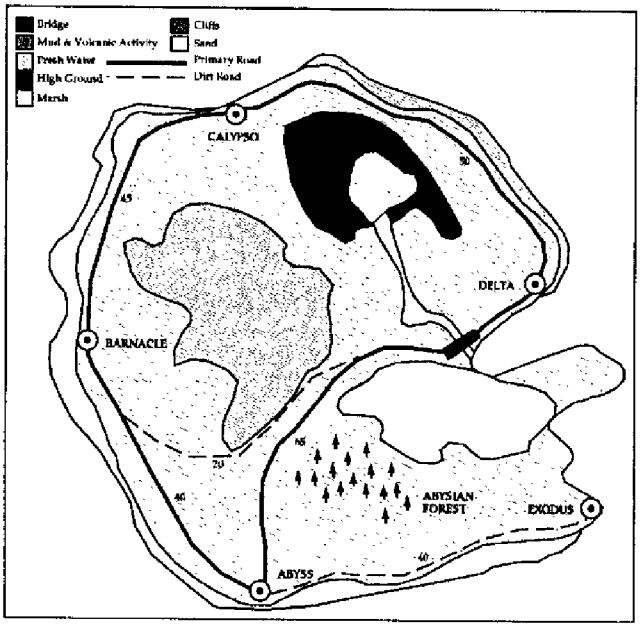


FIG. V.19 – *Pacifica : l'île de l'environnement PRECiS*

V.3.1 L'environnement PRECiS

L'environnement PRECiS [Reece 93] sert de base à un ensemble de problèmes de planification d'opérations d'évacuation de populations non combattantes d'une île appelée Pacifica (figure V.19 page précédente). Cette île fictive accueille plusieurs bases militaires. Le modèle décrit l'ensemble des avions, des bateaux et transports terrestres disponibles ainsi que leurs capacités propres en terme de temps d'autonomie, de charge maximum, de nombre de passagers, *etc.*.

Le but fixé consiste toujours à évacuer les personnes présentes dans chacune des bases. Mais en changeant les conditions initiales (la disponibilité de certains moyens de transports, leur vitesse de déplacement, leurs positions initiales ou le nombre de personnes à évacuer), le nombre de variantes est conséquent.

V.3.2 La modélisation du problème

Voici l'un des plans de plus haut niveau décrit par l'utilisateur dans le formalisme de O-Plan :

```
task Operation_Columbus_Ground_Transports_Only;
  nodes sequential
          1 start,
          parallel
            3 action {transport_ground_transports Honolulu Delta},
            4 action {transport_helicopters Honolulu Delta}
          end_parallel,
          parallel
            5 action {evacuate Abyss 50},
            6 action {evacuate Barnacle 100},
            7 action {evacuate Calypso 20}
          end_parallel,
          parallel
            8 action {fly_passengers Delta Honolulu},
            9 action {transport_ground_transports Delta Honolulu},
           10 action {transport_helicopters Delta Honolulu}
          end_parallel,
          2 finish
        end_sequential;

  effects {location_gt GT1} = Honolulu at 1,
          {location_gt GT2} = Honolulu at 1,
          {in_use_fo GT1} = in_transit at 1,
```

```
            {in_use_for GT2} = in_transit at 1,

            {location_at AT1} = Honolulu at 1,
            {in_use_for AT1} = in_transit at 1,

            {apportioned_forces GT} at 1,

            {at C141} = Honolulu at 1,
            {at C5} = Honolulu at 1,
            {at KC10} = Honolulu at 1,
            {at B707} = Delta at 1,
            {runway_status_at Delta} = clear at 1,
            {runway_status_at Honolulu} = clear at 1,
            {gt_capacity 25} at 1,
            {at_capacity 35} at 1;

    resources
      consumes {resource aviation_fuel_delta_tank1}
                                  = 0 .. 40000 gallons overall,
      consumes {resource aviation_fuel_KC10_tank1}
                                  = 0 .. 30000 gallons overall,
      consumes {resource diesel_fuel_delta_tank2}
                                  = 0 .. 8000 gallons overall;

end_task;
```

On s'aperçoit que la tâche programmée guide fortement O-Plan dans sa recherche et décrit pratiquement complètement le plan recherché. La combinatoire porte sur le choix de la tâche initiale et de ses décompositions. Chacune des actions sera développée en utilisant un schéma d'action de niveau inférieur:

```
schema transport_ground_transports;
;;;
;;; This schema is used to provide ground transportation from
;;; one air base location to another air base location.  This must take
;;; place by using a cargo plane to fly the nominated transportation
;;; vehicles from the source base to the destination base.
;;;
;;; fly all transport explictly from one base to another. No "forall" yet.
;;;
  vars  ?FROM = ?{type air_base},
        ?TO   = ?{type air_base};

  expands {transport_ground_transports ?FROM ?TO};

  nodes 1 action {load ground_transports},
```

```
        2 action {take_off_from ?FROM},
        3 action {fly_to ?TO},
        4 action {land_at ?TO},
        5 action {unload ground_transports};

  orderings 1 ---> 2, 2 ---> 3, 3 ---> 4, 4 ---> 5;

  conditions achieve {at C5} = ?FROM at 1,
             unsupervised {location_gt GT1} = ?FROM at 1,
             unsupervised {location_gt GT2} = ?FROM at 1,
             unsupervised {runway_status_at ?FROM} = clear at begin_of 2,
             supervised {runway_status_at ?FROM} = in_use at end_of 2 from
                                                   begin_of 2,
             unsupervised {runway_status_at ?TO} = clear at begin_of 4,
             supervised {runway_status_at ?TO} = in_use at end_of 4 from
                                                   begin_of 4;
  effects {at C5} = ?TO at 5,
          {location_gt GT1} = ?TO at 5,
          {location_gt GT2} = ?TO at 5,
          {in_use_for GT1} = available at 5,
          {in_use_for GT2} = available at 5,
          {runway_status_at ?FROM} = in_use at begin_of 2,
          {runway_status_at ?FROM} = clear at end_of 2,
          {runway_status_at ?TO} = in_use at begin_of 4,
          {runway_status_at ?TO} = clear at end_of 4;
end_schema;
```

Outre le plan de la tâche, on y voit ses effets ainsi que ses conditions. Une fonctionnalité récente et très intéressante de O-Plan est ici présente. C'est le mot clé `achieve` qui permet de fixé un but qui devra être satisfait. Le choix de l'action (ou de la série d'actions) nécessaire pour y aboutir restant à la charge du système. On retrouve une capacité classique des planificateurs.

Comparons maintenant ces deux descriptions à celle que nous avons fourni à IXTET. Au niveau global nous ne fournissons que la description du but final (ici le nombre de personnes à évacuer). IXTET ne gérant pas encore un but exprimé uniquement par l'état d'une ressource, nous avons dû créer des tâches fictives imposant la consommation à la fin du plan de cette ressource et validant un but final. Cette ressource n'existant pas à l'origine, cela déclenche l'insertion de tâches permettant de la produire. Ces tâches diminuent en fait la quantité de ressources disponibles sur l'île (dans notre cas, le nombre de personnes à évacuer) pour produire de cette ressource (dans notre cas, le nombre de personnes déjà évacuées).

