Modélisation et propagation de contraintes temporelles pour la planification

Jean-François Rit

THESE

présentée à

L'Institut National Polytechnique de Grenoble

Pour obtenir le grade de
Docteur de l'INPG
(arrêté ministériel du 5 juillet 1984)
INFORMATIQUE

par

Jean-François Rit

Modélisation et Propagation de Contraintes Temporelles Pour la Planification

Thèse soutenue le 7 mars 1988 devant la commission d'examen.

G. VEILLON	Président
M. GHALLAB	Rapporteurs
A. QUILLIOT	
J-C LATOMBE	Examinateurs
Y. DESCOTTE	

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Président : Georges LESPINARD **Année 1988**

Professeurs des Universités

BARIBAUD Michel	ENSERG
BARRAUD Alain	ENSIEG
BAUDELET Bernard	ENSPG
BEAUFILS Jean-Pierre	ENSEEG
BLIMAN Samuel	ENSERG
BLOCH Daniel	ENSPG
BOIS Philippe	ENSHMG
BONNETAIN Lucien	ENSEEG
BOUVARD Maurice	ENSHMG
BRISSONNEAU Pierre	ENSIEG
BRUNET Yves	IUFA
CAILLERIE Denis	ENSHMG
CAVAIGNAC Jean-François	ENSPG
CHARTIER Germain	ENSPG
CHENEVIER Pierre	ENSERG
CHERADAME Hervé	UFR PGP
CHOVET Alain	ENSERG
COHEN Joseph	ENSERG
COUMES André	ENSERG
DARVE Félix	ENSHMG
DELLA-DORA Jean	ENSIMAG
DEPORTES Jacques	ENSPG
DOLMAZON Jean-Marc	ENSERG
DURAND Francis	ENSEEG
DURAND Jean-Louis	ENSIEG
FOGGIA Albert	ENSIEG
FONLUPTJean	ENSIMAG
FOULARD Claude	ENSIEG
GANDINI Alessandro	UFR PGP
GAUBERT Claude	ENSPG
GENTIL Pierre	ENSERG
GREVEN Hélène	IUFA
GUERIN Bernard	ENSERG
GUYOT Pierre	ENSEEG
IVANES Marcel	ENSIEG
JAUSSAUD Pierre	ENSIEG
JOUBERT Jean-Claude	ENSPG
JOURDAIN Geneviève	ENSIEG
LACOUME Jean-Louis	ENSIEG
LESIEUR Marcel	ENSHMG
LESPINARD Georges	ENSHMG
LONGEQUEUE Jean-Pierre	ENSPG
LOUCHET François	ENSIEG
MASSE Philippe	ENSIEG
MASSELOT Christian	ENSIEG
MAZARE Guy	ENSIMAG
MOREAU René	ENSHMG
MORET Roger	ENSIEG
MOSSIERE Jacques	ENSIMAG
OBLED Charles	ENSHMG
OZIL Patrick	ENSEEG
PARIAUD Jean-Charles	ENSEEG
PERRET René	ENSIEG
PERRET Robert	ENSIEG
PIAU Jean-Michel	ENSHMG
POUPOT Christian	ENSERG
RAMEAU Jean-Jacques	ENSEEG
RENAUD Maurice	UFR PGP
ROBERT André	UFR PGP
ROBERT François	ENSIMAG
SABONNADIERE Jean-Claude	ENSIEG
SAUCIER Gabrielle	ENSIMAG
SCHLENKER Claire	ENSPG
SCHLENKER Michel	ENSPG
SILVY Jacques	UFR PGP
SIRIEYS Pierre	ENSHMG
SOHM Jean-Claude	ENSEEG
SOLER Jean-Louis	ENSIMAG
SOUQUET Jean-Louis	ENSEEG
TROMPETTE Philippe	ENSHMG
VEILLON Gérard	ENSIMAG
ZADWORNY François	ENSERG

Professeur Université des Sciences Sociales (Grenoble II)

BOLLIET Louis

Personnes ayant obtenu le diplôme d'HABILITATION A DIRIGER DES RECHERCHES

BECKER Monique
BINDER Zdenek
CHASSERY Jean-Marc
CHOLLET Jean-Pierre
COEY John
COLINET Catherine
COMMAULT Christian
CORNUEJOLS Gérard
COULOMB Jean- Louis
DALARD Francis
DANES Florin
DEROO Daniel
DIARD Jean-Paul
DION Jean-Michel
DUGARD Luc
DURAND Madeleine
DURAND Robert
GALERIE Alain
GAUTHIER Jean-Paul
GENTIL Sylviane
GHIBAUDO Gérard
HAMAR Sylvaine
HAMAR Roger
LADET Pierre
LATOMBE Claudine
LE GORREC Bernard
MADAR Roland
MULLER Jean
NGUYEN TRONG Bernadette
PASTUREL Alain
PLA Fernand
ROUGER Jean
TCHUENTE Maurice
VINCENT Henri

Chercheurs du C.N.R.S
Directeurs de recherche 1ère Classe

CARRE René
FRUCHART Robert
HOPFINGER Emile
JORRAND Philippe
LANDAU Ioan
VACHAUD Georges
VERJUS Jean-Pierre

Directeurs de recherche 2ème Classe

ALEMANY Antoine
ALLIBERT Colette
ALLIBERT Michel
ANSARA Ibrahim
ARMAND Michel
BERNARD Claude
BINDER Gilbert
BONNET Roland
BORNARD Guy
CAILLET Marcel
CALMET Jacques
COURTOIS Bernard
DAVID René
DRIOLE Jean
ESCUDIER Pierre
EUSTATHOPOULOS Nicolas
GUELIN Pierre
JOUD Jean-Charles
KLEITZ Michel
KOFMAN Walter
KAMARINOS Georges
LEJEUNE Gérard
LE PROVOST Christian
MADAR Roland
MERMET Jean
MICHEL Jean-Marie
MUNIER Jacques
PIAU Monique
SENATEUR Jean-Pierre
SIFAKIS Joseph
SIMON Jean-Paul
SUERY Michel
TEODOSIU Christian
VAUCLIN Michel
WACK Bernard

Personnalités agrées à titre permanent à diriger des travaux de recherche (décision du conseil scientifique)

E.N.S.E.E.G

CHATILLON Christian
HAMMOU Abdelkader
MARTIN GARIN Régina
SARRAZIN Pierre
SIMON Jean-Paul

E.N.S.E.R.G

BOREL Joseph

E.N.S.I.E.G

DESCHIZEAUX Pierre
GLANGEAUD François
PERARD Jacques
REINISCH Raymond

E.N.S.H.G

ROWE Alain

E.N.S.I.M.A.G

COURTIN Jacques

E.F.P.

CHARUEL Robert

C.E.N.G

CADET Jean
COEURE Philippe
DELHAYE Jean-Marc
DUPUY Michel
JOUVE Hubert
NICOLAU Yvan
NIFENECKER Hervé
PERROUD Paul
PEUZIN Jean-Claude
TAIB Maurice
VINCENDON Marc

Laboratoires extérieurs

C.N.E.T

DEVINE Rodericq
GERBER Roland
MERCKEL Gérard
PAULEAU Yves

ECOLE NATIONALE SUPERIEURE DES MINES DE SAINT-ETIENNE

Directeur : Monsieur M.MERMET
Directeur des Etudes et de la formation: Monsieur J. LEVASSEUR
Directeur des recherches : Monsieur J. LEVY
Secrétaire Général : Mademoiselle M. CLERGUE

PROFESSEURS DE 1ère CATEGORIE

COINDE Alexandre	Gestion
GOUX Claude	Métallurgie
LEVY Jacques	Métallurgie
LOWYS Jean-Pierre	Physique
MATHON Albert	Gestion
RIEU Jean	Mécanique-Résistance des matériaux
SOUSTELLE Michel	Chimie
FORMERY Philippe	Mathématiques Appliquées

PROFESSEURS DE 2ème CATEGORIE

HABIB Michel	Informatique
PERRIN Michel	Géologie
VERCHERY Georges	Matériaux
TOUCHARD Bernard	Physique Industrielle

DIRECTEUR DE RECHERCHE

LESBATS Pierre	Métallurgie

MAITRE DE RECHERCHE

BISCONDI Michel	Métallurgie
DAVOINE Philippe	Géologie
FOURDEUX Angeline	Métallurgie
KOBYLANSKI André	Métallurgie
LALAUZE René	Chimie
LANCELOT Francis	Chimie
LE COZE Jean	Métallurgie
THEVENOT François	Chimie
TRAN MINH Canh	Chimie

Personalités habilitées à diriger des travaux de recherche

DRIVER Julian	Métallurgie
GUILHOT Bernard	Chimie
THOMAS Gérard	Chimie

Professeurs à l'UER de Sciences de Saint-Etienne

VERGNAUD Jean-Maurice	Chimie des Matériaux et Chimie Industrielle

Modélisation et Propagation de Contraintes Temporelles pour la Planification

J-F Rit

Résumé

Nous proposons une représentation du temps pour la planification, fondée sur la manipulations d'intervalles de la droite réelle : les occurrences. Nous abordons la définition et la résolution d'un **problème d'occurrences contraintes**. Celui-ci est posé en considérant un graphe dont les nœuds sont associés à des **domaines d'occurrences possibles** et les arcs à des relations temporelles symboliques, entre intervalles et disjonctives. Résoudre le problème, c'est éliminer, en réduisant les domaines, les occurrences rendues impossibles par les contraintes relationnelles. On peut le faire partiellement au moyen d'un algorithme de propagation de contraintes, dit de **dérivation locale**. On peut améliorer la résolution en exprimant le problème dans des formalismes connus. L'approche est illustrée par l'étude d'un problème d'affectation de systèmes d'armes à bord d'un navire.

Abstract

We propose a representation of time for scheduling, based on handling intervals of the real line: occurrences. In this representation, we attempt to solve a **constrained occurrences problem**, defined as a graph whose vertices are labelled with **domains of possible occurrences** and whose edges are labelled with disjonctive, inter-intervals, symbolic, temporal relations. Solving the problem is withdrawing from the domains every occurrence which is not consistent with the temporal relations. This is partially done by a so-called **local derivation** constraints propagation algorithm. More impossible occurrences can be withdrawn by translating the problem into other well-known formalisms. The approach is illustrated through the study of a ship-born weapon systems assignment problem.

Remerciements

Je remercie les membres du jury de me faire l'honneur de juger cette thèse :

- Monsieur Gérard Veillon, Professeur à l'Institut Polytechnique de Grenoble et Directeur de L'Ecole Nationale Supérieure d'Informatique et de Mathématiques Appliquées,
- Monsieur Malik Ghallab, Chargé de Recherche au CNRS,
- Monsieur Alain Quilliot, Professeur à l'Université Blaise Pascal
- Monsieur Jean-Claude Latombe, Professeur à l'Université de Stanford, qui m'a accueilli dans son équipe lorsqu'il était au LIFIA et qui a dirigé ce travail avec son allant caractéristique,
- Monsieur Yannick Descotte, Directeur du Département Recherches d'ITMI, qui m'a constamment guidé avec rigueur et pénétration de jugement.

Ce travail a été effectué dans le cadre d'une convention CIFRE avec la société ITMI, dont je remercie les dirigeants, J-C Latombe, puis G. Mezin et bien sûr Y. Descotte pour m'avoir fourni tous les moyens nécessaires à son achèvement.

Il a été conduit en partie, dans le cadre d'un contrat[1] avec la DRET, au Centre de Programmation de la Marine, dont je remercie les ingénieurs, P. Grojean et A. Gautier, pour leur collaboration, ainsi que ceux d'ITMI, A.Domissy et P.Fondaneche pour leur aide sur SAFRAN.

Cette recherche n'aurait pu être menée à bien hors d'un laboratoire, foyer privilégié de réception, d'élaboration et de diffusion des idées. Le LIFIA a rempli ce rôle. Que tous les membres de l'équipe IA, et tout particulièrement P.Caloud et T.Granier, les "temporels", en soient remerciés à travers A.Lux qui en est le reponsable.

Je remercie aussi ceux qui ont participé activement à l'élaboration du manuscrit, M.Pasquier, O.Raoult, D.Ziebelin, Y.Lagoude et ceux que j'ai déjà cités.

Enfin, je tiens à remercier, ceux qui m'ont aidés à supporter la pression intense du rush final.

J-F Rit

[1]Contrat 85510

Sommaire

Liste des Figures

Introduction

Position du Problème

Motivation

La conception et la gestion d'un grand projet, la conduite d'un atelier à l'échelle de la journée ou la détermination des actions d'une machine fortement autonome sont autant d'exemples justifiant l'emploi de la planification.

En effet, pouvoir, à l'avance, évaluer les effets conjugués des actions, est un atout certain pour maîtriser un environnement qui évolue, que ce soit sur un chantier de travaux publics ou dans un atelier de mécanique.

Par conséquent, il est utile de s'appuyer sur une représentation explicite du temps, pour concevoir, décrire et construire un système de planification. C'est une telle représentation que nous nous proposons de définir et d'étudier dans le présent mémoire.

Notre Représentation du Temps

Nous avons pris le parti de représenter le temps indépendamment de tout formalisme de planification. Nous cherchons en particulier à préjuger le moins possible de la manière dont on définit une action et dont on décrit ses effets.

Nous supposerons donc seulement la donnée de l'existence "d'objets" que nous appelons des événements, auxquels on associe une occurence qui constitue leur aspect temporel.

Très classiquement, l'occurrence dénote un intervalle de l'ensemble des réels. Nous l'interprétons comme l'ensemble des instants pendant lesquels l'événement se produit. On peut alors définir une relation temporelle entre deux événements à partir d'une relation de position entre les intervalles de leurs occurrences respectives. Par exemple, on peut considérer les événements `la pièce p est percée par la machine M`, `la machine M perce` et la relation `pendant` définie par l'inclusion de l'occurrence du premier événement dans celle du deuxième.

Nous associons, de plus, à chaque événement un domaine d'occurrences possibles. Celui-ci est défini de manière absolue (sans référence aux occurrences des autres événements)

comme un ensemble d'intervalles. Par exemple, on peut associer à l'événement `la machine M perce` le domaine des occurrences qui commencent après 14h00[2], qui finissent avant 16h00 et de durée (ou longueur) supérieure à trente minutes.

Une représentation est utile par ce qu'on peut en tirer. De ce point de vue, un raisonnement temporel consistera, pour nous, à éliminer explicitement des occurrences nécessairement impossibles. Plus précisément, étant donné un ensemble d'événements, liés par des relations temporelles et associés chacun à un domaine d'occurrences possibles, il s'agit d'éliminer explicitement de chaque domaines le maximum d'occurrences impossibles, compte-tenu des relations temporelles et des autres domaines.

Par exemple, si la machine M ne peut percer qu'entre 14h00 et 16h00, et si la pièce p ne peut-être percée par M qu'entre 13h00 et 15h00, alors[3] il est impossible qu'elle soit percée avant 14h00. Autrement dit, p ne peut être percée par M qu'entre 14h00 et 15h00.

Résoudre ce type de problème est à la fois intéressant et difficile. C'est intéressant car un problème de planification est un problème de décision : celui du choix parmi plusieurs actions possibles. Nous nous plaçons dans le cadre où la décision concernant l'existence de chaque action, plus généralement de chaque événement est prise. Il reste à savoir, dès lors, *quand* l'événement peut se produire. Les domaines d'occurrences constituent alors une information d'autant plus fiable que l'on ne connait que des occurrences possibles, mais aussi d'autant plus riche qu'on les connait toutes. En effet, étant donné que l'on se situe à un niveau tout à fait élémentaire du raisonnement temporel, il est prématuré d'éliminer d'autres occurrences que les occurrences nécessairement impossibles.

Le problème est cependant difficile lorsqu'on désire sortir de la classe des domaines d'occurrences à durée fixée et des relations temporelles de précédence. L'étude d'un tel élargissement constitue, sur le plan technique l'essentiel de notre travail.

Limites Conceptuelles de la Représentation

Nous supposons, dans toute notre approche, que la modélisation du temps que nous venons d'esquisser permet d'aborder des problèmes de planification sans hypothéquer le développement d'aspects plus riches du temps. Nous ne reviendrons pas sur ces aspects dans le présent mémoire, aussi les évoquons nous dès maintenant.

En considérant que les propriétés temporelles d'un événement, à savoir son occurrence, ne qualifient que son existence, on laisse en suspens une modélisation satisfaisante de l'évolution. Ainsi, on peut représenter facilement le fait qu'un robot mobile traverse un carrefour dans un certain intervalle de temps et interdit alors la présence d'un autre mobile sur ce carrefour. Mais on ne prévoit pas comment comparer des trajectoires, ce qui permettrait une analyse bien plus fine.

De même, on ne prévoit pas comment le système peut prendre en compte son propre temps

[2]Moyennant la définition d'un système horaire et d'un calendrier, tâche que nous n'abordons pas dans le cadre de ce travail.

[3]On sous-entend ici que les deux événements sont liés par la relation "pendant".

de réflexion. Cette activité, éminemment temporelle, pourrait être envisagée en considérant l'occurrence de l'événement "planifier"[4]. Là encore, nous laissons le problème en suspens dans la représentation, puisque nous ne tenons pas compte de la nature d'un événement.

Enfin, la notion de possibilité a aussi une sémantique minimale qui pourra être enrichie temporellement, en liaison avec la notion de présent qui implique l'introduction d'un sujet dans la représentation. En particulier, on peut distinguer les occurrences possibles d'un événement passé ("le dernier arrêt de la machine pour maintenance préventive") et celles d'un événement futur ("le prochain arrêt pour maintenance préventive"). Dans un cas, le "choix" d'une occurence parmi les possibles ne peut être qu'une déduction ou une supposition, dans l'autre, cela peut correspondre à une action.

Contribution et Plan du Mémoire

Le mémoire est divisé en trois parties constituées chacune de deux chapitres.

La première partie s'appuie sur une étude bibliographique pour dégager les concepts liés à la planification d'une part, et à la représentation du temps d'autre part. Elle montre qu'une représentation du temps indépendante du formalisme de planification, fondée sur la manipulation d'intervalles et la propagation de contraintes est une approche qui unifie beaucoup de travaux concernant la planification et le temps.

La deuxième, plus formelle, est la description de la représentation du temps que nous proposons. Sa principale contribution est d'introduire un degré de généralité qui nous permette de rassembler dans un même formalisme des manipulations numériques et l'expression de relations disjonctives. Son pouvoir de représentation est évalué à travers la comparaison avec des formalismes antérieurs.

La dernière partie revient au cadre plus général de la planification à travers l'étude d'un problème réel d'allocation de ressources : la planification de l'affectation de systèmes d'armes anti-aériennes sur un navire. Il nous permet de mettre en perspective le problème précis de représentation du temps que nous traitons par rapport à celui de la planification et du temps en général. Et il nous permet de montrer qu'au delà des considérations abstraites, la représentation que nous avons développée permet d'aborder un problème concret.

[4]et en évitant les pièges logiques de la récursivité.

Partie I

Planification et Représentation du Temps : l'Approche de l'Intelligence Artificielle

Chapitre 1

La Planification

1.1 Les Systèmes "Classiques" de Planification en IA

Introduction

La planification a depuis longtemps attiré l'attention des chercheurs en IA. C'est devenu un domaine "traditionnel", à tel point que tous les manuels et les ouvrages d'introduction à l'IA décrivent quelques systèmes célèbres et les principes qui les sous-tendent. Ces principes ont donc nourri la réflexion de tous les chercheurs qui ont abordé le domaine de la planification. Leurs travaux font systématiquement référence à ces systèmes, et le présent mémoire n'y échappe pas.

Nous ne donnerons pas dans le texte de références bibliographiques précises. Le lecteur pourra trouver dans [34] et [35] une analyse détaillée de la famille STRIPS. Une bibliographie commentée des articles fondamentaux est donnée dans un numero de SIGART Newsletter [9]. Tate [43] dresse un tableau thématique (suivant les problèmes abordés et les techniques employées pour les résoudre) complet, concis et à jour.

1.1.1 Fondements

Pour les systèmes classiques, un problème de planification consiste en la donnée :

- d'un modèle du monde
- d'un ensemble d'actions, opérateurs modifiant l'état du monde
- d'un état initial et d'un état final (ce dernier n'étant éventuellement que partiellement décrit).

L'activité de planification consiste à décider d'une suite d'actions destinée à faire passer le monde de l'état initial à un état final.

Les premiers travaux concernant la planification utilisent les techniques de démonstration automatique de théorèmes de logique du premier ordre. Le problème posé par la planification était de pouvoir introduire la notion (temporelle) de *changement* dans un tel formalisme. Le monde dans lequel évolue l'agent est alors décrit à l'aide de prédicats comportant une variable interprétable comme une *situation.* Par exemple le fait qu'une lampe l soit allumée dans la situation s s'écrira "Allumée(l,s)".

S'il donne un cadre formel et permet d'utiliser un mécanisme de déduction assez bien connu, le calcul situationnel a l'inconvénient d'être très lourd. En effet, la description et la manipulation de chaque situation, ou état, de l'univers est coûteuse. En particulier, il est nécessaire de noter explicitement *tout* ce qui change *et* tout ce qui ne change pas lors de l'application d'une action. Cette obligation (néfaste) a reçu le nom de *frame problem.* Elle est analogue à l'obligation, dans un dessin animé, de redessiner tout le décor chaque fois qu'un personnage bouge le petit doigt.

Dans le système STRIPS, on formalise l'idée, assez naturelle, de "n'écrire que ce qui change". Une action est définie par ses conditions d'applications et ses effets. Ces derniers sont décrits explicitement par des prédicats faux ou vrais immédiatement après l'exécution de l'action. Par exemple, l'action de prendre un objet A dans la pince du robot, invalide, pour la situation qui suit l'action, le prédicat `libre(A)` et rend vrai le prédicat `dans_pince(A)`. En supposant que *tous* les effets d'une action soient décrits ainsi, on impose[1] que ces effets aient une portée *limitée.* Jamais le planificateur n'envisagera qu'en prenant A il peut "bousculer" au passage les objets B et C, à moins de le définir explicitement dans l'action, ce qui ne fait que repousser un peu plus loin le problème.

Il faut bien noter ici, que le formalisme logique de STRIPS ne traite plus de la notion de situation. Il ne fonctionne que dans un état du monde donné. C'est un mécanisme différent qui fait évoluer le monde à travers l'application des actions.

1.1.2 Les Problèmes Abordés par les Planificateurs Classiques

Planification Hiérarchique

La planification hiérarchique est une stratégie conçue pour maîtriser la complexité de la planification, plus précisément pour s'affranchir de la "tyrannie du détail". Le principe le plus simple est celui du système ABSTRIPS qui commence par planifier dans un monde "de haut-niveau", simple, afin de cerner efficacement l'espace de recherche. Puis le plan est raffiné en intercalant des sous-buts plus détaillés, jusqu'au niveau le plus bas.

Les niveaux d'abstractions d'ABSTRIPS correspondent à une décomposition du domaine de planification, mais la décomposition peut s'appliquer au processus même de planification. Ainsi, dans le système MOLGEN, qui planifie des expériences de génétique, le niveau le plus bas est celui du domaine : le laboratoire ; les objets sont les gènes, les bactéries; les actions sont combiner, séparer...Au dessus, c'est le niveau de la conception du plan, les objets sont

[1]Ceci n'est pas spécifique à STRIPS, c'est vrai dans tous les systèmes classiques.

des contraintes, les actions sont propager, définir des sous-buts ...Au niveau le plus haut, se trouve le contrôle du plan, qui décide quand faire une hypothèse, ou quelles hypothèses défaire si on est bloqué.

Enfin, Cette structure en couches est aussi présente dans le modèle du *blackboard* appliqué à la planification. Le blackboard est une architecture générale, comportant plusieurs sources de connaissance résolvant un même problème. Les sources communiquent par l'intermédiaire d'un tableau noir (blackboard) où chacune lit les données qu'elle désire et inscrit les résultats qu'elle a trouvés. Le tableau noir décrit par B. Hayes-Roth et al. [24] comporte cinq niveaux de représentation. Le passage d'un niveau à un autre se fait par un mécanisme de postage de contraintes. Lorsqu'il se fait vers un niveau inférieur, il correspond à un raffinement du raisonnement et fait de ce système un système hiérarchique.

Les buts interdépendants

Un deuxième moyen de maîtriser la complexité de la planification est de décomposer le but en une conjonction de sous-buts que l'on cherche à atteindre séparément.

Dans STRIPS, en outre, la construction d'un plan se fait en empilant des actions dont les conditions d'application deviennent des buts intermédiaires (planification guidée par les buts). Lorsqu'il y a deux buts conjonctifs, le système cherche à les atteindre successsivement suivant un ordre déterminé a priori. Or, il est possible que ces buts doivent être atteints dans un certain ordre pour que la réalisation du second ne détruise pas le premier. C'est le cas par exemple pour "être propre", qu'on atteint en se lavant, et "avoir une voiture qui marche" qu'on atteint en la réparant...et en se salissant! Les deux buts sont alors dits *interdépendants.*

On peut éviter ce genre de situation en appliquant une stratégie de *moindre engagement*[2], c'est à dire en n'imposant pas d'ordre a priori sur deux buts conjonctifs. Le système NOAH fonctionne ainsi, en produisant un plan dit *non-linéaire* : lorsque deux sous-buts n'ont pas de raison d'être ordonnés a priori (par exemple "prendre(A) et prendre(B)"), ils sont développés séparément. Ensuite, un module de critique de plan analyse les éventuelles interactions nécessitant un ordonnancement.

1.1.3 Les Aspects Non-Traités Par les Planificateurs Classiques

Sacerdoti [37] énumère une série de problèmes peu ou pas abordés par les systèmes classiques de planification. Selon lui, leur résolution conditionne la réalisation d'un robot intelligent au sens de système capable de percevoir et d'agir de la manière la plus autonome possible sur son environnement :

- Tenir compte du temps

[2] A vrai dire, STRIPS peut en éviter certaines, en essayant de nouveau d'atteindre un but détruit. Dans le cas de notre exemple, une telle stratégie marche. Elle échoue cependant dans des cas un peu plus complexes où la réalisation d'un sous-but réclame l'application de plusieurs actions.

Tous les systèmes de planification décrivent un plan comme une succession d'actions. Cependant la durée des actions n'est pas représentée : tout se passe comme si l'univers passait instantanément de l'état antérieur à l'état postérieur à l'action. De même, la simultanéité, donc le parallélisme des actions, n'est pas représentée. Signalons à ce sujet que la non-linéarité d'un plan, dans un système tel que NOAH, ne représente pas forcément la possibilité de l'exécuter en parallèle, mais représente seulement l'indépendance *logique* de certaines branches du plan.

Un certain nombre de représentations du temps ont été proposées pour des planificateurs classiques, nous les évoquons à la section 1.3.

- Autoriser une exécution en parallèle

  L'aspect temporel du parallélisme a été évoqué ci-dessus. Pouvoir faire exécuter un plan à une série d'agents réclame en plus une formalisation des processus de communication.

- Planifier la collecte d'informations

  Les planificateurs classiques sont supposés omniscients, dans un monde où tout est connu et certain. La possession d'informations peut être un acte volontaire mobilisant l'agent, qui doit donc être planifié. La manipulation explicite de la connaissance d'un agent a été étudiée au moyen de la construction de logiques *epistémiques*. Ses relations avec l'action ont aussi fait l'objet d'investigations, mais on n'est allé guère au-delà de la formalisation. Le tableau qu'en dresse S.Rosenschein [36] constitue une introduction concise et claire sur ces travaux.

- Planifier la planification

  Le raisonnement sur l'activité de planification est parfois appelé *meta-planification*. Au terme de "meta", un peu galvaudé, nous préférons celui plus précis de "contrôle explicite de la planification" car il s'agit de manipuler explicitement la structure du plan. On en voit une ébauche dans MOLGEN et le système de blackboard, mais aucun système ne manipule vraiment le plan comme un objet formel.

- Apprendre des plans

  Un système de planification confronté deux fois à la même situation fera deux fois le même travail. Une extension de STRIPS permet au planificateur de stocker les plans déjà formés, mais on ne peut les réappliquer que dans les mêmes conditions exactement, ce qui est trop rare et trop long à vérifier.

- Planifier de manière interactive

  Si le planificateur doit pouvoir résoudre son problème seul, il peut être utile (voire indispensable dans le cas de systèmes "sensibles" forcément contrôlés par un humain) qu'une interaction ait lieu avec un humain lors du processus de résolution. Cet aspect encourage l'expression d'un contrôle explicite sur la planification, compatible avec les modes de représentation humains.

- Amender dynamiquement un plan

Tout plan est élaboré à partir d'une *prévision* de ce que sera l'état de l'univers au moment de l'application de chaque action et de la façon dont elle se déroulera. Or cette prévision peut être fausse ou peu précise, ce qui peut nécessiter une replanification. Décider quand tout replanifier et quand (puis comment) changer localement le plan sont des problèmes clefs de la réussite de l'exécution. L'étude de l'interdépendance entre la planification et l'exécution a donné lieu a la construction de systèmes dits de *planification réactive*. Ce sont eux que nous allons étudier dans la section suivante.

1.2 Planification et Réactivité

1.2.1 Une Nouvelle Problématique de la Planification

Nous avons décrit les planificateurs classiques en considérant la planification comme un problème autonome. Or la planification ne prend son sens que dans l'action et n'est qu'une des fonctions d'un système interagissant avec le monde. Lorsqu'on aborde de front la construction d'un tel système, chacun des problèmes soulevés par la modélisation, la gestion et le traitement de la perception du monde et des actions du système apparaissent si difficiles et si fondamentaux qu'ils semblent invalider d'office tout cadre conceptuel qui ne les prendrait pas en compte de manière prioritaire.

Envisager le problème abstrait — c'est-à-dire déconnecté du cadre de l'exécution — de la planification est donc une approche très réductrice. Par conséquent, les limitations de l'approche classique ont conduit un certain nombre de chercheurs à aborder la planification comme fonction d'un système réagissant avant tout, *sur* et *avec* son environnement. Cette approche porte en général le nom de *planification réactive*[3]. Nous allons évoquer dans cette partie les travaux qui s'y rapportent.

A la section précédente, nous avons esquissé un certain nombre d'aspects fondamentaux mis en évidence par les systèmes classiques. Ces aspects sont présents dans tout problème de planification. On peut même dire qu'ils sont caractéristiques de la planification. Il ne faut donc pas les perdre de vue lorsqu'on se place dans le contexte de la planification réactive.

1.2.2 Systèmes Réactifs et Systèmes de Planification

Il est hors de notre propos de donner ici une définition formelle d'un système réactif ou d'un système de planification. Un tel cadre serait trop rigide pour une analyse bibliographique. Nous donnerons seulement quelques éléments qui permettent, selon nous de caractériser les capacités de réactivité et de planification.

[3]Les travaux sur la planification réactive étant relativement récents, la terminologie que nous employons n'est pas unifiée, on rencontre parfois le terme de *planification opportuniste* avec le sens que nous donnons à planification réactive, mais aussi, parfois, avec un sens différent comme nous le verrons dans la suite.

Ce qu'il faut pour être réactif

Nous venons de souligner que le fondement de la réactivité est la capacité de décider et d'exécuter des actions dans un environnement réel, que n'ont pas les planificateurs classiques. La question de leur réactivité se pose donc seulement si on les "connecte" à un système d'exécution opérant dans le monde réel. Sont-ils alors réactifs ? Supposons qu'un expérimentateur retire périodiquement un cube au sommet d'une des piles construites par STRIPS. Si il retire les cubes à intervalles de temps suffisamment longs, STRIPS pourra recalculer un plan valide après chacune de ses actions. Sinon il sera incapable d'atteindre son but. On a là un premier degré de réactivité, qu'on peut définir objectivement comme la capacité pour un système à obtenir des sorties désirées en fonction d'entrées données.

Du point de vue de la conception de systèmes, ce critère, pour être parfaitement défendable, nous intéresse peu. Nous préférons réserver l'appellation de "réactifs" à des sytèmes qui décident rapidement d'actions grâce à d'autres moyens que la force brute. En particulier ceux qui peuvent compléter, modifier et suspendre un plan lors de son exécution. Les planificateurs classiques qui planifient complètement jusqu'au détail le plus fin[4] que leur modèle du monde le leur permet ne sont pas, dans ce sens, réactifs, même s'ils sont connectés à un système d'exécution.

Il existe une compétition fondamentale entre la possibilité de planifier (plus généralement de prévoir) loin et précisément et celle de fonder ses prédictions sur des données à jour. Le "summum" de la réactivité est pour nous la capacité à gérer dynamiquement le compromis entre ces deux tendances. Ceci suppose bien sûr une architecture de contrôle adéquate et, du point de vue de la planification, une connaissance approfondie de la valeur du résultat que donne un planificateur en fonction des ressources qu'on lui donne. Nous ne connaissons pas, à ce jour, de système doté d'une telle capacité de réactivite'.

Ce qu'il faut pour être un planificateur

Un plan est une représentation, formelle, d'un ensemble d'actions et de descriptions partielles de l'environnement. Cette représentation est structurée en fonction de descriptions prédites du système et de son environnement. Le système est capable de planifier ses actions s'il peut construire et structurer un plan. Nous insistons sur le fait qu'il y a en général *plusieurs* actions, qu'elles sont la plupart du temps *interdépendantes* et reliées à une *prédiction* de leurs effets sur l'environnemnent.

Ainsi, STRIPS est un système doté d'une capacité de planification. Un automate qui pilote une boucle de régulation n'est pas doté de capacité de planification, car le plan trivial à une action n'a pas vraiment de structure intéressante. Nous considérons qu'un système exécutant des actions en fonction d'un arbre de décision n'est pas non plus doté d'une capacité de planification. Il manipule bien un ensemble d'actions, précise même la structure du plan au fur et à mesure de l'exécution, mais ces manipulations se font sans aucune prédiction. Un

[4]Les planificateurs hiérarchiques classiques décomposent leurs actions au moment de la planification. Ils planifient donc aussi jusqu'au détail le plus fin.

système qui n'évalue pas les conséquences de ses actions au-delà de leurs effets immédiats n'est pas capable de planifier.

Systèmes de "planification réactive"

Nous avons choisi de décrire trois grandes classes de systèmes alliant dans une certaine mesure réactivité et planification[5] :

1. Les systèmes planificateur-exécuteur : tour à tour dotés d'une capacité de planification et d'une capacité de réaction, ils consistent en deux modules distincts (le planificateur et l'exécuteur) appelés séquentiellement pour une tâche donnée. Ces modules sont appelés rarement, si possible une seule fois ; seul un "gros" problème provoque l'appel du système de planification lors de l'exécution. Historiquement ce sont les plus anciens, les planificateurs classiques ont été généralement conçus dans le cadre d'une telle architecture. Un tel découpage est encore largement utilisé pour sa simplicité et son efficacité, en particulier pour la planification et la conduite de tâches d'assemblage robotisées.

2. Les systèmes réactifs permettant une planification : contrairement aux précédents, ils peuvent interrompre aisément une exécution et activer un ou plusieurs autres plans. Ces plans sont prédéfinis et sont typiquement des procédures, d'où le nom de systèmes procéduraux qu'ont leur donne parfois. Le "calcul" d'un plan consiste en un empilement opportuniste de procédures, sans que les interactions des effets de ces procédures soient calculées. C'est pourquoi nous disons qu'ils *permettent* seulement la planification, moyennant l'introduction d'un mécanisme d'évaluation des interactions entre procédures. Ce sont les plus récents historiquement.

3. Les systèmes de planification permettant la réactivité : ils sont dotés d'une architecture assez souple pour ne pas tout replanifier en cas d'échec de l'exécution, ou pour adapter leur stratégie de planification aux données d'un problème précis. Cependant, ils n'est pas prévu de manière explicite de mécanismes de perception ou d'interruption. C'est pourquoi ils *permettent* seulement la réactivité moyennant l'introduction de ces mécanismes.

1.2.3 Systèmes Planificateur–Exécuteur

STRIPS, système classique conçu pour une application dans le domaine de la robotique, n'est pas du tout réactif. Cependant, il n'a pas été construit pour piloter un robot. Pour ce faire, on lui adjoint le système PLANEX [33] qui exécute le plan fourni par STRIPS et vérifie après chaque action que la suivante est exécutable. Pour cela, au lieu de ne prendre en compte qu'une liste d'actions, il utilise une structure de données appelée *table triangulaire*, qui code les actions avec leurs préconditions et leurs effets. Plutôt que de recalculer le plan après

[5]Une description plus détaillée et les références sont données dans les sections suivantes.

chaque action, grâce à cette table, PLANEX peut se replacer à un bon endroit, antérieur ou postérieur au point courant. Ainsi, le plan linéaire calculé par STRIPS est transformé en un graphe complet non-orienté dont chaque nœud est une action et que l'on parcourt grâce à la table. Lorsqu'aucun endroit ne convient, STRIPS est rappelé et recalcule un nouveau plan.

