Constraint-based Approaches for Planning and Scheduling : Methods, Tools and Applications

C. Pralet

ONERA

THE FRENCH AEROSPACE LAB

THESE

présentée pour obtenir le titre de

HABILITATION A DIRIGER DES RECHERCHES

délivré par l'Université Paul Sabatier - Toulouse III

Spécialité : Informatique

par

Cédric Pralet

Approches par Contraintes pour la Planification et l'Ordonnancement : Méthodes, Outils et Applications

Constraint-based Approaches for Planning and Scheduling : Methods, Tools and Applications

---

Habilitation présentée le 9 juin 2015, devant le jury composé de :

Martin Cooper (examinateur) - Professeur Université Paul Sabatier

Yves Deville (rapporteur) - Professeur Université Catholique de Louvain

Malik Ghallab (directeur de HDR) - Directeur de Recherche LAAS-CNRS

Pierre Lopez (examinateur) - Directeur de Recherche LAAS-CNRS

Pierre Marquis (rapporteur) - Professeur Université d'Artois

Abdel-Illah Mouaddib (rapporteur) - Professeur Université de Caen

Table des matières

Remerciements

Avant de présenter les diverses contributions réalisées, je tiens en premier lieu à adresser mes remerciements à plusieurs personnes liées de près ou de loin aux activités de recherche décrites dans ce document.

Pour commencer par le commencement, je tiens tout d'abord à remercier les deux personnes qui m'ont lancé sur les voies parfois sinueuses de la recherche, à savoir Thomas Schiex et Gérard Verfaillie, les deux encadrants de ma thèse réalisée de 2003 à 2006. Je reparlerai de Gérard plus loin, mais je tiens à remercier à nouveau Thomas chaleureusement pour ses qualités scientifiques et humaines, et pour l'exemple qu'il peut donner à tout encadrant en devenir.

Je tiens ensuite à remercier les différents collègues, et pour certains amis, côtoyés au sein de l'Onera, plus précisément au sein du département DCSD et de l'unité de recherche Conduite et Décision à Toulouse. Je préfère garder ici une dénomination ensembliste plutôt que d'oublier malencontreusement des noms... mais un élément associé est que le fait de poursuivre des travaux de recherche dans un environnement accueillant et agréable est une chose d'importance.

J'adresse également mes remerciements aux doctorants passés et en cours qui m'ont permis de me frotter aux tâches de direction de recherche. Ce "frottement" a d'abord commencé avec la thèse d'Alexandre Niveau concernant la compilation de connaissances, il s'est poursuivi avec la thèse d'Adrien Maillard traitant de planification pour des satellites, et il s'est enfin matérialisé avec la thèse de Guillaume Casanova concernant l'ordonnancement dans un contexte multi-agents. Merci à tous ces doctorants pour leur écoute et leur envie dans la réalisation de leur thèse. Je remercie aussi vivement les co-encadrants de ces thèses, à savoir respectivement Hélène Fargier, Gérard Verfaillie (dont je reparlerai plus loin) et Charles Lesire. L'encadrement de thèse à plusieurs a été de mon point de vue une expérience réellement enrichissante, la contrepartie pour les doctorants étant de savoir entendre plusieurs influences simultanément.

Pour le passage de l'habilitation à diriger des recherches, je remercie bien évidemment tous les chercheurs qui m'ont fait l'honneur de répondre positivement à mes sollicitations, avec tout d'abord les rapporteurs Yves Deville de l'université catholique de Louvain, Pierre Marquis de l'université d'Artois et Abdel-Illah Mouaddib de l'université de Caen, et avec ensuite les membres du jury Martin Cooper de l'IRIT, et Malik Ghallab et Pierre Lopez du LAAS-CNRS. Merci à toutes ces personnes pour leur lecture attentive de ce manuscrit et pour l'accueil réservé aux travaux présentés le jour de la soutenance. Merci notamment à Malik pour avoir tout de suite accepté de jouer le rôle de directeur de HDR et pour m'avoir guidé dans toute la préparation.

Maintenant est venu le moment de remercier Gérard, compagnon de route au cours de toutes ces années passées à l'Onera, à la fois sur des activités contractuelles et sur des activités de recherche plus "libres". Je lui serai toujours reconnaissant pour sa grande bienveillance à mon égard. Je ne peux que dire que travailler avec Gérard fut pour moi un réel plaisir, que j'aurais bien sûr aimé prolonger si les joies de la retraite ne l'avaient pas attiré !

Je tiens également à remercier toute ma famille, dans laquelle j'inclus également ma belle-famille. Il m'est difficile de formaliser des raisons (malgré mes compétences en formalisation...), mais je dirais tout simplement qu'elle m'apporte toujours de la joie lors de nos rencontres, pas si rares vu nos situations géographiques et le quadrillage de la France que nous avons réussi à réaliser !