Voici la description d'une telle tâche :

```
task Evacuation_finished_Abyss()(start,end)
{
        main evac_status;

        consume(evacuated_from_Abyss():NB_TO_EVACUATE_FROM_ABYSS,start);
        event(evac_status(Abyss):(to_be_evacuated,OK),start);

        (end-start) in [00:00:00.01,00:00:00.01];
}
```

Les tâches permettant de décrire l'évacuation des autres villes sont similaires. Il suffit alors d'insérer les lignes suivantes dans le plan partiel initial pour déclencher l'évacuation de chacune des villes :

```
timepoint evac_OK;

// Initial
explained event(evac_status(Abyss):(?,to_be_evacuated), start);
explained event(evac_status(Barnacle):(?,OK), start);
explained event(evac_status(Calypso):(?,to_be_evacuated), start);

// Goal
hold(evac_status(Abyss):OK, (evac_OK, end));
hold(evac_status(Barnacle):OK, (evac_OK, end));
hold(evac_status(Calypso):OK, (evac_OK, end));

// Constraint
start < evac_OK < end;
```

Voici maintenant la description de la tâche de transports des moyens terrestres.

```
task transport_ground_transports(?from, ?to) (start, end)
{
        timepoint t1,t2;
        ?from != ?to;
        main location_gt;
        main at;

        // Load ground transports
        event(at(C5):(?from,in_use), start);
        hold(at(C5):in_use, (start,end));
        event(at(C5):(in_use,?to), end);
        use(res_vehicle(C5) : 1, (start, end));
```

```
        hold(at(C5):?to, (start, end));

        event(runway_status_at(?from):(clear, in_use), start);
        hold(runway_status_at(?from):in_use, (start,t1));
        event(runway_status_at(?from):(in_use, clear), t1);
        use(res_runway(?from):1, (start, t1));

        event(runway_status_at(?to):(clear,in_use),t2);
        hold( runway_status_at(?to):in_use, (t2,end));
        event(runway_status_at(?to):(in_use,clear),end);
        use(res_runway(?to):1, (t2, end));

        event(location_gt(GT1):(?from, in_use), start);
        hold( location_gt(GT1):in_use, (start,end));
        event(location_gt(GT1):(in_use, ?to), end);
        event(location_gt(GT2):(?from, in_use), start);
        hold( location_gt(GT2):in_use, (start,end));
        event(location_gt(GT2):(in_use, ?to), end);
        use(res_vehicle(GT1):1, (start,end));
        use(res_vehicle(GT2):1, (start,end));

        (t1-start) in [00:10:00,00:10:00];
        (t2-t1) in [03:00:00,03:00:00];
        (end - t2) in [00:10:00,00:10:00];
}
```

V.3.3 Les plans solutions produits

À partir de ces informations, les deux systèmes produisent un plan d'évacuation de l'île. Voici une instanciation du plan obtenu par O-Plan:

```
Plan starts.
       0:00 Start  (transport_ground_transports honolulu delta).
       0:00 Action (load ground_transports).
       0:00 Action (take_off_from honolulu).
       0:00 Action (fly_to delta).
       0:00 Action (land_at delta).
       0:00 Action (unload ground_transports).
       0:00 Finish (transport_ground_transports honolulu delta).
       0:00 Start  (transport_helicopters honolulu delta).
       0:00 Action (load air_transports).
       0:00 Action (take_off_from honolulu).
       0:00 Action (fly_to delta).
       0:00 Action (land_at delta).
       0:00 Action (unload air_transports).
```

```
    0:00 Finish (transport_helicopters honolulu delta).
    0:00 Start  (drive 25 in gt1 from abyss).
    0:00 Start  (drive 25 in gt2 from barnacle).
 3:00:00 Finish (drive 25 in gt1 from abyss).
 3:00:00 Finish (drive 25 in gt2 from barnacle).
 3:00:00 Start  (drive 20 in gt1 from calypso).
 3:00:00 Start  (drive 25 in gt2 from abyss).
 6:00:00 Finish (drive 20 in gt1 from calypso).
 6:00:00 Action (evacuate calypso 20).
 6:00:00 Finish (drive 25 in gt2 from abyss).
 6:00:00 Action (evacuate abyss 50).
 6:00:00 Start  (drive 25 in gt1 from barnacle).
 6:00:00 Start  (drive 25 in gt2 from barnacle).
 9:00:00 Finish (drive 25 in gt1 from barnacle).
 9:00:00 Finish (drive 25 in gt2 from barnacle).
 9:00:00 Start  (drive 25 in gt1 from barnacle).
12:00:00 Finish (drive 25 in gt1 from barnacle).
12:00:00 Action (evacuate barnacle 100).
12:00:00 Start  (fly_passengers delta honolulu).
12:00:00 Action (load passengers).
12:00:00 Action (take_off_from delta).
12:00:00 Action (fly_to honolulu).
12:00:00 Action (land_at honolulu).
12:00:00 Action (unload passengers).
12:00:00 Finish (fly_passengers delta honolulu).
12:00:00 Start  (transport_ground_transports delta honolulu).
12:00:00 Action (load ground_transports).
12:00:00 Action (take_off_from delta).
12:00:00 Action (fly_to honolulu).
12:00:00 Action (land_at honolulu).
12:00:00 Action (unload ground_transports).
12:00:00 Finish (transport_ground_transports delta honolulu).
12:00:00 Start  (transport_helicopters delta honolulu).
12:00:00 Action (load air_transports).
12:00:00 Action (take_off_from delta).
12:00:00 Action (fly_to honolulu).
12:00:00 Action (land_at honolulu).
12:00:00 Action (unload air_transports).
12:00:00 Finish (transport_helicopters delta honolulu).
Plan finishes.
```

Les figures V.21 page 114 et V.22 page 114 montrent deux plans obtenus par IxTeT. Le détail de chacune des tâches primitives décrites par O-Plan est caché dans les tâches de niveau plus élevé d'IxTeT. Le tableau de la figure V.20 page ci-contre montre les performances obtenues par IxTeT pour calculer le second plan en utilisant tout d'abord

	Temps (s)	Nb de nœuds	Retours-arrière
Planification globale	**40**	**218**	**22**
Phase 1	0.1	9	0
Phase 2	6.2	61	0
Union Phases 1 et 2	1.9	19	0
Phase 3	0.1	8	0
Union avec les précédentes	1.3	16	0
Total par Union	**9.6**	**113**	**0**
Phase 1	0.1	9	0
Insertion Phase 2	14.8	101	0
Insertion Phase 3	3.7	51	0
Total par Insertion	**15.6**	**160**	**0**

FIG. V.20 – *Temps de calcul du plan n'utilisant que les moyens terrestres*

une planification globale puis, après avoir découpé le plan en trois phases, en utilisant l'union de plans et enfin l'insertion de buts.

Le temps calcul de ces mêmes plans par O-Plan est de 4 à 5 secondes. Cette rapidité d'exécution s'explique (en partie) par la quantité d'informations supplémentaires fournie à O-Plan qui lui évite toute la phase de recherche non déterministe qu'I_XT_ET doit réaliser.

V.3.4 I_XT_ET vs O-Plan

Nous ne nous aventurerons pas à comparer quantitativement les deux systèmes (concernant O-Plan nous manquons d'ailleurs d'informations sur ce sujet) car les objectifs des deux systèmes ne sont pas les mêmes.