B.R.Fox et K.G.Kempf [16] proposent aussi un système composé d'un planificateur hors-ligne et d'un exécuteur. Il est appliqué à un assemblage effectué par un robot : le planificateur produit un ensemble de gammes d'assemblage qui sont ensuite exécutées. La présence des pièces dans un vrac justifie le besoin de réactivité. Cependant, au lieu —comme dans STRIPS-PLANEX— de calculer un plan linéaire puis d'en déduire une structure moins contrainte (que constitue la table triangulaire), leur système calcule directement un ensemble d'actions le "moins contraint possible" (notion assez vaguement définie). Cet ensemble est structuré par une relation d'ordre partiel et des contraintes disjonctives et conjonctives. On a ainsi un ensemble d'ordres partiels qu'on appelle des *plans partiels* ou plans non-linéaires (cf section 1.1.2). Le nombre de ces plans est en général beaucoup trop élevé pour que le système d'exécution puisse les manipuler. Le système de Fox et Kempf élimine alors les plans qui contiennent des redondances et applique des critères heuristiques prenant en compte la topologie des plans (par exemple, on préfère ceux qui contiennent beaucoup de plans linéaires) et des connaissances dépendant du domaine (par exemple, la stabilité du montage). Ces plans sont communiqués au système d'exécution qui choisit parmi les racines des plans partiels une action que les conditions courantes permettent d'effectuer, puis met à jour tous les plans partiels. Ce travail de choix et de mise à jour est bien sûr d'autant plus coûteux que le nombre de plans linéaires sous-jacents est grand, c'est-à-dire que la "marge de réactivité" est grande. Le problème de réactivité est ainsi ramené à la détermination (laissée à la charge du concepteur) d'une bonne répartition de la complexité du calcul entre le module de planification et celui d'exécution.

1.2.4 Systèmes Réactifs Permettant la Planification

Nous qualifions ces systèmes de *procéduraux*, car ils appliquent des procédures plus qu'ils ne planifient. Ils sont ainsi plus centrés sur l'exécution que les précédents. En effet, pour les systèmes planificateur-exécuteur, la partie la plus difficile est la construction d'un plan à partir d'actions primitives que l'exécuteur exploitera afin d'atteindre, au sens large, un seul but.

Les systèmes procéduraux, eux, disposent déjà d'une "bibliothèque" de plans qui permettent d'atteindre des buts de natures différentes. On invoque ces plans lorsque le besoin s'en fait sentir, plusieurs plans pouvant être actifs en même temps. C'est dans le choix puis la coordination de ces plans que réside la tâche du système.

R.J. Firby [14] propose un modèle simple d'un tel système. La bibliothèque est composée de RAPs (*Reactive Action Packages*) étiquetés chacun par un but et contenant plusieurs plans partiellement ordonnés permettant d'atteindre ce but. Ces plans sont eux-mêmes composés d'actions primitives et de RAPs. Le système est alors organisé autour d'une boucle qui :

- choisit un RAP dans la queue de l'ensemble des RAPs en cours d'exécution,
- vérifie que le but qui lui correspond n'est pas déjà atteint et qu'on désire toujours l'exécution du RAP,
- choisit un plan dans le RAP,
- place les actions et les RAPs du plan dans la queue, puis
- replace le RAP sélectionné au bout de la queue de manière à vérifier ultérieurement que le but a bien été atteint.

Une structure analogue, mais plus complexe, est proposée par M. Georgeff et A. Lansky [22]. C'est une application à la planification réactive des *systèmes experts procéduraux*, lesquels sont destinés à résoudre des problèmes de diagnostic en utilisant des connaissances procédurales [21]. Le rôle des RAPs est ici rempli par des KA (*Knowledge Areas*) qui contiennent la structure d'un plan unique, mais plus complexe qu'un ordre partiel sur un ensemble d'actions.

En effet, les arcs du graphe sous-jacent à une KA ne forment pas forcément un arbre. Ceci permet de représenter des boucles, des branches conditionnelles et des parties récursives. De plus, ces KAs peuvent être invoquées non seulement pour atteindre un but, mais aussi en fonction d'un état courant, ce qui correspond à des "actions reflexes". Enfin, alors que Firby donne aux actions une sémantique liée à la commande d'une machine (réelle), Georgeff leur donne un sens plus général, les actions pouvant modifier la représentation interne du monde et exercer un contrôle sur le raisonnement lui-même.

Cette dernière caractéristique en fait un système plus puissant qu'un simple exécuteur réactif de plans. Nous l'avons cependant décrit dans cette section car il n'y a pas de prévision des conséquences des actions. En effet, l'un et l'autre des systèmes[6] sont purement réactifs dans le sens où il n'y a pas de coordination des plans, les interactions entre ces derniers ne sont évaluées qu'au moment de l'exécution. Ainsi, étant donné un état instantané de la pile des actions, on est incapable de prédire le comportement du système. Ceci n'est pas étonnant. Le problème abordé par les planificateurs classiques étant déjà difficile lorsqu'on cherche à assembler des actions élémentaires, il paraît naturel qu'il le soit encore plus lorsqu'on veut assembler des structures plus riches, surtout si elles sont conditionnelles, bouclées et récursives!

1.2.5 Les Systèmes de Planification Permettant la Réactivité

Ces systèmes sont beaucoup plus orientés vers la planification que vers l'exécution. Simplement, leur architecture *hétérogène* leur permet de tenir éventuellement compte de contraintes apparaissant en cours d'exécution sans recommencer entièrement leur planification.

[6] Firby l'exprime clairement. Il est possible, selon Georgeff, de programmer une procédure qui calcule les interactions entre plans, on n'en sait pas plus.

Le premier système — qualifié par son auteur d'*opportuniste* — est celui de B.Hayes-Roth et al. [24], que nous avons évoqué à la section 1.1.2. L'architecture de "blackboard" permet bien d'avoir une influence retro-active de l'exécution sur la planification, dans la mesure où cette planification peut intégrer des éléments de tous les niveaux. Par conséquent, la "promenade en ville" n'est pas complètement déterminée avant qu'elle ne soit commencée mais pourrait être planifiée au fur et à mesure de l'exécution. Il faut préciser ici que par "opportuniste", on n'entend pas nécessairement "capable d'utiliser les événements qui se présentent pour atteindre le but", mais plus généralement "capable d'utiliser toute espèce d'information (une déduction de haut-niveau comme un fait réel) pour construire le plan".

Dans ce sens, M.Stefik [40] revendique aussi l'appellation d'opportuniste, et avec raison, car son mécanisme d'envoi de contraintes permet de propager tout type d'information sous forme d'une contrainte sur l'ébauche de solution en cours. Un mécanisme analogue d'envoi et de propagation de contraintes fait d'OPIS [25], un système d'ordonnancement d'atelier, un système opportuniste (propriété héritée en fait du système ISIS). Cependant, l'appellation est plutôt justifiée, selon ses concepteurs, par la capacité d'OPIS à adapter la *stratégie* de planification aux caractéristiques du problème particulier qui est posé, selon que l'ordonnancement est plutôt contraint par l'occupation des ressources ou la satisfaction des tâches. En outre, l'ordonnancement du futur "lointain" n'est jamais complètement spécifié, cela coûte trop cher et risque trop de devenir périmé. Au lieu de cela, il peut être contraint au fur et à mesure de l'exécution du plan par d'éventuels événements inattendus.

1.2.6 Les Systèmes Prenant en Compte une Evolution

Parmi tous les systèmes que nous avons décrits, aucun ne détecte de conflits dus à l'environnement. En effet, il n'y a pas *pour le planificateur* une représentation explicite de l'évolution indépendante de l'agent, à mettre en correspondance avec une éventuelle observation de l'évolution réelle. Si la prédiction des effets d'un plan tient seulement compte des effets immédiats dus aux seules actions, cette prédiction est incomplète et la planification est, dans une certaine mesure, aveugle.

Les systèmes que nous décrivons dans cette section tiennent compte d'une évolution parallèle à leurs actions et la comparent véritablement avec la prédiction qui a servi à l'élaboration de leurs plans. Thorndyke [44] et Wesson [48] ont proposé de tels systèmes dans le cadre du contrôle aérien. L'environnement, simulé, est constitué des avions, caractérisés par leur position et leur vitesse. Le but de la planification du contrôleur est que ces avions atteignent leurs aéroports respectifs sans rentrer en collision. La route d'un avion est une succession de manœuvres (changement de pallier, changement de cap) assez faciles à modéliser sous forme d'actions. De plus, les "situations dangereuses" sont assez bien définies par les divers réglements aériens.

Le planificateur extrapole toutes les dix minutes les plans de vols sur les vingt minutes qui viennent. C'est à dire que toutes les dix minutes, la description interne de l'univers est remplacée par la description "réelle". En cas de conflit de routes, il propose des "patches", ou modifications locales, aux plans des avions en instance (virtuelle) de collision. La différence

entre le système de Wesson et celui de Thorndyke est que le dernier est un système distribué : chaque avion construit son propre plan. Thorndyke a testé plusieurs cas de figure suivant que les acteurs sont plus ou moins autonomes et communiquent plus ou moins. Ses résultats confirment l'idée naturelle que moins l'espace de solutions est dense (plus il est difficile de trouver une solution), plus le coût de la planification centralisée et des communications est justifié. Signalons ici que dans ce cas, le système n'a pas un degré de réactivité très élevé : la simulation "s'arrête" lorsque le planificateur se met en marche, ce qui signifie que la planification prend le temps qu'il faut.

Ceci est aussi évident dans le système de Newman et Kempf [32] qui planifie l'emploi du temps d'un robot alimentant les machines d'un atelier. Pendant que le robot exécute une séquence d'actions, le système planifie "faster than real-time" (sic) la séquence d'actions suivante. Mais si cette séquence n'est pas applicable (car calculée à partir de données trop vieilles), le robot attend la suite.

Le système AIRPLAN [30], qui planifie les apontages sur un porte-avion, ne suppose pas que *toute* la séquence de planification se fait à une vitesse infinie (ou "suffisante"). La détection et la résolution de conflits sont interruptibles par l'acquisition de données et la détection de conflits immédiats, qui doivent toutes deux se produire à une vitesse suffisante.

Il faut noter que ces (rares) systèmes prenant en compte une évolution modélisent des phénomènes très simples (par exemple les trajectoires rectilignes uniformes des avions). De plus, l'éventuelle prise en compte du temps de planification est extrêmement rigide et statique : la période et l'horizon sont fixés (sauf pour AIRPLAN[7]).

1.3 Planification et Représentation Explicite du Temps

Comme nous l'avons dit dans la section 1.1 les systèmes classiques de planification ne représentent pas le temps de manière explicite. En effet, toutes les propriétés temporelles d'un plan engendré par un de ces systèmes sont contenues implicitement dans l'ordre d'application des actions. En outre, l'ordre partiel sous-tendu par un plan non-linéaire dénote une indétermination du processus de résolution, et non l'exécution concurrente de branches parallèles du plan.

Cependant un certain nombre de systèmes de planification comportent une représentation explicite du temps. L'objet de cette section est d'en dresser un panorama. Sur la bonne dizaine d'articles publiés en la matière, nous avons choisi 7 systèmes qui couvrent presque tous les aspects abordés dans le domaine : FORBIN de D.Miller et al. [28], DEVISER de S.Vere [45,46], une extension de NNS par M.Ghallab et al. [18], PTWM (planning in a temporal world model) de J.Allen et J.A.Koomen [4], NUDGE d'I.Goldstein [20], NAVAL de H.Delesalle et Y.Descotte [13], OPIS de S.Smith et al. [25,26], NONLIN de A.Tate [42] . Plutôt que d'en donner une description séquentielle, nous en dégagerons les caractéristiques sur trois points :

[7]Encore que la substance bibliographique, extrêmement faible, concernant ce système ne nous permet que des suppositions.

- la nature du problème abordé et le domaine d'application,
- le formalisme de planification et la manière dont il intègre des objets temporels,
- la représentations du temps elle-même.

1.3.1 Nature du Problème Abordé, Domaines d'Application.

Certains systèmes (PTWM, NONLIN) n'ont pas de domaine d'application précis et fonctionnent dans un monde jouet, mais le principal domaine d'application (FORBIN, NNS, OPIS) est celui de l'ordonnancement de tâches et de l'allocation de ressources dans un atelier.

La nature du problème de planification n'est cependant pas toujours la même. On peut dégager deux tendances : les problèmes de séquencement d'actions et ceux d'allocation de ressources. Les deux aspects sont liés et présents dans tous les systèmes, mais en général l'un domine l'autre.

Problèmes de Séquencement

Il s'agit en général de trouver et de combiner des actions primitives de manière à atteindre un but "explicite" (contrairement, par exemple, à un but faisant intervenir un critère d'optimalité), la difficulté résidant dans le grand nombre d'enchaînements d'actions possibles a priori. Un domaine caractéristique de tels systèmes est la robotique d'assemblage. Les primitives y sont les actions de montage et le but une pièce terminée. La dimension temporelle de tels problèmes est plus dans la richesse de la description des actions et de leurs relations (il s'agit d'aller plus loin que la simple séquentialité et permettre, par exemple, la représentation d'actions concurrentes) que dans l'utilisation optimale d'un temps alloué au montage.

C'est l'approche adoptée par NNS qui planifie et fait exécuter par des robots des blocs d'une dizaine d'actions en utilisant des primitives définies hors-ligne auparavant.

On peut étendre cette caractéristique à des domaines autres que la planification dans un atelier. PTWM n'a pas de domaine d'application a priori, mais le monde des blocs utilisé dans les exemples, et le type de problèmes qu'il souleve sont plutôt sequentiels. DEVISER planifie les opérations nécessaires à la prise et l'enregistrement de vues depuis un satellite, ce qui en fait plutôt un système de séquencement, même s'il doit aussi gérer les contraintes de durée posées par la date du rendez-vous.

Problèmes d'Allocation

Dans ce type de problèmes, plutôt que de combiner des actions en fonction de contraintes fortes ("obligatoires"), il s'agit de bien utiliser les ressources disponibles pour des actions déjà trouvées en général. Ce sont donc de problèmes d'optimisation, rendus difficiles par l'absence d'un critère bien défini et la non-convexité de l'espace de recherche.

Un domaine d'application abordé est la répartition de tâches sur les machines d'un atelier dans le cadre d'une fabrication par lots. Un exemple caractéristique de système résolvant ce genre de problème est OPIS qui planifie l'usinage de lames de turbines en fonction notamment des priorités accordées à chaque lot, des dates limites de fabrication et de l'occupation des machines. De même, mais dans le cadre d'un atelier jouet, FORBIN calcule l'emploi du temps d'un robot mobile qui alimente deux machines.

En dehors du cadre de l'atelier, NAVAL planifie l'allocation d'engins à des sites de forage en mer. NUDGE gère les emplois du temps d'un ensemble de personnes.

1.3.2 Le Formalisme de Planification

Les formalismes de planification employés sont en général empruntés à des systèmes "atemporels", auxquels on ajoute des éléments de description supplémentaires, utilisés par un module temporel. On peut distinguer deux familles d'approches : l'utilisation de planificateurs "classiques" et l'utilisation de planificateurs fondés sur la propagation de contraintes.

Systèmes Fondés sur un Planificateur Classique

Ces systèmes ont une démarche similaire à celle de NOAH : chaque clause d'une conjonction de buts (un but est décrit par une proposition logique) est décomposée en actions de manière indépendante. Les interactions entre les préconditions et les effets des actions sont ensuite évaluées, et les actions ordonnées de manière à ce qu'une branche du plan ne détruise pas la précondition d'une action sur autre branche.

La différence avec NOAH est qu'à chaque prédicat intervenant dans la description des actions, on associe un intervalle de validité temporelle (les modalités exactes dépendent de la représentation du temps de chaque système). Les dépendances entre les propositions sont traduites dans un formalisme temporel (par exemple : deux intervalles associés au même prédicat sont égaux ou disjoints, "libre(A)" et "prendre(A)" sont consécutifs). L'analyse des interdépendances des actions fait donc appel, parfois exclusivement, à un module temporel qui en déduit des relations structurant le plan de manière plus riche que le simple ordre partiel calculé par NOAH.

PWTM et NNS confient toutes les vérifications de cohérence au module temporel. NONLIN a un fonctionnement classique et vérifie simplement que la durée totale de son plan est inférieure à une limite fixée ; seules les actions ont donc des attributs temporels. DEVISER a une approche mixte : il utilise le formalisme de NONLIN, mais des intervalles de temps sont aussi associés aux prédicats, et ceci de manière figée par rapport à l'action (les préconditions "contiennent" toujours l'action, les conséquences sont immédiatement consécutives) contrairement à PTWM et NNS qui définissent ce genre de dépendance pour chaque action.

Formalismes Fondés sur la Propagation de Contraintes

Il est beaucoup plus difficile de les décrire de manière synthétique que les précédents. En effet, OPIS, NUDGE et NAVAL n'ont pas de filiation commune avec un système classique. NUDGE et OPIS sont construits sur des systèmes à base de *frames* qui sont des structures hiérarchiques centrées-objet communiquant par envoi de messages, les principaux objets étant les actions (ou tâches). NAVAL a une approche plus hybride. Il travaille sur une représentation de plan qui est ausssi centrée sur des tâches décomposables, mais toute modification de cette représentation passe par l'intermédiaire d'une base de règles (utilisant donc un formalisme proche de la logique du premier ordre).

Dans tous ces systèmes, seules les tâches (et pas les prédicats) ont des attributs temporels.

La fonction de la propagation de contraintes proprement dite est d'assurer la cohérence du plan au fur et à mesure de sa construction. Naturellement, il faut aussi assurer la cohérence temporelle. Cependant, les contraintes non-temporelles correspondent en général à des choix dans un ensemble discret (par exemple l'ensemble des machines sur lesquelles on peut faire une tâche donnée) alors que les contraintes temporelles peuvent avoir un domaine continu (par exemple les différents moments auxquels on peut commencer une tâche). L'introduction d'attributs temporels va donc en général de pair avec celle d'un module temporel qui manipule au moins ces contraintes continues (ou nombreuses si on discrétise le domaine).

NAVAL laisse au mécanisme général la gestion des alternatives temporelles d'ordonnancement, et maintient une représentation spéciale des intervalles numériques de laquelle toutes les relations disjonctives (comme "avant ou apres") sont éliminées. NUDGE a un module temporel appelé BARGAIN qui gère à la fois les occurrences numériques et les décisions d'ordonnancement.

1.3.3 Formalismes de Représentation du Temps

Nous traitons en détail cet aspect dans la suite. Cette section donne donc une vue assez synthétique de la question.

Nous avons souligné à la section précédente qu'une représentation du temps était liée au maintient de la cohérence. En effet, il ne suffit pas de construire un système d'indexation temporel, attribuant une date ou un intervalle à chaque objet, même s'il possède la sophistication d'un calendrier avec anneés, mois, jours et heures. Encore faut-il assurer la cohérence des informations représentées avec un minimum d'efficacité, en s'appuyant sur la structure du formalisme de représentation du temps. De ce point de vue, NUDGE n'a pas une représentation du temps intéressante. En effet, sa stratégie de résolution est de produire des plans totalement instanciés (c'est-à-dire que toutes les variables se voient attribuer une valeur) puis de les modifier de manière à les rendre acceptables (en particulier cohérents). Du point de vue temporel, le nombre de plans produits en changeant simplement les occurrences des activités est potentiellement extrêmement grand. Seul le domaine d'application permet un nombre d'occurrences possibles par activité assez faible (on ne planifie pas sa semaine de travail en la morcelant en secondes), ce qui permet l'énumération. Notons aussi que les

tentatives de représentation du temps dans les bases de données [41] ne semblent pas avoir franchi ce cap.

Comme nous l'avons évoqué à la section précédente, l'objet temporel commun à tous les systèmes est l'intervalle, associé suivant les cas à un prédicat ou à une action. Il y a plusieurs manières de représenter ces intervalles. Parmi les systèmes exposés, deux critères semblent discriminants : d'une part la représentation peut être numérique ou symbolique, d'autre part elle peut manipuler des points ou directement des intervalles.

Les Systèmes Numériques

Ces systèmes utilisent des nombres dans la description d'un intervalle temporel. La représentation temporelle peut utiliser d'autres symboles, dénotant en particulier un relation entre deux intervalles. Cependant, le maintien de la cohérence temporelle passe toujours par la modification de nombres.

OPIS et DEVISER considèrent les intervalles comme objets primitifs dont les domaines de variation sont des fenêtres temporelles. Ce sont les bornes, numériques, de ces fenêtres qui sont modifiées en fonction des autres fenêtres et des relations qui les lient. Notons qu'un autre paramètre intervient dans la représentation : la durée de chaque intervalle qui est, dans ces systèmes, un nombre fixé.

FORBIN, NAVAL et NONLIN manipulent aussi des intervalles, mais la représentation temporelle les décompose en points (début et fin). Ces points sont les nœuds d'un graphe dont les arcs sont calculés à partir des relations entre intervalles et valués numériquement. La cohérence de la représentation est assurée en changeant les valeurs des arcs par un mécanisme analogue à un algorithme PERT.

Les Systèmes Symboliques

Ces systèmes ne manipulent pas de nombre. Les intervalles ou les points sont représentés par des symboles. Le maintien de la cohérence temporelle est assuré en modifiant les relations (elles-mêmes symboliques) entre les intervalles ou les points.

Dans PTWM, les objets de base sont des intervalles. Les relations sont les treize relations possibles entre deux intervalles sur une droite (avant, après, pendant etc.). La cohérence est maintenue grâce à une table de transitivité (inférant, par exemple, que si A est pendant B et B est pendant C, A est pendant C) et un algorithme de fermeture transitive (voir le chapitre suivant pour plus de détails).

Dans NNS, les relations entre intervalles sont traduites au niveau ponctuel, chaque action se décomposant, sur le plan temporel, par rapport à quelques points (typiquement trois à cinq). Le maintien de la cohérence est assuré au moyen d'un treillis, représentant un ordre partiel entre tous les points, mis à jour lors de l'introduction de chaque action.

Chapitre 2

La Représentation du Temps

2.1 Haut-Niveau et Bas-Niveau

O.Costa de Beauregard [10] distingue deux aspects fondamentaux du temps. Le temps *subjectif* et le temps *objectif*. Ces deux termes ne sont pas très explicites, et même si nous y perdons en élégance stylistique, nous nous contenterons des termes *haut-niveau* et *bas-niveau*. En effet, la première notion recouvre l'aspect causal, irréversible des phénomènes inscrits dans le temps. Les chercheurs en IA ont abordé cet aspect sous un angle formel, essayant de "capturer" dans une logique la nature du temps. Le deuxième aspect est plus descriptif. O.Costa de Beauregard le divise en trois notions : succession, simultanéité et durée. Cette section dresse un rapide aperçu bibliographique des représentations du temps de haut niveau (2.1.1) et donne un cadre pour les représentations de bas-niveau fortement liées aux techniques de propagations de contraintes (2.1.2). Comme c'est dans ce dernier cadre que se situe l'essentiel de notre travail concernant la représentation du temps, nous le détaillerons dans deux sections ultérieures concernant la propagation de contraintes (2.2) et les représentation du temps de bas-niveau proprement dites (2.3).

Précisons ici que nous limiterons le cadre de notre analyse aux représentations liées à la planification, en particulier, nous n'aborderons pas les problèmes de description et de manipulation de l'évolution, ni les logiques temporelles dédiées à la représentation de programmes.

2.1.1 Représentations de Haut-Niveau

Le but des représentations de haut-niveau est de formaliser le "raisonnement temporel" dans son sens le plus noble. Chacune s'exprime dans le cadre d'une logique, et leur portée est loin d'être opérationnelle. Nous sommes ici dans un domaine de recherche fondamentale. Deux formalismes ont eu quelque retentissement en IA, nous les présentons ici en nous inspirant des travaux de Y.Shoham [38].

Tout d'abord, ces deux formalismes sont fondés sur des logiques dites *"réifiées"* : ainsi, si

l'on veut dire que la couleur d'une certaine maison est rouge à l'instant t, on écrira `VRAI(t, couleur (maison, rouge))` associant ainsi la proposition reifiée (dans la mesure ou elle est devenue un objet élémentaire dans le cadre de ce formalisme) `couleur (maison, rouge)` à l'instant `t`. Cette séparation syntaxique permet de donner un statut spécial au temps, contrairement à la notation `couleur (maison, rouge, t)`, qui fait de t un paramètre comme les autres.

La Logique temporelle de McDermott

Pour D.McDermott [31], la signification d'un instant est "photographie instantanée de l'univers". A partir de là, la sémantique de sa logique temporelle est définie de manière ensembliste. Un fait est défini par l'ensemble des instants pendant lesquels ce fait est vrai. Ainsi, écrire `Vrai(t, fait)` s'interprète comme $t \in fait$. On pose que l'ensemble des instants est partiellement ordonné, mais seulement vers le futur (le graphe induit par la relation d'ordre est un arbre). De plus, cet ensemble est dense. Un ensemble convexe et ordonné d'instants est un *intervalle*, un intervalle non borné est une chronique, et un événement est un ensemble d'intervalles (suivant une définition analogue a celle de fait). Un instant t est *atteignable* depuis t_0 s'il existe une chaine d'événements reliant t_0 à t. Nous ne rentrerons pas plus avant dans l'évocation de ce formalisme. Signalons seulement qu'au moyen des notions de chronique, atteignable...D.McDermott va jusqu'à la définition d'actions sous la forme d'événements en permettant, provoquant, évitant d'autres.

Ce formalisme n'étant justifié pragmatiquement par aucune implantation, on a toute latitude pour en discuter les présupposés. En particulier, on peut contester la pertinence de considérer les instants comme les seuls objets primitifs et d'en tirer une définition des faits. La conséquence en est que l'absence d'une distinction entre un fait et l'ensemble de ses instants d'occurrence est troublante. Peut-on dire qu'être assis dans son fauteuil tout l'après-midi, est le même fait que lire le journal si les deux ont la même occurrence ? On peut aussi discuter le fait de permettre une "ramification" de la ligne temporelle seulement vers le futur. Pourquoi interdire des chroniques "convergentes" correspondant à plusieurs interprétations possibles du passé ?

La Logique d'Allen

Pour J.Allen [2], les propriétés, les événements et les processus sont des entités temporelles fondamentales. Toutes sont associées à un intervalle (et non à un instant), approche liée à l'élaboration par ailleurs d'un formalisme de manipulation d'intervalles [3]. La différence de nature entre ces entités réside dans leurs propriétés de "divisibilité". Ainsi, une propriété (comme "la maison est rouge") est vraie sur un intervalle si et seulement si elle est vraie sur tout sous-intervalle, alors qu'un événement (comme la balle va de x_1 en x_2) ne peut être vrai sur deux intervalles dont l'un contient l'autre. Un processus ("je marche") peut être vrai sur certains sous-intervalles, la différence intuitive entre un événement et un processus étant que "on peut compter le nombre d'occurrences d'un événement, mais pas d'un processus". Allen utilise lui aussi ces fondements pour élaborer une théorie de la causalité et

de l'action. Notons que contrairement à celui de McDermott, le modèle temporel d'Allen est linéaire. La manipulation d'historiques possibles est définie en dehors du formalisme strictement temporel.

Y.Shoham [38] critique le choix des trois notions de base qui sont définies grossièrement par l'axiomatique d'Allen, en particulier du point de vue de la négation (seules les propriétés peuvent être niées). Il construit une logique réifiée associant un prédicat atemporel à un intervalle. Ceci lui permet de définir a posteriori différentes propriétés de divisibilité, dont celles données par Allen, qui sont toutes accessibles à la négation.

2.1.2 Les Représentations du Temps de Bas-Niveau

Les formalismes de cette famille sont moins ambitieux. Ils ne cherchent pas à justifier la présence d'évolutions, ou pourquoi les effets doivent succéder aux causes. Leur but est plus prosaïque, et le moyen le plus simple de le définir est sans doute de reprendre les termes de D.McDermott et E.Charniak [8] : ces systèmes sont, ou servent à construire des *gestionnaires de graphes temporels* (Time Map Managers). Les nœuds d'un graphe temporel sont des événements, des faits, des états de l'univers, les arcs sont des relations temporelles. Gérer un tel graphe, c'est le modifier de manière à ce que son interprétation corresponde le plus exactement possible à ce qu'il est censé décrire dans le monde réel. Cette définition ne peut qu'être informelle : chaque auteur a son idée sur le sujet, suivant ce qu'il désire représenter, et selon la puissance déductive qu'il désire obtenir.

Tous ces formalismes sont cependant fortement influencés par les logiques réifiées (association d'un terme prédicatif et d'un terme temporel, situation, instant, intervalle de temps...). C'est explicite pour J.Allen [3], ce qui est normal, étant donné ses préoccupations de haut-niveau (voir section 2.1.1) et pour S.Vere [45] qui l'utilise dans le cadre d'un planificateur "classique" . Ça l'est un peu moins pour nous (comme nous le verrons dans la suite) ou pour S.Smith [39], qui ne considèrent pas la signification que l'on peut donner au terme prédicatif ("ce qui se passe") pour se concentrer sur la manipulation des termes temporels ("les moments où ça se passe") qui dénotent en général des intervalles.

Ainsi, on peut manipuler le graphe temporel dans le cadre d'un *module temporel* ne communiquant avec le reste du système que par l'intermédiaire d'interrogations sur l'état du graphe et de relations ou d'objets temporels à ajouter ou à retirer.

En effet, chaque fois que le graphe est modifié localement, il faut en "propager" les effets. Soit, par exemple un graphe comportant trois intervalles A, B, et C, liés par les relations `A avant B`, `B avant ou après C`, `A avant ou après C`, si on ajoute la relation `B avant C`, on peut en déduire `A avant C` et remettre à jour dans le graphe la relation liant A et C. Ne pas le faire rendrait ce graphe *inconsistant*, car A ne peut être après C, autrement dit, on ne peut pas trouver trois intervalles qui obéissent à la description de ce graphe. Si, par la suite, on doit retirer `A avant B`, on en déduit `A avant ou après C`, sinon le graphe serait *incomplet* : il interdirait toute les combinaisons d'intervalles dans lesquelles A est après C[1].

[1]L'exemple que nous donnons ici peut être trompeur dans la mesure où nous supposons que le graphe *est*

Nous verrons, lors de l'étude des différentes représentations temporelles, que la richesse de représentation s'oppose souvent à la possibilité d'effectuer des propagations consistantes et complètes.

La section suivante introduit la propagation de contraintes dans un réseau. Cet ensemble de techniques a été développé (et est encore un domaine très actif) totalement indépendamment du problème de la représentation du temps, mais constitue une base théorique opérationnelle (par opposition à la base conceptuelle qu'est une représentation de haut-niveau) de toutes les représentations actuelles de bas-niveau. C'est pourquoi nous nous permettons cette assez longue digression.

2.2 La Propagation de Contraintes

2.2.1 Généralités

On regroupe sous le terme *propagation de contraintes* un ensemble de techniques de résolution de problèmes largement utilisées en IA, mais aussi en Recherche Opérationnelle (et même par le grand public, à travers les tableurs). Ces techniques correspondent à des aspects si divers qu'il est difficile d'en donner une description formelle. E. Davis [11] propose de les définir de la manière suivante.

Un *réseau de contraintes* est une structure de données déclarative codée sous forme d'un graphe dont les nœuds dénotent des paramètres (valués) et les arcs des relations entre les paramètres. Ce à quoi nous ajoutons que les relations peuvent elles-mêmes être valuées et que le graphe est souvent un hyper-graphe (les arcs dénotent des relations n-aires). Le type principal d'inférence que l'on fait sur un réseau de contraintes consiste à assurer la *consistance globale* du réseau au moyen de mises-à-jour successives et *locales* mises en œuvre par un algorithme dit de *propagation de contraintes* décrit informellement à la figure 2.1.

Les avantages présumés d'un modèle fondé sur les contraintes sont les suivants :

- Un réseau de contraintes est une structure déclarative : l'expression d'une relation ne préjuge pas de la manière dont la relation est exploitée. Par exemple, la relation algébrique[2] $x + y = z$ contraint chaque paramètre en fonction des deux autres sans préciser dans quel ordre et à quel moment on vérifiera la compatibilité des valeurs des trois paramètres.

  De plus, lorsqu'une propagation est interrompue, tous ses effets sont representés sur le réseau. On peut donc les exploiter immédiatement et éventuellement reprendre la propagation sans manipulation préalable.

la représentation temporelle. En général, ce graphe est un simple outil de représentation pour le module temporel. C'est ce dernier qui gère la consistance et la complétude. Il indique au reste du système, le cas échéant, la relation qui lie A et C et qui peut ne pas être codée explicitement dans le graphe. Par souci de simplification, nous referons parfois cette confusion.

[2]qui peut-être représentée sous une tout autre forme dans le réseau

Répéter 1. Prendre un petit nombre de nœuds (et de leurs contraintes associées) connexes dans le réseau *(typiquement deux nœuds reliés par un arc orienté)*

2. Mettre à jour les valeurs de cette partie du réseau compte-tenu de l'information qui y est strictement contenue *(par exemple la valeur du nœud d'arrivée en fonction de celle du nœud de départ)*

jusqu'à ce qu'il n'y ait plus de mise à jour à faire, auquel cas le réseau est dit *au repos*, ou jusqu'à une condition d'arrêt spécifique.

Figure 2.1: Démarche de la propagation de contraintes.

- L'algorithme de propagation agit localement. A un moment donné de la résolution, on n'examine qu'une petite partie du réseau. Ceci permet a priori une structure de contrôle simple et facile à comprendre. C'est aussi un argument possible pour une implantation tirant parti du parallélisme inhérent à l'algorithme.

- Ce mode de représentation se prête bien à des manipulations incrémentales. On peut facilement ajouter des nœuds ou des relations et propager leurs effets sur le reste du réseau.

- Le réseau de contraintes peut posséder une forte analogie avec des objets compréhensibles par l'utilisateur. Un circuit électrique, des particules de matières sont autant d'objets que l'on représente naturellement par un réseau d'éléments interagissant localement. Les mécanismes de propagation sont alors faciles à expliquer à un utilisateur de la représentation.

Toutes ces bonnes propriétés ne sont cependant pas acquises automatiquement. En particulier, la vision très locale de l'algorithme de propagation peut conduire à des inférences d'une puissance insuffisante. Si l'on veut augmenter cette puissance, on peut "globaliser" l'algorithme en lui faisant prendre en compte plus de nœuds à la fois, au prix d'une complexité accrue. Inversement, une grande localité est facile à programmer mais les effets de l'algorithme peuvent n'avoir aucun sens pour l'utilisateur. Si le réseau contient un très grand nombre de nœuds, modifiés un grand nombre de fois (comme ce peut être le cas, par exemple, pour la simulation d'un réseau de neurones) il est illusoire de penser que la trace d'exécution de l'algorithme soit déchiffrable.

E.Davis [11] distingue plusieurs sortes de propagation de contraintes, suivant le type de modification que l'on apporte au réseau. En bref, les valeurs peuvent être uniques, auquel cas on peut chercher à déterminer celles qui sont encore inconnues (inférence de valeurs, un exemple simple est la résolution d'un système triangulaire d'équation). Ou bien on peut changer les valeurs de manière à ce qu'elles soient consistantes (relaxation, c'est une technique peu utilisée en IA). Les valeurs peuvent, au contraire, appartenir à un domaine que l'on réduit de manière à ce qu'il ne contienne que des éléments consistants avec ceux des autres domaines. Les valeurs peuvent être des nombres et seront manipulées

au moyen de l'arithmétique classique, mais elles peuvent aussi être symboliques, auquel cas des manipulations de type algébrique devront être définies. Enfin, la structure du réseau peut être fixe ou variable, dans ce dernier cas, on peut ajouter ou retirer des nœuds ou des relations.

Dans ce qui suit, nous nous intéressons à un type particulier de problèmes utilisant la propagation de contraintes : les problèmes d'étiquetage.

2.2.2 Problèmes d'Etiquetage

Soit $\mathcal{X} = (X_i)_{i=1,n}$ un ensemble de variables et $(V_i)_{i=1,n}$ leurs domaines. Une *contrainte p-aire* sur un ensemble de p variables est un sous-ensemble du produit cartésien de ces variables. Elle peut être définie en extension, au moyen de la liste des p-uplets possibles, ou en compréhension, au moyen d'une condition nécessaire et suffisante la caractérisant.

Un *étiquetage* de $\mathcal{X}$, consistant avec $\mathcal{R}$, ensemble de contraintes sur des parties de $\mathcal{X}$, est un n-uplet dont les projections sur les parties de $\mathcal{X}$ vérifient les contraintes de $\mathcal{R}$ (c'est-à-dire appartiennent à $\mathcal{R}$). On peut se poser différents problèmes, suivant que l'on désire : savoir si $\mathcal{X}$ admet un étiquetage consistant avec $\mathcal{R}$, trouver un étiquetage consistant, ou bien trouver tous les étiquetages consistants avec $\mathcal{R}$. Nous considérons dans la suite que le problème est de trouver *tous* les étiquetages consistants. La plupart des études concernant ce problème supposent que les domaines ont un nombre fini d'éléments. C'est pourquoi nous ferons aussi cette hypothèse dans le reste de cette section. Un grand nombre de problèmes peuvent se formuler de cette manière : coloriage de graphes, analyse d'indices de scène, etc.

L'approche qui consiste à énumérer tous les n-uplets de valeurs possibles, puis à les vérifier est combinatoire (dans le sens où sa complexité est exponentielle). On peut considérer deux familles de techniques destinées à en diminuer la complexité : les méthodes d'énumération/choix/retour-arrière, et celles de moindre-engagement/propagation de contraintes. Les premières consistent à essayer tour à tour les valeurs possibles, suivant un ordre en général fonction de critères heuristiques, et à revenir "intelligemment" en arrière en cas d'échec, de manière à en essayer le moins possible. Cette technique est nettement plus intéressante lorsqu'on désire obtenir un seul étiquetage. Les méthodes de moindre engagement consistent à réduire les domaines de valeurs, grâce à une propagation de contraintes, sur une base locale, c'est-à-dire à éliminer les valeurs qui, moyennant la vérification d'un petit nombre de variables, ne peuvent appartenir à aucune solution.