Enfin, je réserve mes derniers remerciements aux trois êtres qui m'entourent au quotidien, Elyssa mon amour et mes deux petites filles Mathilde et Julia, qui deviennent de jour en jour de moins en moins petites, et qui n'ont très certainement aucune idée de tout ce qu'elles peuvent m'apporter.

Chapitre 1

Introduction

1.1 Contexte de la recherche

Les recherches que je mène s'intéressent au *contrôle de systèmes* par des techniques de *décision automatique*. Ces recherches sont appliquées principalement au monde aéronautique et spatial, notamment à la conduite de satellites, de drones, ou de sous-systèmes embarqués sur un aéronef. Partant d'un modèle formel de la dynamique du système considéré et de l'environnement dans lequel il s'insère, l'objectif est de produire des stratégies permettant d'optimiser le comportement du système tout en satisfaisant un certain nombre de contraintes, par exemple des contraintes physiques comme des limitations sur les ressources disponibles (temps, énergie...) ou des contraintes traduisant des exigences utilisateurs. Les stratégies recherchées peuvent être :

- ou bien des stratégies dites *réactives*, prenant la forme de plans d'actions construits en faisant des hypothèses d'évolution déterministe du système et de l'environnement, ces plans étant reconstruits/adaptés en cas de déviation entre ces hypothèses et la réalité ;
- ou bien des stratégies dites *proactives*, prenant la forme de politiques de décision qui spécifient les actions à réaliser pour plusieurs scénarios envisageables.

Les décisions visées, souvent discrètes, correspondent à un contrôle dit de haut niveau, par opposition au contrôle bas niveau relatif au domaine de l'automatique continue. En sortie, l'ambition est d'obtenir, sur la base de modèles formels, des systèmes qui soient à la fois sûrs et performants.

Un point particulier de mes travaux de recherche est qu'ils s'inscrivent dans une démarche combinant développements méthodologiques et traitements d'applications du monde industriel. Ces applications sont abordées au travers d'activités contractuelles avec différents partenaires (CNES, ASTRIUM, TAS, AIRBUS, ESA). Elles génèrent également des collaborations avec des partenaires académiques (LAAS-CNRS, IRIT, ISTC-CNR, Politechnico di Milano). Les applications rencontrées nourrissent la réflexion en exhibant des contraintes systèmes et des contextes opérationnels variés : par exemple, suivant la mission considérée, la décision peut devoir être prise hors ligne (avant la mise en situation du système) ou en ligne (au cours de l'exécution), elle peut devoir être réalisée avec des ressources de calcul plus ou moins limitées, elle peut devoir être réalisée sur un calculateur

embarqué, elle peut devoir être réalisée en temps plus ou moins contraint, elle peut devoir être réalisée dans un contexte mono-engin ou multi-engins, elle peut nécessiter des raisonnements sur des horizons temporels en avant plus ou moins longs, elle peut nécessiter de prendre en compte les déplacements des engins, elle peut devoir être faite sur la base de modèles hybrides mêlant état discret et état continu... A un niveau plus fondamental, un objectif est d'identifier les caractéristiques communes aux problèmes rencontrés pour définir des cadres de représentation appropriés, des algorithmes de décision optimisés, et des outils génériques. Le traitement d'applications variées conduit par contre parfois à une certaine disparité dans les sujets traités et dans les contributions techniques.

1.2 Domaines scientifiques

Mes activités se situent à l'intersection de plusieurs domaines scientifiques. Premièrement, les travaux effectués possèdent des liens forts avec le domaine de l'*optimisation combinatoire*. Ce domaine a pour objet l'étude de techniques permettant de fouiller un ensemble de décisions possibles de manière efficace, en repoussant au maximum les limites de l'explosion combinatoire. Plus spécifiquement, le cadre classique sur lequel reposent mes travaux est celui de la *programmation par contraintes* [234], un paradigme de programmation déclarative dans lequel le modélisateur déclare des variables de décision pouvant prendre certaines valeurs, des contraintes portant sur ces variables, et éventuellement un critère à optimiser. Des algorithmes génériques sont ensuite disponibles pour trouver automatiquement une solution au problème posé, en utilisant notamment des mécanismes dits de *propagation de contraintes* capables de raisonner sur le modèle pour élaguer vite et bien l'espace de recherche d'une solution. D'autres cadres sont aussi utilisés plus ponctuellement dans mes travaux, tels que la théorie des graphes, la programmation linéaire et linéaire en nombres entiers, la satisfiabilité de formules logiques et la recherche opérationnelle. Le lien avec la communauté optimisation combinatoire se traduit par des publications dans des journaux et dans des conférences françaises et internationales de ce domaine : RAIRO - Operations Research, CP (International Conference on Principles and Practice of Constraint Programming), CPAIOR (International Conference on Constraint Programming, Artificial Intelligence, and Operations Research) et JFPC (Journées Francophones de Programmation par Contraintes).