O-Plan est un système qui, outre le planificateur, intègre (ou intégrera) tout un ensemble de logiciels interactifs d'aide à la spécification, de distribution de l'information, de mise au point de plans, *etc.* Sur ce point, O-Plan est très largement en avance sur le développement d'I_XT_ET. Mais, si le besoin de ces outils s'est fait rapidement sentir dans O-Plan, cela est aussi dû à la diversité et au nombre d'informations que l'on doit décrire à O-Plan.

Le langage d'O-Plan semble d'ailleurs très riche mais ceci parce que l'approche

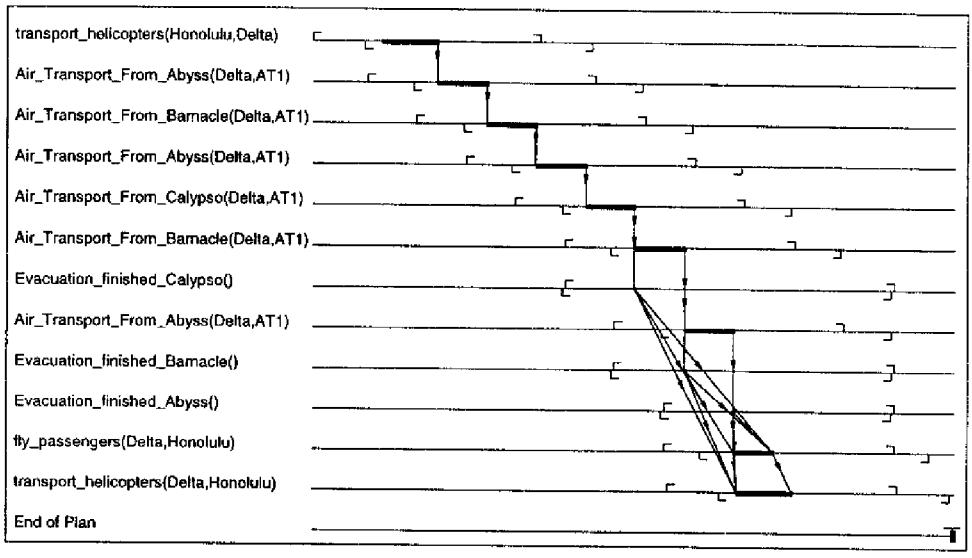


FIG. V.21 – *Plan d'évacuation n'utilisant que les hélicoptères*

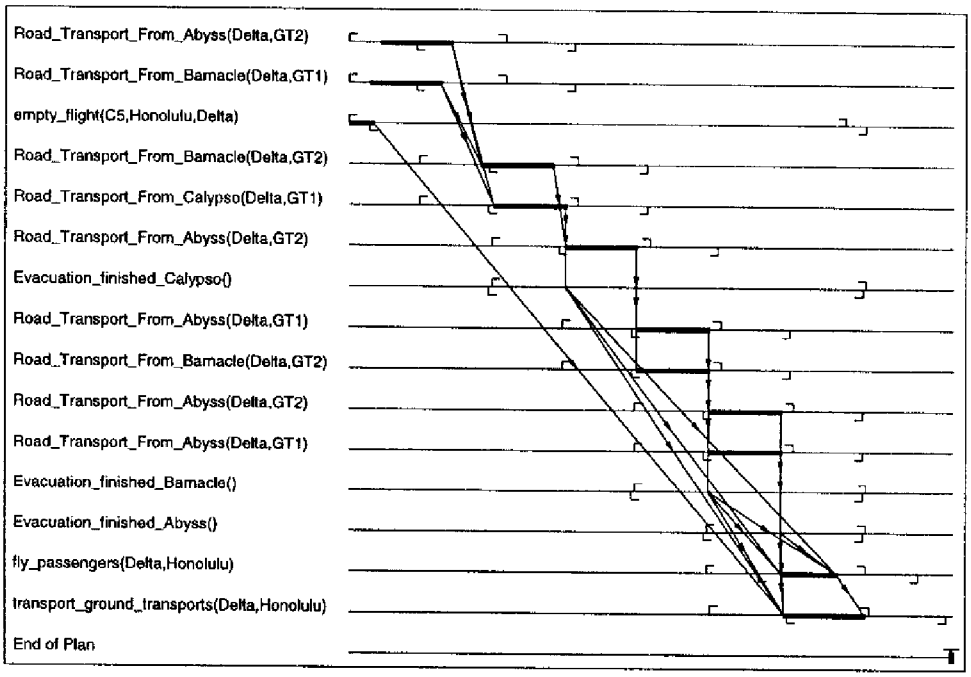


FIG. V.22 – *Plan d'évacuation n'utilisant que les moyens terrestres*

adoptée par O-Plan exige de spécifier beaucoup de chose ; il y a beaucoup à programmer pour réduire la combinatoire dans l'approche par décomposition de tâches. IxTeT qui syntaxiquement offre moins de constructeurs a été suffisamment expressif pour spécifier un problème du type « évacuation de Pacifica » en tenant compte des nombreuses contraintes de l'environnement PRECiS. Le codage de cet exemple n'a nécessité que deux jours de travail (phase de mise au point inclue). Un plan pour lequel O-Plan nécessite 5 pages de description, est décrit en deux pages dans le formalisme d'IxTeT.

D'autre part IxTeT n'a aucunement besoin de la description des tâches de plus haut niveau (qui décrivent des modèles de plans complets) ni des celles des tâches les plus primitives. Il est capable de retrouver seul l'ensemble des actions nécessaires pour satisfaire les buts fixés. O-Plan en serait peut-être capable en utilisant son prédicat `achieve` mais nous n'avons pas d'informations sur ses capacités à ce niveau. D'un autre côté, la possibilité pour l'utilisateur de fournir au système une trame générale de plan lors qu'elle est connue est une possibilité très intéressante de O-Plan qu'IxTeT ne possède pas actuellement. Avec IxTeT, grâce aux opérateurs d'union de plans et d'insertion de buts que nous avons créés pour IxTeT, l'utilisateur peut faire profiter le planificateur de cette connaissance de haut niveau qu'il possède sur le plan général pour accélérer le processus d'élaboration de plans. L'utilisation de ces opérateurs est donc bien un apport réel au système IxTeT pour le rendre plus fonctionnel. Mais il faudrait enrichir le langage de spécification d'IxTeT afin de l'amener vers un véritable langage d'algèbre de plans. Cela permettrait de spécifier des problèmes construisant peu à peu un plan général en utilisant différents types d'opérateurs de composition de plans en fonction des résultats des phase précédentes.

Il serait réellement important de connaître les mécanismes internes de O-Plan afin de comparer de manière plus formelle ces deux systèmes qui « opposent » une méthode par décomposition de tâches et une méthode par raffinement de plans partiels. Les apports de l'un vers l'autre seraient alors très enrichissants.

Conclusion et Perspectives

Le but que nous nous étions fixé était d'obtenir une distribution de la planification entre différents agents coopérant en utilisant un système de planification prédictif général.