Les deux types de méthodes sont complémentaires : l'énumération garantit que l'on élimine toutes les inconsistances au prix d'une complexité exponentielle et la propagation de contraintes est en général d'une complexité polynômiale mais ne détecte pas toutes les inconsistances. C'est pourquoi on utilise en général la dernière comme un *filtre* réduisant la complexité de l'énumération. Dans la suite, nous nous focaliserons sur les méthodes de propagation de contraintes.

2.2.3 Etiquetage d'un Réseau de Contraintes Binaires

Toutes les études de réseaux de contraintes se restreignent à l'étude de contraintes binaires[3]. U.Montanari [29] montre que l'on peut associer à toute contrainte n-aire un ensemble de contraintes binaires qui sont obtenues en projetant les n-uplets vérifiant la contrainte sur tous les couples de domaines. On transforme ainsi un produit cartésien de n domaines en $(n-1)!$ produits cartésiens de deux domaines. Comme on peut s'y attendre en présence d'une projection, l'ensemble de contraintes binaires n'est qu'une approximation de la contrainte originale car il induit une contrainte n-aire qui contient la première. C'est cependant la meilleure approximation, car le nombre de n-uplets "parasites" introduit par la reconstruction d'une contrainte n-aire est minimal (par rapport au nombre de parasites introduits par une autre approximation).

Le problème d'étiquetage consistant est alors reformulé par Montanari en *Problème central.* Il ne consiste pas à trouver directement tous les n-uplets solutions de la contrainte n-aire formulée par le réseau de contraintes binaires, mais à éliminer toutes les valeurs[4] qui n'appartiennent à aucun n-uplet solution. Ceci revient à *expliciter* la contrainte globale (n-aire) exercée par le reste du réseau sur les valeurs de la variable dont on restreint le domaine. Moyennant une telle élimination, on peut construire une solution "de proche en proche" sur une base locale, donc non-combinatoire. Etant donné que la construction d'une solution est elle-même d'une complexité exponentielle, il en va de même pour le problème central. On est donc amené à résoudre le problème plus faible qui consiste à éliminer *certaines* des valeurs impossibles, de manière à alléger l'étape d'énumération.

2.2.4 Degrés de Consistance d'un Réseau de Contraintes

La consistance d'arcs (ou 2-consistance) est la propriété que possède un réseau de contraintes lorsque toutes les valeurs v_i d'une variable X_i sont compatibles avec au moins une valeur de toutes les variables avec lesquelles X_i est en relation :

$$\forall i,j \in [1,n] X_i R X_j \Rightarrow \forall v_i \in V_i, \exists v_j \in V_j / (v_i, v_j) \in R \qquad (2.1)$$

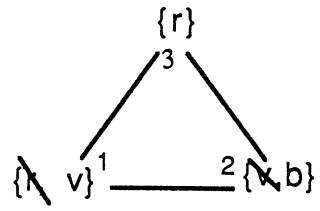
où R, contrainte binaire, est aussi, par abus de notation, définie comme un sous-ensemble du produit cartésien $X_i \times X_j$, conformément à la définition donnée à la section 2.2.2.

Il est possible de rendre un graphe 2-consistant en comparant chaque couple de variables et en éliminant les v_i qui ne vérifient pas la relation 2.1. Chaque fois qu'un domaine V_i est modifié, les couples comprenant X_i doivent être vérifiés. Les domaines étant finis, l'algorithme termine en $O(n^2N^2)$ étapes où n est le nombre de variables et N la taille maximale d'un domaine. Une telle méthode est souvent appelée filtrage de Waltz, ce dernier l'ayant employée pour filtrer des valeurs attachées aux indices de scène d'une image.

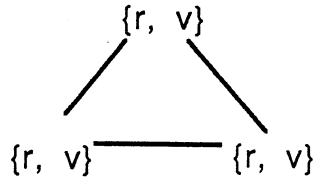
[3]Et aussi à celle, triviale, des contraintes unaires

[4]En fait, pour Montanari, étant donné sa formalisation du problème, ce sont des paires qu'on élimine. Il n'y a pas de différence fondamentale avec l'élimination de valeurs.

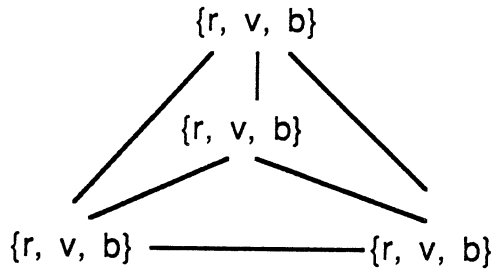
Il s'agit d'étiqueter chaque graphe en associant une couleur à chaque noeud de manière à ce que deux noeuds voisins n'aient pas la même couleur.
Sur la figure, on donne pour chaque noeud le domaine des couleurs admissibles a priori (par exemple {r, v} signifie que le noeud peut être rouge ou vert).



Consistance d'arcs (2-consistance)
En comparant les variables 2 à 2, on déduit que, en comparant 1 et 3, 1 ne pas être rouge, puis en comparant 1 et 2 2 ne peut pas être vert.



Consistance de chemins (3-consistance):
La consistance d'arcs est vérifiée sur ce graphe. Mais si l'on fixe deux valeurs consistantes, aucune valeur n'est possible sur le troisième noeud. Ce réseau n'est donc pas 3-consistant.



Consistances d'ordre supérieur: ce graphe est évidemment globalement inconsistant. Cependant il est 2 et 3-consistant. Si l'on fixe 3 valeurs 3-consistantes, on ne peut trouver de valeur possible sur le quatrième noeud. Faire ce genre de vérification permet de prouver l'éventuelle 4-consistance du réseau. On peut généraliser et définir une propriété de k-consistance.

Figure 2.2: Niveaux de Consistance

Le premier dessin de la figure 2.2 montre comment les modifications locales imposées par la vérification de la consistance d'arcs (en l'occurrence l'élimination de toutes les couleurs qui sont *forcément* identiques à celle d'un nœud voisin) permettent de rendre un réseau globalement consistant. Le second dessin montre que la consistance d'arcs n'est pas une condition suffisante de consistance globale. En particulier, il ne vérifie pas la propriété suivante (exprimée en français dans la légende du dessin) :

$$\forall i,j,k \in [1,n],$$
$$X_i R_{ij} X_j \quad X_j R_{jk} X_k \quad X_i R_{ik} X_k \quad \Rightarrow \qquad (2.2)$$
$$\forall (v_i, v_j) \in R_{ij}, \exists v_k \in V_k / (v_i, v_k) \in R_{ik} \wedge (v_j, v_k) \in R_{jk}$$

Cette propriété est appelée 3-consistance, car elle vérifie une consistance "locale" sur trois variables à la fois. Rendre le réseau 3-consistant consiste à éliminer tous les v_i (ou les v_j) qui n'appartiennent à aucun couple (v_i, v_j) vérifiant la relation 2.2. Montanari travaille directement sur les relations ce qui lui permet d'éliminer les *couples* de valeurs (v_i, v_j) inconsistants. Cette opération d'élimination peut être vue comme le calcul de la contrainte entre X_i et X_j *induite* par R_{ik} et R_{kj}. Lorsqu'on peut exprimer facilement cette opération, il s'ensuit un algorithme de fermeture transitive qui permet de rendre le réseau 3-consistant. La complexité de l'algorithme est cubique par rapport au nombre de variables.

E.C. Freuder [17] a généralisé la propriété de k-consistance. Nous en donnons seulement un énoncé informel : si l'on étiquette $k-1$ variables quelconques par des valeurs satisfaisant toutes les contraintes existant entre ces variables, toute k-ième variable peut être étiquetée par une valeur telle que les k valeurs satisfassent les contraintes entre les k variables. Freuder donne aussi un algorithme permettant de rendre un réseau k-consistant en augmentant itérativement le degré de consistance.

Domaines Continus

Si l'on donne aux variables des domaines numériques continus, le problème d'étiquetage se formule de la même manière, ainsi que les différentes propriétés de consistance. Cependant, la plupart des algorithmes (Montanari, Freuder...) reposent sur une comparaison énumérative des domaines (locale, certes) et sont donc inapplicables tels quels. Toutefois, on peut souvent utiliser avec profit la structure d'ordre qui existe sur un domaine numérique, et sa compatibilité avec l'expression algébrique des contraintes.

2.3 Etude Détaillée des Représentations du Temps de Bas-Niveau

2.3.1 Les Systèmes Symboliques

Ce type de système, introduit par J.Allen [3] rentre très bien dans le cadre d'un formalisme de graphe temporel. Les nœuds sont des symboles qui font référence à des intervalles, et les

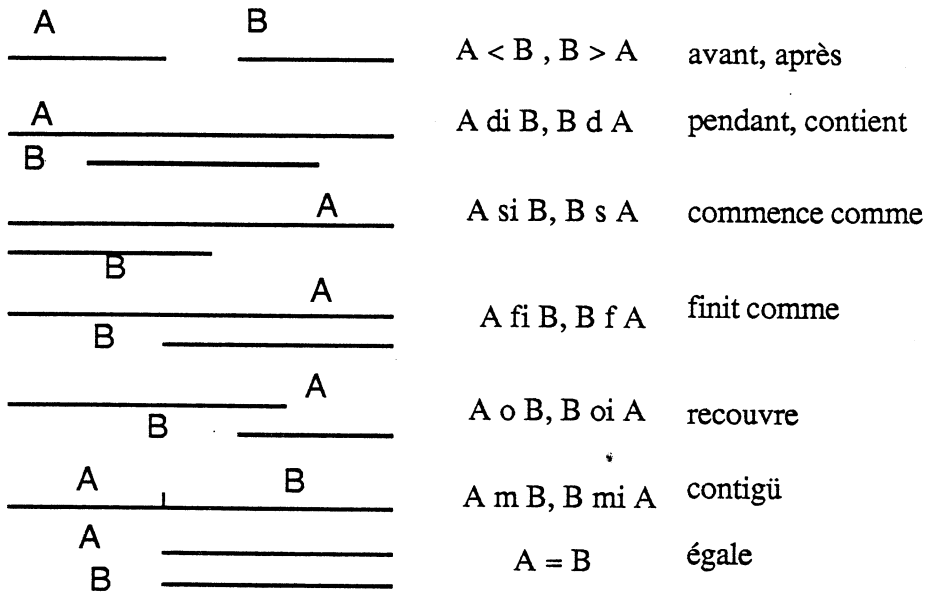


Figure 2.3: les treizes primitives reliant deux intervalles

arcs sont des relations temporelles disjonctives (comme "pendant ou après ou avant") dont nous expliquerons le formalisme plus en détail par la suite.

Les déductions aboutissent au retrait de certaines disjonctions des relations, ce qui revient à les préciser. C'est cette caractéristique qui nous fait donner le nom de *symbolique* à ce type de système, par opposition aux systèmes *numériques*, que nous verrons par la suite, dont les déductions conduisent à modifier des contraintes numériques sur des intervalles — comme une date de fin au plus tard —.

Principes du système d'Allen

J.Allen considère un graphe temporel dont chaque nœud dénote un intervalle. Allen a défini treize relations primitives définissant les positions relatives de deux intervalles (voir fig. 2.3).

Ces relations sont mutuellement exclusives et permettent de relier tout couple d'intervalles. Ainsi, tout couple d'intervalles datés est relié par une et une seule de ces relations primitives. Elles sont donc les composants élémentaires des relations constituant les arcs du graphe temporel. En effet, ces arcs — que nous appellerons relations *composées* ou *disjonctives* — sont des *ensembles* de primitives que l'on peut définir de la manière suivante (en utilisant la notation d'Allen, $R = (r_i)_{i=1,n}$ est une relation disjonctive entre A et B) :

$$A(r_i)_{i=1,n}B \Leftrightarrow \bigvee_{i=1}^{n} A\, r_i\, B$$

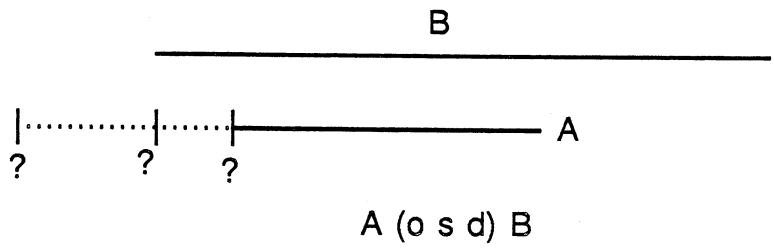


Figure 2.4: Un exemple de relation composée

La figure 2.4 donne un exemple de relation composée : A peut être pendant B, ou commencer comme B, ou enfin recouvrir B.

Le fait de considérer ces relations comme des ensembles permet de définir très facilement la conjonction et la disjonction de relations composées comme, respectivement, l'intersection et l'union des ensembles qu'elles constituent. Ainsi, par exemple :

$$A(<>)B \wedge A(<\ d)B \Rightarrow A(<)B$$

et

$$A(<>)B \vee A(<\ d)B \Rightarrow A(<>\ d)B$$

Le pouvoir de déduction du système d'Allen repose sur l'établissement d'une table de transitivité 13×13 entre les relations primitives, permettant de calculer r_3 dans la formule :

$$A\, r_1\, B \wedge B\, r_2\, C \Rightarrow A\, r_3\, C$$

Par exemple, si r_1 et r_2 valent $<$, r_3 vaut $<$ (voir en annexe B la table de transitivité complète).

En général, la relation r_3 est composée, car la connaissance de r_1 et r_2 ne donne pas toujours assez d'information pour que r_3 soit suffisamment précise. Par exemple :

$$A < B \wedge B\, d\, C \Rightarrow A(<\ o\ m\ d\ s)C$$

Lorsque r_1 et r_2 sont elles-mêmes composées, l'algorithme de calcul de r_3 est très simple :

1. Décomposer r_1 et r_2 en une disjonction de relations élémentaires
2. Développer tous les termes en appliquent la distributivité de $\wedge$ par rapport à $\vee$. On obtient ainsi des clauses qui sont des paires conjonctives.
3. Calculer grâce à la table le résultat de chaque paire conjonctive
4. Les conjonctions ayant disparu, composer les clauses disjonctives en une seule relation composée.

Cet algorithme définit sur l'ensemble des relations disjonctives une loi de composition interne que nous appellerons $\circ$.

Sur un graphe entier, $\left(\{S_i\}_{i\in[1,n]}, \{R_{ij}\}_{i,j\in[1,n]}\right)$, l'algorithme d'Allen a pour objet d'expliciter toutes les relations entre tous les nœuds. Il s'applique de la manière suivante (où Q est l'ensemble des paires[5] d'indices dénotant une relation en attente de propagation) :

Initialisation Pour $i, j \in [1, n], i \neq j$:

$Q \leftarrow \{i, j\}$

Propagation Tant que $Q \neq \emptyset$

- $\{i, j\} \leftarrow$ dépiler(Q) *On prend*[6] *un arc.*
- Pour $k \in [1, n], k \neq i, j$
  - *On calcule l'influence de l'arc tiré sur les autres arcs :*
    $R'_{ik} \leftarrow (R_{ij} \circ R_{jk}) \wedge R_{ik}$
    $R'_{kj} \leftarrow (R_{ki} \circ R_{ij}) \wedge R_{kj}$
  - Si $R'_{ik} = \emptyset \vee R'_{kj} = \emptyset$ Alors STOP *Contradiction*
  - *Si la nouvelle relation diffère (en fait elle est nécessairement inférieure ou égale) de l'ancienne, il faut la propager :*
    Si $R'_{ik} \neq R_{ik}$ Alors empiler $\{i, k\}$ dans Q ; $R_{ik} \leftarrow R'_{ik}$
    Si $R'_{kj} \neq R_{kj}$ Alors empiler $\{j, k\}$ dans Q ; $R_{kj} \leftarrow R'_{kj}$

Cet algorithme ne permet que d'*ajouter* des relations qui précisent de plus en plus le graphe en *éliminant* des disjonctions. Etant donné que le nombre de primitives et la taille du graphe sont finis, l'algorithme s'arrête, soit parce qu'il n'y a plus de relations apportant une information nouvelle, soit parce qu'une relation composée devient vide, auquel cas on a détecté l'*incohérence* des données de départ.

Complexité de l'algorithme, les intervalles de référence

Le calcul d'un graphe de n sommets suivant la méthode que nous venons d'exposer a le coût suivant :

(Nombre de relations composées à mettre à jour) $\times$ (nombre de mises à jour par relation) $\times$ (coût d'une mise à jour)

Le premier terme est égal au nombre de relations dans le graphe : $n(n-1)/2$ puisque le graphe est complet. Le deuxième terme est borné par 13, puisqu'il y a au maximum treize primitives et que chaque mise a jour enlève au moins une primitive[7]. Pour faire une mise à jour, il faut, pour chaque nœud, appliquer le premier algorithme que nous avons donné, soit

[5] L'ordre des indices n'est pas important, car chaque relation est inversible

[6] Q n'est pas nécessairement une pile, on peut prendre n'importe quel arc.

[7] En pratique, ce terme vaut beaucoup moins que 13

$O(n) \times O(r^2)$, où r est le nombre "moyen" de primitives composant une relation composée (r est borné par 13).

En resumé, la complexité de cet algorihme par rapport au nombre de sommets est $O(n^3)$. Notons que le terme en $O(r^2)$, bien que borné (par $169 = 13^2$), implique que moins il y a de disjonctions, c'est à dire plus le problème est contraint, plus l'algorithme est rapide.

En fait beaucoup de problèmes ont une structure temporelle organisée en "groupements" d'événements faiblement interdépendants. J.Allen propose donc de définir un certain nombre d'*intervalles de référence* dont chacun contient un de ces "groupements". Par exemple, tous les événements contenus dans l'intervalle de référence `hier` sont calculés indépendamment de ceux contenus dans l'intervalle `aujourd'hui`. Allen propose donc une modification de sa méthode, qui consiste à ne calculer complètement que ces "sous-graphes" temporels. On ne calcule les relations entre deux intervalles appartenant à des sous-graphes différents qu'en passant par les intervalles de référence, en essayant d'exploiter des relations immédiates de transitivité (par exemple lorsque les intervalles de référence ne sont liés que par des relations de succession).

L'intérêt d'une telle méthode, en termes de complexité est évident : étant donné que la complexité est en $O(n^3)$, si on divise un graphe en deux, celle-ci est réduite par $2^3/2 = 4$. L'inconvénient est qu'en général, en "forçant" une telle structure, on perd de l'information. Une relation assez forte entre deux intervalles ne peut être exprimée en passant par les relations intermédiaires avec les intervalles de référence.

Consistance, Liens avec les Problèmes d'étiquetage

Nous avons vu que l'algorithme s'arrêtait lorsqu'il détectait une incohérence, mais on peut se poser la question suivante : "s'arrête-t-il *chaque fois* qu'il y a une incohérence dans les données ?".

La réponse est non : il existe des graphes, parfaitement cohérents selon l'algorithme d'Allen, pour lesquels on ne peut associer un intervalle à chaque nœud et qui obéisse aux relations définies dans le graphe [3].

La raison de cette inconsistance est plus facile à voir si on regarde ce problème comme un problème d'étiquetage consistant. Cependant, l'étiquetage d'un nœud du reseau d'Allen par un intervalle est implicite. Tout ce qu'on peut dire est que les propriétés des relations temporelles sont compatibles avec cette interprétation. Car les seuls objets du formalisme sont les relations, primitives et disjonctives, l'ensemble de ces dernières étant muni de la loi d'intersection et de la loi de composition. Or cet ensemble possède la structure utilisée par Montanari pour exprimer son algorithme de propagation.[8].

[8]Structure utilisée aussi par Aho, Hopcroft et Ulmann [1] sous le nom de *Closed Semi-Ring*, ici $(\mathcal{P}(P), \wedge, \circ, P, \{=\})$ où P est l'ensemble des primitives, $\mathcal{P}(P)$ l'ensemble des parties de P et $\circ$ la loi de composition entre relations disjonctives. L'algorithme de propagation, dit de *fermeture transitive*, fonctionne un peu différemment de celui d'Allen : on reconstruit le réseau en y réintroduisant les nœuds un à un et en veillant à chaque introduction que le tout soit 3-consistant. On obtient cependant le même résultat, avec une complexité théorique équivalente.

En effet, la formalisation de Montanari [29] ne manipule pas directement les étiquettes, mais les relations de compatibilité entre deux variables x et y représentées par des matrices dont l'élément r_{ij} vaut 1 si et seulement si les ièmes et jèmes valeurs du domaine de x et y sont compatibles. Cela permet à Montanari une formalisation élégante s'appuyant sur les propriétés du calcul matriciel, mais on peut toujours repasser à une manipulation directe des étiquettes. Par contre, en ne s'appuyant que sur les relations, Allen peut tourner le fait que les domaines de valeurs sont infinis, et même donner suffisamment de généralité à son formalisme pour que la question de savoir si les domaines d'intervalles sont infinis ou non, n'ait pas d'importance pour la propagation des contraintes.

L'expression précise du problème d'Allen en termes d'étiquetage a été faite par T.Granier [23] au moyen d'un formalisme très proche de celui de Montanari.

Le problème central de Montanari étant NP-complet, il est légitime de s'interroger sur la complexité du problème d'Allen, vu comme un problème d'etiquetage. Le résultat de NP-complétude a été établi par Vilain et Kautz [47]. Pour éliminer toutes les configurations incohérentes, il faudrait énumérer tous les graphes composés uniquement de primitives et vérifier leur cohérence. Le nombre de ces graphes est exponentiel par rapport au nombre de sommets ($O(r^{n^2})$ si r est le nombre moyen de primitives par arc).

Vilain et Kautz ont proposé un formalisme comparable à celui d'Allen, mais dont la 3-consistance garantit la consistance globale : on considére un graphe dont les nœuds ne dénotent pas des intervalles mais des *points*. Il existe alors trois relations élémentaires ($<$, $>$ et $=$) dont on peut construire la table de transitivité et la méthode de propagation est exactement la même que celle d'Allen. Cependant, ce formalisme est plus pauvre : en "traduisant" un réseau d'Allen en réseau de Vilain, on perd de l'information. En particulier, la relation $A(<>)B$ est intraduisible dans un réseau de points, or c'est une relation clef de l'ordonnancement. Ceci met donc en évidence le fait qu'il y a des disjonctions "fortes", comme $(<>)$ qui imposent une énumération et des disjonctions "faibles", comme $(m\ <)$ que l'on peut exprimer dans un formalisme ponctuel pour lequel il suffit de vérifier la 3-consistance. T.Granier [23] propose un algorithme qui s'appuie sur la détection des disjonctions faibles (qu'il appelle "contraintes compatibles" — avec un formalisme ponctuel —) et énumère les disjonctions fortes, ce qui lui permet d'obtenir une consistance globale.

Richesse et Limitations de la Représentation

La représentation d'Allen, manipulant des intervalles, permet de traiter des relations disjonctives entre intervalles. Cependant, seules certaines disjonctions peuvent être considérées. Il est impossible[9] d'exprimer la relation "$A < B \vee A < C$", par exemple. En effet, une telle relation est ternaire, or le graphe temporel ne contient que des relations binaires. Pour représenter des relations d'un ordre supérieur, il faudrait manipuler non pas un graphe, mais un hypergraphe, faisant intervenir des relations entre les arcs, comme le "ou" dans notre exemple.

[9] A dire vrai c'est possible, mais en projetant cette contrainte sur un réseau binaire (cf. section 2.2.3), on obtient la contrainte triviale, la plus générale, entre A, B et C.

La théorie de la manipulation et de la construction des primitives (détaillée par J.Allen et P.Hayes [5]) et des relations composées est à la fois rigoureuse et élégante, par sa considération d'un petit nombre de symboles. Cependant, dans [5], toutes les primitives sont construites à partir des notions temporelles de succession et de simultanéité (rassemblées dans la relation "meet"). Jamais celle de durée n'apparait, et en effet, on ne peut pas représenter une relation du type "A dure moins longtemps que B". Ceci est à rapprocher de la faiblesse du formalisme d'Allen à manipuler les concepts métriques (après tout, durée = fin - début).

2.3.2 Les Systèmes Numériques

Contrairement au parti pris d'Allen, les systèmes numériques manipulent explicitement des nombres (réels en général), ce qui leur permet de manipuler l'aspect métrique du temps.

Les Systèmes Fondés sur la Programmation Linéaire

Ces systèmes partent de la constatation que la plupart des relations temporelles peuvent s'exprimer de façon linéaire en fonction des dates de début et de fin d'un événement. Ceci suppose bien sûr que l'on associe un intervalle à un événement.

Systèmes Utilisant l'Algorithme du Simplexe L'algorithme du simplexe optimise n'importe quelle fonction linéaire sur un domaine exprimé au moyen de contraintes linéaires. Dans le problème qui nous occupe, il n'y a a priori pas de fonction à optimiser. On cherche seulement l'expression des domaines de validité des variables. Le principe du simplexe peut être utilisé de nombreuses manières, et la façon dont il peut servir à trouver ces domaines n'est pas toujours très claire. C.Barrouil [6], par exemple, l'utilise en ajoutant itérativement les contraintes $L(T) \geq 0$ sous forme de fonction $L(T)$ à optimiser. Si il n'existe pas de solution positive, l'ensemble des contraintes est incohérent. De plus, on a toujours "sous la main" une solution totalement explicitée, mais les domaines des variables, eux, ne sont pas explicites.

Ce genre de méthode permet d'utiliser toute la puissance de représentation de l'algèbre linéaire. Un point révélateur est que nous ne la décrivons pas du point de vue d'un graphe temporel : on peut en effet manipuler des relations n-aires intraduisibles dans le formalisme essentiellement binaire d'un graphe. On ne peut pas vraiment parler de la consistance de la méthode, dans la mesure où les domaines ne sont pas explicites. L'algorithme ne fait qu'exhiber *une* solution qui est, il est vrai, toujours consistante.

Cependant, on ne peut pas exprimer de disjonctions, C.Barrouil, par exemple, utilise dans son module temporel la puissance de PROLOG pour énumérer les propositions constituant les disjonctions et gérer le retour arrière en cas de problème.

Quant à la complexité de cette méthode, elle est fortement alourdie par le coût théoriquement exponentiel de l'algorithme du simplexe, qu'il faut faire fonctionner chaque fois que l'on ajoute une contrainte.

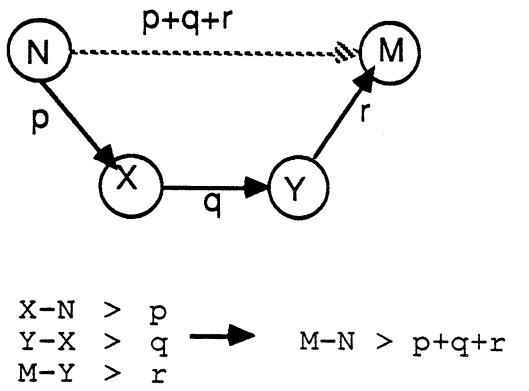


Figure 2.5: chemin étiquetté

Systèmes Utilisant des Contraintes Linéaires Simples Non seulement beaucoup de relations temporelles s'expriment de façon linéaire, mais parmi celles-ci, un grand nombre s'expriment de la manière simplifiée suivante :

$$N - M \geq p \tag{2.3}$$

où N et M sont des variables et p est un paramètre positif. Par exemple, "A pendant B" se traduit en *début(A) - début(B)* ≥ 0 *et fin(B) - fin(A)* ≥ 0

L'équation 2.3 traduit une relation binaire que l'on peut représenter dans un graphe temporel dont les nœuds dénotent des instants, en particulier les instants de début et de fin d'un événement. Les arcs de ce graphe relient les nœuds entre lesquels existe une relation comme 2.3 et sont étiquetés par la valeur p. Cet arc signifie que les deux nœuds (que nous appellerons N et M) doivent être distants d'au moins p. Une déduction consiste à expliciter de combien N et M doivent être distants. En effet, si on trouve entre ces nœuds un chemin formé par des arcs de valeur p, q, r par exemple, alors N et M doivent être écartés d'au moins $p + q + r$ (fig. 2.5).

Le plus long chemin ainsi trouvé a donc pour valeur la distance minimale imposée par le graphe entre M et N. Le graphe est incohérent si et seulement si il contient un circuit[10] de longueur strictement positive. Si on introduit dans le graphe un nœud O (pour "origine") lié vers tous les autres par un arc valant 0, la valeur du plus long chemin entre O et un nœud M est égale à la "date au plus tôt" de M.

En général, il est souhaitable de fixer une date limite globale de fin de planning. Ceci est impossible avec le mécanisme que nous venons de donner. Cependant, on peut raffiner le modèle en ajoutant des arcs à valeur négative, dont la valeur absolue dénote la distance maximale entre les nœuds. L'opposé du chemin le plus court entre un nœud M et le nœud origine est égal à la date au plus tard de M.

[10]chemin qui arrive à son point de départ

Ainsi, on peut calculer un intervalle de variation pour chaque nœud. On peut aussi calculer des durées minimales et maximales : si D et F sont des nœuds dénotant le début et la fin d'un intervalle (d'un événement), le plus long chemin de D à F est la longueur minimale de cet intervalle (ou durée minimale de l'événement). De même, l'opposé du plus court chemin de F à D est la longueur maximale (durée maximale).

La richesse et la complexité de ce formalisme est contenue dans les qualificatifs de

- binaire, puisqu'on peut le décrire au moyen d'un graphe.
- ponctuel, chaque nœud représente un point. Donc les limitations d'un formalisme ponctuel que nous avons évoquées à propos du formalisme de Vilain et Krautz s'appliquent ici : assez peu naturel (les événements sont en général des intervalles), et n'autorisant pas les disjonctions comme "A et B ne se recouvrent pas". Par contre comme celui de Vilain, le formalisme est consistant et complet, pour une complexité[11] en $O(n^3)$
- linéaire : sous leur plus simple expression (d'où une richesse inférieure aux méthodes fondées sur le simplexe), on peut tout de même exprimer des contraintes numériques en particulier sur les durées (plus simples que linéaires, même, puisque ce sont uniquement des valeurs extrémales) d'où l'avantage sur le formalisme de Vilain.

Notons pour finir qu'il est possible d'utiliser un formalisme encore plus simple, en considérant uniquement des événements à durée fixe. Dans ce cas, un seul nœud suffit pour représenter un événement.

Les Systèmes Manipulant Explicitement les Domaines d'Intervalles

Principes : Cette classe de systèmes est très proche d'une formalisation du type problème d'étiquetage consistant (sans que leurs concepteurs n'en fassent mention toutefois). Nous étudierons ici les systèmes de S.Vere [45] et de S.Smith et C.Le Pape [39,26]. Les objets primitifs sont des intervalles numériques, associés à un prédicat pour Vere, à une tâche pour Smith et Le Pape. Ils ne sont cependant pas manipulés directement, mais à travers leur domaine, défini par des bornes numériques (début et fin au plus tôt et au plus tard, la durée est fixée). Ainsi les intervalles sont des variables dont le domaine, infini, d'étiquettes est une *fenêtre* temporelle. Ce domaine est donc manipulé explicitement, contrairement aux représentations précédentes.

Les variables constituent les nœuds d'un graphe dont les arcs sont étiquetés par des relations temporelles qui ne sont pas changées lors de la propagation[12]

[11] D'ailleurs, en y regardant bien, c'est le même algorithme de *fermeture transitive* utilisé par les deux méthodes, en étiquetant un graphe numérique avec des >, on obtient un graphe de Vilain.

[12] Précisons : lors de la propagation *temporelle*, elles sont changées dans le cadre plus global du système de planification utilisant la représentation temporelle, en particulier par le système de Le Pape et Smith (OPIS [25]) qui est fondé sur la propagation de contraintes de toutes natures.

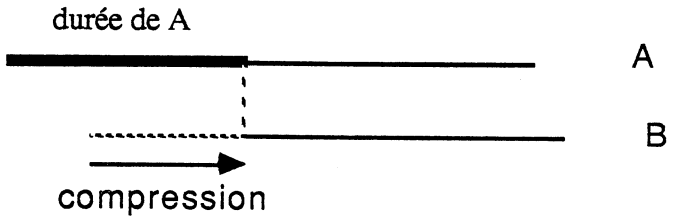


Figure 2.6: Effet de la fenêtre de A sur la fenêtre de B lorsque `A avant B`

Ces relations temporelles, symboliques, comme dans la représentation d'Allen sont pour Vere comme pour Smith, la conséquence directe d'une structure de plan. Vere décrit un plan a la manière de STRIPS, les conditions d'application d'un action doivent "contenir" cette dernière au sens temporel et ses effets lui sont consécutifs. Quant à Smith, il s'appuie sur la décomposition hiérarchique et l'ordre des tâches pour poser les contraintes temporelles (par exemple une tâche de haut-niveau commence avant et finit après toutes les tâches filles). Suivant les choix d'implémentation, les relations temporelles sont représentées plus ou moins explicitement : totalement implicitement dans le système de S.Vere, explicitement dans le système de S.Smith.

La vérification de la compatibilité de deux domaines liés par une relation temporelle se traduit par l'élimination des étiquettes impossibles. Comme on ne peut énumérer ces étiquettes et les rejeter explicitement, on modifie les bornes du domaine.

Par exemple, si le nœud A est défini (entre autres) par un début au plus tôt D_A et une durée d_A, si le nœud B est défini par un début au plus tôt D_B et si, enfin, A est relié à B par la relation **avant**, alors la relation $D_B \geq D_A + d_A$ doit être vérifiée (v. fig 2.6).

Comme le domaine du début de B ne peut qu'être réduit, le seul moyen de rendre cette inégalité vraie est d'augmenter D_B, c'est-à-dire de "comprimer" la fenêtre de B.

A l'échelle du graphe tout entier, une compression peut en entraîner une autre. On propage donc ces modifications jusqu'à ce que le réseau soit au repos.

Consistance et Complexité de ces Méthodes Un tel mécanisme est essentiellement *binaire* : une compression ne met en jeu que deux domaines. Au contraire, ceux que nous avons décrits précedemment étaient ternaires : deux nœuds en contraignaient un troisième, ou deux relations en contraignaient une troisième (le tout mettant en jeu trois nœuds). Autrement dit, la propagation de telles contraintes permet d'obtenir, un consistance d'arcs, plus faible a priori que la consistance de chemins ou la consistance globale obtenue par les méthodes précédentes.

Vere et Lepape–Smith n'ont pas fait part de leurs considérations à ce sujet. Nous nous appuierons donc, par anticipation, sur les résultats de la section 4.1.3. Mais il faut dire

que la consistance n'est pas pour eux un problème crucial. En effet, lorsqu'il n'y a pas de relations disjonctives et que les durées sont fixées, hypothèses que fait S.Vere dans [45], la consistance d'arcs implique la consistance globale.

Lepape et Smith autorisent les relations disjonctives et donc laissent subsister des inconsistances. Cependant, leur système est capable de revenir en arrière en cas d'échec (i.e. de mise en évidence de l'inconsistance de l'ensemble de solutions courant). Par conséquent, de telles inconsistances augmentent le nombre de retour-arrières par rapport à ceux dûs à des raisons non temporelles, mais sont prises en compte.

Quant à la complexité des propagations, elle ne provoque pas de commentaire de la part des auteurs. Nous verrons à la section 4.1.3 que cet aspect n'est pas sans problèmes : il est assez difficile de l'évaluer et, lorsqu'il y a contradiction symbolique des relations (comme $A < B$ et $B < A$) on peut construire des cas de figure pour lesquels la propagation est arbitrairement longue. On doit donc supposer, même si c'est peu satisfaisant, que les systèmes en question filtrent de telles contradictions et ont une complexité pratique acceptable.

Synthèse

Les travaux historiques menés en IA sur la planification d'actions reposent sur l'hypothèse que l'environnement est entièrement contrôlé par l'agent, ce qui diminue énormément la nécessité de considérations temporelles. Ces dernières n'apparaissent donc qu'indirectement, en particulier à travers la nécessité d'ordonner certaines actions, nécessité mise en évidence par l'existence de buts interdépendants. Cette mise en évidence et les solutions du problème qu'on peut y apporter constituent, avec l'étude de la planification hiérarchique un apport indéniable des systèmes classiques à la maîtrise de la complexité de la planification.

Un bref examen des systèmes réactifs, qui prennent en compte une interaction entre planification et exécution, lèvent l'hypothèse d'un univers statique en l'absence d'agent, se révèle assez décevant. En effet, on constate, dans les travaux existants, que lorsque ces systèmes imposent à l'agent de réagir au bon moment, ils perdent leur capacité de planification. En particulier ils n'examinent pas (ou peu) l'interdépendance des buts et des actions. Inversement, les systèmes capables de planifier ne sont pas vraiment réactifs.

Un certain nombre de systèmes récents reprennent des formalismes de planification — atemporels et non-réactifs en général — et leur adjoignent une représentation du temps. Dans tous les cas, cette représentation est fondée sur la manipulation d'intervalles (de temps), associés à une tâche, un but, un prédicat...

On remarque là une approche commune de la représentation du temps, que nous qualifions de "bas-niveau". En effet, elle n'aborde pas la formalisation de notions telles que l'évolution ou la causalité. De telles formalisations ont été proposées (nous les qualifions de "haut-niveau"), compatibles avec la manipulation d'intervalles. Cependant, elles n'ont pas donné lieu à des développements concrets et elles ne sont pas inattaquables sur le fond.

Si l'on examine les représentations de bas-niveau, on constate que la plupart associent à un même objet temporel, non pas un, mais plusieurs intervalles possibles. Chaque ensemble d'intervalles possibles est lié à plusieurs autres et exerce par là une influence sur eux, une contrainte.