Le deuxième domaine lié à mes recherches est celui de la *planification d'actions* et de l'*ordonnancement de tâches*. Pour la planification d'actions, le but est, étant donnée une description formelle des actions réalisables par un système et de leurs effets, de produire un plan, c'est-à-dire un ensemble d'actions à appliquer en séquence ou en parallèle, permettant d'atteindre un objectif donné. Les stratégies de décision produites peuvent également prendre la forme d'éléments appelés des politiques, qui associent à tout état atteignable par le système une décision à prendre dans cet état. Pour l'ordonnancement, le but est d'agencer un ensemble de tâches connues initialement tout en prenant en compte des contraintes sur le temps et les ressources consommées par les différentes tâches. Les travaux que j'effectue dans ce domaine se traduisent par des publications dans les conférences ICAPS (International Conference on Automated Planning and Scheduling) et JFPDA (Journées Francophones sur la Planification, la Décision et l'Apprentissage pour la conduite de systèmes).

Le troisième domaine lié à mes recherches est celui de l'*intelligence artificielle*, au travers de la problématique générale de la décision séquentielle dans l'incertain. Cette problématique consiste à définir et étudier des cadres de représentation de problèmes dans lesquels des décisions doivent être prises pour mener un système dans un certain ensemble d'états, tout en prenant en compte des incertitudes sur les effets possibles des actions ou sur les événements pouvant se produire dans l'environnement du système. Les travaux effectués dans ce domaine, notamment au cours de ma thèse de 2003 à 2006, se sont concrétisés par des publications dans les journaux JAIR (Journal of Artificial Intelligence Research), RIA (Revue d'Intelligence Artificielle) et TSI (Technique et Science Informatiques), et dans les conférences ECAI (European Conference on Artificial Intelligence) et UAI (International Conference on Uncertainty in Artificial Intelligence).

Enfin, comme déjà indiqué précédemment, mes activités de recherche sont appliquées au domaine *aéronautique et spatial*. Le but est d'apporter des solutions techniques à des problèmes réels, basées ou bien sur des travaux existants de la littérature, ou bien sur des travaux issus des recherches effectuées par des chercheurs de mon équipe ou par moi-même. Dans cette optique, les développements effectués nécessitent des connaissances applicatives sur la manière dont sont conçus les systèmes, la manière dont ils sont utilisés opérationnellement, et parfois sur la physique régissant le comportement de tel ou tel aspect. Cette activité se traduit par diverses publications dans des manifestations scientifiques applicatives dédiées, à savoir les manifestations IWPSS (International Workshop on Planning and Scheduling for Space), iSAIRAS (International Symposium on Artificial Intelligence, Robotics, and Automation in Space) et, plus ponctuellement, ERTS (European Congress on Embedded Real Time Software).

1.3 Organisation du rapport

Ce rapport retrace les principaux travaux de recherche effectués. Il laisse de côté les détails techniques pour se concentrer sur le cheminement global. Dans le chapitre 2, nous commençons par décrire des applications pour lesquelles nous avons dû construire des planificateurs. Dans le chapitre 3, nous décrivons les premiers travaux effectués pour appliquer la programmation par contraintes à la gestion de systèmes dynamiques à événements discrets, systèmes dont l'état évolue en fonction de l'arrivée d'événements instantanés au cours du temps. Dans le chapitre 4, nous décrivons les travaux réalisés pour traiter spécifiquement des problèmes dans lesquels la question centrale est d'ordonnancer un ensemble de tâches prédéfinies avec des contraintes sur le temps et sur les ressources, et non de faire évoluer l'état d'un système par des actions. Dans le chapitre 5, nous décrivons des techniques à base de contraintes développées dans un contexte de décision plus proactive, dans lequel les différentes situations rencontrables à l'exécution sont anticipées hors ligne et compilées sous la forme d'une politique de décision. Dans le chapitre 6, nous décrivons les travaux effectués pendant la période de thèse, portant sur la définition d'un cadre et d'algorithmes génériques capables de couvrir plusieurs formalismes de décision séquentielle dans l'incertain. Le chapitre 7 donne enfin des perspectives de recherche. Pour faciliter la lecture du document, les références propres sont mentionnées avec une étoile dans la marge par exemple comme suit : [222]. *

Chapitre 2

Problèmes de planification du domaine aéronautique et spatial

Dans ce chapitre, nous décrivons des problèmes de planification et d'ordonnancement que nous avons traités au cours des années précédentes. Nous considérons principalement des problèmes issus du domaine spatial dans lesquels le but est de contrôler les activités de satellites d'observation et de surveillance. Nous donnons une vue globale de ce domaine applicatif et nous montrons comment la nature des problèmes abordés a motivé nos développements de nouvelles techniques. Ces dernières sont présentées dans des chapitres ultérieurs dédiés aux méthodes.