Nous avons donc enrichi le planificateur IxTeT pour faire quelques pas sur la route menant vers cet objectif ambitieux :

- en enrichissant le mécanisme interne d'explication des propositions d'IxTeT par la notion d'assertions primordiales ;
- en validant la méthode de remise en cause des liens causaux et en montrant comment on pouvait limiter ses effets sur la complexité du rétablissement de la cohérence par des méthodes adéquates de filtrage.
- en décrivant formellement des opérateurs d'union de plans partiels et d'insertion de buts qui ont été intégrés à IxTeT ;
- en proposant un algorithme réparti permettant de propager un ensemble de contraintes temporelles numériques distribuées entre plusieurs agents ;
- en exhibant deux algorithmes utilisant les opérateurs de composition de plans permettant de réaliser de la planification incrémentale et de la planification distribuée dans un contexte ne tenant pas compte de l'exécution des plans ;
- en montrant leur utilisation et les gains réels qu'ils apportent sur différents exemples.

Ces quelques pas nous ont permis de planifier des problèmes complexes dont la résolution ne semblaient pas envisageables jusqu'à présent sauf par des planificateurs du type d'O-Plan mais auxquels il faut fournir un grand nombre d'informations supplémentaires. Nous avons pu ainsi poser quelques jalons pour aborder le problème de la

planification multi-agents grâce à l'aspect incrémental. La mise en œuvre des exemples nous a enfin amené à montrer les grandes qualités expressives du langage de description d'IxTeT.

Mais, le chemin à parcourir est encore long. Tout au long de ce document, nous avons présenté quelques-uns des obstacles restant en proposant parfois une piste permettant peut-être de les surmonter. Voici les futurs développements ou extensions qui nous semblent les plus importants ainsi que quelques problèmes non résolus que nous avons rencontrés lors de nos expérimentations :

- une meilleure caractérisation des paramètres de contrôles de l'heuristique guidant le planificateur lors de sa recherche (actuellement la détermination des « bonnes » valeurs de ces paramètres est encore trop dépendantes du problème et surtout des buts) ;
- la validation de l'approche proposée pour résoudre les problèmes distribués sur des exemples d'une plus grande ampleur tant en nombre d'agents qu'en taille du plan propre à chaque agent (ceci est lié fortement au point précédent) ;
- la nécessité d'intégrer une gestion des contraintes temporelles contingentes afin de produire des plans contrôlables (en utilisant éventuellement des travaux sur les CSP mixtes?) ;
- le besoin de déterminer des contraintes de synchronisation réalistes (c.-à-d. contrôlables) entre les plans de différents agents (en intégrant directement cette notion dans la phase de planification?) afin de déterminer les messages nécessaires à la bonne exécution des plans ;
- des méthodes de caractérisation des problèmes solubles par des opérateurs de combinaison de plans et permettant de détecter les buts que l'on peut découpler (temporellement et/ou logiquement) ;
- l'extension du paradigme de fusion de plans proposé par [Robert 96] afin de le valider dans le cadre d'une utilisation d'un système de planification général gérant les contraintes temporelles numériques et les ressources partageables.

Plusieurs travaux de recherche visant à étendre les capacités d'IxTeT sont actuellement en cours au LAAS : tout d'abord la possibilité d'intégrer dans la description des tâches des quantificateurs sur les variables et des effets conditionnels dépendant du contexte (ce dernier point nécessite la prise en compte de la négation dans IxTeT) ;

d'autres travaux concernent la possibilité d'intégrer à IxTeT des planificateurs spécialisés permettant de résoudre des problèmes pour lesquels on dispose d'algorithmes spécifiques efficaces (par exemple la recherche d'un chemin dans un graphe topologique explicite) ; enfin la mise en œuvre d'un superviseur de plans permettant d'exécuter réellement des plans produits par IxTeT. Tous ces travaux amènerons certainement de nombreuses modifications à la version actuelle. L'une des difficultés sera de conserver une cohérence globale entre toutes ces améliorations afin de garantir leur bon fonctionnement en commun.

A notre avis, le véritable défi pour les planificateurs du type d'IxTeT (systèmes de planification prédictif garantissant la complétude de leur exploration, gérant différents types de contraintes...) est leur intégration à un système réel et donc la prise en compte des interactions entre planification et exécution. L'autre défi est le développement d'une expertise permettant de formaliser et de spécifier facilement les tâches dans le langage fourni en choisissant à chaque fois le codage le plus adapté selon le type du problème.

Annexes

Annexe A

Algorithmes de gestion des contraintes temporelles

Dans cette annexe sont réunis tous les algorithmes de gestion des contraintes temporelles. Tout d'abord l'algorithme de propagation complet (en $O(n^3)$) puis l'algorithme de propagation incrémentale (en $O(n^2)$) et enfin les deux algorithmes qui permettent l'ajout d'une contrainte et l'interrogation du réseau de contraintes.

Alg. A.1 PROPAGATION COMPLÈTE DES CONTRAINTES TEMPORELLES

```
- ∴ - On traite N instants.
- ∴ - K est une matrice d'intervalles décrivant les contraintes initiales non propagées
- ∴ - Les contraintes manquantes sont initialisées par défaut à ]−∞, +∞[
Début
    Pour k ← 1 à N Faire
        - ∴ - A chaque itération, on traite les chemins de longueur k
        Pour i ← 1 à N Faire
            Pour j ← 1 à N Faire
                - ∴ - On calcule la contrainte entre t_i et t_j
                K(i,j) ← (K(i,k) ⊕ K(k,j)) ∩ K(i,j)
            Fin Pour
        Fin Pour
    Fin Pour
Fin
```

Alg. A.2 PROPAGATION INCRÉMENTALE DES CONTRAINTES TEMPORELLES(i_0, j_o)

```
- ∴ - On traite N instants
- ∴ - On suppose que la matrice des contraintes K est complètement propagée
      sauf pour pour les instants d'indices i_0 et j_0
- ∴ - On applique donc l'algorithme de propagation globale sur ces seuls instants
Début
    Pour i ← 1 à N Faire
        K(i,j_o) ← (K(i,i_o) ⊕ K(i_0,j_0)) ∩ K(i,j_o)
    Fin Pour
    Pour i ← 1 à N Faire
        Pour j ← 1 à N Faire
            K(i,j) ← (K(i,j_0) ⊕ K(j_0,j)) ∩ K(i,j)
        Fin Pour
    Fin Pour
Fin
```

Alg. A.3 AJOUT OU MODIFICATION D'UNE CONTRAINTE TEMPORELLE(i_0, j_0, C_0)

– $\therefore$ – *On veut ajouter la nouvelle contrainte temporelle C_0 entre les instant t_{i_0} et t_{j_0}*
– $\therefore$ – *Le résultat est Ok ou Erreur selon la cohérence de l'ensemble des contraintes*

Début
  $Test \leftarrow$ INTERROGATION DES CONTRAINTES TEMPORELLES(i_0, j_0, C_0)
  Si $Test$ = Redondance **Alors Retour** (Ok)
  Si $Test$ = Inconsistance **Alors Retour** (Erreur)
  $K(i_0, j_0) \leftarrow K(i_0, j_0) \cap C_0$
  PROPAGATION INCRÉMENTALE DES CONTRAINTES TEMPORELLES(i_0, j_o)
  Retour (Ok) – $\therefore$ – *La propagation s'est obligatoirement bien passée*
Fin

Alg. A.4 INTERROGATION DES CONTRAINTE TEMPORELLES(i_0, j_0, C_0)