Tenir compte du temps revient à calculer quels intervalles sont possibles, moyennant les contraintes qui s'exercent sur eux. Ce calcul, la propagation de contraintes, peut être formalisé indépendamment de considérations sur le temps. Il a été étudié, dans le cas d'ensembles discrets, à travers les problèmes dits d'étiquetage. L'étude montre qu'en général, l'élimination de tous les objets impossibles requiert une complexité équivalente à celle de l'énumération de l'ensemble des solutions.

Cependant, l'unité des représentations du temps de bas-niveau disparaît sur le plan technique. Certains systèmes, que nous appelons *numériques* utilisent des nombres et donnent donc des résultats quantitatifs. Les représentations *symboliques* ne manipulent que des relations. Dans les deux cas, et suivant chaque représentation, on travaille ou bien directement sur les intervalles, ou bien après une décomposition en points, en prenant les extrémités des intervalles.

On peut alors délimiter l'ensemble des représentations par deux extrêmes : les représentations numériques et ponctuelles d'une part, et les représentations symboliques d'intervalles d'autre part. Les premières sont fondées sur la recherche de plus long chemin dans un graphe. Ce sont celles qu'on voit le plus souvent lorsqu'il s'agit de résoudre des problèmes concrets, car on peut difficilement se passer de manipulations numériques efficaces. Les secondes, introduites par J.Allen, reposent sur une énumération exhaustive des relations symboliques possibles entre deux intervalles. Elles sont séduisantes par leur élégance sur le plan théorique, en particulier dans le traitement des disjonctions.

La partie qui va suivre consiste à rassembler ces deux perspectives dans un même formalisme. Ainsi, nous introduisons des relations disjonctives dans une représentation numérique, en manipulant directement des intervalles. Cette filiation nous conduit à adopter les mêmes hypothèses fondamentales que nos prédécesseurs : la représentation est indépendante du formalisme de planification, elle repose sur la manipulation d'intervalles, et plus précisément sur la propagation de contraintes sur des intervalles possibles.

Partie II

Définition d'un Formalisme de Représentation et de Propagation de Contraintes Temporelles

L'objet de cette partie est la description d'un formalisme de représentation de bas-niveau du temps. Ce formalisme est fondé sur la manipulation d'intervalles numériques au moyen de propagation de contraintes temporelles symboliques. Il est destiné à permettre d'élaborer la partie principale d'un module temporel pour un système de planification. La tâche principale d'un tel module est de maintenir les domaines de validité temporelle d'un certain nombre d'événements, moyennant des contraintes absolues (numériques) et relatives (symboliques). Le but est de pouvoir répondre aux questions du type "Quand tel événement est-il possible?". En effet, pouvoir répondre à cette question permet aussi de répondre à "Tel événement est-il possible à tel moment?" et, par contraintes successives "Donner une suite d'événements possibles".

Prenons un exemple :

1. Paul veut rencontrer Jean pendant la pause du déjeuner, mais pas pendant son repas.
2. Paul va déjeuner entre 11h30 et 13h30 Il ne peut s'absenter de son travail plus d'une heure, et il lui faut au moins une demi-heure pour prendre son repas. L'entrevue doit durer au moins une demi-heure pour être fructueuse, mais doit se terminer avant 12h30.
3. Par conséquent, la réunion peut avoir lieu au plus tôt à partir de 11h30, Paul n'aura qu'une demi-heure pour prendre son repas, à 13h00 au plus tard, il sera rentré.

Le but d'un module de propagation de contraintes temporelles est de pouvoir faire la déduction 3 à partir des données décrites dans 1 et 2. En effet :

1. La première phrase définit les entités de base du problème, que nous appelons des événements : rencontre, pause, repas. Elle définit aussi des relations temporelles entre ces événements : rencontre pendant pause, repas pendant pause, pause et repas disjoints.
2. La seconde phrase définit des contraintes numériques absolues (c'est-à-dire ne mettant en jeu qu'un événement) sur les moments pendant lesquels chaque événement peut avoir lieu : entre 11h30 et 13h30, pendant moins d'une heure, etc.
3. Les informations, décrites dans la troisième phrase sont aussi des contraintes numériques absolues mais plus précises, car elles tiennent compte des relations temporelles et des contraintes sur les autres événements.

Un exemple d'utilisation de module temporel est donné au chapitre 6.

Cette partie est scindée en deux chapitres. Le premier est *descriptif* : nous y définissons le type de problème temporel que nous abordons, que nous désignons par *problème d'occurrences contraintes* ou POC, et comment on peut le représenter. Le second chapitre montre comment *manipuler* les objets temporels définis au chapitre précédent et propose un algorithme de résolution d'un problème d'occurrences contraintes. La résolution n'est que partielle, nous expliquons dans quelle mesure elle l'est. Enfin, nous comparons le formalisme que nous avons défini avec d'autres, plus classiques, et montrons comment passer de l'un à l'autre.

Chapitre 3

Définition d'un Problème d'Occurrences Contraintes

3.1 Formulation du Problème

3.1.1 Evénements et DOPs

Les Evénements sont les entités élémentaires du module temporel. La description d'un événement (qui peut être une action, un processus, etc) n'entre pas dans le cadre de ce module. On peut considérer, que c'est un symbole.

A chaque événement, on associe une occurrence qui est un intervalle (fermé) sur la droite réelle. Sa signification intuitive est "les moments pendant lesquels se passe l'événement" . Nous n'envisageons donc pas le cas d'événements intervenant sur un "intervalle non convexe" comme *déjeuner* qui arrive tous les jours. On peut supposer que de tels événements sont décomposés à un plus haut niveau de raisonnement en événements dont l'occurrence est un intervalle.

L'intérêt d'un module temporel est de manipuler des événements dont l'occurrence n'est pas fixée, mais est simplement partiellement connue grâce à une contrainte numérique. Une telle contrainte permet à l'occurrence de varier dans un domaine défini par un ensemble de relations mettant en jeu le début et la fin de la dîte occurrence. Nous appelons ce domaine *Domaine des Occurrences Possibles ou DOP*[1]. Un DOP est un ensemble d'occurrences possibles mutuellement exclusives dont on sait qu'une seule occurrence sera éventuellement tirée, d'où le terme de "domaine" par analogie avec le domaine d'une variable. Dorénavant, nous associerons donc un DOP à chaque événement.

[1]Nous utilisons aussi parfois le sigle SOPO (Set Of Possible Occurrences) pour des raisons historiques.

3.1.2 Relations Temporelles

Les contraintes numériques absolues ne sont pas les seules données que l'on possède sur les occurrences des événements. Il faut aussi prendre en compte l'information de type relationnel. Cette dernière est codée sous forme de liens temporels qui sont le plus souvent déduits d'informations de nature différente.

Si l'on reprend l'exemple du début, l'événement *repas* est relié à *pause déjeuner* par la relation *pendant* qui dénote une dépendance hiérarchique : la pause déjeuner comprend le repas. La relation de non-recouvrement entre le repas et la réunion peut découler d'un raisonnement non temporel (par exemple, Paul veut montrer des documents qu'il ne serait pas facile de déballer dans un restaurant). Au niveau de ce module, toutes les relations temporelles sont mises au même niveau et leur origine est oubliée.

Les relations temporelles sont construites à partir des primitives qui permettent de décrire la relation symbolique qui unit deux intervalles (avant, après, pendant...). Ce sont des applications définies sur les DOPs de la manière suivante :

$$\begin{array}{rccc} R_{ij} : & O_i \times O_j & \rightarrow & \{0,1\} \\ & (o_i, o_j) & \mapsto & R_{ij}(o_i, o_j) \end{array}$$

Où R_{ij} est une relation temporelle, O_i et O_j deux DOPs, et o_i, o_j deux occurrences. Cette relation exprime une contrainte liant le début et/ou la fin de o_i et o_j. Par exemple, R_{ij} est la relation *avant* ssi :

$$R(o_i, o_j) = 1 \Leftrightarrow \text{début}(o_j) \geq \text{fin}(o_i)$$

3.1.3 Les Problèmes d'Occurrences Contraintes (POC)

Quelques définitions

Le problème d'occurrences contraintes est la structure de base manipulée par le module temporel. De ce point de vue, on peut dire que la fonction du module temporel est de résoudre des problèmes d'occurrences contraintes. L'objet de ce qui suit est de définir précisément ce que nous entendons pas là.

POC Un problème d'occurrences contraintes (ou POC) est un couple $(\mathcal{O}, \mathcal{R})$ où $\mathcal{O}$ est un l'ensemble des DOPs obtenu canoniquement à partir d'un ensemble d'événements $\mathcal{E}$, et $\mathcal{R}$ un ensemble de relations entre les éléments de $\mathcal{O}$. Par abus de langage, nous appellerons aussi POC le couple $(\mathcal{E}, \mathcal{R})$.

On peut aussi voir un POC comme un graphe dont les nœuds sont valués par un DOP et les arcs par une relation temporelle.

Choix d'Occurrences Soit $\mathcal{O} = (O_i)_{i=1,n}$ un ensemble de DOPs ; on appelle choix d'occurrences (ou choix) de $\mathcal{O}$ un n-uplet :

$$(o_i)_{i=1,n} \ / \ \forall i \in [1,n],\ o_i \in O_i$$

Autrement dit, c'est ce que l'on obtient en fixant une occurrence dans chaque domaine d'occurrences possibles.

Solution Opérationnelle Soit $\mathcal{P} = \left((O_i)_{i=1,n}, (R_{jk})_{j,k\in[1,n]}\right)$ un POC. Un choix d'occurrences $(o_i)_{i=1,n}$ de $(O_i)_{i=1,n}$ est une solution opérationnelle (ou solution) ssi :

$$\forall j,k \in [1,n],\ R_{jk}(o_j, o_k) = 1$$

Intuitivement, une solution est la description d'un monde où chaque événement a une occurrence précise et où toutes les relations temporelles sont vérifiées.

Consistance et Complétude Un POC $((O_i)_{i=1,n}, \mathcal{R})$ est dit :

dérivé de $((O'_j)_{j=1,n}, \mathcal{R})$ ssi

$$\forall k \in [1,n],\ O_k \subset O'_k$$

(dériver un POC signifie donc restreindre un ou plusieurs DOPs)

admissible si il existe un choix qui soit une solution

consistant si toute occurrence d'un DOP appartient à une solution

hyper-consistant si tout choix est une solution

complet par rapport à un POC $\mathcal{P}'$ si $\mathcal{P}$ est dérivé de $\mathcal{P}'$ et si l'ensemble des solutions de $\mathcal{P}'$ est égal à l'ensemble des solutions de $\mathcal{P}$.

Résolution d'un POC

L'idéal en présence d'un POC serait de trouver tous les choix qui sont des solutions. On ne peut cependant pas expliciter toutes ces solutions : les DOPs étant des domaines continus, le nombre de ces solutions est a priori infini. Même si les domaines sont discrétisés, le nombre de solutions a de grandes chances d'être très élevé.

Cependant, on peut considérer qu'on a trouvé les solutions d'un POC P de manière implicite si :

1. On montre l'admissibilité de P.
2. On exhibe un POC P' dérivé de P, complet par rapport à P et consistant.

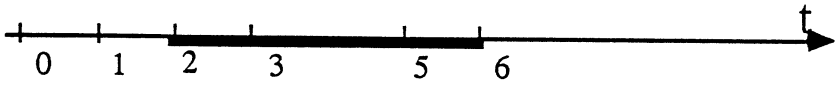


Figure 3.1: Représentation uni-dimensionnelle des occurrences

Nous appellerons cette opération de dérivation la résolution du POC P. Dès lors qu'on sait résoudre un POC, pour trouver une solution particulière, il suffit de choisir une occurrence dans un DOP de P'. Cela permet d'obtenir un nouveau POC dérivé que l'on résout à son tour et pour lequel on choisit une occurrence dans un autre DOP. Si on itère ce processus sur tous les DOP on est sûr d'obtenir une solution en ayant à chaque étape une liberté de choix maximale.

Résoudre un POC, c'est donc éliminer toutes les occurrences qui n'appartiennent à aucune solution[2] . On peut voir le processus de dérivation d'un POC P comme l'explicitation au niveau de chaque DOP de l'information contenue dans les contraintes relationnelles exprimées partout ailleurs dans P.

Nous avons défini dans cette section les Problèmes d'Occurrences Contraintes de manière très abstraite, comme un problème d'étiquetage. En effet, on peut en éliminer toutes les références à un raisonnement temporel. On peut même considérer, par exemple qu'il spécifie de la même manière un "problème des couleurs contraintes" pour lequel il s'agirait de déterminer "l'ensemble des couleurs possibles" des différentes "parties d'un costume" moyennant des "relations chromatiques", binaires (respectivement, les DOPs des événements moyennant des relations temporelles).

Dans la section suivante, nous précisons ce que sont les relations temporelles, les occurrences et les DOPs et comment on peut les représenter.

3.2 Domaines d'Occurrences Possibles et Relations temporelles

3.2.1 Représentations 1-D et 2-D des DOPs

Une occurrence est un intervalle fermé de $\Re$. La droite réelle est donc un moyen naturel de représentation (fig 3.1). Cependant, une représentation uni-dimensionnelle des DOPs est ambiguë car elle impose qu'on les représente au moyen d'intervalles alors que ce sont des

[2]La résolution d'un POC est analogue au *problème central* défini par Montanari dans le cas de domaines discrets (section 2.2.3).

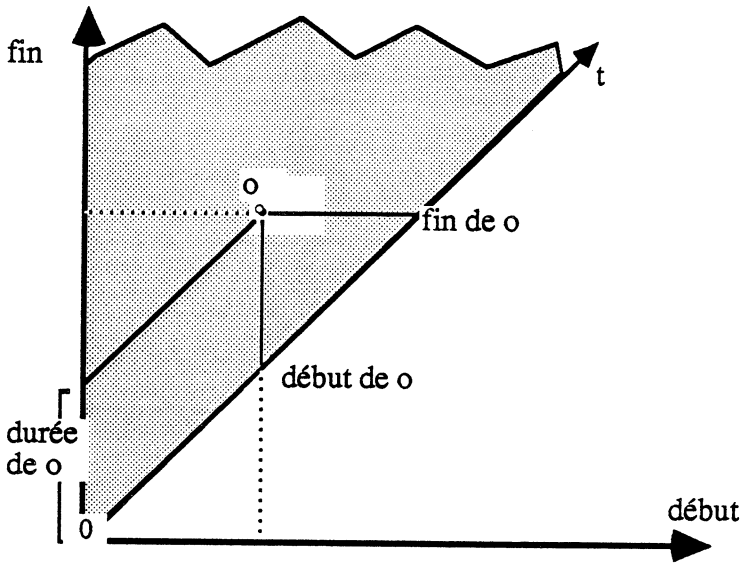


Figure 3.2: Le Plan des Occurrences

domaines (des ensembles) d'intervalles. Ainsi, l'intervalle de la figure 3.1 pourrait représenter le domaine des occurrences commençant après 2 et finissant avant 6, ou le domaine des mêmes occurrences de durée égale à 3, ou encore le domaine des occurrences dont le début n'est pas dans [4,5].

Cette ambiguïté provient de la nature bi-dimensionnelle d'une occurrence : cette dernière est complètement déterminée par deux paramètres : le début et la fin. C'est pourquoi nous utilisons, comme support graphique, le *plan des occurrences* (fig 3.2) d'axes (début,fin), dans lequel une occurrence est un point et un DOP est une région. Comme les occurrences commencent avant de finir, elles se trouvent forcément dans la region grisée de la figure 3.2. La diagonale principale a une signification particulière : c'est le lieu des occurrences dont le début égale la fin, c'est-à-dire le lieu des dates. Ceci relie les représentations bi-dimensionnelle et uni-dimensionnelle de la manière suivante : le début et la fin d'une occurrence o sont des dates et peuvent être projetés sur la diagonale (fig 3.2). Le segment défini par ces dates est l'ensemble des instants qui forment o, c'est donc la représentation uni-dimensionnelle de o.

La figure 3.3 est la représentation non-ambiguë des DOPs que la figure 3.1 ne peut représenter.

DOPs disjonctifs dans le cas linéaire

Nous n'avons pas besoin, en général, de faire d'hypothèse sur la forme des domaines d'occurrences. Cependant, signalons ici que nous nous sommes placés dans le cadre linéaire lors de la mise en œuvre (voir section 3.3). Dans ce cadre, on peut définir analytiquement chaque DOP par un système conjonctif et disjonctif d'équations et d'inéquations linéaires

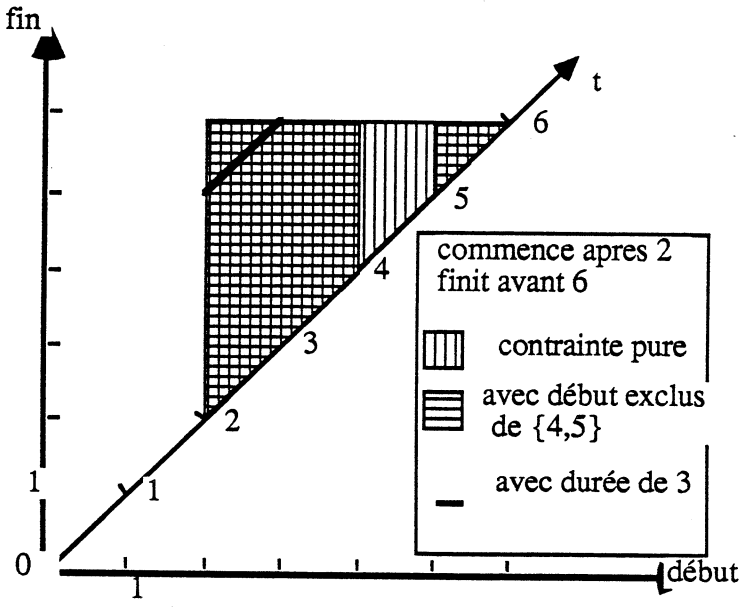


Figure 3.3: Les DOPs de la figure 3.1 dans une représentation bi-dimensionnelle

des variables début et fin[3].

De même, nous nous situerons implicitement dans ce cadre lorsque nous ferons allusion au DOPs *disjonctifs*. En effet, nous distinguons deux types de DOPs , les DOPs convexes, et les autres que nous qualifions de disjonctifs. La raison en est qu'une conjonction d'inéquations linéaires a toujours un ensemble convexe (éventuellement vide!) de solutions. Par conséquent, un DOP non-convexe, disjonctif, ne peut s'exprimer analytiquement sans disjonction. Il est "intrinsèquement" combinatoire.

3.2.2 Les Relations d'Allen

Les relations temporelles que nous considérons sont celles introduites par J.Allen (voir section 2.3.1). Nous faisons ici un bref rappel des propriétés que nous utilisons.

Allen a défini treize primitives définissant les positions relatives de deux intervalles (voir fig. 2.3).

Ces relations sont mutuellement exclusives et permettent de relier tout couple d'intervalles : tout couple d'intervalles datés est relié par une et une seule de ces relations primitives.

A partir des primitives, on peut construire des relations plus complexes : en effet, ces relations — que nous appellerons relations *composées* ou *disjonctives* — sont des *ensembles* de primitives que l'on peut définir de la manière suivante (en utilisant la notation d'Allen,

[3]début et fin de l'occurrence de l'événement associé au DOP que l'on définit.

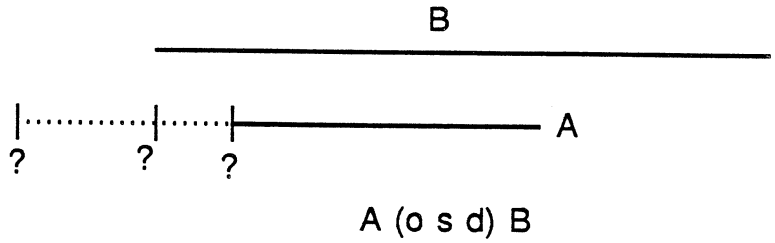


Figure 3.4: Un exemple de relation composée

$R = (r_i)_{i=1,n}$ est une relation disjonctive entre A et B) :

$$A(r_i)_{i=1,n}B \Leftrightarrow \bigvee_{i=1}^{n} A\, r_i\, B$$

C'est-à-dire, en utilisant le formalisme de la section 3.1.2 si a et b sont deux occurrences de A et B :

$$R(a,b) = 1 \Leftrightarrow \bigvee_{i=1}^{n} r_i(a,b) = 1$$

La figure 3.4 donne un exemple de relation composée.

Le fait de considérer ces relations comme des ensembles permet de définir très facilement la conjonction et la disjonction de relations composées comme, respectivement, l'intersection et l'union des ensembles qu'elles constituent. Ainsi, par exemple :

$$A(<>)B \wedge A(< d)B \Rightarrow A(<)B$$

et

$$A(<>)B \vee A(< d)B \Rightarrow A(<> d)B$$

3.2.3 Région Permise par une Occurrence et une Relation

On peut interpréter une relation temporelle R par rapport à une occurrence o au moyen de l'ensemble des occurrences x vérifiant R avec o :

$$\mathcal{P}(o,R) = \{x/R(o,x) = 1\}$$

Cet ensemble définit ce que nous appelons la région permise par o et R. En effet, la région étiquetée par le symbole $<$, par exemple, sur la figure 3.5 est celle de toutes les occurrences qui peuvent être après[4] o.

La figure 3.5 montre une "cartographie" du plan des occurrences obtenue à partir de toutes les primitives d'Allen. On pourrait imaginer d'autres moyens de partitionner ce plan, avec plus ou moins de primitives, nous avons gardé celui-là.

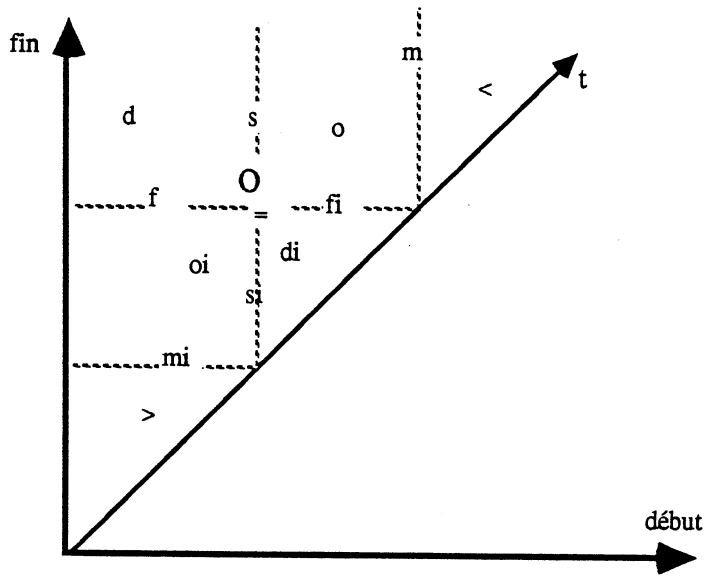


Figure 3.5: Interprétation des primitives d'Allen à l'aide de régions permises

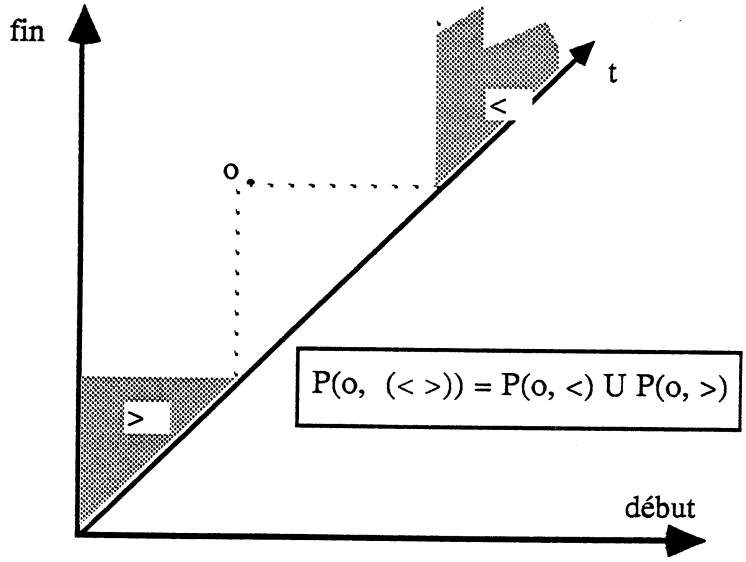


Figure 3.6: Région permise par une contrainte disjonctive

En terme de régions, une conjonction de relations peut être définie par l'intersection des régions permises par les clauses de la conjonction (fig 3.7). Une relation disjonctive peut être représentée par la réunion des régions permises par les clauses de la disjonction (fig 3.6). Ceci nous permet de distinguer les relations fortement, ou intrinsèquement, disjonctives des autres. Une relation est dite fortement disjonctive lorsqu'elle définit une région permise non convexe. L'expression analytique d'une telle relation, lorsqu'elle est linéaire, comporte nécessairement une disjonction. Ainsi, (<>) est fortement disjonctive, (< *om*) l'est "faiblement".

[4]Le symbole est <, car c'est la région des occurrences telles que *o* peut être *avant* elles.

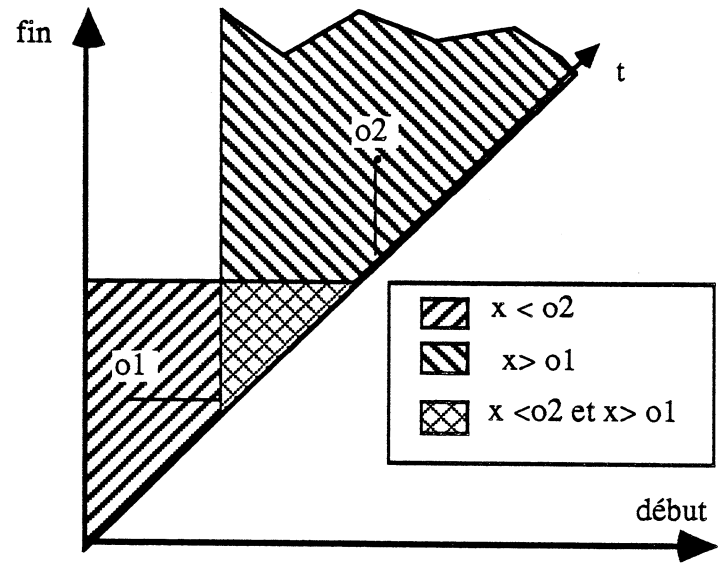


Figure 3.7: Région permise par une contrainte conjonctive

3.3 Remarques sur la mise en œuvre

Cette section contient un certain nombre de remarques relatives aux problèmes soulevés lors d'une mise en œuvre. Ces problèmes sont en général d'ordre conceptuel ("que va-t-on représenter pratiquement ?") mais leur résolution n'est pas nécessaire au reste de l'exposé. Le niveau de ces remarques est donc différent de ce qui précède et de celui du chapitre suivant, le lecteur pourra les sauter en première lecture.

3.3.1 Domaine d'Occurrences Possibles

Le Plan des Occurrences : Outil Conceptuel ou Outil d'Interface?

Nous utiliserons constamment dans ce texte le plan des occurrences comme un outil conceptuel de représentation. La tentation est grande de le considérer aussi comme un outil de représentation de plans pouvant être intégré dans une interface graphique. C'est ce que nous avons fait en réalisant l'interface graphique du module temporel. L'expérience est assez révélatrice : le problème de la représentation bi-dimensionnelle est que les DOPs ont tendance à se recouvrir et diminuent ainsi énormément la lisibilité. Pour pallier cet inconvénient il faudrait utiliser des outils graphiques plus sophistiqués que les fonctionnalités standard dont nous disposons. A cet égard, la figure 3.3 est assez représentative du niveau d'information nécessaire. Elle utilise des trames différentes, et souligne le DOP-segment d'un trait gras pour améliorer l'aspect du graphique.

En ce qui concerne l'interface du système de planification PASAN (voir la troisième partie), nous avons réalisé en plus une représentation unidimensionnelle "améliorée".

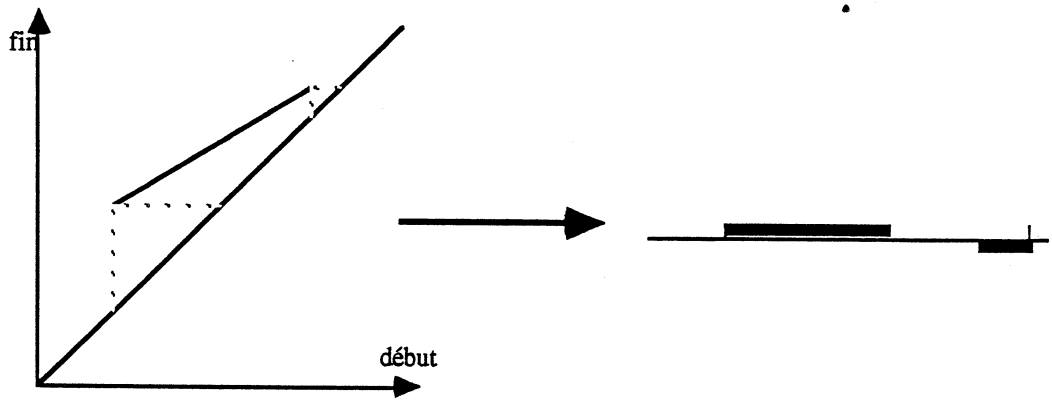


Figure 3.8: Une représentation uni-dimensionnelle de DOPs

L'amélioration provient d'une application du principe de projection : on projette un DOP sur la droite réelle, le transformant ainsi en "fenêtre", et on surcharge la fenêtre en portant sur le début et la fin de la fenêtre les durées correspondant à la première et à la dernière occurrence (fig 3.8). Comme dans ce cas particulier, la durée varie de manière monotone, un minimum d'information est perdu (bien que les DOPs disjonctifs soient assez mal représentés).

Des DOPs de Types Différents

Un DOP tel que nous l'avons formellement défini est un ensemble absolument quelconque. Autrement dit, c'est un ensemble quelconque de points du plan des occurrences : un patatoïde, un nuage de points, une courbe...

En pratique, on considère des classes particulières, ou types, d'ensembles. La raison principale est qu'une propagation de contraintes suppose beaucoup de manipulations sur les DOPs. De telles manipulations sont relativement aisées sur des régions simples (un point, un segment de droite, un rectangle...) mais pas sur des régions complexes (comme une courbe quelconque). La seconde raison est qu'en général, le domaine d'application ne met en jeu que certains types de contraintes temporelles, si bien que les DOPs ont un grand nombre de caractéristiques communes. Enfin, en typant un DOP, on peut utiliser un algorithme ad hoc pour le manipuler. Ainsi une intersection point-point mettra en jeu un algorithme beaucoup plus simple qu'une intersection rectangle-rectangle.

Nous considérons deux catégories de DOPs : les DOPs *composés* et les DOPs *élémentaires*. Les algorithmes "géométriques" d'intersection et d'union sont définis entre les DOPs élémentaires. Les DOPs composés sont des réunions de DOPs élémentaires. L'union et l'intersection de deux DOPs composés sont calculées en les décomposant en DOPs élémentaires.

Suivant le pouvoir de représentation et l'efficacité des opérations de manipulations que l'on désire obtenir, on peut définir des types plus ou moins riches de DOPs élémentaires.

A titre indicatif, nous donnons ici ceux que nous avons implantés et qui nous servent dans la troisième partie :

Les points Ce sont les DOPs réduits à une occurrence.

Les boites Ce sont des rectangles, définis par un début et une fin au plus tôt et au plus tard. Le système est assez robuste pour manipuler des boites qui débordent de la région des occurrences permises.

Les segments Dans une représentation à une dimension, on peut les voir comme des fenêtres, contenant des occurrences dont la durée est contrainte. Lorsque le durée est fixée, le segment est parallèle à la diagonale, sinon, il a une orientation quelconque.

Les segments disjonctifs Un segment disjonctif est l'union de segments ayant la même droite pour support. Nous les appelons aussi segments "troués". La figure 4.1 est construite à partir d'un segment disjonctif à durée fixée.

3.3.2 Relations Temporelles

Domaines Ouverts ou Fermés ?

Les primitives d'Allen sont définies au moyen d'inégalités strictes. Par exemple :

$$A(<)B \Leftrightarrow \text{fin}(A) < \text{debut}(B)$$

Les régions permises que ces relations définissent sont "ouvertes". Or nous avons trouvé commode d'avoir des DOPs fermés (qui contiennent leurs bords). En particulier parce que un DOP réduit à une occurrence est fermé et un segment de droite ne peut être ouvert et est fermé s'il contient ses extrémités. Pour éviter d'avoir à manipuler des fermés et des ouverts, nous avons choisi de ne manipuler que des ensembles fermés.

Ceci nous oblige a redéfinir les régions permises, donc la signification des primitives par rapport à celle de la figure 3.5. Ainsi "A avant B" autorise le début de B à être égal à la fin de A. Donc, $<$ dans notre implantation dénote $(< m)$ dans le formalisme que nous décrivons dans ce mémoire. Les autres primitives peuvent ainsi être redéfinies : di en $(di, si, fi, =)$, o en $(o, fo, s, =, m)$, f en $(= f)$ etc. on notera que ces relations sont disjonctives, mais le sont faiblement. Les opérations d'union et d'intersection conservant la propriété de fermeture, on ne manipulera que des ensembles fermés.

L'inconvénient d'un tel choix est une perte dans le pouvoir de représentation. On ne peut pas représenter de précédence stricte, puisqu'on vient de la remplacer. Plus généralement, les nouvelles primitives n'étant pas mutuellement exclusives, le calcul symbolique sur les relations disjonctives est compliqué si on ne revient pas dans le formalisme d'Allen. De plus, la composition de deux nouvelles primitives peut en donner une ancienne : par exemple,

$m \circ m =<$, si on remplace $<$ par $(< m)$, on laisse passer une inconsistance. Enfin, il n'y a pas de négation : si non$(<)$ avec les primitives implantées dénote non$(< m)$ au sens d'Allen, la négation vaut $(>, mi, oi, si, di, fi, =, f, d, s, o)$ qui ne peut s'exprimer avec de nouvelles primitives.

En pratique, on peut tourner la difficulté en s'appuyant sur une discrétisation sous-jacente, peu élégante. Par exemple, "après 14h00 strictement" s'exprimera en fait "après 14h01 largement".

Les délais

Nous avons enrichi les primitives temporelles de la notion de délai, de manière à pouvoir coder directement des relations comme *se baigner une heure après avoir mangé.*

Une telle construction n'ajoute rien au pouvoir "brut" de la représentation, il constitue simplement une amélioration de la souplesse d'utilisation.

Intuitivement, ajouter un délai δ à une relation entre A et B revient à intercaler un événement fictif de durée δ entre A et B. Formellement :

- On note $(R\ \delta)$ une primitive avec délai, R étant une primitive temporelle et δ un nombre réel positif ou nul, $(R) \equiv (R\ 0)$
- Si $R \in \{o, m, <, fi, s\}$ (primitives de type "avant")

  $$A\,((R\ \delta))\,B \Leftrightarrow A(m)A' \wedge A'(R)B$$

  où A' est un événement de durée δ.
- Si $R \in \{oi, mi, >, f, si\}$ (primitives de type "après")

  $$A\,((R\ \delta))\,B \Leftrightarrow B\,\left((R^{-1}\ \delta)\right)\,A$$

  où R^{-1} est la relation inverse de R.
- Si $R \in \{d, di, =\}$

  $$A\,((R\ \delta))\,B \Leftrightarrow A(R)B$$

  Autrement dit, le délai n'a pas d'effet pour ces primitives (car nous n'avons pas trouvé très claire la signification qu'il aurait).

On n'a pas intérêt, dans une implantation, à introduire des événement fictifs. De tels événements alourdissent le mécanisme de propagation de contraintes temporelles s'ils sont trop nombreux, et on peut s'en passer en codant directement l'effet du délai dans la définition de chaque primitive[5].

[5] C'est a dire dans le calcul de la région permise (voir le chapitre suivant).

Chapitre 4

Résolution d'un Problème d'Occurrences Contraintes, Comparaison avec d'autres Formalismes

4.1 Résolution d'un Problème d'Occurrences Contraintes

La fonction d'un module de propagation de contraintes temporelles est de résoudre le POC qu'on lui fournit[1]. L'algorithme que nous allons décrire nous semble le plus naturel compte-tenu de la formalisation du problème et du mode de représentation que nous avons adoptés.

4.1.1 Région Permise, Raffinement, Dérivation Locale

Région permise

On peut étendre la notion de région permise par une occurrence à celle de région permise par un DOP O_1 et une relation R :

$$P(O_1, R) = \{x / \exists o \in O_1 / R(o, x) = 1\}$$

qui est aussi la réunion des régions permises par les occurrences de O_1. La figure 4.1 montre la région permise par un DOP — ici un segment disjonctif — et chaque relation primitive.

Si $R = R_{12}$ est définie entre les DOPs O_1 et O_2, toute occurrence de O_2 qui n'appartient pas à cette région n'appartient à aucune solution. Par conséquent, on peut éliminer de

[1]En regard de cette fonction, le terme "module de propagation de contraintes temporelles" est un peu abusif, car la propagation de contraintes ne recouvre qu'une partie des méthodes algorithmiques que l'on peut utiliser pour résoudre un POC.