2.1 Mission des satellites d'observation et de surveillance

2.1.1 Description générale

La mission des satellites d'observation et de surveillance consiste à collecter des données concernant certaines cibles, en utilisant un ou plusieurs instrument(s) embarqué(s), et à transmettre ces données vers des stations de réception sol. Les cibles concernées peuvent être par exemple des zones terrestres devant être balayées par un instrument optique ou radar. Elles peuvent également correspondre à des points au sol où un phénomène particulier a été détecté. Dans le cas des satellites d'observation de l'univers, les cibles peuvent être des zones de l'espace (étoiles, exo-planètes...), ou encore des événements électromagnétiques tels que des sursauts gamma.

Dans ce qui suit, nous nous concentrons sur des satellites d'observation et de surveillance en orbite autour de la Terre, sur des orbites circulaires ou elliptiques, et ne restant pas toujours à la verticale du même point (satellites dits *défilants*). Voir la figure 2.1 pour une illustration. Ces satellites doivent exécuter plusieurs types d'activités :

1. des activités d'*acquisition*, au cours desquelles les données collectées par les instruments embarqués sont enregistrées à bord du satellite ;

2. des activités de *vidage*, au cours desquelles les données enregistrées à bord du satellite sont vidées vers des stations de réception sol ; ces stations sont distribuées à la surface de la Terre ; leur nombre varie suivant les missions, et il existe souvent de longues

plages temporelles au cours desquelles le satellite ne voit aucune station, et donc au cours desquelles il ne peut pas vider les données enregistrées à bord ;

3. des activités de *manœuvres*, utilisées pour modifier la direction de pointage du satellite ; ces manœuvres sont par exemple utilisées pour permettre au satellite de passer d'un pointage vers une zone à acquérir à un pointage vers la zone suivante ;

4. des activités de *contrôle d'orbite*, servant à maintenir le satellite sur son orbite de référence ;

5. des activités de *pointage spécifique*, par exemple de pointage géocentrique (pointage vers le centre de la Terre pour mettre le satellite dans une position de sécurité), ou de pointage héliocentrique (pointage dans une direction opposée au soleil pour optimiser la recharge des batteries via les panneaux solaires).

En plus de cela, les satellites envoient automatiquement des informations de base permettant au sol de suivre leur comportement. Ces informations sont envoyées vers des stations de contrôle. Ces dernières sont aussi utilisées par le sol pour envoyer des informations vers le satellite, par exemple des plans à exécuter à bord. Ces plans sont habituellement produits par une entité appelée le centre de mission. Un centre appelé le centre d'opération est également présent pour assurer le bon fonctionnement de la plate-forme.

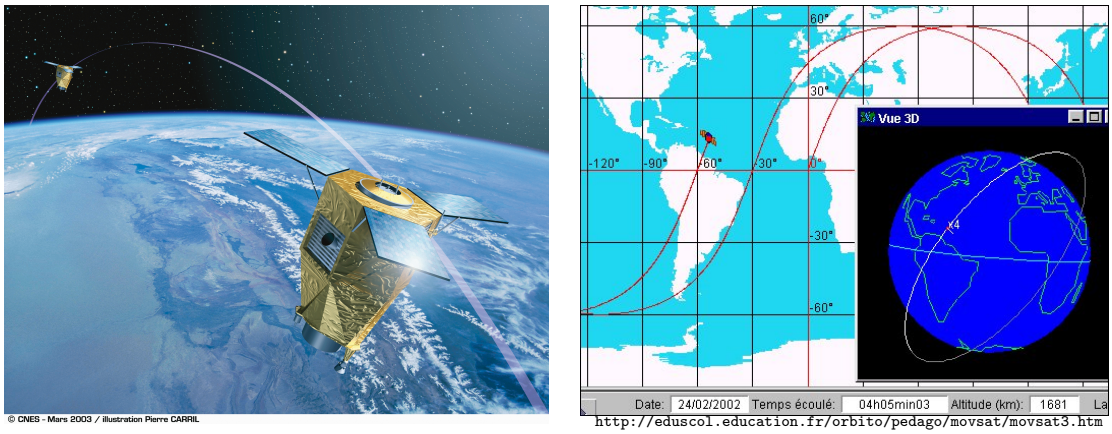


FIGURE 2.1 : A gauche : satellite d'observation Pléiades ; à droite : trajectoire d'un satellite défilant

2.1.2 Rôle de la planification

Les activités d'un satellite d'observation ne sont pas toutes le résultat d'un processus décisionnel réalisé par un planificateur. Par exemple, les activités de contrôle d'orbite permettant de maintenir le satellite sur son orbite de référence sont assurées par le SCAO (Système de Commande d'Attitude et d'Orbite), un système embarqué spécifique dans lequel la planification n'intervient pas. En pratique, dans la majorité des problèmes de planification d'activités de satellites que nous avons rencontrés, les deux tâches principales étaient de décider du plan d'acquisition et de décider du plan de vidage de données.