– $\therefore$ – *On suppose que la matrice des contraintes K est déjà propagée*
– $\therefore$ – *On s'interroge sur la compatibilité d'une contrainte C_0 entre les instants t_{i_0} et t_{j_0}*
– $\therefore$ – *Le résultat est :*
  Redondance : si l'ajout ne modifie pas K
  Inconsistance : si l'ajout rend l'ensemble des contraintes inconsistant
  Propagation : si l'ajout est possible et nécessite une propagation

Début
  Si $K(i_0, j_0) \subset C_0$ **Alors Retour** (Redondance)
  Si $K(i_0, j_0) \cap C_0 = \emptyset$ **Alors Retour** (Inconsistance)
  Retour (Propagation)
Fin

Annexe B

Algorithmes de gestion des contraintes sur les variables

Dans cette annexe sont réunis tous les algorithmes de gestion des contraintes sur les variables. Les quatre premiers (de B.1 à B.4) permettent l'ajout de contraintes et testent la 2-consistance du réseau de contraintes (algorithmes B.5 et B.6) en temps polynômial. Le dernier algorithme (B.7 page 130) valide complètement le réseau.

Alg. B.1 INSÉRER UNE CONTRAINTE D'ÉGALITÉ($?v_i$, $?v_j$)

– $\therefore$ – *$?v_i$ et $?v_j$ sont les deux variables dont on veut imposer l'égalité*

Début

$\epsilon(?v_i) \leftarrow \epsilon(?v_i) \cup \epsilon(?v_j)$

$D_{\epsilon(?v_i)} \leftarrow D_{\epsilon(?v_i)} \cap D_{\epsilon(?v_j)}$

$E \leftarrow E - \{\epsilon(?v_j)\}$

PROPAGER LES MODIFICATIONS DE CLASSES DE VARIABLES($\{\epsilon(?v_i)\}$)

Fin

Alg. B.2 RESTREINDRE LE DOMAINE D'UNE VARIABLE($?v, D$)

– $\therefore$ – *La valeur de la variable $?v$ doit être dans D*

Début

$D_{\epsilon(?v)} \leftarrow D_{\epsilon(?v)} \cap D$

PROPAGER LES MODIFICATIONS DE CLASSES DE VARIABLES($\{\epsilon(?v)\}$)

Fin

Alg. B.3 INSÉRER UNE CONTRAINTE D'INÉGALITÉ($?v_i, ?v_j$)

– $\therefore$ – *$?v_i$ et $?v_j$ sont les deux variables qui ne doivent pas être égales*

Début

Ajouter dans $K_{\neq}$ la contrainte Diff($\epsilon(?v_i), \epsilon(?v_j)$)

PROPAGER LES MODIFICATIONS DE CLASSES DE VARIABLES($\{\epsilon(?v_i), \epsilon(?v_j)\}$)

Fin

Alg. B.4 INSÉRER UNE CONTRAINTE DE DÉPENDANCE($?v_i, D_i, ?v_j, D_j$)

– $\therefore$ – *Si la valeur de $?v_i$ appartient à D_i alors la valeur de $?v_j$ appartient à D_j*

Début

Ajouter dans $K_{\Rightarrow}$ la contrainte Dep($\epsilon(?v_i), D_i, \epsilon(?v_j), D_j$)

PROPAGER LES MODIFICATIONS DE CLASSES DE VARIABLES($\{\epsilon(?v_i), \epsilon(?v_j)\}$)

Fin

Alg. B.5 PROPAGER LES MODIFICATIONS DE CLASSES DE VARIABLES(M)

– ∴ – M est l'ensemble des classes dont il faut propager la modification

Début
  $K \leftarrow$ les contraintes de $K_{\neq}$ et de $K_{\Rightarrow}$ portant sur une ou plusieurs classes de M
  Tant Que K n'est pas vide **Faire**
    Extraire de K une contrainte $C(\epsilon_i, \epsilon_j)$ *– ∴ – C contraint les classes ϵ_i et ϵ_j*
    Si PROPAGER VIA LA CONTRAINTE($C(\epsilon_i, \epsilon_j)$) **Alors**
      $K \leftarrow K \cup$ les contraintes de $K_{\neq}$ et de $K_{\Rightarrow}$ portant sur ϵ_i ou ϵ_j
    Fin Si
  Fin Tant Que
Fin

Alg. B.6 PROPAGER VIA LA CONTRAINTE($C(\epsilon_i, \epsilon_j)$)

– ∴ – $C(\epsilon_i, \epsilon_j)$ est la contrainte dont il faut propager la modification

Début
  Modification ← Faux
  Si $C \in K_{\neq}$ **Alors**
    – ∴ – C impose $Val(\epsilon_i) \neq Val(\epsilon_j)$
    Si $|D_{\epsilon_i}| = 1$ **et** $D_{\epsilon_i} \subset D_{\epsilon_j}$ **Alors**
      $D_{\epsilon_j} = D_{\epsilon_j} - D_{\epsilon_i}$; Modification ← Vrai
    Fin Si
    Si $|D_{\epsilon_j}| = 1$ **et** $D_{\epsilon_j} \subset D_{\epsilon_i}$ **Alors**
      $D_{\epsilon_i} = D_{\epsilon_i} - D_{\epsilon_j}$; Modification ← Vrai
    Fin Si
  Sinon *– ∴ – C appartient à $K_{\Rightarrow}$*
    – ∴ – C impose $Val(\epsilon_i) \in D_i \Rightarrow Val(\epsilon_j) \in D_j$
    Si $D_{\epsilon_j} \cap D_j = \emptyset$ **Alors**
      $D_{\epsilon_i} \leftarrow D_{\epsilon_i} - D_i$; Modification ← Vrai
    Fin Si
    Si $D_{\epsilon_i} \subset D_i$ **Alors**
      $D_{\epsilon_j} \leftarrow D_{\epsilon_j} \cap D_j$; Modification ← Vrai
    Fin Si
  Fin Si
  Retour (Modification)
Fin

Alg. B.7 VALIDATION DES CONTRAINTES SUR LES VARIABLES(V, G)

- $\therefore$ - *V est l'ensemble des variables associées à leurs domaines*
- $\therefore$ - *G représente un réseau de contraintes entre ces variables*

Début
  Si $\exists\ ?v \in V$ tel que $D_{\epsilon(?v)} = \emptyset$ **Alors**
    Retour (Inconsistance)
  Fin Si
  Si toutes les variables sont instanciées **Alors**
    Retour (Ok)
  Fin Si
  Choisir une variable $?v_i \in V$ tel que $\texttt{Card}(D_{\epsilon(?v_i)}) > 1$
  Pour chaque valeur C de $D_{\epsilon(?v_i)}$ **Faire**
    $(V', G') \leftarrow (V, G)$
    RESTREINDRE LE DOMAINE D'UNE VARIABLE($?v_i$, C) dans (V', G')
    Si VALIDATION DES CONTRAINTES SUR LES VARIABLES(V', G') = Ok **Alors**
      Retour (Ok)
    Fin Si
  Fin Pour
  Retour (Échec)
Fin

Annexe C

Un exemple complet de planification

Voici le contenu intégral des trois fichiers spécifiant le problème de planification du robot R3 (voir § V.2 page 92). Le premier fichier décrit les constantes, les attributs et les ressources du monde. Le second contient les tâches exécutables par le robot. Le troisième contient un plan partiel initial spécifiant le problème à résoudre.