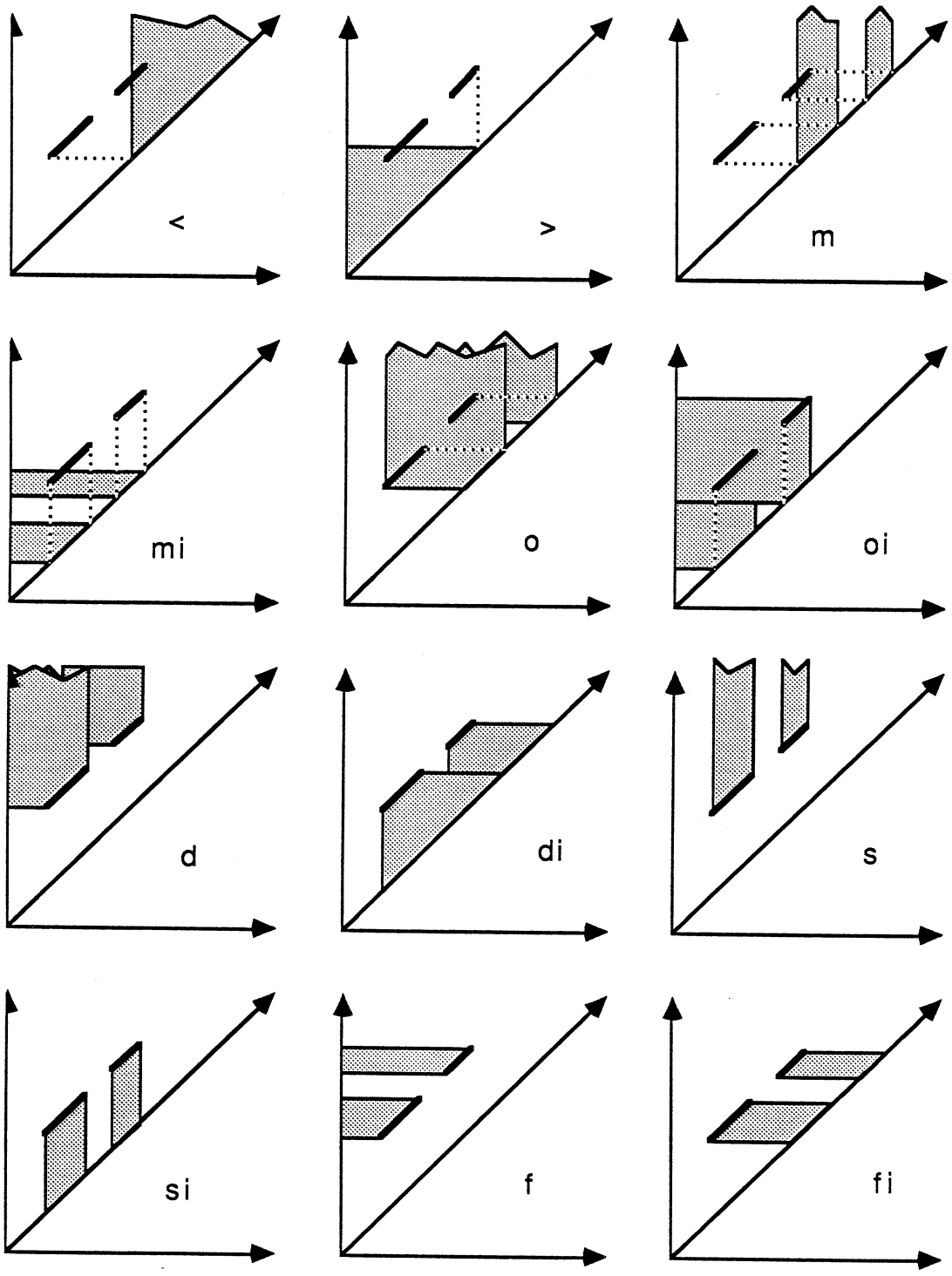


Figure 4.1: Exemples de régions permises

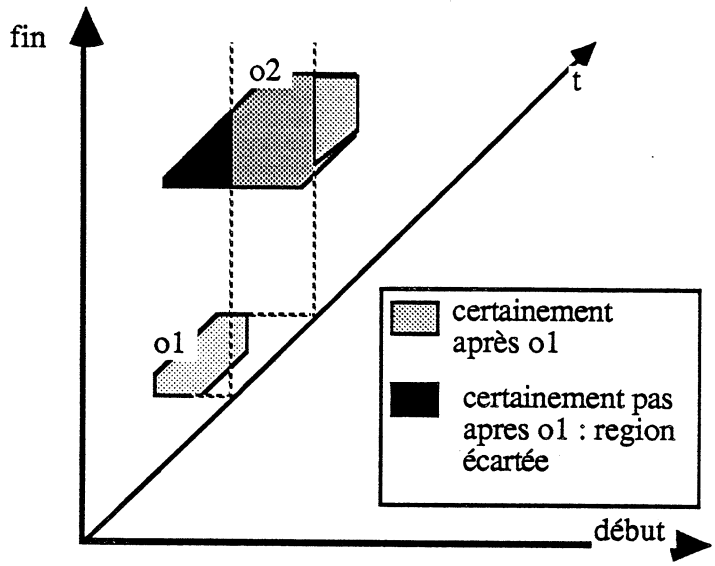


Figure 4.2: Propagation d'une relation temporelle sur un DOP

telles occurrences lors du processus de résolution, en étant sûr de garder la propriété de complétude du POC dérivé et en améliorant sa consistance. Nous appellerons *raffinement* cette opération ρ d'intersection de O_2 avec la région permise par O_1 et R :

$$\rho(O_1, R_{12}, O_2) = \mathcal{P}(O_1, R_{12}) \cap O_2$$

On pourrait aussi considérer la région "hyper-permise" par O_1, qui est l'intersection des régions permises par les occurrences de O_1. Construire l'opération de raffinement sur cette région serait une dérivation hyper-consistante, mais probablement pas complète. Elle peut cependant être intéressante si O_1 est le DOP d'un événement sur lequel on a aucun contrôle. En effet, avec une telle dérivation, toutes les solutions sont "insensibles" au choix d'une occurrence dans O_1.

Propagation de Contraintes

Nous appelons propagation de la contrainte R_{12} l'application de l'opération de raffinement de O_1 sur O_2. Le principe de l'algorithme de résolution est que chaque fois qu'un DOP est modifié, les régions permises par lui le sont aussi et que donc cette modification doit être propagée par les opérateurs de raffinement. Cet algorithme permet par définition de donner au POC la propriété de consistance d'arcs.

D'où l'algorithme de dérivation locale du POC $\left((O_i)_{i=1,n}, (R_{ij})_{i,j\in[1,n]}\right)$, en notant A l'ensemble de contraintes en attente de propagation :

- $A \leftarrow \{R_{ij}\}_{i,j\in[1,n]}$ *Initialisation de* A.
- Tant que $A \neq \emptyset$

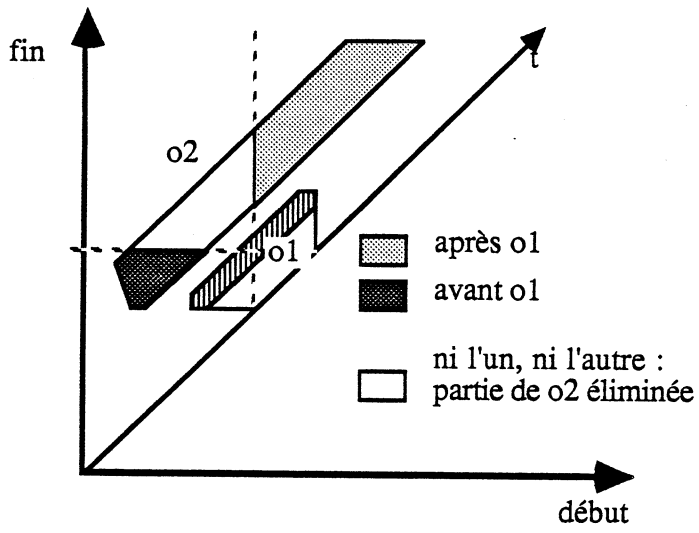


Figure 4.3: Propagation d'une Contrainte Disjonctive Créant un "Trou" dans un DOP

1. Retirer un élément R_{ij} de A
2. $O'_j \leftarrow \rho(O_i, R_{ij}, O_j) \cap O_j$, *Raffinement de* O_j
3. Si $O_j \neq O'_j$, alors
   - $A \leftarrow A \cup \{R_{jk}, k \in [1,n]\}$, *La modification est propagée.*
   - $O_j \leftarrow O'_j$

Notons qu'avant d'effectuer l'étape 3, on peut vérifier si O'_j est vide, auquel cas on peut s'arrêter tout de suite : l'un des événements ne peut avoir d'occurrence. Si on continue quand même, l'algorithme s'arrête quand tous les DOPs sont vides. Notons aussi que lors d'une propagation, il n'est pas nécessaire d'ajouter dans A tous les R_{jk}. Seules les relations susceptibles de propager réellement la modification effectuée sur O_j doivent être ajoutées. Ainsi, si O_j est un rectangle dont seule la fin au plus tôt est modifiée, les relations à propager sont $<$, $=$, m, o, d, s, f et fi.

Une caractéristique intéressante de cet algorithme[2] est qu'il permet, à travers les régions permises, de propager, à partir de DOPs éventuellement disjonctifs, des contraintes disjonctives. D'ailleurs, ces dernières, transforment généralement les DOPs convexes en DOPs disjonctifs (fig 4.3). L'information propagée à partir d'un DOP disjonctif est alors en quelque sorte commune à toutes les composantes du DOP, et peut être très vague si ces composantes sont éloignées les unes des autres.

[2]Cette caractéristique en est vraiment une : notre algorithme est le seul, à notre connaissance, qui propage véritablement des disjonctions.

4.1.2 Complexité

Nous avons vu lors de la section précédente que le principe de l'algorithme était de propager chaque modification de DOP. La complexité de l'algorithme est donc :
nombre de modifications $\times$ coût d'une propagation
$\simeq$ (nombre de DOPs modifiés $\times$ nombre de modifications par DOP) $\times$ (nombre de relations par DOP $\times$ coût d'un raffinement)

$$\simeq n \times e \times r \times c$$

En fait, l'évaluation de cette expression n'est pas très aisée :

n Représente le Nombre d'événements qui est fixé.

e Mesure l'efficacité d'un raffinement, $e = 1$ signifie que chaque DOP est modifié une seule fois. Malheureusement, la propagation peut mal se passer en cas de relations contradictoires. Par exemple, sur la figure 4.4 $e = 4$ pour A et B : il faut appliquer quatre fois A *avant* B et B *avant* A avant de détecter la contradiction.

r est le nombre moyen de relations concernant un événement. Si r est grand, il y aura plus de contrainte à propager, mais elles ont plus de chance d'être efficace, ce qui diminue e.

c dépend de la "forme" des DOPs et du nombre moyen de primitives, c est ainsi directement lié au degré de disjonction du problème. Ce paramètre dépend aussi fortement de l'efficacité des algorithmes "géométriques" (intersection et réunion de DOPs) mis en œuvre.

A titre d'exemple, dans PASAN (voir chapitre 6), la complexité est de l'ordre de cn^2, car e est presqu'égal à 1 et les POCs sont des graphes complets (de disjonctions de deux clauses). L'ordre de grandeur de c est de 0.2 secondes, une propagation sur dix nœuds prend donc environ 20 secondes.

Il faut cependant insister sur le problème théorique que pose le cœfficient e. Il est évident, d'après la figure 4.4 que si les DOPs ne sont pas bornés, l'algorithme ne termine pas. De plus, même si les DOPs sont bornés, il est possible de construire, par exemple en rapprochant A et B sur la figure 4.4, une propagation arbitrairement longue, voire interminable (fig 4.5). Il est par conséquent nécessaire d'introduire un paramètre de granularité ε, strictement positif, fixant une limite inférieure de modification de DOP. Moyennant ces deux restrictions, le fait que les DOPs sont modifiés de manière monotone garantit la terminaison.

4.1.3 Consistance

Nous avons appelé l'algorithme que nous venons de donner algorithme de résolution *locale*. En effet, chaque DOP/nœud du POC/graphe n'est influencé explicitement que par ses voisins. L'effet des nœuds plus éloignés, c'est-à-dire des contraintes implicites peut ainsi être

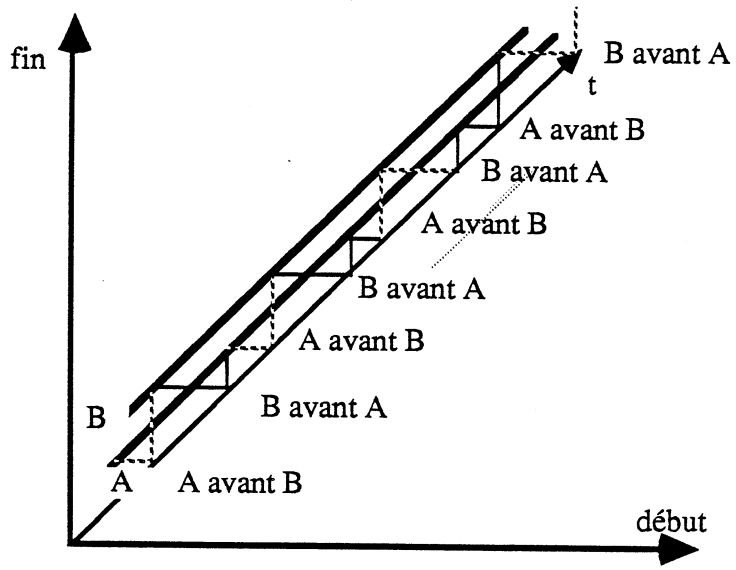


Figure 4.4: Une contradiction alourdit la propagation

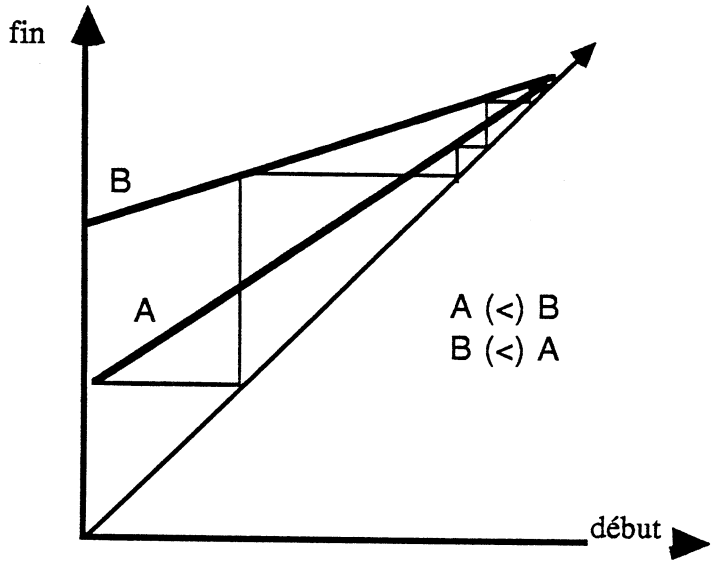


Figure 4.5: Un cas où l'algorithme ne termine pas

masqué par les voisins, ce qui peut laisser des occurrences qui n'appartiennent à aucune solution. En un mot, la dérivation locale qui permet d'obtenir une consistance d'arcs est plus faible, comme le montre la figure 4.6 que la résolution qui implique une consistance globale.

Pour des raisons de concision, nous ne donnons que des dessins. Le lecteur est invité à vérifier pour chacun des deux POCs de la figure 4.6 que :

- Les POCs sont localement consistants : tous les DOPs sont bien inclus dans l'intersection de régions qui leur sont permises.

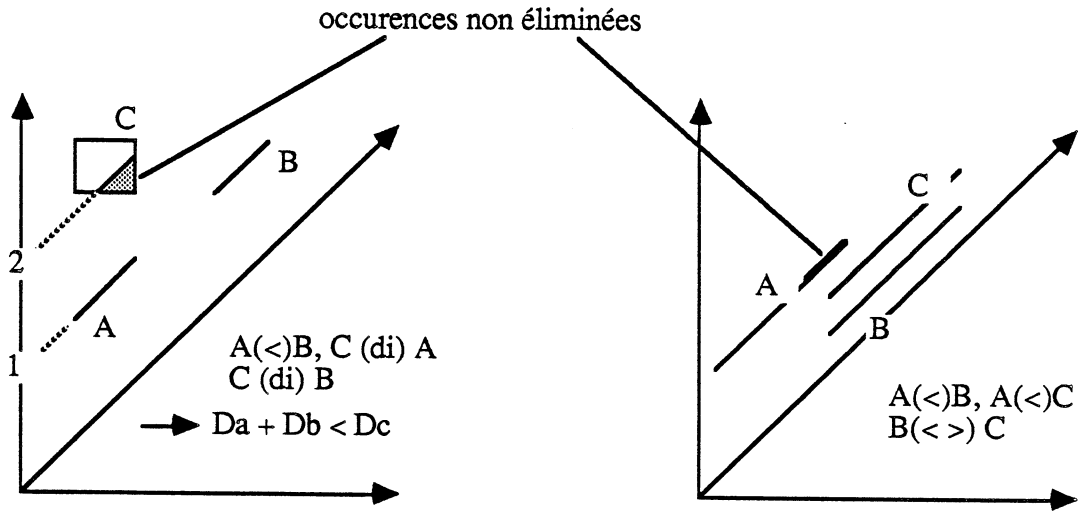


Figure 4.6: Occurrences inconsistantes non détectées par une dérivation locale

- Mais ils ne sont pas globalement consistants : les occurrences signalées comme impossibles sur la figure n'appartiennent à aucune solution. Ceci peut être montré en constatant que la propagation du choix d'une occurrence impossible conduit à vider un des DOPs[3].

L'intérêt d'une méthode de propagation de contraintes comme celle que nous venons de décrire est donc d'agir comme un filtre en éliminant la plus grande partie des inconsistances. Un tel point de vue suppose donc que les algorithmes qui utilisent les résultats d'une propagation de contraintes soient suffisamment robustes pour éliminer quelques inconsistances et soient capables de bénéficier du dégrossissage opéré par la propagation. On peut voir à la section 6.3.2 comment PASAN résout ce problème de robustesse.

Une autre possibilité est de ne se placer que dans les cas de figure pour lesquels la dérivation locale est suffisante pour assurer la consistance. Par exemple (Annexe A) si les DOPs sont des segments "montants" et en l'absence de relations disjonctives, la consistance d'arcs implique la consistance globale. De même, si les DOPs sont des fenêtres dont les éléments sont contraints seulement par une durée minimale et maximale, en l'absence de relations disjonctives, on peut appliquer les méthodes évoquées à la section 2.3.2.

Dans tous ces cas, il faut que les DOPs aient une forme précise, et surtout, que l'énumération des disjonctions ne conduise pas à une explosion combinatoire.

[3]Le lecteur pourra aussi remarquer que *toutes* les occurrences impossibles sont signalées. Si on retire les parties en gras, les POCs résultants sont consistants globalement.

4.2 Comparaison du formalisme DOP avec les autres

Nous nous sommes déjà efforcés de souligner les avantages et les défauts de chacune des représentations du temps que nous avons exposées. Encore qu'il semble difficile de comparer sur un même plan une représentation symbolique comme celle d'Allen, qui vise essentiellement à *structurer* un ensemble d'intervallles, et une représentation numérique comme la formulation PERT qui permet de calculer des quantités sur une structure déjà établie.

Nous nous lancerons toutefois dans une telle comparaison, mais en adoptant une démarche constructive et utilitaire, en donnant des algorithmes qui permettent de passer d'une représentation à l'autre :

- Tous les formalismes n'étant pas équivalents, les algorithmes de passage de l'un à l'autre ne conservent pas toutes les caractéristiques de la représentation d'un "problème temporel". Mettre en évidence ces caractéristiques est un aspect important de cette partie. En outre, certaines d'entre elles peuvent être traduites, mais de manière lourde ou maladroite car les intentions des concepteurs ne coïncident pas. Trouver ces "lourdeurs" nous permet donc de cerner plus finement ces intentions que par l'examen direct.

- Chaque formalisme ayant ses mérites, l'existence d'algorithmes de traduction nous permet d'envisager l'adoption de plusieurs points de vue différents pour mieux résoudre un problème temporel complexe. Envisager seulement, car la coordination efficace de plusieurs modes de représentation constitue un problème en soi. Nous espérons seulement en poser quelques bases pour le raisonnement temporel.

Nous n'avons pas voulu faire une comparaison deux à deux de tous les formalismes que nous avons évoqués, cela nous conduirait à un exposé trop long. Notre étude sera donc centrée sur le formalisme que nous venons d'exposer (DOP). En effet, d'une part il est nouveau donc il est intéressant de le confronter à des représentations plus éprouvées; d'autre part, le formalisme DOP intègre des aspects assez généraux donc la comparaison avec d'autres représentations a plus de chances d'avoir un sens.

Nous avons donc choisi de comparer la représentation DOP avec la représentation d'Allen, symbolique, et une représentation numérique fondée sur des techniques classiques de programmation linéaire. Les deux sont suffisamment contrastées pour que l'étude soit féconde, elles sont assez puissantes pour que les algorithmes de traduction aient éventuellement un intérêt pratique et enfin, elles sont décrites avec soin dans la littérature.

4.2.1 Comparaison avec le Formalisme d'Allen

Pourquoi Passer d'un Formalisme à l'autre?

La représentation d'Allen constitue un des éléments que nous avons utilisés pour construire DOP. En effet, elle s'appuie sur un graphe dont les arcs sont étiquetés par une relation symbolique disjonctive et les nœuds dénotent des intervalles. La représentation DOP aussi,

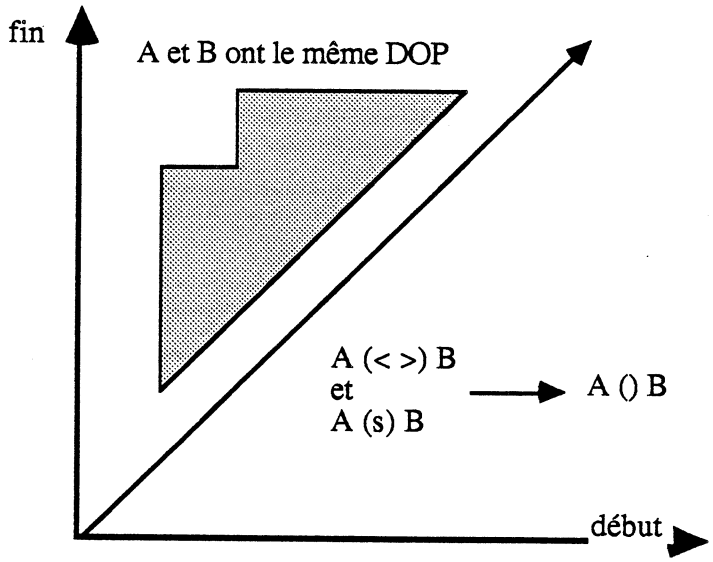


Figure 4.7: Une contradiction non détectée par l'algorithme de résolution locale mais détectable par l'algorithme d'Allen

mais de plus, on associe aux nœuds les bornes numériques d'un domaine de valeurs pour les intervalles. Il est donc naturel de considérer, à première vue, que la formalisation d'un problème temporel par une représentation d'Allen est la simplification de la formulation du même problème avec une représentation DOP. Sous cet angle, la représentation d'Allen contient la partie structurelle de la représentation DOP.

Par conséquent, traduire la formulation d'Allen d'un problème en formulation DOP n'a aucun intérêt. La transformation est triviale : le graphe est conservé, et on ajoute à chaque nœud le domaine maximal (par exemple {début $>$ 0 $\wedge$ fin $<$ fin-des-temps}). Le seul effet est d'employer une représentation plus compliquée que l'originale pour des résultats moins bons, l'algorithme de résolution locale étant moins puissant que celui d'Allen lorsqu'aucune information numérique n'est exploitable.

Par contre, passer d'une formulation DOP à une formulation d'Allen est intéressant pour deux raisons :

- Du point de vue de la représentation des connaissances : lorsque l'on calcule un plan, on est souvent plus intéressé par la structure du plan résultant — qui est une information synthétique — que par les domaines d'occurrences des actions, même si ces derniers ont été nécessaires pour calculer la première[4].

- Du point de vue de la propagation de contraintes : les algorithmes utilisés dans l'une et l'autre représentation sont différents et celui d'Allen détecte des contradictions que

[4]Au risque bien sûr de commettre les grosses erreurs de raisonnement que l'on fait en utilisant un résultat (la structure du plan) en "oubliant" les hypothèses qui ont permis de le calculer!

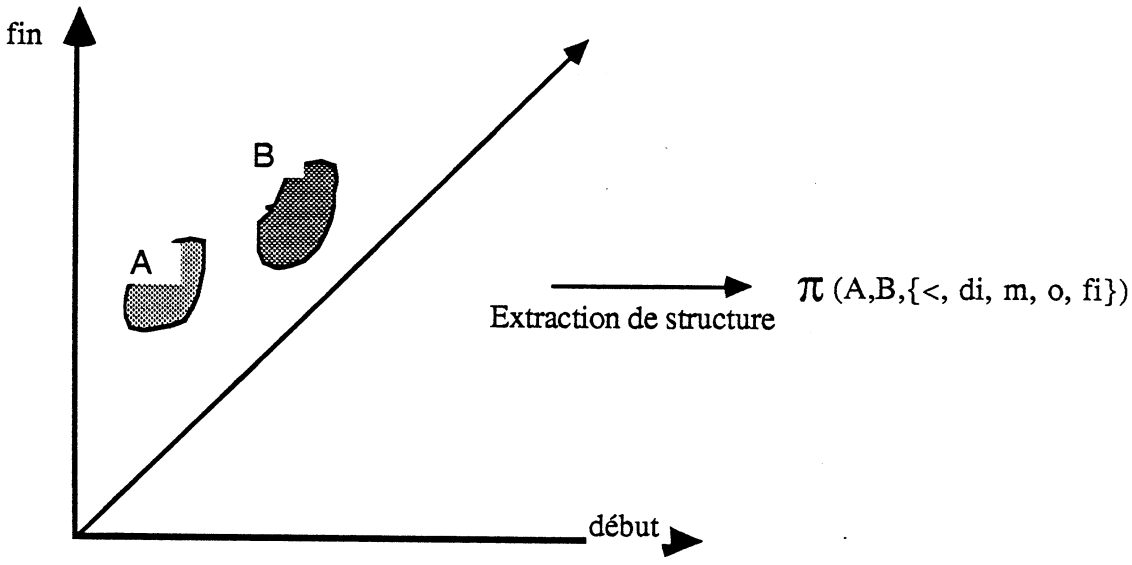


Figure 4.8: Exemple d'Extraction de la Relation Symbolique entre deux DOPs

l'algorithme de dérivation locale reconnait péniblement (fig. 4.4) ou pas du tout (fig. 4.7).

Nous abordons donc dans la suite le problème de traduction d'un formalisme DOP en formalisme d'Allen.

Algorithme Trivial et Extraction de Structure

L'*algorithme trivial* consiste à dupliquer le graphe du POC en retirant les DOPs. Mais en n'utilisant que l'algorithme trivial, on perd toute "l'information" contenue dans l'expression des domaines. Considérons par exemple le POC $(\{A, B\}, \{(<>)\})$ où A et B sont les DOPs de la figure 4.8. L'algorithme trivial établira la relation $(<>)$ entre A et B, mais les DOPs sont tels que A ne peut être après B.

L'*algorithme d'extraction de structure* qui est une conséquence immédiate de la définition d'une région permise par un DOP et une relation temporelle permet d'expliciter les relations symboliques entre deux DOPs que l'on peut déduire de leurs frontières numériques :

Notations Soit un POC $\mathcal{P} = \left((O_i)_{i\in[1,n]}, (R_{ij})_{i,j\in[1,n]}\right)$. On notera r une relation primitive, R^{-1} la "réciproque" d'une relation R ($ARB \Leftrightarrow BR^{-1}A$) et $\rho(O, R)$ la région permise par le DOP O et la relation R.

Algorithme *Cet Algorithme construit, à partir des DOPs de $\mathcal{P}$, un graphe d'Allen :*

$$\left((a_i)_{i\in[1,n]}, (A_{ij})_{i,j\in[1,n]}\right)$$

- Créer les n nœuds du graphe d'Allen
- Pour $i = 1, n-1$
  - Pour $j = i+1, n$
    - $A_{ij} \leftarrow \emptyset$
      - Pour $r \in \{<, >, m, mi, d, di, s, si, o, oi, f, fi, =\}$
      - Si $O_j \cap \rho(O_i, r) \neq \emptyset$ Alors $A_{ij} \leftarrow A_{ij} \cup \{r\}$
    - $A_{ji} \leftarrow A_{ij}^{-1}$

Justification

- On définit la fonction booléenne π prenant comme arguments respectivement deux DOPs et une relation (d'Allen), que l'on interprète comme *il est possible que* $O_i R O_j$, par :

$$\pi(O_i, O_j, R \cup R') = \pi(O_i, O_j, R) \wedge \pi(O_i, O_j, R') \quad (4.4)$$

$$\pi(O_i, O_j, \{r\}) \Leftrightarrow \exists (o_i, o_j) \in O_i \times O_j / r(o_i, o_j) = 1 \quad (4.5)$$

- Les propositions suivantes sont alors équivalentes :

$$\pi(O_i, O_j, \{r\}) \quad (4.6)$$

$$O_j \cap \rho(O_i, r) \neq \emptyset \quad (4.7)$$

$$O_i \cap \rho(O_j, r^{-1}) \neq \emptyset \quad (4.8)$$

- L'algorithme précédent calcule, pour chaque paire de DOPs O_i, O_j le plus grand R tel que $\pi(O_i, O_j, R)$.

La figure 4.8 donne un exemple d'extraction de toutes les relations possibles entre les DOPs A et B. Le lecteur est invité à vérifier que pour chaque primitive P qui est un argument de π, on peut trouver une occurrence de A et une occurrence de B liées par P. Une manière de trouver ces occurrences est d'en choisir une dans l'intersection de B (par exemple) et de la région permise par A et P.

4.2.2 Comparaison avec un Formalisme Linéaire Simple

Nous comparons dans cette section le formalisme DOP avec le formalisme fondé sur la recherche des plus longs chemins dans un réseau, dont les arcs sont valués par un réel. Ce dernier a déjà été évoqué à la section 2.3.2 . Nous le désignerons dans la suite par "formalisme de réseau" ou "formalisme linéaire", même si ces termes doivent être pris ici dans un sens restreint.

Contrairement à la comparaison avec le formalisme d'Allen, nous ne commencerons pas par donner un exemple qui montre les lacunes de DOP par rapport au formalisme de réseau, indiquant ainsi l'intérêt pratique de passer d'un formalisme à l'autre. Un tel exemple est donné dans la suite (fig. 4.12) et le lecteur peut le consulter dès maintenant, mais nous pensons qu'il sera mieux compris après l'exposé des méthodes de traduction.

Traduction d'un POC en Réseau d'Arcs Valués

Le formalisme DOP distingue les contraintes individuelles, numériques, et les contraintes relationnelles, symboliques. Nous diviserons donc la traduction d'un POC en deux parties : traduction des relations puis des domaines.

1 : Traduction des Relations Ce problème est équivalent à la traduction d'un réseau d'Allen en formalisme linéaire. On peut le découper en deux étapes consécutives :

1. Traduction d'un réseau d'Allen en réseau symbolique ponctuel
2. Traduction du réseau symbolique ponctuel en réseau numérique.

Rappelons que dans un réseau d'Allen, les nœuds dénotent des intervalles alors que dans un réseau ponctuel ils dénotent des points. Les relations symboliques valuant les arcs de ces réseaux sont donc différentes. Le lecteur pourra se reporter à la section 2.3.1 pour plus de détails.

La traduction d'un réseau d'Allen en réseau ponctuel est un problème déjà résolu par T. Granier [23]. Nous en rappelons seulement les grandes lignes :

1. Création des sommets : our tout sommet, dénotant un intervalle du graphe d'Allen, on crée deux sommets ponctuels dénotant le début et la fin de cet intervalle.
2. Etablissement des liens :
   - Le début et la fin de chaque intervalle sont reliés par la relation (<=)
   - Traduction des relations entre intervalles : Ces relations étant binaires, le problème se ramène à la traduction de liens entre deux intervalles. Une relation entre deux intervalles est transposée en quatre relations entre les deux débuts et les deux fins de ces intervalles.

     La traduction d'une primitive est effectuée grâce à une table établie par T. Granier (donnée en annexe B).

     La traduction d'une relation disjonctive entre intervalles, qui est une réunion de primitives, s'effectue facilement en calculant l'union des relations entre points issues de la traduction de chacune des primitives. Il faut noter ici que la traduction de certaines disjonctions entraine une perte d'information. Par exemple, la traduction de (<>) entre deux intervalles donne le même résultat, sur le plan ponctuel, que celle de la relation indéterminée.

L'étape suivante, qui consiste en la traduction d'un réseau ponctuel symbolique en réseau numérique est facile. Elle consiste *grosso modo* à établir entre deux nœuds A et B un arc direct si $A(<)B$, un arc inverse (de B vers A) si $A(>)B$ et les deux si $A(=)B$, tous ces arcs étant valués par 0. L'absence de relation, (dénotée par la relation symbolique (<>=), peut

$A(R)B$	Arc de A vers B	Arc de B vers A
(<)	ε	$-\infty$
(>)	$-\infty$	ε
(<=)	0	$-\infty$
(>=)	$-\infty$	0
(<>=)	$-\infty$	$-\infty$
(=)	0	0

On peut remplacer ε par 0 — auquel cas on perd le fait que l'inégalité est stricte — et $-\infty$ par une absence d'arc.

Figure 4.9: Table de conversion entre relations symboliques et relations numériques

être représentée par une absence d'arc ou un arc valué par $-\infty$. Un autre détail à régler est la prise en compte des inégalités strictes dénotées par le graphe symbolique (comme c'est le cas, par exemple, pour la relation $A(<)B$) alors qu'il est préférable que le graphe numérique dénote des inégalités larges. Si on désire conserver le sens de ces inégalités strictes, on peut introduire une valeur ε dénotant un réel positif infiniment petit à condition de définir son comportement vis-à-vis de l'addition et de l'ordre sur les réels.

La table de conversion complète est donnée à la figure 4.9.

2 : Traduction des Domaines On peut, là aussi, procéder en deux étapes : traduction des contraintes sur la durée et de celles sur le début et la fin. Ces contraintes sont en fait les bornes (min et max) encadrant ces trois paramètres. Géométriquement, ces contraintes correspondent à une enveloppe à six côtés du domaine que l'on veut traduire (fig. 4.10) ; dans le cas général, elle est bien sûr plus large que domaine en question.

La représentation de contraintes sur la durée dans le formalisme de réseau consiste à établir, entre les points dénotant le début et la fin d'un intervalle, un arc direct valué par la durée minimale des occurrences du DOP de cet intervalle[5] et un arc inverse valué par l'opposé de la durée maximale.

Les contraintes sur le début et la fin nécessitent l'introduction d'un point de référence correspondant à l'origine O du plan des occurrences possibles. Les points de début et de fin correspondant à chaque DOP sont alors reliés à l'origine conformément à la figure 4.10.

[5]On remarquera que cette opération rend redondant l'établissement entre le début et la fin d'un arc valué par (<), lors de l'opération de traduction des relations entre intervalles en relations symboliques entre points. Une telle redondance permet que le graphe ponctuel symbolique soit plus "complet", au cas où l'on désire travailler sur cette structure intermédiaire.

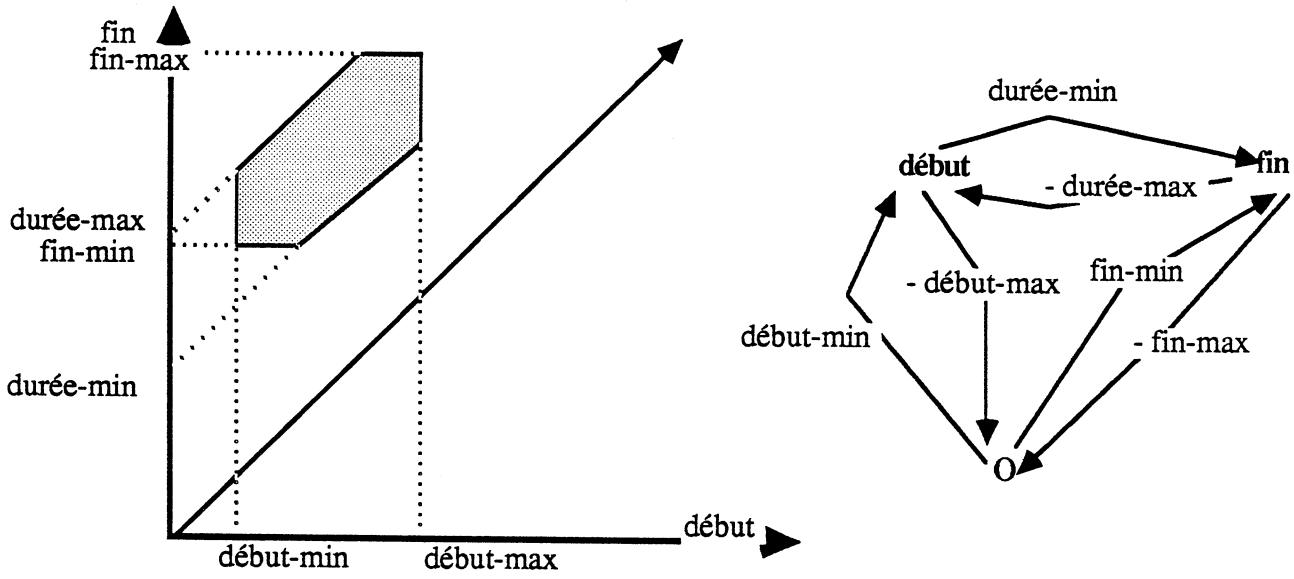


Figure 4.10: Représentation d'un DOP sur un graphe ponctuel valué

Traduction d'un Réseau d'arcs valués en POC

La traduction d'un réseau numérique ponctuel en POC est décomposée de la même façon que la transformation réciproque que nous venons de voir : traduction des relations et des domaines. Nous supposons cependant que dans l'ensemble des points constituant le réseau, on a déjà identifié des intervalles, c'est-à-dire des couples de points (début,fin). Ceci n'empêche pas que le réseau contiennent aussi des points "isolés" dénotant des instants. Nous supposons même qu'il en existe au moins un, considéré comme origine, antérieur (c'est-à-dire lié par un arc positif) à tous les autres.