Ces deux types de plans sont illustrés à la figure 2.2. La construction de ces plans est le fruit de plusieurs décisions : une *sélection* des tâches à réaliser parmi les tâches candidates, une *allocation* des tâches aux ressources disponibles, un *ordonnancement* des tâches sélectionnées, et un *placement temporel* de ces tâches. Pour beaucoup de missions, la planification des acquisitions est traitée en amont de la planification des vidages. En cas d'impasse (pas suffisamment de ressources pour vider les acquisitions programmées), un retour sur le plan d'acquisition est effectué. Pour d'autres missions, les activités d'acquisitions et de vidages sont planifiées simultanément, la motivation étant que ces activités sont couplées du fait des ressources présentes à bord du satellite en quantité limitée.

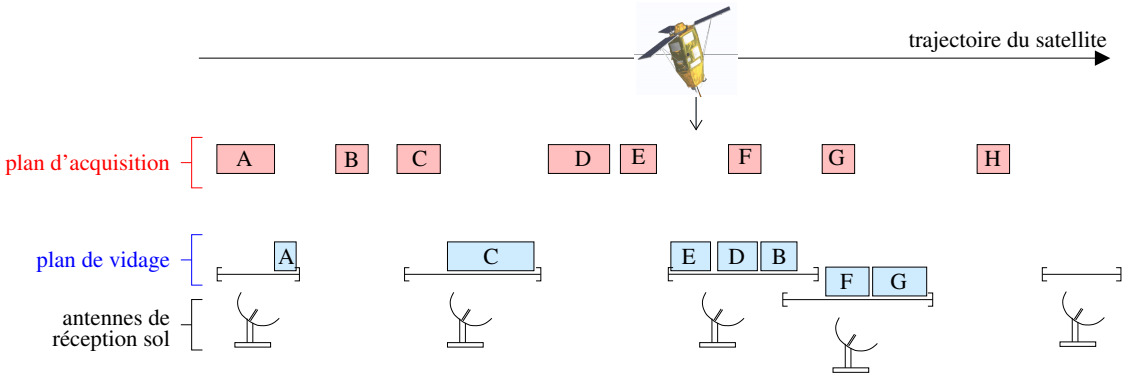


FIGURE 2.2 : Activités d'acquisition et de vidage d'un satellite d'observation ; les vidages sont réalisés au cours de fenêtres de visibilité d'antennes sol ; acquisition et vidage peuvent éventuellement avoir lieu en parallèle

Enfin, la planification des activités peut concerner soit un seul satellite soit plusieurs. Un exemple de mission impliquant une constellation de satellites pour la surveillance des mers du globe est fournie à la figure 2.3. Dans certains cas, les activités des satellites doivent être coordonnées, comme dans les missions TerraSAR-X et TanDEM-X impliquant deux satellites en formation étroite à quelques centaines de mètres l'un de l'autre [143].

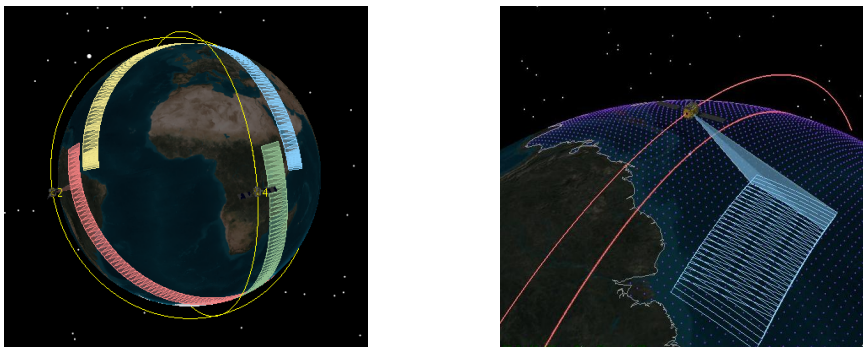


FIGURE 2.3 : Balayage réalisé par les instruments radar d'une constellation de 4 satellites sur un horizon de 45 minutes (partie gauche), et maillage et balayage des océans réalisé par l'instrument radar d'un satellite sur une courte période (partie droite)