C.1 Constantes, Attributs et Ressources

```
// Fichier de description des constantes, attributs et ressources
// +--------------+--------------+--------------+
// |              |              |              |
// |              |              |              |
// |  Bureau 1    |  Bureau 2    |   Bureau 3   |
// |              |              |              |
// |              |              |              |
// +----      ----+-----     -----+-----     ----+
// |                                            |
// |                                          +--|
// |                 Couloir                  |: | <- Recharge
// |               de capacite 2             +--|
// |                                            |
// +----------+---     -----+--     ---+---------+
//            |             |          |
//            | Imprimante  | Courier  |
//            |             |          |
//            +-------------+----------+

//-------------------------------------------------------------
// Macros pour definir une ressource productible
// a capacite maximum
#define DefDRes(ResName, CapMax, CapInit) \
     resource FreeCapacity##ResName() {\
        capacity = CapMax - CapInit;\
     }\
     \
     resource Capacity##ResName() {\
        capacity = CapInit;\
     }
#define Consomme(ResName, Quant, When) \
     consume(Capacity##ResName(): Quant, When);\
     produce(FreeCapacity##ResName(): Quant, When)
#define Produit(ResName, Quant, When) \
     produce(Capacity##ResName(): Quant, When);\
     consume(FreeCapacity##ResName(): Quant, When)

//-------------------------------------------------------------
// Pour definir les attributs et les ressources utilisees
#define RES_PUISSANCE
#define ATT_ETATROBOT
#define RES_COULOIR

//-------------------------------------------------------------
```

```
constant PLAN_DUREE_MIN = 0:00:00;
constant PLAN_DUREE_MAX = 2:30:00;
constant MOVE_DUREE = 3:00;
constant ACTION_DUREE = 1:45;
constant CHARGE_DUREE = 5:00;

constant CONSO_DEPLACEMENT = 10;
constant CONSO_OP_COURIER = 5;
constant PROD_RECHARGE = 20;
constant PUIS_UTILISEE_RECHARGE = 30;

constant ChargeBatterieMax = 50;
constant ChargeBatterieR1 = 25;
constant ChargeBatterieR2 = 25;
constant ChargeBatterieR3 = 25;

constant CapacityCouloirMax = 4;
constant CapacityCouloirInit = 4;
constant OccupCouloir = 2;

constant ROBOTS = {R1, R2, R3, R4};
constant BUREAUX = {Bureau1, Bureau2, Bureau3};
constant DESTINATAIRES = {Personne1, Personne2, Personne3};
constant SERVICES = {Imprimante, Courier};
constant COULOIRS = {Couloir};
constant SALLES = BUREAUX | SERVICES;
constant POSITIONS = SALLES | COULOIRS;
constant PRISES = {PriseBureau1, PriseCouloir};
constant CHARGES = {CourierPourPersonne1,
    CourierPourPersonne2,
    CourierPourPersonne3,
    Aucune};
constant ETATS_COULOIR = {Occupe, Libre};
constant POSCOURIER = {Ailleurs, Bac, R1, R2, R3, Bureau1, Bureau2, Bureau3};
constant PERSONNES = {Personne1, Personne2, Personne3};
constant ETATS_ALIM = {InUse, Free};

#ifdef ATT_ETATROBOT
attribute EtatRobot(?robot) {
    ?robot in ROBOTS;
    ?value in ETATS_ROBOT;
}
#endif //- of ifdef ATT_ETATROBOT

attribute PosCourier(?personne) {
    ?value in POSCOURIER;
    ?personne in PERSONNES;
```

```
}

attribute Position(?robot){
    ?robot in ROBOTS;
    ?value in POSITIONS;
}

attribute ChargeRobot(?robot) {
    ?robot in ROBOTS;
    ?value in CHARGES;
}

#ifdef RES_COULOIR
DefDRes(Couloir, CapacityCouloirMax, CapacityCouloirInit)
#endif //- of ifdef RES_COULOIR

resource BacACourier() {
    capacity = 1;
}

#ifdef RES_PUISSANCE
resource PuissancePrise() {
    capacity = 80;
}
attribute AlimentationCouloir() {
    ?value in ETATS_ALIM;
}
DefDRes(BatterieR1, ChargeBatterieMax, ChargeBatterieR1)
DefDRes(BatterieR2, ChargeBatterieMax, ChargeBatterieR2)
DefDRes(BatterieR3, ChargeBatterieMax, ChargeBatterieR3)
#endif //- of ifdef RES_PUISSANCE
//-------------------------------------------------------------
```