1 : Traduction des domaines La distinction entre relations et domaines est moins explicite dans un réseau numérique que dans le formalisme DOP. Les arcs reliant le début et la fin d'un même intervalle, ainsi que les arcs passant par l'origine permettent d'exprimer des domaines. Les autres arcs relient des points appartenant à des intervalles différents et servent à exprimer des relations entre domaines.

La traduction elle-même est facile : à chaque intervalle identifié dans le réseau, on associe un DOP défini par des bornes sur le début, la fin et la durée conformément au schema de la figure 4.10.

2 : Traduction des relations Le principe de cette partie est analogue à celui de la partie réciproque. Il fait intervenir les deux mêmes étapes : passage du formalisme numérique ponctuel à un formalisme symbolique ponctuel puis à un formalisme symbolique d'intervalles.

La première étape peut être atteinte au moyen de la table de la figure 4.9. Un problème

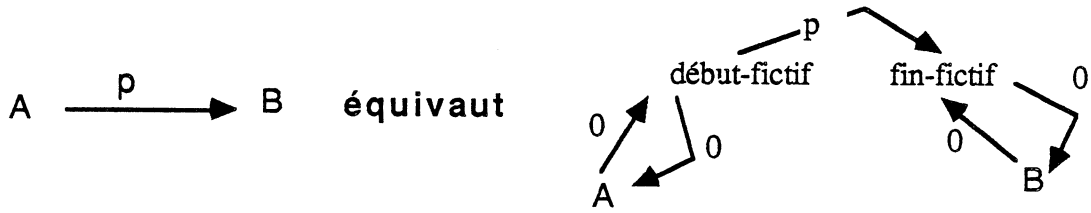


Figure 4.11: Représentation d'un arc valué, par un intervalle explicite fictif

reste cependant à régler : le lecteur peut remarquer que cette table ne comprend que des arcs valués par 0, ε ou $-\infty$. Par conséquent, si on n'utilise que cette table, on perd les informations numériques relationnelles. Par exemple, "la voiture démarre au moins cinq secondes après le passage du feu au vert" est traduit au mieux par "la voiture démarre après le passage du feu au vert". Un moyen de ne pas perdre cette information est d'introduire des intervalles correspondant à des événements fictifs à la place des arcs valués qui relient des points appartenant à deux intervalles distincts (figure 4.11). La valeur numérique de la relation devient alors une contrainte sur un domaine (plus précisément sur sa durée) que l'on peut représenter en formalisme DOP. En reprenant l'exemple précédent, on peut dire que l'on introduit un intervalle correspondant à l'événement "réaction du conducteur". Cet événement dure au moins cinq secondes, est consécutif au passage du feu au vert et le démarrage lui est consécutif.

La seconde étape est aussi possible grâce à une table établie par T.Granier. Cette table donne, pour chacune des relations primitives ponctuelles possibles sur un des quatre arcs reliant le début ou la fin de deux intervalles, la relation disjonctive possible entre les deux intervalles en question (voir en annexe B).

le Pouvoir Représentatif de DOP est le plus Fort

En effet, nous affirmons (sans le démontrer ici) qu'un réseau d'arcs valués quelconque peut être traduit en formalisme DOP puis retraduit en un réseau équivalent, au sens où il dénote les mêmes intervalles reliés par les mêmes relations, au moyen des méthode que nous venons de donner. Par contre, il existe des POCs qui ne peuvent être traduits en réseau puis retraduits en un POC équivalent (et nous supposons avec cette seconde affirmation que les méthodes de traduction que nous avons données sont les meilleures en termes de représentation). Ceci est vrai tant en ce qui concerne les domaines que les relations :

- Un DOP représenté par un segment de droite, traduisant une dépendance linéaire entre le début d'une occurrence et sa durée n'est pas représentable dans un réseau. On peut l'encadrer par ses durées minimales et maximales, par exemple (c'est ce qui se passe lorsque la méthode de traduction que nous avons donnée est employée (fig. 4.10)), mais

la transformation inverse ne redonnera plus un segment. Les DOPs de mobilisation des systèmes d'armes (section 6.2.3 et figure 6.3) sont des exemples de tels DOPs.

- Une relation disjonctive telle que (<>) a la même traduction en réseau ponctuel que la relation la plus générale (c'est-à-dire l'absence de relation). La perte est évidente.

Comparaison des Algorithmes : Avantage aux Réseaux

Nous venons d'affirmer que, du point de vue de la représentation explicite, le formalisme DOP "contient" le formalisme de réseau. Mais qu'en est-il des algorithmes de manipulation d'un POC (ou d'un réseau) qui explicitent les occurrences possibles ?

Tout d'abord, le formalisme DOP profite de son avantage de représentation. Considérons, par exemple, deux événements, liés par une disjonction (<>) créant un "trou" sur l'un des DOPs (comme sur la figure 4.3); la disjonction n'étant pas représentable dans le formalisme de réseau, le trou ne sera pas créé et les occurrences inconsistantes qu'il représente ne seront pas éliminées par un algorithme de recherche des plus longs chemins[6].

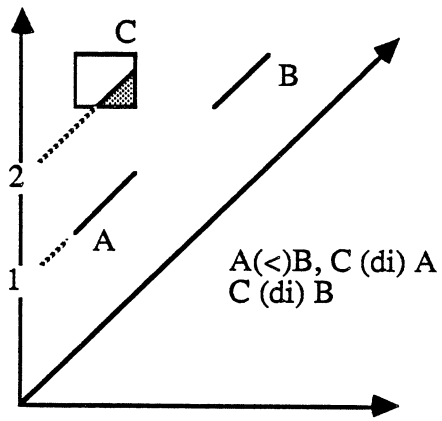
Pour comparer plus justement l'algorithme de dérivation locale et l'algorithme RLC, il faut considérer les POCs traduisibles "parfaitement" dans le formalisme de réseau. Sans parler d'efficacité, le "pouvoir" d'un algorithme de résolution de POC sera d'autant plus grand qu'il y aura d'occurrences impossibles éliminées. La figure 4.12 donne un exemple de POC traduisible en formalisme de réseau et pour lequel l'algorithme RLC élimine des occurences indétectées par l'algorithme de dérivation locale. Par conséquent, moyennant la limitation de son pouvoir de représentation, l'algorithme RLC est plus puissant que l'algorithme de dérivation locale.

On peut "transposer" l'algorithme de dérivation locale sur le formalisme de réseau en constatant que l'intersection d'un DOP avec une région permise est une mise à jour de ses frontières, c'est à dire de certains arcs du réseau. On peut alors donner deux explications à la faiblesse de cet algorithme :

1. Le lecteur peut remarquer sur la figure 4.12 que le plus long chemin associé à la contrainte "pertinente" sur la durée de C passe par les points définissant A et B. Il fait intervenir en tout trois intervalles. Or l'algorithme de dérivation locale n'en fait intervenir que deux (outre l'origine) et n'examine jamais l'analogue du chemin en question. Techniquement parlant, l'algorithme RLC examine *tous* les triplets de points, pas l'algorithme de dérivation locale.

   On pourrait alors imaginer un algorithme de résolution qui mette en jeu trois DOPs (faisant intervenir "la région permise à C par A compte tenu de B"). Ceci ne suffirait pas : un contre-exemple construit à partir de la figure 4.12 en ajoutant un troisième DOP *D* à A et B qui soit "pendant" C, serait analogue. Le nouvel algorithme de résolution ne détecterait pas que la durée de C doit être supérieure à la somme des durées de A,B et D, car cela ferait intervenir *quatre* DOPs. D'où le point suivant :

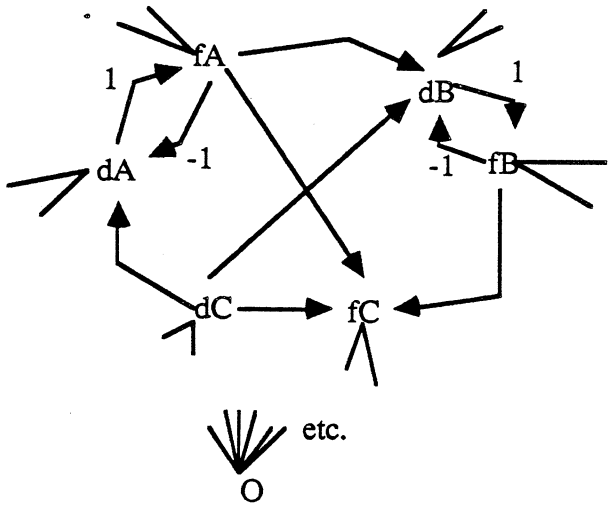
[6]Nous le désignerons dans la suite par algorithme RLC.



L'algorithme de résolution n'élimine pas les occurrences inférieures à 2

Ce POC peut être traduit en le réseau ci-dessous :

(Les arcs sans etiquette sont implicitement
valués par 0 et les arcs passant par O sont seulement indiqués)



Dans ce réseau, le chemin (dC, dA, fA, dB, fB, fC) a une longueur de 2. Par conséquent, l'algorithme de mise à jour, fondé sur la recherche des plus longs chemins, peut donner la valeur 2 à l'arc (dC, fC). la durée minimale de C est donc (au moins) 2. Si on retraduit ce réseau en formalisme DOP, la partie grisée de C, correspondant à des occurrences impossibles n'apparaitra plus. Par conséquent l'algorithme fonctionnant sur les réseaux est plus performant (au sens où il élimine plus d'occurrences impossibles) que l'algorithme de résolution locale.

Figure 4.12: Un exemple de POC mieux résolu par la recherche de plus longs chemins que par l'algorithme de dérivation locale

2. L'algorithme RLC met à jour des arcs entre événements, pas l'algorithme de dérivation locale. En effet ce dernier ne modifie pas les relations mais seulement les DOPs. De plus, les relations ne possèdent pas de descripteurs quantitatifs (des "délais"). Nous avons bien donné ci-dessus un moyen de traduire de tels descripteurs en introduisant des événements fictifs. Mais un tel artifice fonctionne pour traduire statiquement un réseau, par pour remplacer des arcs créés dynamiquement : si on remplace un arc valué entre deux points des événements A et B par un événement fictif F, F ne doit pas être traité comme A et B (ce qui conduirait par exemple à le comparer avec B, créant ainsi un autre événement fictif, et ainsi de suite jusqu'à l'infini).

   Un bref, le formalisme DOP ne permet pas facilement de donner à l'ensemble des relations temporelles une structure qui permette d'utiliser un algorithme de fermeture transitive.

Synthèse

Récapitulation

Nous nous intéressons dans ce chapitre à la définition et à la résolution d'un *problème d'occurrences contraintes.* Un tel problème peut être posé en considérant un graphe dont les nœuds sont des ensembles d'intervalles possibles, des *domaines d'occurrences possibles.* Chaque domaine est contraint numériquement, indépendamment des autres domaines. Les relations du graphe sont des relations symboliques, entre intervalles et disjonctives, telles que définies par J.Allen. Résoudre un problème d'occurrences contraintes, c'est éliminer, en réduisant les domaines, les occurrences rendues impossibles par les contraintes relationnelles.

Nous avons introduit une représentation géométrique du problème, en associant à chaque occurrence un point du plan de coordonnées (début, fin). Un domaine d'occurrences possibles O est donc une région du plan. La contrainte qu'il exerce, à travers la relation R sur un domaine D tel que $O\ R\ D$, est vue comme une région permise par O et R pour D. Les parties de D qui ne sont pas dans cette région permise sont nécessairement constituées d'occurences impossibles pour D.

Il en découle naturellement un algorithme d'élimination d'occurrences impossibles. Il consiste à calculer l'intersection de chaque domaine avec les régions qui lui sont permises, tant qu'il y a lieu de le faire. Cet algorithme n'élimine que des occurrences impossibles. Cependant il ne les élimine pas toutes, excepté dans le cas particulier d'un problème sans disjonctions où tous les domaines sont des segments.

Si on traduit un problème d'occurrences contraintes dans le formalisme d'Allen, on perd beaucoup de l'information contenue dans la forme des domaines. Cependant, on peut, alors, détecter des contradictions qui ne sont pas mises en évidence par notre algorithme.

De manière analogue, si on traduit un problème d'occurrences contraintes dans un formalisme ponctuel fondé sur la recherche de plus longs chemins, on perd l'information de nature disjonctive ; mais on peut éliminer des occurrences impossibles indétectées par notre algorithme.

Bilan: Que se passe-t-il quand on introduit des contraintes disjonctives dans une représentation numérique ?

Précisons d'abord que les contraintes disjonctives sont déjà présentes dans les représentations numériques "classiques". En effet, on y manipule, implicitement, des ensembles d'intervalles possibles et les intervalles d'un même ensemble sont mutuellement exclusifs : on n'en choisira qu'un en fin de compte. Cependant les contraintes disjonctives que nous appelons "fortes" ne peuvent être traduites dans ces représentations.

En présence d'un problème comportant des contraintes disjonctives fortes, deux attitudes sont possibles : on peut décomposer ces contraintes disjonctives ou les traiter directement.

Les décomposer signifie se ramener, à des ensembles de contraintes conjonctives qui définissent autant de POCs (de même taille que le POC initial), puis à réunir les solutions de ces POCs. On tombe alors, pour chaque problème, dans le domaine "classique" de la programmation linéaire. Dans le cas où les domaines peuvent s'exprimer par des bornes sur le début, la fin et la durée des occurrences, un algorithme fondé sur les plus longs chemins permet d'éliminer toutes les occurrences impossibles. Dans le cas contraire, on peut utiliser un algorithme, fondé sur celui du simplexe, qui permet seulement d'exhiber une solution (c'est à dire une affectation d'occurrence a chaque événement).

La décomposition de ces contraintes disjonctives est cependant combinatoire : le nombre de problèmes conjonctifs est exponentiel par rapport au nombre de contraintes disjonctives. Les méthodes précédentes doivent reposer sur des heuristiques d'énumération pour ne pas être intolérablement coûteuses. La voie que nous avons ouverte est l'adoption d'une stratégie de moindre engagement qui limite la complexité d'une éventuelle énumération.

Le principe consiste à ne décomposer les contraintes disjonctives que sur une base *locale*. On décompose chaque contrainte, en limitant les problèmes conjonctifs à deux nœuds seulement (ceux qui interviennent dans la contrainte), puis on réunit les solutions avant de décomposer une autre contrainte.

Le problème est que cette méthode n'élimine pas, en général, toutes les occurrences impossibles. Pire, même en l'absence de contraintes disjonctives fortes, dans les conditions d'application des autres méthodes, l'élimination n'est pas complète.

Que faire alors des occurrences résiduelles ? Une première solution consiste à les laisser et, par la suite, à s'adapter à l'apparition de ces occurrences.

On peut aussi choisir décomposer, finalement, les contraintes disjonctives. Le travail déjà fait aura servi à dégrossir le problème, en décelant des contradictions ou en éliminant, de fait, certains termes des contraintes disjonctives (et nous avons donné un algorithme permettant de les expliciter dans le formalisme d'Allen). Dans ce cas, on peut terminer en traduisant le tout dans un formalisme linéaire classique (et nous avons indiqué comment le faire). Naturellement, les passages d'une représentation à l'autre sont coûteux, une stratégie globale efficace nécessite une étude algorithmique serrée.

Problèmes ouverts

Terminons par quelques problèmes ouverts qui constituent autant de directions de recherche.

D'abord, un point de vue qui est une direction non pas de recherche, mais plutôt de développement : l'efficacité des algorithmes géométriques. En effet, on appréhende facilement de petits croquis mais l'implantation des opérations géométriques élémentaires coûte cher. D'autant plus qu'on a besoin d'un niveau de généralité assez élevé, même en se limitant au cas linéaire. Il s'agit, en effet, de manipuler des polygônes non convexes et même non connexes, mais aussi, et en même temps, des segments et des points. Les problèmes algorithmiques de géométrie dans le plan constituent un domaine bien étudié, en raison, en particulier, de l'essor de la CAO, nous n'avons pas cherché à le creuser.

Le deuxième point est plus spécifiquement temporel. Il concerne l'élimination des occurrences rendues impossibles par des contraintes de capacité, telles que celles apparaissant dans l'exemple suivant : soient dix tâches d'une heure, qui ne peuvent se recouvrir, et qui doivent être effectuées en l'espace de trois heures. Notre algorithme de propagation ne comparant les tâches que deux à deux, il constate que deux tâches d'une heure peuvent très bien se faire en trois heures sans se recouvrir. Donc aucune inconsistance n'est détectée, alors qu'il est bien évident que le problème n'a pas de solution. Signalons qu'il est résolu par un algorithme de plus longs chemins lorsqu'un ordre sur les tâches est fixé.

En ce qui concerne l'algorithme de propagation lui-même, on peut chercher à le généraliser en élargissant l'horizon d'une propagation. En effet, nous avons vu que les effets des clauses d'une disjonction sont calculés séparément en considérant seulement deux nœuds, puis sont immédiatement réunis. L'extrême inverse est représenté par les méthodes reposant sur une décomposition complète dont on calcule l'effet sur tous les domaines. Adopter un moyen terme suppose que l'on adopte une méthode de contrôle permettant de décider quand il est intéressant de rassembler les déductions sur deux clauses d'une disjonction.

Enfin, toujours en ce qui concerne la propagation, on peut élargir encore notre point de vue en cherchant à défaire des contraintes au lieu d'en ajouter. En effet, on a, avec la méthode exposée dans les pages précédentes, la propriété suivante. Supposons qu'on élimine des occurrences impossibles d'un problème d'occurrences contraintes donné. Si on ajoute à ce graphe un nœud ou une relation, l'élimination est toujours valide, car on a procédé par conditions nécessaires. On peut donc reprendre les domaines en l'état et calculer l'effet des nouvelles contraintes introduites. Par contre, si on retire un nœud ou une relation, il faut réintroduire les occurrences rendues, auparavant, impossibles par ce qui est retiré. Dans l'état actuel de la méthode, le seul moyen est de reprendre le problème depuis le début et d'éliminer toutes les occurrences impossibles. Le travail de la propagation précédente est donc perdu. Pour pouvoir défaire "juste ce qu'il faut", un moyen est de garder une "trace" des causes de chaque élimination. Si la contrainte relachée n'a pas de lien avec ces causes, l'élimination est toujours valide. Ce que nous venons de décrire très informellement fait partie d'une classe plus générale de méthodes de résolution dites de "maintenance de vérité". Ces méthodes concernent principalement la production de clauses propositionnelles (donc en nombre discret et fini). Il n'est pas évident de les adapter à des domaines continus

dont les influences mutuelles passent par des relations qui peuvent aussi être relachées.

Partie III

Un Système de Planification pour l'Affectation de Systèmes d'Armes à bord d'un Navire

Introduction

La partie précédente traite des techniques de propagation de contraintes temporelles. On peut dire en un mot que ces techniques concernent le placement dans le temps d'événements interdépendants. On peut les implanter dans un module regroupant un ensemble de fonctionalités utiles à la planification en général. Ce dernier est donc un outil pour la planification conçu dans un cadre général.

Le point de vue adopté dans cette partie est diamétralement opposé. Il s'agit d'étudier sur un exemple précis — la planification de l'affectation de systèmes d'armes sur un navire[7]— un problème de planification complet. En effet, la fonction de planification proprement dite appartient à un système situé dans un univers évoluant indépendamment de lui (simulé par SAFRAN, voir la section 6.1.1) et qui est modifié par ses actions. Depuis la perception de cet univers, jusqu'à l'action, toutes les fonctions d'un agent autonome auquel on réclame une certaine intelligence peuvent être introduites.

Illustration des techniques exposées dans la partie précédente, PASAN n'est ni un problème jouet, ni une application opérationnelle.

Ce n'est pas un problème jouet dans le sens où l'affectation des systèmes d'armes, on l'a dit, est un problème réel et complet. Les différentes hypothèses, conduisant à une formalisation plus simple seront, autant que faire se peut, explicitées, voire justifiées et éventuellement provisoires. La partie effectivement résolue (au chapitre 6), avec toutes les simplifications qui sont faites peut, elle, être considérée comme un jouet, mais l'analyse du problème complet met en évidence les extensions nécessaires pour résoudre le reste.

Ce n'est pas une réalisation opérationnelle. D'abord, nous ne possédont pas toutes les données nécessaires à une telle réalisation. Nous avons bénéficié de l'aide des experts du Centre de Programmation de la Marine sur bien des points, mais nous n'avons jamais accédé à des données, protégées par le secret militaire, sans lesquelles notre système n'a aucune valeur opérationnelle[8]. Nous estimons que le problème est toujours réaliste, mais il est fort possible qu'il soit assez loin de *la* réalité.

La deuxième incompatibilité de PASAN avec une application opérationnelle est que notre système utilise uniquement des techniques d'IA pour résoudre des problèmes d'IA. Or, en admettant que l'on puisse séparer physiquement la part de l'IA dans un système logiciel et matériel, le travail que représentent les interfaces avec les systèmes d'acquisition et ceux de commandes (y compris d'éventuelles interfaces homme–machine) est considérable et il était hors de questions que nous l'abordions.

Cette partie est scindée en deux chapitres. Le premier constitue l'analyse du problème et la mise en évidence d'un certain nombre de questions fondamentales qu'il soulève du point

[7]Nous utiliserons dorénavant le sigle PASAN (Planification de l'Affectation de Systèmes d'Armes à bord d'un Navire) pour désigner à la fois cette activité et le logiciel qui l'effectue.

[8]Par conséquent, il va de soi que toutes les opinions émises dans ce texte n'engagent que leur auteur et n'ont pas de rapport avec la doctrine d'utilisation des systèmes d'armes en vigueur dans la Marine Nationale

de vue de la planification. Le second chapitre décrit l'architecture et le fonctionnement du système lui-même. Gardant en toile de fond les questions soulevées au chapitre précédent sans y apporter de réponse définitive, son but est de montrer l'intégration d'un module de propagation de contraintes temporelles dans un système conçu avec des préoccupations beaucoup plus larges.

Chapitre 5

Analyse du Problème

5.1 Description du Domaine

Dans cette section nous posons le problème de l'affectation des systèmes d'armes comme un problème de planification. Nous en décrivons les aspects sans préjuger de la manière dont il peut être résolu. Nous n'envisageons même pas, ici, une résolution automatique. Le lecteur peut considérer que cette section contient les renseignements qu'on lui donnerait s'il devait lui-même décider de l'affectation des systèmes d'armes. Il va de soi que les détails ne sont pas abordés (de toutes façons nous ne les connaissons pas!), mais les quelques pages qui suivent sont un peu techniques.

5.1.1 L'Affectation des Systèmes d'Armes : Pourquoi, Comment et Quand?

Description Succinte du Problème

Un bâtiment de guerre possède plusieurs systèmes d'armes (entre 2 et 5 environ) destinés à assurer sa protection. Ces systèmes, du canon au missile, ont à peu près tous une portée différente. Un bâtiment peut être soumis à l'attaque d'une dizaine d'engins hostiles (avions, missiles) arrivant généralement par vagues successives. Le but est de détruire ou de leurrer ces engins hostiles en utilisant au mieux les ressources du navire.

L'affectation des systèmes d'armes (SA) est actuellement prise en charge "à la volée" par le système SENIT[1]. Ce système prend en entrée une situation instantanée décrite sous la forme d'un ensemble de pistes. Ces dernières sont les echos radar des objets entourant le bâtiment, dotés d'informations cinématiques et d'informations d'identification (permettant par exemple de ne pas tirer sur ses propres avions). Compte-tenu de cette situation, le SENIT décide *d'une* interception pour chaque SA, sans se préoccuper des interceptions futures (du moins pas explicitement). On peut dire que le système SENIT calcule pour chaque SA un

[1]Système d'Exploitation Navale des Informations Tactiques

"plan" d'une action. Le problème abordé est l'étude d'un système qui produise un plan de plusieurs interceptions permettant ainsi une utilisation plus efficace des SA.

La suite de cette section décrit comment et quand on affecte une piste à un système d'armes destructif[2], ou comment on peut la leurrer.

Description d'un Système d'Armes

Séquence de tir Une fois l'objectif détecté par un radar de veille, il peut être désigné à un SA. La séquence se compose d'une phase d'acquisition (recherche + poursuite) de l'ordre de 10 s, puis de la phase de vol dans le cas d'un missile, d'une durée de l'ordre de la minute. En cas d'échec de l'interception, on peut éventuellement retirer immédiatement, sans avoir à refaire d'acquisition.

Dans le cas d'un tir au canon, seule la phase d'acquisition occupe le SA puisqu'il n'y a pas de conduite durant le vol.

Conditions du Tir On ne peut pas tirer n'importe où ni n'importe quand. Les conditions du tir peuvent être synthétisées par des zones spatio-temporelles de tir interdites. Ces zones sont l'effet de l'influence de facteurs extérieurs sur le tir. Elles peuvent être statiques (cas par exemple d'une partie du bâtiment masquant le SA), mais aussi dynamiques (il vaut mieux, en général que deux missiles ne se croisent pas). Nous reviendrons sur ces zones après la description des systèmes de Guerre Electronique.

La Guerre Electronique

Si le tir d'un engin destructeur caractérise l'action d'un SA, l'action d'un système de guerre électronique est le tir et surtout la présence d'un leurre destiné à tromper l'engin hostile sur la position du bâtiment.

L'émission d'un leurre se fait suivant la séquence suivante : le tir du leurre proprement dit, puis un fort brouillage destiné à aveugler l'engin hostile et enfin activation du leurre qui après le brouillage pourra être pris pour le bâtiment. Notons qu'au moment du brouillage les missiles lancés par le bâtiment lui-même sont *aussi* susceptibles d'être perturbés. C'est même là que réside probablement l'interaction la plus forte entre la guerre électronique et les autres SA.

Un leurre magnétique (nuage de paillettes) reste actif une dizaine de minutes, il est parfois doublé par un leurre infra-rouges (feu de bengale) qui lui dure une dizaine de secondes. Pendant ces dix minutes, un leurre magnétique, poussé par le vent, a le temps de dériver d'une distance de l'ordre du kilomètre. Le vent a aussi une importance dans l'orientation du leurre, dont l'efficacité dépend du diamètre apparent.

[2]que nous appellerons dans la suite système d'armes tout court

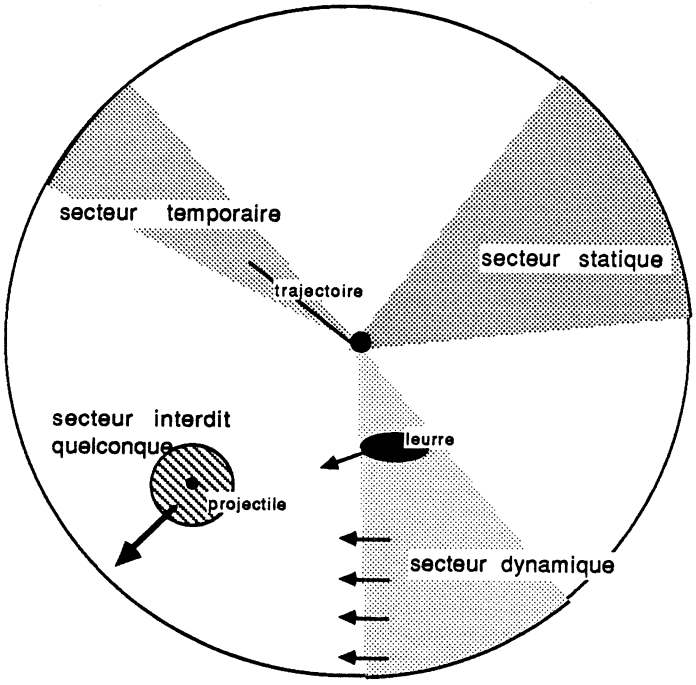


Figure 5.1: secteurs interdits

Un leurre a aussi ses zones de présence interdites. Si, par exemple, le bâtiment et le leurre sont tous deux proches de la trajectoire du missile, ce dernier, bien que leurré pourra quand même atteindre le bateau (donc, ne pas tirer le leurre entre le bâtiment et le missile!).

Suivant que le missile a accroché ou non le bâtiment, on devra lancer le leurre près ou loin du bateau. Loin pour forcer le missile a choisir entre plusieurs cibles (dont une réelle), près pour que la confusion soit possible.

Dans les deux cas, le résultat est assez aléatoire, et surtout, lorsqu'on s'apperçoit de l'échec, il est souvent trop tard. C'est pourquoi une action de guerre électronique est en général doublée par une action "dure", c'est à dire une interception.

Calcul des Disponibilités Dynamiques des Systèmes d'Armes

Le problème qui nous occupe maintenant est le suivant : étant donnée une piste, quand peut-elle être affectée à un système d'armes donné (sans souci d'optimisation)?

Le fait d'occuper un SA est très étroitement lié au fait d'occuper un secteur de l'espace pendant un certain temps. L'espace entourant le bâtiment est le milieu par lequel passent tous les calculs de réservation qui se ramènent ainsi au calcul de l'intersection d'une zone potentielle de tir avec une zone de tir interdit. Ces zones de tir interdit sont, du plus simple au plus compliqué (fig 5.1) :

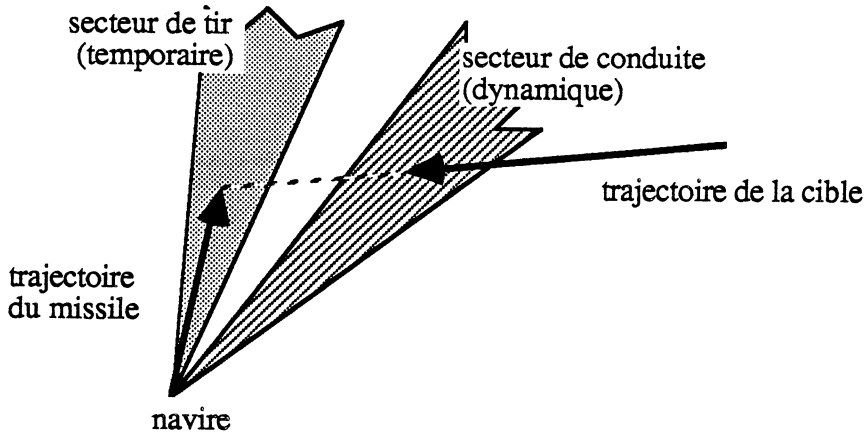


Figure 5.2: Les deux types de secteur mobilisé pendant une interception

- L'extérieur d'un disque ($\rho \geq \rho_{max}$) qui correspond à la portée des SA.
- Un secteur statique : portion de quadrant centrée sur le bâtiment ($\theta \in [\theta_1, \theta_2]$), correspondant, par exemple, à un secteur masqué par partie du bâtiment.
- Un secteur temporaire : portion de quadrant ayant une durée de vie limitée ($\theta \in [\theta_1, \theta_2]$, $t \in [t_1, t_2]$) : cas d'un secteur occupé par un tir radial et interdit aux autres tirs.
- Un secteur dynamique : portion "mobile" de quadrant ($\theta \in \theta_1(t), \theta_2(t)]$) : cas d'un secteur masqué par un leurre se déplaçant.
- Une zone quelconque dynamique de l'espace : cas d'un projectile ou d'un leurre dont la proximité avec un autre projectile est indésirable.

Précisons ici que l'on peut distinguer deux sortes de zones de tir (fig 5.2) : la zone du tir proprement dit est un quadrant temporaire (en supposant la trajectoire radiale) et la zone de conduite est un quadrant dynamique. Dans tous les cas, c'est en calculant l'intersection de l'une de ces zones avec les zones de tir ou de conduite interdit que l'on détermine la possibilité de prise en charge de la piste par un système d'arme.

Perturbations et Replanification.

Une caractéristique du problème est que les situations de planification sont fortement sujettes à changement. Les principales causes de perturbation sont :

- un changement de cap de la cible
- l'échec du tir

- l'apparition de nouvelles cibles

En dehors de l'obligation d'interrompre une interception déjà commencée (d'où une perte de temps), le coût d'une replanification est psychologique. Un opérateur sous la pression d'une attaque généralisée verra d'un très mauvais œil un système qui change d'avis toutes les secondes. Notons aussi ici, qu'un plan trop vague (ou un ensemble de plans considérés trop grand) est inacceptable psychologiquement pour les mêmes raisons. Un plan clair et stable, même de qualité moyenne est préferable à une solution vague et changeante, même si c'est la meilleure.

5.1.2 Caractéristiques du Problème

Même si PASAN est un cas précis, il est utile d'en dégager les caractéristiques afin de cerner une classe de problèmes que l'on pourrait résoudre avec une architecture et des techniques analogues.

L'Aspect Allocation de Ressources

PASAN est un problème d'allocation de ressources non partageables. Une tâche est un tir qui mobilise le SA pendant un certain temps. Les tâches traitées par un même SA ne peuvent pas se recouvrir et doivent donc être ordonnées. Cependant, il n'existe pas (ou très peu) de contraintes d'ordonnancement a priori, ce qui confère une nature fortement disjonctive et combinatoire au problème.

Le nombre de tâches est relativement limité. Chaque tâche peut, et doit être considérée de manière individuelle. Des notions comme le flux des tâches ou l'établissement d'un régime permanent sur une ressource — qui apparaissent par exemple si l'on gère la production d'un grand nombre de pièces du même type sur quelques machines — sont ici sans objet. Cependant, cela ne signifie pas que l'on n'utilise pas de critères globaux de charge pour guider de manière heuristique le calcul d'un ordonnancement.

L'Aspect Spatio-Temporel

Assurer la cohérence d'un plan demande des manipulations plus numériques que logiques. Il n'y a pas de contraintes fortes sur l'enchainement des actions comme dans la planification de gammes ou la robotique d'assemblage. Un tir est une action relativement élémentaire (dans le sens peu décomposable) liée aux autres actions par une relation simple d'ordonnancement. Ceci rend inutile un formalisme dérivant, comme dans STRIPS, les préconditions nécessaires à la réalisation d'une action et ses effets. Par contre, la description des actions met en jeu des contraintes numériques nombreuses et complexes recouvrant les aspects géométriques et temporels. Par exemple, lorsqu'on impose que le tir sur p_1 ne doive pas recouvrir le tir sur p_2, les contraintes symboliques de non-recouvrement sont trop pauvres

pour être utiles à la validation d'un plan. Par contre la relation liant la position de la piste, sa vitesse et le moment du tir est extrêmement active lors du processus de validation de plan.

Les Critères d'Allocation Sont Aussi Plutôt Numériques On peut diviser les critères de choix (ou d'optimalité) d'allocation en deux familles : les critères de "classification" et les critères de "modélisation". Les critères de classification mesurent souvent la valeur et l'adéquation technique des tâches et des ressources. Par exemple, en ce qui concerne l'affectation de systèmes d'armes, de tels critères interviennent dans des règles du type : *Le tir au canon est inutile sur tel type de missiles.* L'évaluation du respect de ces critères revient en gros à comparer des attributs de ressources et de tâches.

Par contre, l'évaluation des critères de modélisation est plus compliquée que la consultation de la valeur d'un attribut. C'est le cas des critères spatio-temporels, et plus généralement, des critères mettant en jeu plusieurs variables, souvent continues, liées entre elles par des lois plus ou moins connues. Pour tenir compte de ces critères (intervenant dans des règles du type *Le tir au canon est autorisé dans tel secteur*) il faut manipuler ces lois. En bref il faut (quand c'est possible) faire des calculs. Par exemple, lorsqu'on envisage de tirer au canon sur une cible donnée, il faut tenir compte des éléments cinématiques, éventuellement dynamiques pour calculer *quand* elle sera dans le secteur autorisé, ce qui est plus complexe que vérifier si la piste est bien d'un type tel qu'on peut la tirer. Il se trouve que PASAN fait plutôt intervenir des critères de modélisation.

L'Aspect de l'Incertain

L'univers instantané est connu, mais il évolue de façon incertaine L'évolution de l'univers donne aux tâches des caractéristiques seulement partiellement connues. Ceci est une forme d'incertitude, que l'on peut qualifier de temporelle car elle est différente de l'incertitude sur les caractéristiques *instantanées* de l'univers qui dans le cas qui nous occupe, sont supposées parfaitement connues : on sait précisément quelles sont les pistes présentes et où elles sont mais on ne peut que conjecturer leur évolution. Ceci a deux implications sur la planification : d'abord cela peut compliquer énormément la description des contraintes géométriques et temporelles sur les tâches (nous disons "peut" car ce problème est ignoré dans l'implantation). Ensuite, cela justifie une planification en ligne. La situation future n'étant que conjecturée, le plan n'est pas forcément exécutable. Il faut donc le réactualiser régulièrement, alors que l'on est déjà en train de l'appliquer. Nous verrons que, même traité de façon minimale, cet aspect du problème a des répercussions à l'échelle d'une planification instantanée.