2.1.3 Diversité des missions

D'un point de vue global, les missions des satellites d'observation et de surveillance se ressemblent dans le sens où les types d'activités à réaliser sont grosso modo toujours les mêmes, à savoir collecter des données et vider ces données vers des stations sol. Cependant, chaque mission possède des caractéristiques qui lui sont propres. Plusieurs missions peuvent être considérées à titre illustratif, comme la mission SPOT nécessitant une planification sol pour l'observation de la Terre dans le domaine optique, la mission ENVISAT nécessitant une planification sol pour des acquisitions dans le domaine radar, ou encore la mission EO-1 (Earth-Observing One [46, 47]), un précurseur dans la mise en œuvre d'une planification à bord pour un satellite d'observation. Dans le reste de ce chapitre, nous considérons principalement six missions spatiales pour lesquelles nous avons effectivement développé des planificateurs, et nous montrons comment le contact avec ces missions a guidé nos travaux sur les méthodes et les outils. Les six missions en question sont les missions **HotSpot**, **Integral**, **Samson**, **Ecoute**, **AgataOne** et **Pflex**, décrites à la figure 2.4. Des descriptions précises de ces missions et des planificateurs que nous avons développés pour les traiter sont disponibles dans les références indiquées dans le tableau.

	Mission	Objectif	Partenaires
*	**HotSpot** [205, 248]	Mission de surveillance de points chauds à la surface du globe par un satellite autonome : planification bord des acquisitions et des vidages de données vers le sol.	CNES
*	**Integral** [207, 244]	Observation de l'univers par un satellite : planification des activités d'acquisition du satellite INTEGRAL sur un horizon d'une année.	ESA, VEGA, ISTC-CNR, Politechnico Di Milano
*	**Samson** [215, 179, 216]	Mission de surveillance des mers du globe par une constellation de satellites ; planification sol des activités d'acquisition et planification sol des réservations de fenêtres de communication avec les stations de réception.	Thales Alenia Space, CNES
*	**Ecoute** [212]	Mission de surveillance électromagnétique : planification bord des vidages (pour un plan d'acquisition donné), avec gestion des incertitudes sur le volume des données générées.	CNES
*	**AgataOne** [197, 196]	Mission de surveillance électromagnétique utilisant deux satellites proches, l'un faisant des détections et l'autre des acquisitions : planification bord des acquisitions et des vidages.	CNES
*	**Pflex** [224, 149, 150]	Mission d'observation avec un satellite Post-Pléiades, utilisant une architecture physique complexe : planification sol ou bord des vidages, avec incertitudes sur le volume des données générées.	CNES, ASTRIUM, LAAS-CNRS

FIGURE 2.4 : Exemples de missions traitées

2.2 Ingrédients des problèmes de planification

Nous donnons ci-après une vue globale des différents aspects rencontrés dans les problèmes de planification que nous avons traités, avec les contraintes à considérer, les critères, et le contexte de la décision. Nous explicitons également les liens avec des modèles de représentation classiques en planification et en ordonnancement.

2.2.1 Contraintes temporelles

Le premier type de contraintes présent dans les problèmes de planification d'activités de satellites est un ensemble de contraintes temporelles à satisfaire.

Contraintes temporelles sur les acquisitions Les contraintes temporelles les plus élémentaires, que nous avons rencontrées sur toutes les missions traitées, portent sur chaque acquisition prise individuellement. Elles se traduisent, pour chaque acquisition candidate, par une durée d'acquisition et par un ensemble de fenêtres temporelles possibles pour réaliser l'acquisition.

Le second jeu de contraintes temporelles à considérer correspond à des contraintes de précédence spécifiant que la durée entre deux acquisitions successives doit être suffisante pour pouvoir effectivement passer de la première à la seconde. Pour les premiers satellites d'observation de la Terre tels que SPOT, qui avaient la capacité à réaliser des acquisitions sur la gauche ou sur la droite de la trace au sol du satellite grâce à un miroir de visée orientable, la durée minimale requise entre deux acquisitions données était une constante fonction de la distance angulaire entre acquisitions selon l'axe du roulis du satellite. Dans ce cas, la contrainte temporelle obtenue est une contrainte assez classique de non chevauchement avec temps de mise au point (*setup times*) entre tâches [190]. Nous avons utilisé une telle modélisation dans notre traitement de la mission HotSpot [205]. *

Ensuite sont apparus des satellites tels que SPOT6, SPOT7 ou Pléiades, capables de se mouvoir autour de leur centre de gravité grâce à des roues à réaction ou à des actionneurs gyroscopiques. Ces satellites, qui sont dits *agiles*, peuvent pointer non seulement sur la gauche ou sur la droite, mais aussi vers l'avant ou vers l'arrière, ce qui a permis d'offrir de la latitude sur la date de réalisation des acquisitions. L'apparition de l'agilité a par contre complexifié les contraintes temporelles à satisfaire concernant l'enchaînement des acquisitions. Plus précisément, pour un satellite agile, la durée minimale requise pour passer d'un pointage vers une zone à un pointage vers une autre zone devient dépendante de la date à laquelle la transition est enclenchée. On parle dans ce cas d'une durée de transition *time-dependent* [44, 84], dans le sens où la durée de la transition dépend de la date à laquelle la transition est enclenchée. Ce phénomène est notamment dû au fait que le satellite bouge sur son orbite. Voir la figure 2.5 pour une illustration.