C.2 Tâches de R3

```
// Fichier de description des tâches
// Les relations du domaines
#include "Domaine-Relations"

task R3SortDe(?de) (start, end) {

    ?de in SALLES;

    timepoint SortDe;
    timepoint EntreVers;

    main Position;

#ifdef ATT_ETATROBOT
    event(EtatRobot(R3): (Oisif, EnMouvement), start);
    hold(EtatRobot(R3): EnMouvement, (start, end));
    event(EtatRobot(R3): (EnMouvement, Oisif), end);
#endif // ATT_ETATROBOT

    hold(Position(R3): ?de , (start , SortDe));
    event(Position(R3): (?de, Couloir), SortDe);
    hold(Position(R3): Couloir, (SortDe, end));

#ifdef RES_COULOIR
    Consomme(Couloir, OccupCouloir , SortDe);
#endif // RES_COULOIR

#ifdef RES_PUISSANCE
    Consomme(BatterieR3, CONSO_DEPLACEMENT, end);
#endif // RES_PUISSANCE

    (end - SortDe) in [MOVE_DUREE, MOVE_DUREE];
    (SortDe - start) in [MOVE_DUREE, MOVE_DUREE];

}

task R3EntreDans(?vers) (start, end) {

    ?vers in SALLES;

    timepoint EntreVers;

    main Position;
```

```
#ifdef ATT_ETATROBOT
    event(EtatRobot(R3): (Oisif, EnMouvement), start);
    hold(EtatRobot(R3): EnMouvement, (start, end));
    event(EtatRobot(R3): (EnMouvement, Oisif), end);
#endif //- of ifdef ATT_ETATROBOT

    hold(Position(R3): Couloir, (start, EntreVers));
    event(Position(R3): (Couloir, ?vers), EntreVers);
    hold(Position(R3): ?vers, (EntreVers, end));

#ifdef RES_COULOIR
    Produit(Couloir, OccupCouloir, EntreVers);
#endif //- of ifdef RES_COULOIR

#ifdef RES_PUISSANCE
    Consomme(BatterieR3, CONSO_DEPLACEMENT, end);
#endif //- of ifdef RES_PUISSANCE

    (end - EntreVers) in [MOVE_DUREE, MOVE_DUREE];
    (EntreVers - start) in [MOVE_DUREE, MOVE_DUREE];

}

task R3PrendCourier()(start, end) {

    timepoint PriseCourier;

    hold(Position(R3): Courier, (start, end));

    main PosCourier;

#ifdef ATT_ETATROBOT
    event(EtatRobot(R3): (Oisif, Actif), start);
    hold(EtatRobot(R3): Actif, (start, end));
    event(EtatRobot(R3): (Actif, Oisif), end);
#endif //- of ifdef ATT_ETATROBOT

    hold(ChargeRobot(R3): Aucune, (start, PriseCourier));
    event(ChargeRobot(R3): (Aucune, CourierPourPersonne3), PriseCourier);
    hold(ChargeRobot(R3): CourierPourPersonne3, (PriseCourier, end));

    hold(PosCourier(Personne3): Bac, (start, PriseCourier));
    event(PosCourier(Personne3): (Bac, R3), PriseCourier);
    hold(PosCourier(Personne3): R3, (PriseCourier, end));

#ifdef RES_PUISSANCE
    Consomme(BatterieR3, CONSO_OP_COURIER, end);
```

```
#endif //- of ifdef RES_PUISSANCE

    use(BacACourier():1, (start, end));

    (end - start) in [ACTION_DUREE, ACTION_DUREE];
}

task R3PoseCourier() (start, end) {

    timepoint PoseCourier;

    main PosCourier;

    hold(Position(R3): Bureau3, (start, end));

#ifdef ATT_ETATROBOT
    event(EtatRobot(R3): (Oisif, Actif), start);
    hold(EtatRobot(R3): Actif, (start, end));
    event(EtatRobot(R3): (Actif, Oisif), end);
#endif //- of ifdef ATT_ETATROBOT

    hold(ChargeRobot(R3): CourierPourPersonne3, (start, PoseCourier));
    event(ChargeRobot(R3): (CourierPourPersonne3, Aucune), PoseCourier);
    hold(ChargeRobot(R3): Aucune, (PoseCourier, end));

    hold(PosCourier(Personne3): R3, (start, PoseCourier));
    event(PosCourier(Personne3): (R3, Bureau3), PoseCourier);
    hold(PosCourier(Personne3): Bureau3, (PoseCourier, end));

#ifdef RES_PUISSANCE
    Consomme(BatterieR3, CONSO_OP_COURIER, end);
#endif //- of ifdef RES_PUISSANCE

    (end - start) in [ACTION_DUREE, ACTION_DUREE];
}

#ifdef RES_PUISSANCE
task R3RechargeAlimentationCouloir() (start, end) {


    main CapacityBatterieR3;

#ifdef ATT_ETATROBOT
    event(EtatRobot(R3): (Oisif, EnCharge), start);
    hold(EtatRobot(R3): EnCharge, (start, end));
    event(EtatRobot(R3): (EnCharge, Oisif), end);
#endif //- of ifdef ATT_ETATROBOT
```

```
    hold(Position(R3): Couloir, (start, end));

    event(AlimentationCouloir():(Free, InUse), start);
    hold(AlimentationCouloir(): InUse, (start, end));
    event(AlimentationCouloir(): (InUse, Free), end);

    Produit(BatterieR3, PROD_RECHARGE, end);

    use(PuissancePrise(): PUIS_UTILISEE_RECHARGE, (start, end));

    (end - start) in [CHARGE_DUREE, CHARGE_DUREE];
}
#endif //- of ifdef RES_PUISSANCE

//-------------------------------------------------------------
```

C.3 Plan Initial

```
#include "Domaine-Relations"

task R3Init()(start, end) {

    //- Situation Initiale

    explained event(Position(R3): (Couloir, Bureau3), start);
    explained event(PosCourier(Personne3): (Ailleurs, Bac), start);
    explained event(ChargeRobot(R3): (CourierPourPersonne3, Aucune), start);

#ifdef RES_PUISSANCE
    explained event(AlimentationCouloir(): (InUse, Free), start);
#endif //- of ifdef RES_PUISSANCE

#ifdef ATT_ETATROBOT
    explained event(EtatRobot(R3): (Actif, Oisif), start);
#endif //- of ifdef ATT_ETATROBOT

    //- Goal

    timepoint DistCourier, FinDistCourier;
    hold(PosCourier(Personne3): Bureau3, (DistCourier, FinDistCourier));
    (FinDistCourier - DistCourier) in [50.00, 50.00];

    timepoint ArriveDansBureau1;
    hold(Position(R3): Bureau1, (ArriveDansBureau1, end));

    DistCourier < ArriveDansBureau1;

    (end - start) in [PLAN_DUREE_MIN, PLAN_DUREE_MAX];
}
```

Table des figures

Liste des algorithmes

Références bibliographiques

[Aho 74] A.V. Aho, J.E. Hopcroft & J.D. Ullman. The Design and Analysis of Computer Algorithms. Addison-Wesley Publishing Company, 1974.

[Alaoui 90] A. M. Alaoui. *Raisonnement temporel pour la Planification et la Reconnaissance de situations.* Thèse de l'Université Paul Sabatier, Toulouse (France), Laboratoire d'Automatique et d'Analyse des Systèmes (C.N.R.S.), Octobre 1990.

[Allen 81] J. F. Allen. *An Interval-based representation of temporal knowledge.* In 7th International Joint Conference on Artificial Intelligence (IJCAI), Vancouver (Canada), August 1981.

[Allen 83a] J. F. Allen. *Planning using a temporal world model.* In 8th International Joint Conference on Artificial Intelligence (IJCAI), Karlsruhe (RFA), August 1983.

[Allen 83b] J.F. Allen. *Maintaining Knowledge About Temporal Intervals.* Communications of the ACM, vol. 26(11), pages 832–843, 1983.

[Allen 91] J. Allen. Reasoning about Plans, chapter 1: Temporal Reasonning and Planning. J. Allen, H. Kautz, R. Pelavin and J. Tenenberg editors, Morgan Kaufmann San Mateo (CA), 1991.

[Chapman 87] D. Chapman. *Planning for Conjunctive goals.* Artificial Intelligence, vol. 32, pages 333–377, 1987.

[Chater 95] M. Chater. *De la Planification Incrémentale vers la Planification Multi-Robots.* Rapport de DEA Systèmes Intelligents de

Paris-IX Dauphine, Laboratoire d'Automatique et d'Analyse des Systèmes (C.N.R.S.), Toulouse (France), Octobre 1995.

[Chatila 91] R. Chatila, R. Alami, B. Degallaix, V. Pérébaskine, P. Gaborit & P. Moutarlier. *An Architecture for Task Refinement and Execution Control for Intervention Robot: preliminary experiment.* In 2nd International Symposium on Experimental Robotics, (ISER '91), Toulouse, France, 1991.

[Currie 91] Currie & Tate. *O-plan: the open planning architecture.* In Artificial Intelligence, 52:49-86, 1991.

[Dousson 93] C. Dousson, P. Gaborit & M. Ghallab. *Situation recognition: representation and algoritms.* 13th International Joint Conference on Artificial Intelligence, IJCAI'93, Chambéry(France), 28 Août - 3 Septembre 1993, pp. 166-172, 1993.

[Dousson 94] C. Dousson. *Suivi d'évolutions et Reconnaissance de Chroniques.* Thèse de Doctorat, Université Paul Sabatier, Toulouse, Septembre 1994.

[Fargier 95] H. Fargier, J. Lang & T. Schiex. *Mixed Constraint Satisfaction : a Framework for Decision Problems under Incomplete Knowledge.* In Proc. IJCAI'95, Montréal, Canada, August 1995.