5.2 Planification : Problèmes Liés à la Réactivité

Il est tout à fait évident que face à une situation très évolutive et assez incertaine, planifier c'est aussi replanifier sans cesse, le dernier plan établi devenant ainsi une donnée aussi importante que les données cinématiques pour la planification en cours.

Nous aborderons le problème par trois de ses aspects : la prise en compte du plan précédent par la planification courante, la définition du niveau d'abstraction pour la planification et enfin, la méthode de planification courante elle-même.

5.2.1 Le plan : une donnée de la planification

Jusqu'à présent, nous entendions par *données* tout renseignement concernant le monde extérieur au système. Cependant, à partir du moment où l'on replanifie sans cesse, le plan courant — contenant les dates d'interception — est un élément de la situation. Au point de vue statique d'une part : certains SA à certains moments peuvent être plus ou moins occupés et donc la menace que représente une piste peut être accrue si celle-ci se présente à une époque chargée. Mais aussi au point de vue dynamique : étudier les transformations possibles du plan (par exemple en évaluant des marges de manœuvre) est utile à l'évaluation de la stabilité de ce plan.

5.2.2 Quel niveau de représentation pour la planification?

On sait que la situation cinématique change constamment, remettant ainsi le plan en cause. Cependant, il y a des des changements radicaux du plan et d'autres que l'on considère comme des modifications bénignes. De telles modifications mineures peuvent être assimilées à des *déformations* du plan, par exemple des glissements des intervalles d'interception ou des augmentations de durées. Ceci suppose que l'on considère qu'un plan qui se déforme est au fond, ou à un certain niveau d'abstraction, le même plan. Ainsi, à un plan de niveau élevé correspondent plusieurs plans possibles de niveau opérationnel (c'est-à-dire décrivant une exécution de plan et une seule).

L'avantage de faire la planification à ce niveau abstrait est le suivant : Les changements de situation considérés comme mineurs et néanmoins fréquents ne "remontent" pas jusqu'à ce niveau. Ils sont ignorés, ce sont autant de calculs en moins à faire.

Cependant, maintenir un plan uniquement à haut niveau suppose que l'on soit capable de mettre en place une architecture telle que :

- La répartition de la tâche globale de planification soit bien équilibrée entre les deux niveaux. Par exemple, si en pratique, tout se fait au bas niveau, le découpage ne fait qu'alourdir la représentation, ce qui est nuisible à l'efficacité.
- Le calcul d'un plan opérationnel à partir d'un plan de haut niveau (agissant comme un

ensemble de contraintes sur les affectations de SA et les intervalles possibles d'interception) soit faisable, et que les problèmes qu'il soulève soient indépendants des problèmes relatifs à la planification abstraite (c'est une condition de bon fonctionnement de toute stratégie de décomposition de problèmes).

- Une éventuelle impossibilité de calculer une solution opérationnelle — parce que la solution abstraite est inexécutable car les réservations ne "rentrent pas" — soit :
  - ou bien détectable, dans le cas où ce type de contadiction ne peut être détecté que lors de cette étape numérique
  - ou bien impossible, grâce à une modélisation ad hoc (pas de disjonction, durées bien choisies...)
- Le calcul indépendant de deux solutions opérationnelles à t et t' proches produise des solutions proches (hypothèse de continuité du calcul d'une solution opérationnelle, garantissant une certaine forme de stabilité)

Il s'agit donc de trouver un niveau de planification ayant les avantages décrits au début, et de trouver les techniques de calcul d'une solution opérationnelle ayant les propriétés ci-dessus. Afin de fixer les idées, voici une possibilité concernant le problème de représentation.

Le niveau abstrait fixe l'affectation des pistes aux SA et éventuellement leur ordre de traitement (par SA) qui sont les contraintes de *structure* du plan. L'objet élémentaire est donc à ce niveau un intervalle d'interception pour les SA et un intervalle de présence de leurre pour la Guerre Electronique.

Quant à l'extraction d'une solution opérationnelle, on peut d'une part appliquer une règle de choix très simple parmi les libertés que laissent les contraintes (par exemple on prend le premier intervalle disponible de la première interception à planifier). Ou bien, on peut manipuler les ensembles d'intervalles d'interception possibles, en adoptant une stratégie de moindre engagement. La première méthode est plus rapide, sauf en univers contraint, car il faut alors de nombreux essais pour trouver une solution. La seconde méthode est plus lente à trouver une solution mais élimine plus de contradictions et marche mieux en univers très contraint. L'idéal serait évidemment de combiner les deux.

5.2.3 Le cycle Elaboration-Evaluation-Modification

Il y a deux sortes de contraintes difficiles à traiter pour la planification. En premier lieu, les contraintes fortes, interdépendantes et temporelles : c'est typiquement le cas des secteurs interdits temporaires et dynamiques réservés par un tir. Ces contraintes sont difficiles à prendre en compte avant la planification complète car leur valeur dépend du moment où elles sont planifiées.

En second lieu, les critères globaux, par exemple ceux relatifs à la stabilité du plan, à la charge des SA, au nombre total de cibles traitées, sont difficiles à évaluer avant que le plan ne soit entièrement construit.

Pour arriver à prendre en compte ces contraintes de manière efficace, il faut "faire des essais", c'est-à-dire, en gros, faire un plan, voir ce que cela donne, et corriger ce qui ne va pas. Ceci met en évidence trois fonctionnalités : l'élaboration, l'évaluation et la modification de plan. Cette démarche correspond à une optimisation locale à l'intérieur de l'espace des solutions.

Les critères globaux de charge et de stabilité relèvent plutôt du niveau de planification le plus abstrait. En effet, une charge, par exemple, se corrige en proposant une répartition différente des pistes parmi les SA. Par contre, la résolution des incompatibilités spatio-temporelles doit d'abord être tentée au "bas-niveau".

5.2.4 Le Problème de la Planification Distribuée

Nous supposons dans cette section que chaque SA possède un planificateur intégré[3] ou local. Nous supposons aussi qu'il y a un "planificateur général" qui s'occupe au moins de la répartition des cibles et de la gestion des conflits entre les plans locaux.

Dans le cadre d'une planification distribuée, le problème clef est de résoudre les conflits provoqués par les planificateurs locaux. Ce problème peut se diviser en deux :

- détecter le conflit : problème de communication entre agents
- le résoudre : problème de la coopération.

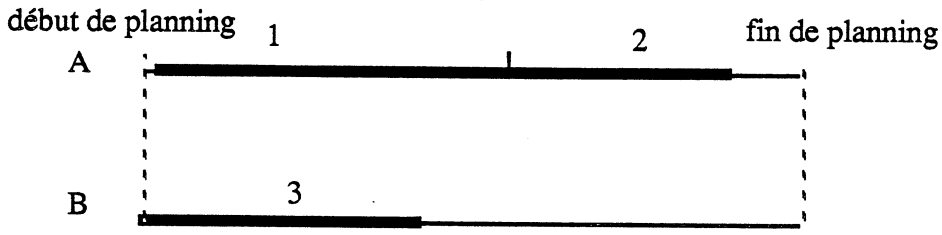
Détection

Pour détecter un conflit, il faut connaître les différents plans puis comparer leurs effets. On peut supposer que les plans sont tous accessibles au planificateur global. Cependant, nous avons vu qu'on pouvait avoir deux niveaux de planification. Chaque niveau gère une catégorie de conflits : les conflits majeurs impliquent une modification de la structure du plan, et les conflits mineurs (et fréquents) impliquent une modification des modalités d'exécution du plan — allongement ou raccourcissement d'une interception, glissements...—. Pour reconnaître un conflit majeur, il faut connaître les *marges de glissement* que possède le plan, donc une information plus complète que le plan daté. Ceci suppose que les planificateurs individuels possèdent et soient capables de communiquer cette information.

Résolution

Quels sont les moyens d'action dont on dispose pour résoudre un conflit? Le moyen le plus faible — qui est le seul dont le planificateur global disposera surement — est de changer la répartition des cibles entre SA, en espérant que le conflit détecté disparaîtra et que d'autres n'apparaitront pas.

[3]Cette hypothèse n'est pas gratuite. Elle correspond à une proposition d'étude du CPM.



On suppose que A et B placent leurs actions au plus tôt
et que 1 ne peut recouvrir ni 2 ni 3.
Aucune répartition ne convient. Il est nécessaire de
planifier 3 APRES 1 ou d'imposer à A de planifier 1 APRES 2.

Figure 5.3: La nécessité de pouvoir exprimer des contraintes d'ordre

Si le planificateur général connait la manière dont réagiront les planificateurs individuels, et si l'espace de solutions est assez dense, une telle méthode est viable. Par contre, si les conflits sont fréquents, une répartition différente a peu de chances (car il s'agit bien de chance lorsqu'on a un contrôle si faible) de les faire disparaître, les problèmes pouvant n'être dus qu'à des contraintes de délai ou d'ordonnancement (voir fig. 5.3).

Par conséquent, il apparait utile dans beaucoup de cas que les planificateurs soient capables de prendre en compte des contraintes d'interdiction temporelles sur certaines plages de temps (de manière absolue, ou sélectivement suivant les cibles) ou des contraintes d'ordonnancement. Il est alors nécessaire de définir un mécanisme de gestion de ces interdictions, afin d'appliquer les contraintes les moins coûteuses, en termes de perte de solutions, d'éviter les situations de bouclage etc.

Chapitre 6

Architecture et Fonctionnement de PASAN

6.1 Présentation du Système PASAN

6.1.1 Fonction et intégration dans SAFRAN

La tâche du système PASAN est de décider de l'affectation des systèmes d'armes dans un univers simulé par SAFRAN.

Brève description de SAFRAN

SAFRAN est un logiciel développé par la société ITMI pour le compte du Centre de Programmation de la Marine [7]. Il est destiné à tester le comportement d'un bâtiment isolé face à une attaque aérienne. Le comportement du bâtiment est représenté par un ensemble de règles, depuis l'évaluation de la menace que représentent les pistes, jusqu'au déroulement des séquences de tir. L'attaque est représentée par un scénario prédéfini par l'utilisateur et par des règles régissant le comportement des pistes, ces dernières pouvant modifier dynamiquement le scénario.

Une session de SAFRAN consiste à :

1. définir des règles de comportement, regroupées dans plusieurs bases de règles, et un ou des scénarios d'attaque.
2. exécuter le scénario, c'est-à-dire le confronter avec le comportement du bâtiment.

L'exécution est organisée autour de la boucle, simulant un pas de temps, suivante :

1. Mise à jour de la position et de la vitesse des pistes en fonction du pas de temps et conformément au scénario.

2. Evaluation des règles modifiant le comportement du bâtiment et éventuellement le scénario si les pistes réagissent.

3. Intervention éventuelle de l'utilisateur qui peut modifier le pas de temps, demander des explications, examiner les paramètres cinématiques des pistes etc.

Fonction de PASAN

La fonction de PASAN est, à un instant donné et pour un ensemble de pistes donné, de calculer un plan d'affectation des pistes pour chaque système d'armes.

On suppose, au moment de la planification, que les pistes ont une trajectoire rectiligne uniforme.

En ce qui concerne la réactivité, PASAN calcule un plan à chaque pas de temps de SAFRAN. Il ne retient du pas de temps précédent que le plan calculé à ce moment, dit *plan courant.* C'est là le niveau de fonctionnalité minimal pour rester compatible avec l'évolution du monde et de ses propres actions.

L'aspect "temps réel" du système n'est pas abordé : on suppose que le temps de planification et d'application des ordres d'exécution est d'un ordre très inférieur à celui du pas de temps de SAFRAN. Pratiquement cela revient à arrêter momentanément la simulation pendant la planification.

6.1.2 Organisation

La tâche de PASAN est décomposée suivant trois étapes principales qui interviennent lors de chaque itération de la boucle de SAFRAN (fig. 6.1).

Acquisition Elle consiste à transformer les données concernant l'état instantané du monde — situation des pistes, état du SA et du navire — en données temporelles pour la planification, c'est-à-dire en DOPs. Ces DOPs représentent des ensembles d'intervalles possibles de mobilisation des SA.

Planification Le but de cette phase est de fournir un plan d'occupation des SA.

La planification est organisée autour d'une boucle Proposition – Elaboration – Evaluation – Nouvelle Proposition[1]. En effet, la vérification de contraintes temporelles complexes, comme les secteurs de tir temporaires, et l'optimisation de critères globaux, comme la charge des SA ou le nombre et la qualité des pistes traitées, sont des processus mal connus qui nécessitent une stratégie d'essais-erreurs.

[1]Nous n'avons pas gardé ici la formulation Elaboration – Evaluation – Modification de la section 5.2.3. Dans la mesure où nous n'avons pas précisé s'il y aurait une véritable capacité d'amendement de plan, nous avons remplacé le terme de "modification" par "proposition-nouvelle proposition".

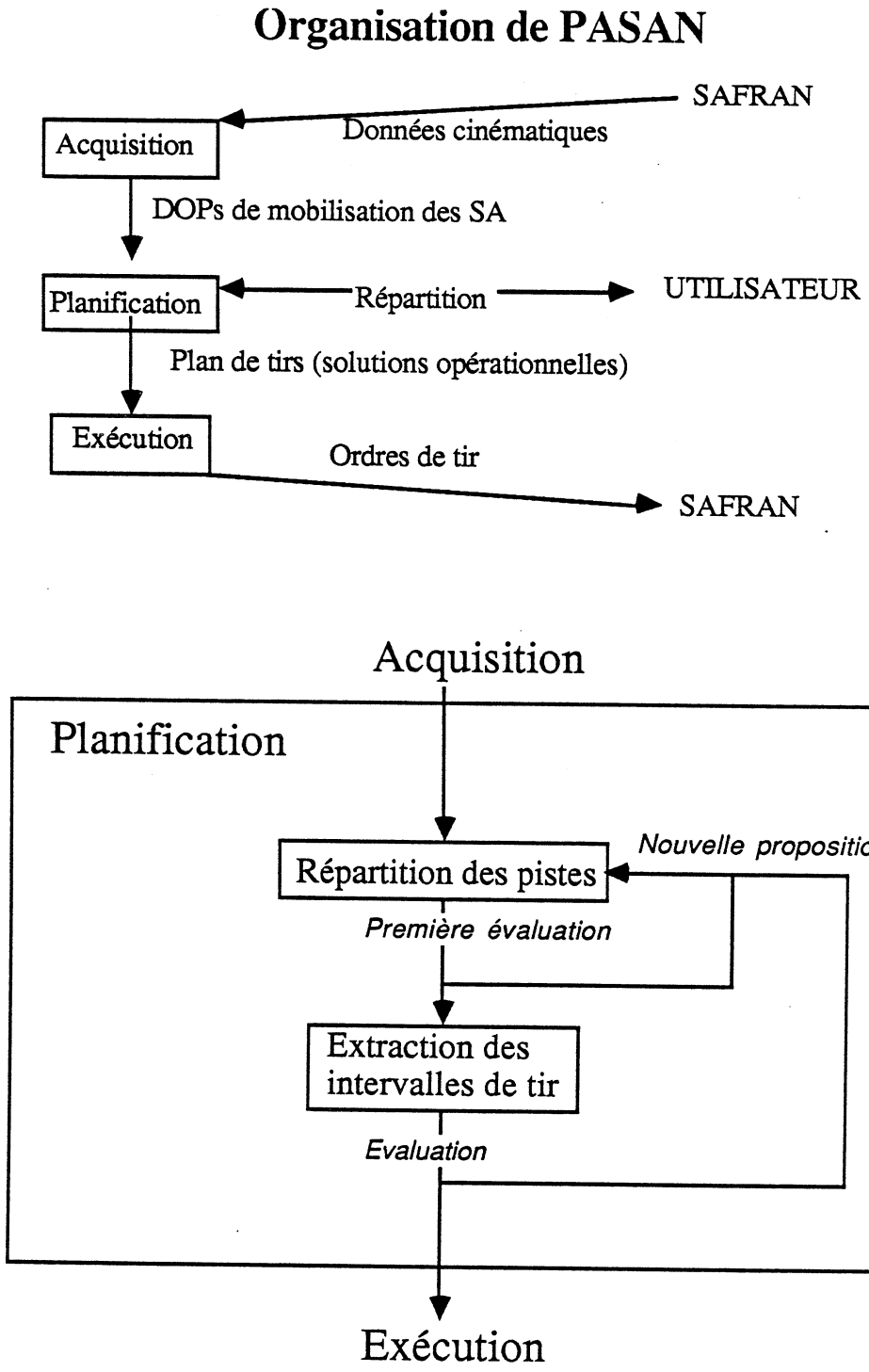


Figure 6.1: Organisation de PASAN et du Module de Planification

Nous faisons coïncider l'aspect hiérarchique et l'aspect distribué en considérant deux niveaux de planification, dont le second correspond à une autonomie de planification des SA :

1. la répartition des pistes correspond à un niveau global et symbolique de la planification,
2. l'extraction d'occurrences correspond à un niveau numérique et local, c'est-à-dire au niveau de chaque SA.

Répartition A partir des DOPs calculés lors de la phase d'acquisition, on répartit entre chaque SA les pistes qu'il devra traiter. Cette répartition implique, sur le plan temporel, l'établissement de contraintes entre les DOPs. Ces contraintes sont propagées en utilisant les techniques exposées au chapitre précédent. Le résultat de cette propagation correspond à une évaluation provisoire. Si cette denière n'est pas satisfaisante, en particulier si elle est incohérente, il est possible de s'échapper dès ce niveau de la boucle de planification et de proposer une autre répartition sans procéder à l'extraction.

Extraction Pour une répartition, et pour chaque SA, indépendamment des autres, on extrait un unique élément de chaque ensemble d'intervalles possibles de tir. Le produit de l'extraction est une proposition de plans qui peut être acceptée ou rejetée. Si, après évaluation de la qualité de chaque plan et de la qualité globale, la proposition est acceptée, le plan est exécuté. Si elle est rejetée, il faut proposer une nouvelle répartition et le cycle de la planification recommence.

Exécution L'exécution consiste à entériner la proposition de plan. On suppose qu'elle est appliquée, de manière à pouvoir tenir compte des actions en cours lors des planifications suivantes (en particulier, de manière à ne pas les replanifier).

6.1.3 Implantation

Interface avec SAFRAN

Le comportement du SENIT, qui décide des allocations, étant modélisé dans SAFRAN par des bases de règles, nous avons retiré ces dernières pour les remplacer par PASAN. A dire vrai, pour le moment nous supprimons toutes les règles. Mais il est indispensable que nous gardions la possibilité de les rétablir, par exemple pour modéliser le comportement des pistes réagissantes. Le rôle de SAFRAN est donc, pour l'instant, strictement réduit à l'exécution d'un scénario fournissant périodiquement les données nécessaires à la phase d'acquisition de PASAN.

Planificateurs Locaux

En toute rigueur, la seule donnée communiquée aux planificateurs locaux lors de l'étape d'extraction devrait être la répartition. A charge au planificateur local de faire sa propre

acquisition et son propre calcul des intervalles possibles de tir, en prenant éventuellement en compte des paramètres et des contraintes différentes du niveau global auquel on fait la répartition.

Ne possédant aucun renseignement précis sur ces planificateurs locaux, qui sont conçus ailleurs qu'à ITMI, nous utilisons les mêmes algorithmes et les mêmes données pour l'extraction que pour la répartition. Ceci se traduit par le fait qu'on utilise pour l'extraction les DOPs calculés à la répartition.

Boucle Proposition-Elaboration-Evaluation

Seule la partie "élaboration" de la boucle est implantée. La proposition est donnée "à la main" par l'utilisateur, que ce soit la proposition initiale ou une proposition consécutive à un échec de l'élaboration. De même, l'évaluation du plan produit consiste dans l'implantation à détecter d'éventuelles contradictions au niveau de chaque SA *pris individuellement.* L'évaluation des incompatibilités entre SA, et celle de critères plus évolués (marge, stabilité) sont aussi laissées à la charge de l'utilisateur.

La version implantée de PASAN agit donc non pas comme un système autonome de planification, mais comme un outil d'aide à la décision. Elle permet de dégrossir le problème, en calculant les DOPs de mobilisation puis en effectuant l'extraction à partir de la répartition. La suite de ce chapitre décrit strictement ce qui est implanté, sauf mention explicite du contraire.

6.2 L'Acquisition

6.2.1 Acquisition de l'Etat Instantané du Monde

Cette étape consiste à recenser les objets du monde réel intéressant la planification. En l'occurrence, ces objets sont : les pistes, le navire et les SA; le monde réel est le monde de SAFRAN. Cette étape constitue donc l'interface SAFRAN-PASAN. PASAN possède du monde son propre modèle, mais ce dernier est bien évidemment calqué sur la modélisation de SAFRAN.

Le Navire et les SA. Le seul paramètre du Navire acquis à chaque pas de temps est son cap. Ce dernier permet de calculer le secteur battant de chaque SA. Le cap est une variable dynamique uniquement en vue d'éventuels développements. En effet, pour l'instant, aucune action ne permet de changer le cap.

Les Pistes. PASAN acquiert toutes les pistes considérées par SAFRAN. Il acquiert la position et le vecteur vitesse de chaque piste, et leur attribue un nom, le même que dans

SAFRAN. Celui-ci permet d'identifier la piste d'un pas de temps à l'autre comme étant l'écho radar du même objet.

6.2.2 Détermination de l'Ensemble des Points Envisageables d'Interception

Pour une cible et un SA donnés, un point envisageable d'interception est un point de l'espace où la cible peut être interceptée par un objet lancé par le SA. Le but de cette étape est de trouver l'ensemble de ces points.

En fait, nous abusons du flou qui existe autour du terme envisageable. Un interception est envisageable compte-tenu des hypothèses sur l'évolution de la piste, et avant toute considération sur la planification. En effet, la planification va réduire cet ensemble, jusqu'à le réduire à un seul point envisagé, correspondant à l'action planifiée.

L'ensemble initial des points envisageables est égal à l'intersection du secteur battant du SA, de son disque de portée et de l'ensemble des points où peut se trouver la cible. Nous faisons l'hypothèse que la cible suit une trajectoire rectiligne uniforme. Cela permet de relier de manière univoque un point d'interception, une direction de tir et un intervalle temporel d'interception. En l'absence de raisonnement sur l'évolution, la trajectoire rectiligne uniforme est la plus logique car la plus simple. Les limitations de cette hypothèse sont assez évidentes, en particulier, quelques zigzags de la trajectoire réelle provoquent des ensembles de points d'interceptions très éloignés et ont toutes les chances de produire un plan instable. Une question importante est donc ici posée, celle de la robustesse des plans produits lorsque l'hypothèse rectiligne uniforme est totalement fausse.

La trajectoire de la piste est limitée par le point courant (qui est la position au moment de l'acquisition) et le CPA[2]. Elle est limitée par le point courant, car on ne peut pas intercepter la piste dans le passé[3]. La limite du CPA est plus discutable. La raison pragmatique est que cela simplifie les calculs et améliore l'approximation linéaire qui est faite sur le calcul des DOPS. Conceptuellement, on peut le justifier en disant que tirer sur une cible qui s'approche est un acte de nature différente de tirer sur une cible qui s'éloigne. En effet, une cible qui s'éloigne est peu menaçante pour le navire, et surtout est difficilement atteignable, à moins d'avoir un projectile beaucoup plus rapide que la cible. Par conséquent, il est justifié de considérer la partie de trajectoire qui s'éloigne comme appartenant à une autre piste dont on peut planifier séparément l'interception et avec une priorité moins élevée. Pour le moment, cette priorité est tellement faible qu'on ne traite pas ce type de cible. On peut tronque donc la trajectoire au-delà du CPA.

En conclusion, l'ensemble des points d'interception envisageables est un segment de

[2]CPA : Closest Point of Approach, le point le plus proche du Navire de la trajectoire, supposée rectiligne, de la piste. Le TCPA est le temps mis par la piste pour aller jusqu'à ce point à vitesse constante.

[3]Ce n'est d'ailleurs pas une contrainte assez forte, car une interception au point courant suppose un temps d'acquisition nul et une vitesse infinie du projectile. Le raffinement de cette contrainte — temporelle — sera fait au moment du calcul des DOPs.

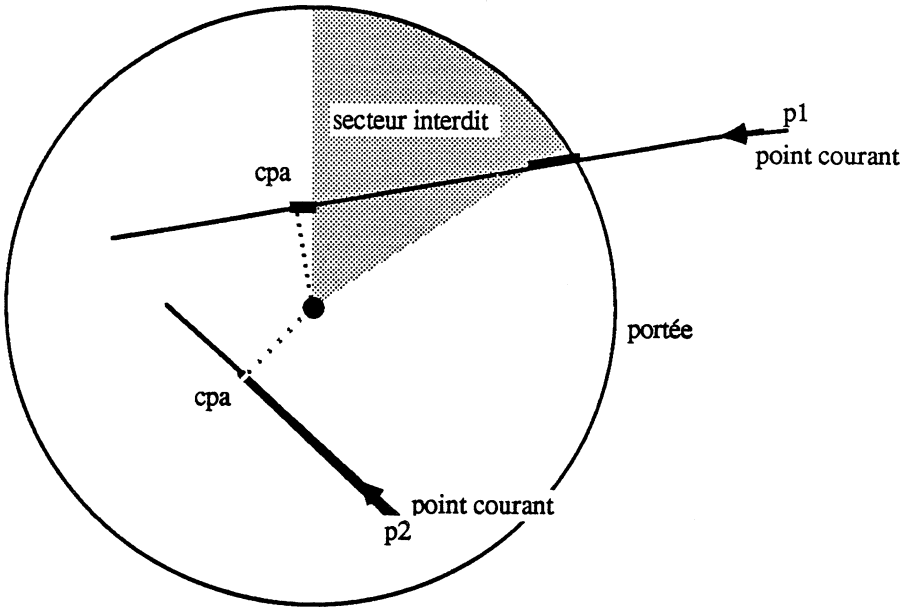


Figure 6.2: Ensembles de Points Envisageables d'Interception

droite, éventuellement troué par le secteur interdit du SA (voir fig 6.2).

6.2.3 Calcul des DOPs de mobilisation

A chaque point d'interception, on peut associer une date t_i d'interception, calculée à partir de l'hypothèse d'évolution. Cette date correspond à la fin de la durée de vol du projectile, durée égale à d_i/v_p où d_i égale la distance du navire au point d'interception et v_p égale la vitesse du projectile. Par conséquent, si on note t_a la durée d'acquisition du SA, ce dernier est mobilisé pendant l'intervalle $[t_i - d_i/v_p - t_a \quad t_i]$, si le projectile est conduit par le SA pendant le vol (ce qui est le cas pour certains missiles), et $[t_i - d_i/v_p - t_a \quad t_i - d_i/v_p]$ sinon.

A l'ensemble des points d'interception peut ainsi être associé un ensemble d'intervalles de mobilisation, que nous appellerons *DOP de mobilisation.* Lorsque le SA est mobilisé pendant la durée du vol, l'équation d'un DOP de mobilisation est :

$$t_d = t_i - \sqrt{(d_c/v_p)^2 + (v_c/v_p)^2 (t_i - t_c)^2} - t_a$$

où t_d est le début de la mobilisation, t_i la fin, t_c le TCPA, d_c la distance du navire au CPA et v_c la vitesse de la cible. La figure 6.3 montre quelques DOPS de mobilisation dans le plan (t_d, t_i), c'est à dire (début, fin), pour lesquels t_a est nul et $t_c = 100$. En général, les DOPs sont des arcs du second degré, croissants, convexes et de dérivée inférieure à 1 (la durée de vol décroit lorsque le début du tir croit).

Garder cette forme pendant la propagation temporelle serait assez lourd, d'autant plus que c'est déjà une approximation reposant sur l'hypothèse rectiligne uniforme. Nous l'approximons donc au segment reliant les deux extrémités de l'arc.

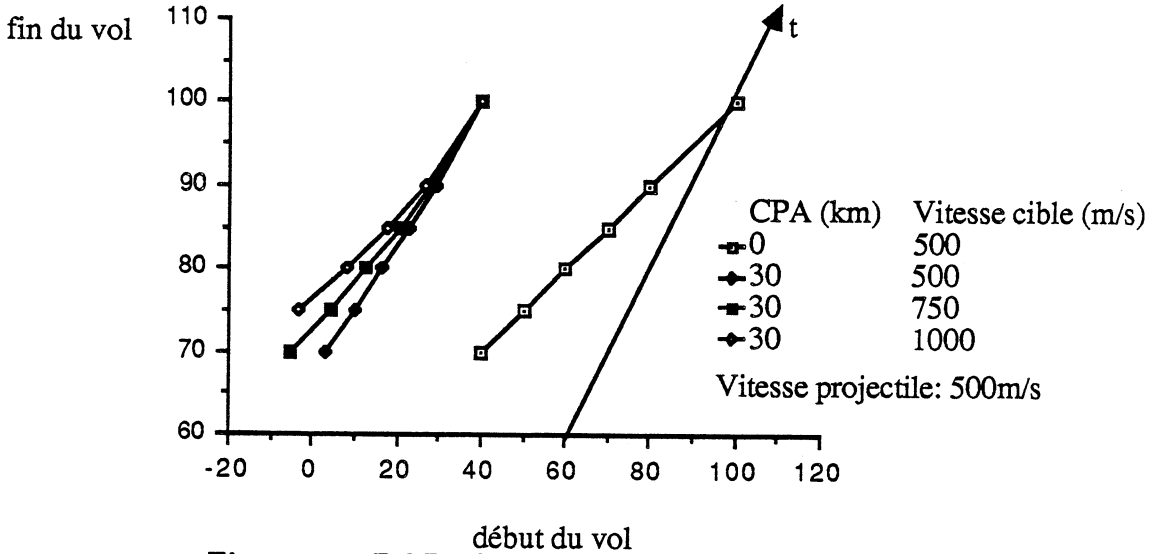


Figure 6.3: DOPs de vol du projectile

Ce dernier étant convexe, cela revient à surestimer la durée de vol et donc de mobilisation. Etant donné le type des contraintes du problème (contraintes de non-recouvrement), cela revient à surcontraindre ce dernier, c'est-à-dire à rejeter d'emblée certaines éventuelles solutions (ou aussi à dériver un POC incomplet).

Il faut noter que la convexité est d'autant plus prononcée (et donc l'approximation linéaire moins bonne) que la distance au CPA et le rapport des vitesses de la cible et du projectile sont grands (fig 6.3). Ceci correspond à des cas d'interception à la fois difficiles et peu fréquents (dans le cadre de l'auto-défense). A l'opposé, dans le cas d'une trajectoire radiale, le DOP de mobilisation est rigoureusement un segment de droite.

Si l'ensemble des points d'interception est constitué d'une réunion de segments, le calcul précédent est appliqué à chacun d'eux.

Après le calcul des DOPs de mobilisation, il est nécessaire de leur appliquer la contrainte "après l'instant courant" car une action ne peut pas être commencée dans le passé[4].

A l'issue de cette étape, une piste est dite affectable à un SA si le DOP de mobilisation du SA sur cette piste n'est pas vide.

[4]A moins que ce ne soit une action en cours d'exécution, nous étudierons ce cas à la section 6.4.

Notons aussi que cette contrainte est plus forte que celle, précédemment appliquée, de la troncature du segment d'interception au point courant. En effet, ce dernier correspond seulement à la fin de la mobilisation.

6.3 La Planification

6.3.1 La Répartition

C'est à ce niveau que l'on décide quelles pistes on traite avec quels systèmes d'armes. On calcule aussi certaines conséquences de ce choix sur les ensembles d'intervalles possibles d'interception .

Définition d'une Répartition

Pour un SA donné, une répartition associe à chaque piste affectable une des valeurs *prioritaire, annexe, non-traitée.* Le SA ignore les pistes non-traitées, il doit traiter les pistes prioritaires, et essaye de traiter le maximum de pistes annexes.

La répartition courante est la répartition que l'on est en train de planifier.

Calcul du POC de la Répartition Courante

Ce calcul permet de comprendre les implications d'une répartition en termes d'intervalles possibles de mobilisation. Il permet une boucle locale élaboration - évaluation - modification dans la mesure où l'utilisateur peut proposer une nouvelle répartition sans aller jusqu'au bout du processus de planification. On peut dire que la répartition est faite de manière semi-automatique par le système qui choisit lui-même les pistes annexes qui seront traitées.

Pour chaque SA, on construit un problème d'occurrences contraintes, ou POC (voir 3.1.3) à partir des pistes prioritaires. Les nœuds temporels de ce POC sont nommés d'après les pistes, et les DOPs attachés sont les DOPs de mobilisation du SA. Chaque nœud est relié aux autres par une contrainte de non-recouvrement ($< >$). On applique à ce POC l'algorithme de dérivation locale. Si, lors de la propagation, un DOP de mobilisation devient vide cela signifie qu'une piste prioritaire n'est pas traitable, donc le calcul s'arrête et il faut proposer une nouvelle répartition.

Si ce n'est pas le cas, on introduit une à une les pistes annexes en ajoutant à chaque fois au POC que l'on vient de calculer, un nœud temporel. Ce nœud est aussi lié aux autres par une contrainte de non-recouvrement que l'on propage. Cependant, contrairement aux pistes prioritaires, un DOP qui se vide au cours de la propagation ne provoque pas d'invalidation de la répartition. L'introduction de la piste provoquant la contradiction est rejetée, et on modifie la répartition courante de manière à ce que finalement, le plus possible de pistes annexes soient introduites. Pour ce faire, on développe l'arbre des répartitions possibles en gardant, lors de l'introduction des pistes annexes, les POCs intermédiaires qui y correspondent. L'expansion de l'arbre est guidée par la minimisation d'une fonction d'évaluation qui vaut le nombre de nœuds rejetés.

Qualité de l'Elimination : Le Problème des Contraintes sur les Durées

En calculant les intervalles possibles de mobilisation, on peut tester la validité d'une répartition (si aucun DOP ne se vide), et un éventuel ordre partiel implicite entre ces intervalles. La qualité d'une telle évaluation est bien entendu fonction de l'acuité avec laquelle on élimine les occurrences impossibles. Nous avons vu au chapitre précedent que l'algorithme de propagation employé ne permettait pas d'éliminer à coup sûr toutes les occurrences impossibles, en particulier lorsqu'il y a des disjonctions. Or il y a beaucoup de disjonctions dans PASAN, et l'évaluation ne peut donc être qu'approximative. Il est donc intéressant de savoir dans quelle mesure.

L'expérience montre que la qualité du calcul est nettement insuffisante. En effet, l'algorithme d'élimination ne sait pas détecter des situations dans lesquelles plus de trois pistes sont traitées par le même SA, alors que la somme des durées de mobilisation est supérieure à la longueur de la "fenêtre globale", formée du début au plus tôt et de la fin au plus tard de toutes les mobilisations.

Ne pas pouvoir détecter ce genre d'inconsistance signifie donc que l'on laisse passer fréquemment une répartition inadmissible, d'où la mauvaise qualité du calcul. Ceci n'est pas dramatique, car l'algorithme d'extraction peut quand même tirer une solution de ce genre de répartition. Cependant, cela le conduit à faire retours en arrière qui, lorsqu'ils sont trop fréquents, finissent par coûter cher.

6.3.2 L'Extraction d'une Solution Opérationnelle

C'est à ce niveau que l'on décide à quel moment les systèmes d'armes tirent sur les pistes qui leur sont attribuées. On est aussi ammené à décider de ne pas traiter certaines pistes, lorsqu'on ne peut pas les placer dans le plan.

Définition

Pour un SA donné, une solution opérationnelle est un ensemble de couples (intervalle temporel, piste).

Un couple correspond à une intention de mobilisation du SA pendant l'intervalle temporel pour tirer sur la piste. L'ensemble de ces couples est donc un plan d'actions, au sens où on l'entend généralement, précisément daté.

Rappelons que dans cette implantation, l'extraction prend en entrée, en plus de la répartition, le POC de la répartition courante restreint à chaque SA.

La fonction d'extraction consiste à calculer une solution opérationnelle à partir d'une répartition. Cela consiste à "extraire" de chaque DOP du POC de la répartition courante une occurrence et une seule. Cette fonction correspond à celle que remplirait un planificateur local.

Calcul d'une Solution Opérationnelle

Le principe de ce calcul est de choisir une occurrence dans un DOP, de propager les conséquences de ce choix, puis de choisir une occurrence dans un autre DOP, jusqu'à ce que tous soient traités.

Choisir une occurrence signifie remplacer le DOP original par un DOP réduit à une seule occurrence. Cette opération peut-être vue comme l'application d'une contrainte, qui doit être propagée sur les autres DOPs du POC de la répartition courante. Une telle élimination constitue la conséquence du choix de l'occurrence.

On est donc ammené à résoudre successivement les trois problèmes suivants :

1. Dans quel DOP choisir une occurrence ?

   L'ordre dans lequel on réduit les DOPs influe sur la solution finale. En effet, les occurrences des autres DOPS, que l'on élimine lors de la propagation, ne pourront plus être choisies. Par conséquent, plus un DOP est réduit tôt, plus on a le choix de l'occurrence.

   Nous avons classés, pour l'extraction, les DOPs dans l'ordre croissant de leur début au plus tôt. On peut le justifier par le désir de garder le plus de latitude possible dans le choix de l'occurrence lorsque le DOP dans lequel on choisit appartient au futur proche.

2. Quelle occurrence choisir ?

   Plus le tir commence tard, plus il est court, donc plus il est efficace, mais plus il finit tard. Nous avons simplifié l'aspect d'efficacité en l'intégrant dans la portée, que l'on suppose permettre une efficacité "acceptable". Nous avons donc opté pour un choix de tir au plus tôt.