Dans certains travaux, des simplifications sont faites concernant l'agilité [256] : les durées minimales à respecter entre deux acquisitions données sont approximées par des durées fixes. Nous avons aussi utilisé une telle approximation pour la mission Integral [207], * pour laquelle les durées de manœuvre du satellite n'étaient pas significatives par rapport à la durée des autres tâches du plan. Pour d'autres missions, nous avons modélisé l'agilité de manière plus fine [197, 224], la motivation étant que si les temps de transition entre *

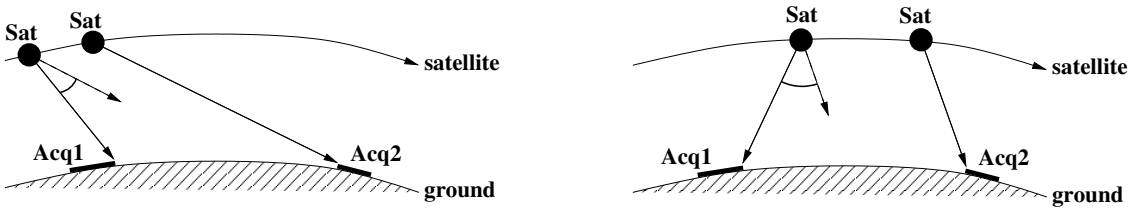


FIGURE 2.5 : Illustration de comment le mouvement en attitude réalisé par un satellite agile pour transiter d'une acquisition à une autre dépend de la date à laquelle la transition est réalisée ; dans le second cas (partie droite), le mouvement angulaire à réaliser est plus important que dans le premier cas (partie gauche)

acquisitions sont estimés par des durées fixes, alors les plans produits peuvent être inapplicables en cas de sous-estimation des durées de transition réelles, et sous-optimaux en cas de surestimation de ces durées. Cela nous a amené à mêler au sein de la planification des raisonnements sur des aspects discrets (ordonnancement des acquisitions) et des considérations sur des quantités continues (direction de pointage et vitesses de rotation du satellite). Ces quantités continues ont été traitées en faisant appel, au sein du processus de planification, à des bibliothèques dédiées de calculs pour satellites agiles prenant en compte le mouvement du satellite sur son orbite, la rotation de la Terre sur elle-même, et des contraintes cinématiques sur les vitesses angulaires maximales du satellite. Comme les contraintes temporelles *time-dependent* étaient ressenties comme des contraintes pouvant apparaître dans d'autres domaines, nous avons de plus réalisé un travail méthodologique particulier sur la définition d'un cadre générique pour traiter ces contraintes, le cadre des
* *Time-dependent STN* [213] décrit au chapitre 4.

Contraintes temporelles sur les vidages Côté vidage, on retrouve tout d'abord, de manière similaire aux acquisitions, des fenêtres temporelles autorisées pour réaliser des envois de données. Ces fenêtres correspondent aux plages temporelles au cours desquelles le satellite est dans le cône de visibilité d'une station de réception sol et où la station de réception est disponible pour recevoir des données en provenance du satellite.

On retrouve ensuite des contraintes temporelles sur l'enchaînement des tâches de vi-
* dage [255]. Pour les premières missions que nous avons étudiées [212], ces contraintes correspondaient à de simples contraintes de non chevauchement entre vidages successifs, représentant le fait que l'instrument d'émission de données ne pouvait pas vider plusieurs acquisitions simultanément. Dans les missions considérées aujourd'hui, la complexification envisagée pour les architectures physiques des satellites induit une complexification des contraintes temporelles associées au vidage. Premièrement, il peut exister plusieurs ressources utilisables en parallèle pour réaliser les vidages, complexification que nous avons
* abordée lors de nos travaux sur la mission Pflex [224]. L'architecture bord utilisée pour cette mission est décrite à la figure 2.6. En termes d'ordonnancement, cette architecture physique nous a conduit à un problème de type *Flexible Open Shop Scheduling* [190], impliquant deux types de ressources non partageables : des canaux d'émission servant à vider les données et des banques mémoires servant à stocker les données, qui ne pouvaient être lues à tout instant que par un seul canal d'émission. Deuxièmement, il peut exister des