[Fikes 71] R.E. Fikes & N.J. Nilsson. *STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving.* Artificial Intelligence, vol. 2, pages 189–208, 1971.

[Fleury 96] S. Fleury. *Architecture de contrôle distribuée pour robots mobiles autonomes : principes, conception et applications.* Thèse de Doctorat, Université Paul Sabatier, Toulouse, Février 1996.

[Gaborit 90] P. Gaborit, A. Potet & C. Sayettat. *Semantics and Validation Procedures of a Multi-modal Logic for Formalization of Multi-agent Universes.* In Proceedings of European Conference on Artificial Intelligence, 1990.

[Gardiès 79] J.L. Gardiès. Essai sur la logique des modalités. Presses Universitaires de France,Philosophie d'aujourd'hui, 1979.

[Ghallab 91] M. Ghallab, P. Grandjean, P. Gaborit & E. Dekneuvel. *SKIDS Project: interim report.* Contrat CEE-ESPRIT, Projet SKIDS P1560, Janvier 1991, 39p. 91044, Laboratoire d'Automatique et d'Analyse des Systèmes (C.N.R.S.), Toulouse (France), January 1991.

[Henriksson 96] A. Henriksson. *Models and Operations on Plans.* Master's Thesis, Kunglica Tekniska Högskolan, Stockholm, July 1996.

[Hertzberg 93] J. Hertzberg & E. Rutten. *Temporal Planner = Nonlinear Planner + Time Map Manager.* AI-Communications, vol. 6, no. 1, pages 18–26, March 1993.

[Konolige 84] K. Konolige. *A Deduction Model of Belief and its Logic.* Technical note 326, SRI International, 1984.

[Laborie 95] P. Laborie. *I_XT_ET: une Approche Intégrée pour la Gestion de Ressources et la Synthèse de Plans.* Thèse de Doctorat, École Nationale Supérieure des Télécommunications, Paris, Décembre 1995.

[Laruelle 94] H. Laruelle. *Planification Temporelle et Exécution de Tâches en Robotique.* Thèse de Doctorat, Université Paul Sabatier, Toulouse, Avril 1994.

[Marois 95] F. Marois. *Apprentissage de Scénarios.* Thèse, CNAM, Toulouse, Juin 1995.

[McAllester 91] D. McAllester & D. Rosenblitt. *Systematic nonlinear planning.* In Proceedings AAAI-91, pages 634–639, 1991.

[Perebaskine 92] V. Perebaskine. *Une Architecture Modulaire pour le Contrôle d'un Robot Mobile Autonome.* Rapport technique 1152, Thèse de l'Université Paul Sabatier, Toulouse (France), Avril 1992.

[Reece 93] Reece, Tate, Brown, Hoffman & Burnard. *The PRECiS Environment.* In Proceedings of the National Conference on Artificial Intelligence (AAAI-93) ARPA-RL Planning Initiative Workshop, Washington, DC, 1993.

[Robert 96] F. Robert. *Coopération multi-robots par insertion incrémentale de plans.* Thèse de Doctorat, Institut Nationale Polytechnique de Toulouse, Toulouse, Juin 1996.

[Sacerdoti 75] E.D. Sacerdoti. *A Structure for Plans and Behaviours.* technical note 109, SRI, 1975.

[Shoham 87] Y. Shoham. *Temporal Logics in AI: Semantical and Ontological Considerations.* Artificial Intelligence, vol. 33, pages 89–104, 1987.

[Vere 83] S.A. Vere. *Planning in Time: Windows and Durations for activities and goals.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 5, no. 3, May 1983.

[Vidal 95] T. Vidal. *Le Temps en Planification et en Ordonnancement : vers une gestion complète et efficace de contraintes hétérogènes et entachées d'incertitude.* Thèse de Doctorat, Université Paul Sabatier, Toulouse, Septembre 1995.

[Vidal 96] T. Vidal & M. Ghallab. *Dealing with Uncertain Durations in Temporal Constraint Networks dedicated to Planning.* ECAI-96, European Conference in A.I., Budapest (H), August 1996.

[Vilain 86] M. Vilain & H. Kautz. *Constraint Propagation Algorithms for Temporal Reasoning.* In Proceedings of AAAI-86, Philadelphia, Pa. AAAI, August 1986.

[Wilkins 88] D.E. Wilkins. Practical Planning: Extending the Classical AI planning paradigm. Morgan Kaufman, 1988.

Planification Distribuée pour la Coopération Multi-Agents

Permettre à plusieurs agents de planifier et de coordonner leurs activités de manière distribuée tel est l'objectif des travaux présentés dans ce mémoire. L'approche proposée s'appuie sur des opérateurs de composition de plans. Afin de gérer au mieux les interactions entre différents agents, leurs plans sont produits par IxTeT, un système de planification permettant la prise en compte de contraintes temporelles numériques et gérant le parallélisme des tâches ainsi que le partage de ressources.

Les deux premiers chapitres décrivent le formalisme logique utilisé par IxTeT ainsi que le fonctionnement du planificateur lui-même et les améliorations qu'il est possible d'y apporter.

Le troisième chapitre détaille alors formellement les méthodes et algorithmes permettant de réaliser des opérateurs de composition de plans : union de plans, insertion de nouveaux buts dans un plan existant. On y démontre leurs limites théoriques.

Le quatrième chapitre décrit la mise en œuvre de ces opérateurs en exhibant des algorithmes tant pour améliorer les performances de la planification incrémentale mono-agent que pour réaliser un système distribué de planification multi-agents. Dans ce système, un plan global est élaboré par composition successive de plans individuels. Ce plan global reste implicite et n'est donc jamais centralisé. On présente ensuite les problèmes spécifiques rencontrés lorsque planification et exécution sont réalisées simultanément. Ces problèmes ouverts sont cruciaux dans un contexte multi-agents.

Le document se termine par une illustration et une évaluation sur des exemples appliqués au domaine multi-robots permettant d'apprécier les avantages mais aussi les limites de l'utilisation de ces opérateurs de composition de plans et par une comparaison avec un autre système de planification distribuée.

Mots-clés: Planification, Planification Distribuée, Composition de Plans, Multi-Agents, Coopération.

Distributed Planning for Multi-Agents Cooperation

The purpose of this thesis is to enable several agents plan and coordinate their activities in a distributed way. The proposed approach uses plan composition operators. We rely on a planning system IxTeT which manages numerical temporal constraints, parallel tasks and resource sharing. IxTeT is used to generate each agent's plan, while managing their interactions.

The first part describes the logical formalism used in IxTeT, its algorithms and possible improvements.

A second part describes in detail formal methods and algorithms used to implement plans composition operators: union, insertion of new goals in an existing plan; it demonstrates their theoretical limits.

In the next part, these operators are used for incremental mono-agent planning and multi-agents distributed planning. The corresponding algorithms are integrated into a distributed planning system in which the global plan is never centralised; it remains implicit and results from successive compositions of individual plans. The simultaneity of planning and execution leads to specific insolved problems, crucial for multi-agents cooperation.

The last part presents and evaluates examples in a multi-robots context, illustrating the advantages and limits of these planning composition operators. The proposed system is then compared to another planning system.

Keywords : Planning, Distributed Planning, Plans Composition, Multi-Agents, Cooperation.