3. Que faire si la propagation échoue ?

   La propagation échoue si un des DOPs se vide. Cela signifie que l'occurence choisie n'appartient à aucune solution du POC courant. Une telle situation est possible dans la mesure où les propagations antérieures n'assurent pas la consistance globale. Elle devrait cependant être rare.

   Il est nécessaire, en cas d'échec, de revenir sur un des choix précédents. Ce peut être sur l'occurrence, que l'on peut choisir ailleurs dans le DOP, sur le choix du DOP, que l'on peut décider de réduire plus tard (en espérant que d'ici là on aura éliminé l'occurrence fautive et ses voisines), ou enfin sur l'existence même du DOP. C'est cette dernière solution que nous avons implantée : si le choix d'une occurrence dans un DOP provoque l'apparition d'une contradiction, on rejette entièrement le DOP. C'est là une solution assez radicale, c'est pourquoi nous esquissé les autres possibilités. Elle est tempérée par le fait que l'on cherche à garder le maximum de DOPs, grâce à un algorithme analogue à celui qui cherche à introduire le maximum de pistes annexes lors de la répartition.

   Notons, à ce sujet, qu'un retour-arrière y est plus coûteux qu'à la répartition. En effet, lorsque l'on revient sur la décision d'introduire un DOP, il suffit de revenir sur le

POC dans l'état qu'il avait avant l'introduction, quand il y avait un nœud de moins. Un tel état peut être "sauvé" à chaque introduction. Dans le cas de l'extraction, tous les nœuds sont présents dès le départ. Lorsqu'on revient sur une extraction, on revient *aussi*, dans notre implantation, sur l'existence du DOP correspondant. Or cette existence est une hypothèse dans *toutes* les propagations précédentes. Revenir sur elle invalide donc ces propagations, autrement dit, il faut les recommencer depuis le début, sur le POC amputé du DOP rejeté.

6.4 L'Exécution

6.4.1 Le module d'Exécution

L'exécution consiste à entériner — avec l'accord de l'utilisateur — une solution opérationnelle qui devient alors *Le plan en cours d'exécution.* Cette donnée n'est disponible que pour PASAN lui-même. Un module symétrique à l'acquisition et constituant une interface PASAN-SAFRAN n'est pas implanté. Par conséquent, PASAN croit que le navire tire mais le monde extérieur (SAFRAN) ne le sait pas. Cela ne remet pas en cause l'activité de planification, mais nuit seulement au réalisme de la simulation. En particulier, on ne peut pas simuler les réactions des pistes aux tirs, alors que SAFRAN le permettrait, si les pistes avaient un moyen de savoir qu'elles sont poursuivies.

6.4.2 La Prise en Compte du Plan en Cours d'Exécution

Pour améliorer la clarté de l'exposé, nous regroupons ici les effets, sur PASAN, de l'existence d'un plan en cours d'exécution.

Effet sur l'Acquisition

Acquisition du Monde Extérieur A chaque pas de temps, PASAN acquiert les actions en cours d'exécution. Pour chaque SA, l'ensemble des actions en cours d'exécution est l'ensemble des couples *(intervalle temporel, piste)* tels que l'instant courant appartient à l'intervalle temporel.

DOPs de Mobilisation Nous avons vu qu'à chaque DOP de mobilisation on applique la contraintes "après l'instant courant" car une action ne peut commencer dans le passé. Nous affinons ici ce raisonnement.

La figure 6.4 montre l'espace des occurrences divisé en trois régions par l'instant courant :

- La région des occurrences futures : dans cette région, chaque action est liée à un DOP qui a une certaine étendue. Celle-ci correspond à l'éventail des possibilités de choix du planificateur et à l'incertitude sur l'évolution.

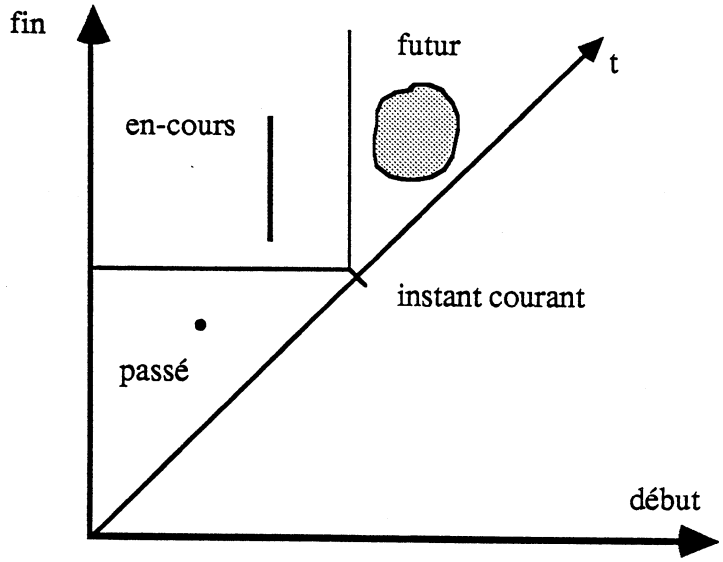


Figure 6.4: Effet de l'instant courant sur la forme des DOPs.

- La région des occurrences présentes : Les actions ayant un DOP dans cette région ont déjà commencé. Il est raisonnable de penser que PASAN sait quand une action de tir a commencé. Donc les DOPs de cette région ont un début parfaitement déterminé, et ce sont des segments de droite verticaux. Dans l'implantation, nous simplifions ce cas en ne considérant qu'un point. Qui plus est, ce point n'est pas recalculé à chaque pas de temps, ce qui permettrait de tenir compte des changements cinématiques de la piste à laquelle il correspond.

- La région des occurrences passées : les actions ayant un DOP dans cette région sont terminées. On suppose qu'on en connait donc le début et la fin. Donc les DOPs de cette région sont des points.

Par conséquent, pour tenir compte de l'ensemble des possibilités offertes au planificateur, le DOP de mobilisation est la réunion du DOP "futur" et du DOP (réduit ici à une occurrence) présent. Ces deux parties correspondent aux deux termes de la disjonction suivante : ou bien la piste est traitée par le tir en cours, ou bien elle est traitée par un autre tir, auquel cas le tir en cours est annulé.

Influence sur la Planification

Dans un DOP de mobilisation, les deux termes de la disjonction *DOP futur - DOP présent* ne représentent pas des possibilités équivalentes. En effet, il est en général préférable de terminer une action commencée plutôt que de l'interrompre. Nous avons donc utilisé des algorithmes qui, implicitement, "préfèrent" l'occurrence en cours à toutes les autres :

- Lors de la répartition : Toutes les pistes dont le DOP de mobilisation contient une occurrence en cours sont nécessairement prioritaires. Cela revient à dire qu'un tir ne peut être interrompu pour maximiser le nombre de pistes traitées.

- Lors de l'extraction : L'ordre d'extraction des DOPs est celui du début au plus tôt. Donc si un DOP contient une action en cours, son début au plus tôt est inférieur à celui de tous les autres, tronqués à l'instant courant. Donc c'est de lui que l'on extraiera la première occurrence. Comme, dans un DOP, on choisit l'occurrence de plus petit début, c'est l'occurrence en cours qui sera choisie.

Synthèse

Nous avons montré dans cette partie que notre formalisation n'est pas limitée à des descriptions abstraites ou à la résolution de problèmes jouets.

Nous avons tenu à replacer dans un cadre concret, et plus large, le problème précis abordé à la partie II. Ceci nous a conduit à évoquer des problèmes externes à ceux relatifs à notre représentation du temps. Ils ont cependant un intérêt à être posés dans la mesure où ils concernent la planification et le temps ou, plus précisément, la planification en contexte fortement évolutif.

Nous avons mis en évidence un problème d'occurrences contraintes. Ce problème correspond au cas précis de la planification des interceptions pour un système d'armes, à un instant donné, et moyennant une hypothèse donnée d'évolution de la cible.

En effet, dans ce cas, pour chaque cible, chaque point d'interception possible correspond à un instant d'interception. Cet instant est la fin d'un intervalle de vol du projectile, c'est à dire d'une occurrence de tir. On peut, avec une marge d'erreur assez faible, approximer le domaine des occurrences de tir possibles à un segment de droite. Ce segment traduit une dépendance linéaire de la durée du tir par rapport à son début ; plus on tire tard, plus la cible est proche et plus courte est la trajectoire du projectile. Les relations temporelles sont exclusivement des relations, fortement disjonctives, de non-recouvrement : un système d'armes ne conduit qu'un tir à la fois.

Notre formalisme est adapté au problème de l'affectation des systèmes d'armes, principalement parce que c'est un problème d'allocation de ressources. En effet, on associe facilement, sur une ressource, un intervalle à une tâche (ici un tir) indépendamment d'un formalisme de planification. Les contraintes assez fortes qui s'exercent sur les domaines et la présence de disjonctions sont autant de facteurs favorables à l'utilisation de ce formalisme.

En ce qui concerne les perspectives de l'utilisation de cette représentation du temps, on peut noter la manifestation concrète des problèmes évoqués à la fin de la partie précédente :

L'efficacité des algorithmes de géométrie s'avère être effectivement un élément déterminant, en ce qui concerne aussi bien le temps d'exécution que le temps d'implantation. Nous n'y avons pas apporté un soin particulier, considérant que notre problème de recherche se situait ailleurs. La propagation et le calcul d'une contrainte sur un DOP prend actuellement, sur une machine lisp de type Explorer, environ 0.2 secondes CPU, ce qui permet de mesurer les progrès qui restent à faire en ce domaine.

L'élimination des occurrences rendues impossibles par des contraintes de capacité est aussi un point important. En effet, avant de commencer l'extraction d'un plan, c'est à dire le choix des occurrences, il importe de savoir s'il n'y a pas plus de cibles que ne peut en traiter le système d'armes. De plus, il serait intéressant d'avoir une estimation du nombre de pistes qu'on peut espérer traiter, avant de chercher à calculer comment on va le faire.

La capacité à revenir en arrière, en particulier à retirer des nœuds d'un problème d'occurrences contraintes, serait aussi bien utile. En effet, plutôt que d'introduire les pistes une à une, il est naturel ici de les retirer une à une. En effet, devant une vague d'attaque, dans le cadre de l'auto-défense, on cherche d'abord à éliminer tous les agresseurs. Si c'est impossible, et seulement alors, on se résout à ne pas tirer sur les moins dangereux.

Conclusion

Nous renvoyons le lecteur à la fin de chaque partie pour une récapitulation technique synthétique et pour une mise en évidence de quelques problèmes précis que notre représentation laisse ouverts. Nous tenterons ici d'évaluer avec autant de recul que possible notre travail.

La valeur de notre approche repose sur trois thèses. Elles sont encore largement discutables et nous prétendons seulement avoir posé les fondements d'une argumentation solide.

1. Il est utile de construire une représentation explicite du temps pour la planification et indépendamment du formalisme de planification.

   Ce n'est pas, en tout cas, une opinion qui va à contre-courant : dans la première partie, nous avons décrit un certain nombre de représentations du temps indépendantes d'un formalisme de planifications mais utilisées par des planificateurs. Dans la deuxième partie nous avons construit une représentation du temps, que nous avons utilisée, dans la troisième partie, pour résoudre un problème de planification. Certes, c'est un problème particulier, qui plus est d'affectation de ressources, ce qui permet une modélisation bien plus simple que si on devait construire un système de génération de plans : une tâche est facilement associée à un intervalle d'occurrence et elle est liée aux autres tâches par des relations simples, par exemple la relation de non-recouvrement temporel.

2. Il est utile de garder explicitement toutes les occurrences possibles.

   C'est peut-être là le point le plus faiblement justifié. Nous le faisons dans l'introduction en affirmant qu'il serait prématuré, à un niveau aussi élémentaire du raisonnement, de supprimer des occurrences si on ne peut montrer qu'elles sont impossibles. Or nous ne mettons en évidence, ni dans une construction théorique, ni dans l'exemple, un niveau supérieur qui ait besoin d'utiliser la représentation explicite de toutes les occurrences possibles.

3. Il est utile de propager des contraintes disjonctives sans les décomposer entièrement.

   Nous montrons que c'est possible et que c'est applicable. Cependant, nous montrons aussi qu'une telle propagation ne permet pas de résoudre complètement, en général, un problème d'occurrences contraintes. Pour ce faire, il faut, de toutes façons, décomposer le disjonctions. On peut facilement construire des cas où une propagation est faite en pure perte. Quant à la complexité pratique, il serait nécessaire de comparer les résultats

de notre implantation avec ceux d'une implantation "classique" qui décompose les disjonctions au lieu de les propager.

Finalement, même si ces thèses ne sont pas démontrées, nous donnons une formalisation propre du problème et un exemple montrant qu'il est, sur le plan pratique, intéressant de le résoudre. La technique de dérivation locale est trop faible pour le faire dans le cas général, aussi bien d'un point de vue théorique (le problème est au moins NP-dur de toutes façons), que pratique (où un raisonnement de capacité semble indispensable). Ce n'est pas la seule qu'on puisse utiliser, il appartient à des travaux ultérieurs d'en formaliser d'autres qui la complètent ou qui soient directement meilleures.

Annexe A

Quand la dérivation locale Suffit-elle pour Avoir une Consistance Globale?

A.1 Le Résultat

Théorème 1 *Soit* $\mathcal{P} = \left((O_i)_{i=1,n}, (R_{ij})_{i,j\in[1,n]}\right)$ *un POC. Si :*

H1 *Les relations* R_{ij} *ne sont pas disjonctives :*

$$\forall i,j \in [1,n], R_{ij} \in \{(<),(>),(m),(mi),(d),(di),(o),(oi),(s),(si),(f),(fi),(=)\}$$

H2 *Les DOPs sont des segments "montants" :*

Pour tout i *:*

- *les occurences*

$$o_i^m = \left(\min_{o\in O_i} début(o), \min_{o\in O_i} fin(o)\right)$$

et

$$o_i^M = \left(\max_{o\in O_i} début(o), \max_{o\in O_i} fin(o)\right)$$

appartiennent à O_i*)*

- $\forall o,p,q \in O_i, p \neq q \Rightarrow \exists \lambda \in \Re / o = \lambda(p-q)+p$

H3 *Les DOPs sont convexes :*

$$\forall o,p \in O_i, \forall \lambda \in [0,1], \lambda o + (1-\lambda)p \in O_i$$

Alors l'algorithme de dérivation locale résout $\mathcal{P}$*.*

Soit $\mathcal{P}$ un POC vérifiant H1, H2 et H3 et arc-consistant montrons que :

1. toute occurrence
2. appartient à une solution

Considérons pour tout $\lambda \in [0,1]$ le choix d'occurrences :

$$\left(\lambda o_i^m + (1-\lambda) o_i^M\right)_{i\in[1,n]}$$

Intuitivement, cet ensemble correspond au choix dans chaque DOP d'une occurrence à une distance relative donnée des extrémités.

1. En raison de H2 et H3, les points du choix appartiennent bien aux DOPs de $\mathcal{P}$: ils appartiennent au segment reliant les deux extrémités définies par H2. De plus, d'apres H2, toute occurrence appartient à un choix.

2. Les occurrences du choix vérifient toutes les relations :

   Ceci réclame une vérification, fastidieuse pour toutes les relations (élémentaires) aussi ne le feront nous ici que pour la relation $<$. Soit donc $R_{ij} = (<)$ alors la vérification de la consistance d'arc implique que

   $$\min_{o\in O_j} \text{début}(o) > \min_{o\in O_i} \text{fin}(o) \wedge \max_{o\in O_j} \text{début}(o) > \max_{o\in O_i} \text{fin}(o)$$

   Donc, comme $\lambda \in]0,1[\Rightarrow \lambda > 0 \wedge 1-\lambda > 0$ en multipliant les inégalités ci-dessus par λ et $1-\lambda$ respectivement et en les additionant membre à membre on obtient :

   $$\lambda \min_{o\in O_i} \text{fin}(o) + (1-\lambda) \max_{o\in O_i} \text{fin}(o) < \lambda \min_{o\in O_j} \text{début}(o) + (1-\lambda) \max_{o\in O_j} \text{début}(o)$$

   Les cas où λ égale 0 ou 1 sont aussi trivialement vérifiés. Cette dernière inéquation signifie que la fin de l'occurrence choisie dans O_i est inférieure au début de l'occurrence choisie dans O_j donc que R_{ij} est bien vérifiée.

   La vérification des autres relations obeit aux mêmes principes : la consistance d'arcs implique des inégalités (ou des égalités) entre les débuts et fins minimaux ou entre les débuts et fins maximaux. En multipliant les membres de ces inégalités par λ ou $1-\lambda$ et en les additionnant, on obtient les inégalités sur les débuts et fins des choix d'occurrences qui caractérisent la relation temporelle qui doit être vérifiée.

A.2 Discussion des Hypothèses

Le théorème précédent donne une condition suffisante (H1 et H2 et H3) pour que la consistance locale implique la consistance globale. Cette condition n'est pas nécessaire. On peut facilement construire un POC ne vérifiant pas l'une des trois hypothèses et que "par chance" la dérivation locale résout globalement. Cependant, on peut discuter la *pertinence* des hypothèses, à savoir : le théorème est-il toujours vrai si on enlève l'une d'elle? Cette section

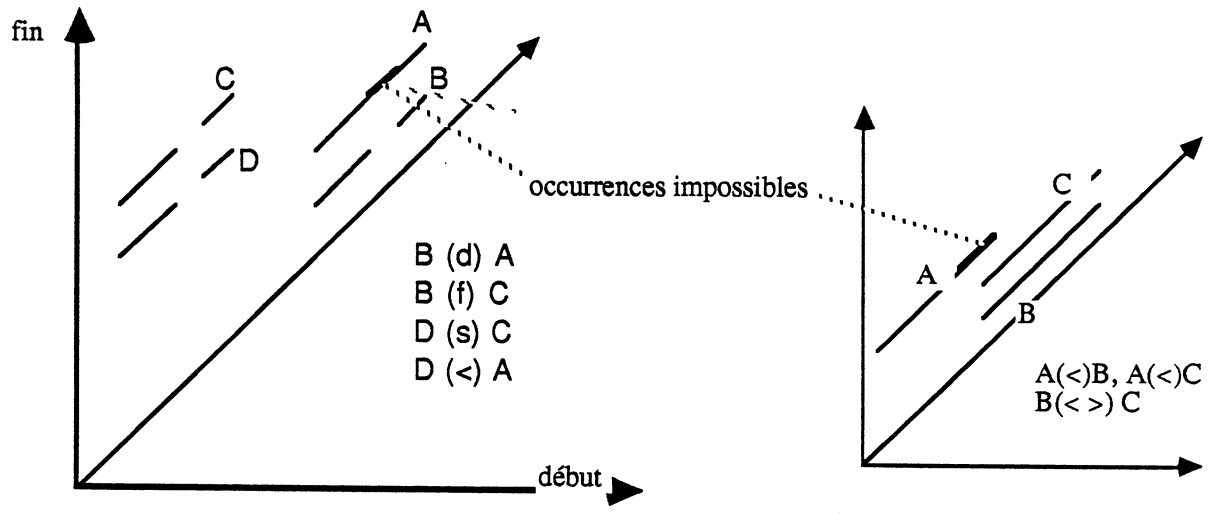


Figure A.1: Lorsqu'on lève H3 ou H1, il n'y a plus de consistance globale.

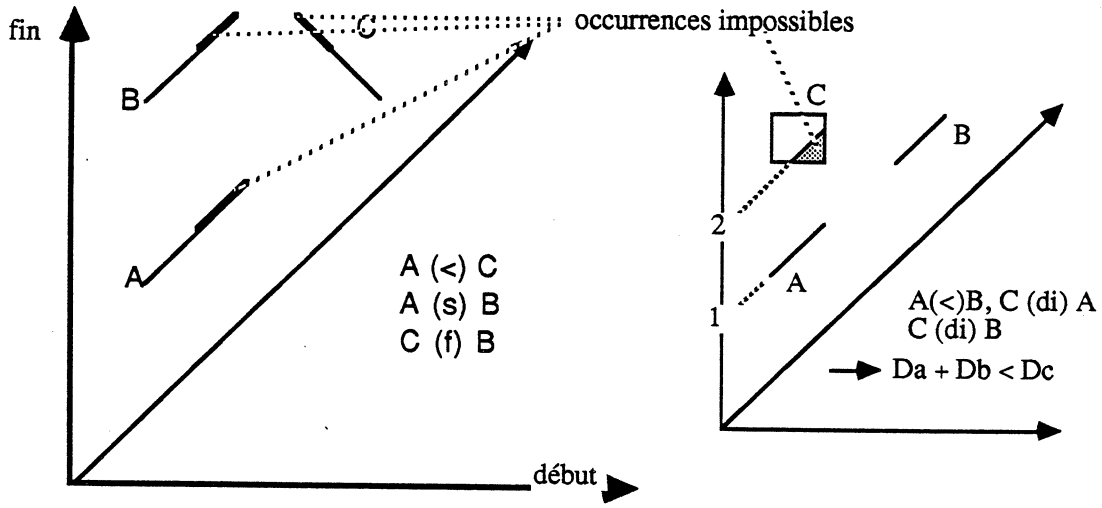


Figure A.2: Lorsqu'on lève H2, il n'y a plus de consistance globale.

montre que non, d'où la pertinence des hypothèses. Nous donnons dans cette section un contre-exemple pour chaque cas de figure (soit quatre, si on décompose H2 en deux parties). Pour des raisons de concision, nous ne donnons que des dessins. Le lecteur est invité à vérifier pour chaque POC représenté que :

- Les POCs sont localement consistants : tous les DOPs sont bien inclus dans l'intersection de régions qui leur sont permises.

- Mais ils ne sont pas globalement consistants : les occurrences signalées comme impossibles sur la figure n'appartiennent à aucune solution. Ceci peut être montré en constatant que la propagation du choix d'une occurrence impossible conduit à vider un des DOPs[1].

Le deuxième dessin de la figure A.1 montre que lever H1, c'est-à-dire autoriser les relations disjonctives, suffit pour que la 2-consistance n'implique pas la 3-consistance (et a fortiori la consistance globale). On peut cependant affaiblir un peu H1, en autorisant les contraintes disjonctives faibles (voir section 3.2.2). En effet, ces dernières peuvent s'exprimer par une conjonctions d'égalités et d'inégalités et la démonstration du théorème est identique.

Le premier dessin de la figure A.1 montre que lever H3, c'est-à-dire autoriser des trous dans les DOPs, suffit pour que la 2-consistance n'implique pas la 3-consistance. On peut remarquer que H1 et H3 sont liées dans le sens où seules des relations non-disjonctives (plus précisément faiblement disjonctives) conservent la convexité des DOPs, c'est-à-dire ne créent pas de trous dans les segments.

La figure A.2 montre la pertinence de l'hypothèse H2. On remarquera qu'ici, le problème n'est pas lié à la convexité.

[1]Le lecteur pourra aussi remarquer que *toutes* les occurrences impossibles sont signalées. Si on retire les parties en gras, les POCs résultants sont consistants globalement (et constituent autant de preuves de la non necessité de H1 H2 et H3).

Annexe B

Tables de Primitives Symboliques

B.1 Table d'Allen

Cette table (fig. B.1), tirée de [3], définit la loi $\circ$ pour des relations primitives entre intervalles (voir section 2.3.1) :

$$Ar_1B \wedge Br_2C \Longrightarrow A(r_1 \circ r_2)C$$

Elle se lit comme suit :

	r_2
r_1	$r_1 \circ r_2$

Les symboles bef, aft et Γ sont équivalents respectivement à $<$, $>$, et ($< > =$ d di o oi m mi s si f fi).

B.2 Tables de Conversion Intervalles-Instants

Ces tables (fig. ??) ont été établies par T. Granier [23]. Les relations entre points sont notées, avec un sens évident, same, prec et fol.

- Intervalles $\rightarrow$ instants : à une relation primitive r entre deux intervalles A et B (première ligne du tableau), on fait correspondre quatre relations entre, respectivement, le début de A et celui de B (deuxième ligne), le début de A et la fin de B (troisième ligne), la fin de A et le début de B (quatrième ligne), la fin de A et la fin de B (cinquième ligne).

- Instants $\rightarrow$ intervalles : inversement, à toute relation primitive entre deux points dénotant le début ou la fin de deux intervalles A et B (valeur de la relation sur la première ligne, dénotation des points sur la première colonne), on fait correspondre une relation disjonctive entre A et B. La relation finale est la conjonction des relations obtenues à chaque ligne. On remarquera que les lois de conjonction et de disjonction

	bef	aft	d	di	o	oi	m	mi	s	si	f	fi
bef	(bef)	Γ	(bef o m d s)	(bef)	(bef)	(bef o m d s)	(bef)	(bef o m d s)	(bef)	(bef)	(bef o m d s)	(bef)
aft	Γ	(aft)	(aft oi mi d f)	(aft)	(aft oi mi d f)	(aft)	(aft oi mi d f)	(aft)	(aft oi mi d f)	(aft)	(aft)	(aft)
d	(bef)	(aft)	(d)	Γ	(bef o m d s)	(aft oi mi d f)	(bef)	(aft)	(d)	(aft oi mi d f)	(d)	(bef o m d s)
di	(bef o m di fi)	(aft oi di mi si)	(o oi d s f di si fi equ)	(di)	(o di fi)	(oi di si)	(o di fi)	(oi di si)	(o di fi)	(di)	(oi di si)	(di)
o	(bef)	(aft oi di mi si)	(o d s)	(bef o m di fi)	(bef o m)	(o oi d s f di si fi equ)	(bef)	(oi di si)	(o)	(o di fi)	(o d s)	(bef o m)
oi	(bef o m di fi)	(aft)	(oi d f)	(aft oi di mi si)	(o oi d s f di si fi equ)	(aft oi mi)	(o di fi)	(aft)	(oi d f)	(aft oi mi)	(oi)	(oi di si)
m	(bef)	(aft oi di mi si)	(o d s)	(bef)	(bef)	(o d s)	(bef)	(f fi equ)	(m)	(m)	(o d s)	(bef)
mi	(bef o m di fi)	(aft)	(oi d f)	(aft)	(oi d f)	(aft)	(s si equ)	(aft)	(oi d f)	(aft)	(mi)	(mi)
s	(bef)	(aft)	(d)	(bef o m di fi)	(bef o m)	(oi d f)	(bef)	(mi)	(s)	(s si equ)	(d)	(bef o m)
si	(bef o m di fi)	(aft)	(oi d f)	(di)	(o di fi)	(oi)	(o di fi)	(mi)	(s si equ)	(si)	(oi)	(di)
f	(bef)	(aft)	(d)	(aft oi di mi si)	(o d s)	(aft oi mi)	(m)	(aft)	(d)	(aft oi mi)	(f)	(f fi equ)
fi	(bef)	(aft oi di mi si)	(o d s)	(di)	(o)	(oi di si)	(m)	(oi di si)	(o)	(di)	(f fi equ)	(fi)

Figure B.1: Table de composition des 13 primitives entre intervalles

de contraintes permettent de ne pas construire explicitement la table de *tous* les motifs de relations entre quatre points.

Si, par exemple, les points de fin sont reliés par (prec same) et ceux de début par (same), les intervalles correspondants sont reliés par :

$((< \text{o m d s}) \vee (= \text{f fi})) \wedge (= \text{s si}) = (= \text{s})$.

IP	=	<	>	*d*	*di*	*o*	*oi*	*m*	*mi*	*s*	*si*	*f*	*fi*
dd	*same*	*prec*	*fol*	*fol*	*prec*	*prec*	*fol*	*prec*	*fol*	*same*	*same*	*fol*	*prec*
df	*prec*	*prec*	*fol*	*prec*	*prec*	*prec*	*prec*	*prec*	*same*	*prec*	*prec*	*prec*	*prec*
fd	*fol*	*prec*	*fol*	*fol*	*fol*	*fol*	*fol*	*same*	*fol*	*fol*	*fol*	*fol*	*fol*
ff	*same*	*prec*	*fol*	*prec*	*fol*	*prec*	*fol*	*prec*	*fol*	*prec*	*fol*	*same*	*same*

intervalles → instants

	same	prec	fol
dd	(=,s,si)	(<,di,o,m,fi)	(>,d,oi,mi,f)
df	(mi)	(=,<,d,di,o,oi,m,s,si,f,fi)	(>)
fd	(m)	(<)	(=,>,d,di,o,oi,mi,s,si,f,fi)
ff	(=,f,fi)	(<,d,o,m,s)	(>,di,oi,mi,si)

instant → intervalles

Figure B.2: Tables de conversion intervalles ⟷ instants

Bibliographie

[1] Aho, Hopcroft, Ullman. Section 5.6 *Path-Finding Problems* of Chapter 5 *Algorithms on Graphs.* in *The Design and Analysis of Computer Algorithms.* Addison-Wesley 1974.

[2] J.F.Allen. *Towards a General Theory of Action and Time.* Artificial Intelligence, (23):123–154, 1984.

[3] J.F. Allen. *Maintaining Knowledge about Temporal Intervals.* Communications of the ACM, 26(11):832–843, November 1983.

[4] J.F.Allen and J.A.Koomen. *Planning Using a Temporal World Model.* In Proceedings of the eighth IJCAI, pages 741–747, 1983.

[5] J.F. Allen and P.J. Hayes. *A Commonsense Theory of Time.* In Proceedings of the ninth IJCAI, Los Angeles, pages 528–531, Aug 1985.

[6] C.Barrouil *Contrôle de la Cohérence d'un Plan par la Méthode du Simplexe.* In Proceedings 4eme Congrès Reconnaissance des Formes et Intelligence Artificielle, Paris, pages 531–542, Jan 1984.

[7] P.Bessiere, P.Grojean, P.Fondaneche. *SAFRAN* Huitièmes Journées Internationales sur les Systèmes Experts et leurs Applications, Avignon, Mai 1988 (a paraître).

[8] E.Charniak and D.McDermott. *Managing Plans of Action.* Chapter 9 of "Introduction to Artificial Intelligence". Addison-Wesley Publishing Company, 1985.

[9] A.D.Christiansen. *The History of Planning Methodology: An Annotated Bibliography.* SIGART Newsletter, Number 94, page 44, October 1985. 1

[10] O. Costa de Beauregard. *Temps.* Encyclopedia Universalis, 1985.

[11] E. Davis. *Constraint Propagation with Interval Labels.* Artificial Intelligence 32, pages 281–331, 1987.

[12] A.Dechter and R.Dechter. *Removing Redundancies in Constraint Networks.* In Proceedings of the sixth National Conference on Artificial Intelligence (AAAI-87), pages 105-109, 1987.

[13] H.Delesalle and Y.Descotte. *Une Architecture de Système Expert pour la Planification d'Activité.* In Procedings of the sixth International Workshop on Expert Systems and their Applications, Avignon, pages 903–916, April 1987.

[14] R.J. Firby. *An Investigation into Reactive Planning in Complex Domains.* In Proceedings of the sixth National Conference on Artificial Intelligence (AAAI-87), pages 202–206, 1987.

[15] B.R.Fox and K.G.Kempf. *Complexity, Uncertainty and Opportunistic Scheduling.* In Proceedings of the second International Conference on Artificial Intelligence, IEEE 1985.

[16] B.R.Fox and K.G.Kempf. *A Representation for Opportunistic Scheduling.* In Proceedings of the third International Symposium on Robotics Research, Gouvieux (France), pages 111–117, October 1985.

[17] E.C. Freuder. *Synthesizing Constraint Expressions.* Communications of the ACM (21)11, pages 958–966, Nov. 1978.

[18] M.Ghallab, R.Alami and R.Chatila. *Real-Time Reasoning and Acting in Robotics.* Draft Version. In Proceedings of the fourth International Symposium on Robotics Research, Santa Cruz (California), August 1987.

[19] I.Goldstein. *Bargaining Between Goals.* In Proceedings of the fourth IJCAI, pages 175–180, 1975.

[20] I.Goldstein. *Nudge, A Knowledge-Based Scheduling Program.* In Proceedings of the fifth IJCAI, pages 257–263, 1977.

[21] M. Georgeff and U. Bonollo. *Procedural Expert Systems.* In Proceedings of the eighth IJCAI, pages 151–157, 1983.

[22] M. Georgeff and A. Lansky . *Reactive Reasonning and Planning.* In Proceedings of the sixth National Conference on Artificial Intelligence (AAAI-87), pages 677–682, 1987.

[23] T.Granier. *Etude de la Cohérence des Réseaux d'Intervalles Temporels.* Rapport de Recherche IMAG A paraitre. 1988

[24] B.Hayes Roth et al. *Modeling Planning as an Incremental, Opportunistic Process.* In Proceedings of the sixth IJCAI, page 375, 1979.

[25] C.Le Pape. *OPIS1 Un Système d'Ordonnancement Opportuniste.* Rapport Final de bourse INRIA, Septembre 1986.

[26] C.Le Pape and S.Smith. *Management of Temporal Constraints for Factory Scheduling.* In Proceedings of the Temporal Aspects in Information Systems Conference, AFCET, Sophia Antipolis, France, pages 165–176, May 1987.

[27] A.K.Mackworth. *Consistency in Networks of Relations.* Artificial Intelligence (8), pages 99–118, 1977.

[28] D.Miller, R.J.Firby and T.Dean. *Deadlines, Travel Time and Robot Problem Solving.* In Proceedings of the ninth IJCAI, pages 1052–1054, 1985.

[29] U. Montanari. *Networks of Constraints: Fundamental Properties and Applications to Picture Processing.* Information Sciences (7), pages 95–132, 1974.

[30] S. Masui, J. Mc Dermott, and A. Sobel. *Decision-making in Time-critical Situations.* In Proceedings of the eighth IJCAI, pages 233–235, August 1983.

[31] D.McDermott. *A Temporal Logic for Reasoning about Processes and Plans.* Research Report #196, Department of Computer Science, Yale University, March 1981.

[32] P.A.Newman and K.G.Kempf. *Opportunistic Scheduling for Robotic Machine Tending.* In Proceedings of the second International Conference on Artificial Intelligence, IEEE 1985.

[33] N.Nilsson. *A Hierarchical Robot Planning and Execution System.* Technical Note 76, Stanford Research Institute, April 1973.

[34] N.Nilsson. *Basic Plan Generating Systems.* Chapter 7 of "Principles of Artificial Intelligence". Tioga Publishing Company, 1980.

[35] N.Nilsson. *Advanced Plan Generating Systems.* Chapter 8 of "Principles of Artificial Intelligence". Tioga Publishing Company, 1980.

[36] S.J. Rosenschein. *Formal Theories of Knowledge in AI and Robotics.* New Generation Computing, 3 (1985), pages 345–347.

[37] E.Sacerdoti. *Plan Generation and Execution for Robotics.* Technical note 209 SRI, April 1980.

[38] Y.Shoham. *Reified Temporal Logics : Semantical and Ontological Considerations.* In Proceedings of the 7th ECAI, Brighton UK, pages 390–397, 1986.

[39] S.Smith. *Exploiting Temporal Knowledge to Organize Constraints.* Technical Report CMU-RI-TR-83-12, Carnegie-Mellon University, The Robotics Institute, July 1983.

[40] M.Stefik. *Planning with Constraints.* Artificial Intelligence (16), pages 111–170, 1981.

[41] *Temporal Aspects in Information Systems.* Proceedings of the Conference. AFCET. Sophia Antipolis, France, May 1987.

[42] A.Tate. *Generating Project Networks.* In Proceedings of the fifth IJCAI, pages 888–893, 1977.

[43] A.Tate *A Review of Knowledge-based Planning Techniques.* Knowledge Engineers's Review 2(1), June 1985.

[44] W. Thorndike, D. McArthur and S. Cammarata. *AUTOPILOT : a Distributed Planner for Air Fleet Control.* In Proceedings of the seventh IJCAI, pages 171–177, August 1981.

[45] S. Vere. *Planning in time : Windows and duration for activities and Goals.* Technical Report, Jet Propulsion Laboratory, November 1981.

[46] S.Vere. *Temporal Scope of Assertions and Window Cutoff.* In Proceedings of the ninth IJCAI, pages 1055–1059, August 1985.

[47] Marc Vilain and Henry Kautz. *Constraint Propagation Algorithms for Temporal Reasoning.* In Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86), Philadelphia PA, pages 377–382, August 1986.

[48] R. B. Wesson. *Planning in the world of the air traffic controler.* In Proceedings of the fifth IJCAI, pages 473–479, 1977.

Thèse de Doctorat de l'INPG

Auteur : Jean-François Rit

Etablissement : Laboratoire d'Informatique Fondamentale et d'Intelligence Artificielle

Titre : Modélisation et Propagation de Contraintes Temporelles pour la Planification.

Résumé : Nous proposons une représentation du temps pour la planification, fondée sur la manipulations d'intervalles de la droite réelle : les occurrences. Nous abordons la définition et la résolution d'un **problème d'occurrences contraintes**. Celui-ci est posé en considérant un graphe dont les nœuds sont associés à des **domaines d'occurrences possibles** et les arcs à des relations temporelles symboliques, entre intervalles et disjonctives. Résoudre le problème, c'est éliminer, en réduisant les domaines, les occurrences rendues impossibles par les contraintes relationnelles. On peut le faire partiellement au moyen d'un algorithme de propagation de contraintes, dit de **dérivation locale**. On peut améliorer la résolution en exprimant le problème dans des formalismes connus. L'approche est illustrée par l'étude d'un problème d'affectation de systèmes d'armes à bord d'un navire.

Mots clés : Intelligence Artificielle, Temps, Planification, Allocation de Ressources, Propagation de Contraintes.