effets *time-dependent* également pour le vidage. Nous avons abordé cet aspect à nouveau au travers de la mission Pflex [224]. Dans cette mission, la durée d'un vidage dépendait de la date à laquelle le vidage était enclenché, à cause de débits de vidage fonctions de la distance satellite-station, distance variable au cours du temps du fait du mouvement du satellite sur son orbite. De plus, la mission faisait intervenir une antenne d'émission orientable (non fixe sur le satellite), et la gestion de cette antenne mobile à bord d'un satellite lui-même agile induisait également des contraintes *time-dependent* pour gérer les durées minimales requises par l'antenne pour passer d'un pointage vers une station à un pointage vers une autre station. Au final, les problèmes d'ordonnancement obtenus s'écartaient des problèmes d'ordonnancement classiques, et les plans de vidage obtenus étaient plus complexes que celui illustré à la figure 2.2. Pour traiter ces problèmes, nous avons pu réutiliser nos travaux génériques sur les *Time-dependent STN* décrits au chapitre 4. *

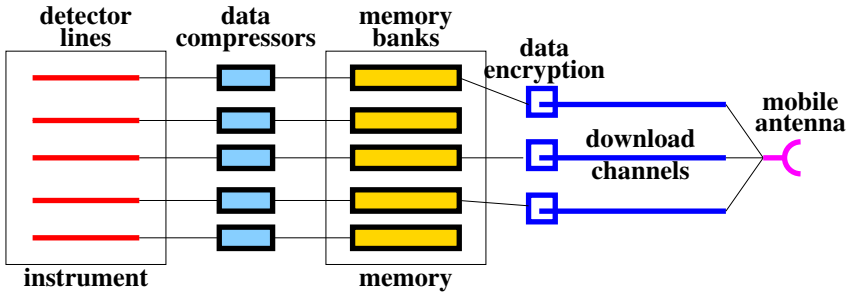


FIGURE 2.6 : Architecture bord de la mission Pflex [224] : de gauche à droite, l'architecture se décompose en un ensemble de barrettes de capteurs constituant l'instrument d'observation, un ensemble de compresseurs mémoire utilisés pour compresser les données, un ensemble de banques mémoire utilisées pour enregistrer les données à bord, un ensemble de canaux d'émission, et une antenne d'émission orientable vers les stations de réception. *

Contraintes temporelles liant acquisition et vidage Enfin, des contraintes temporelles lient acquisition et vidage. On retrouve tout d'abord dans les problèmes toujours la contrainte basique selon laquelle un vidage ne peut avoir lieu avant l'acquisition correspondante. Dans le cas des satellite agiles, on trouve ensuite des contraintes plus complexes dues au fait qu'un dépointage fort du satellite requis pour réaliser une acquisition n'est pas toujours compatible avec un vidage en parallèle vers une station donnée, même si le satellite est dans le cône de visibilité de la station. Dans la mission AgataOne [197], nous avons traité ce point en calculant d'abord un plan d'acquisition, puis en restreignant les fenêtres temporelles utilisables pour le vidage. *

2.2.2 Gestion des ressources

Le deuxième élément présent dans les problèmes de planification associés aux satellites est un ensemble de ressources requises pour réaliser effectivement les tâches, comme la mémoire et l'énergie disponibles à bord, la température des instruments, le nombre de on/off réalisés sur les instruments, ou la durée maximale d'acquisition par orbite. En termes d'ordonnancement, nous avons rencontré des ressources de différents types.

Ressources consommables Le premier type de ressource rencontré correspond aux ressources classiquement appelées des ressources *consommables*. La consommation globale de ces ressources est limitée au cours du temps. Nous avons par exemple eu à considérer ce
* genre de ressource dans la mission Samson [215, 179], pour laquelle le nombre journalier de on/off réalisés sur un instrument d'acquisition radar était limité, pour des raisons
* de fiabilité à long terme de l'instrument. Dans une mission telle que Integral [207], nous avons eu à considérer une ressource consommable représentant la durée cumulée maximale d'acquisition sur chaque révolution du satellite ; cette durée ne devait pas dépasser une valeur donnée de manière à garder du temps d'observation en cas d'arrivée d'un événement inattendu nécessitant une acquisition immédiate. Pour traiter ces ressources, il nous a suffi d'introduire dans les modèles des compteurs de consommation de ressources.

Ressources renouvelables Le deuxième type de ressource rencontré correspond aux ressources classiquement appelées des ressources *renouvelables*, consommées par une ou plusieurs tâches à un instant donné et redevenant disponibles en quantité égale une fois cette ou ces tâches terminée(s). Un exemple de ressource renouvelable que nous avons rencontrée dans le domaine spatial est la ressource mémoire, utilisée par chaque acquisition a réalisée pendant toute la période de stockage de a à bord du satellite. D'un point de vue ordonnancement, la ressource mémoire est une ressource *cumulative*, c'est-à-dire utilisable par plusieurs tâches en parallèle dans la limite du niveau de mémoire maximal disponible. Combiner la prise en compte de la ressource mémoire avec la prise en compte de contraintes
* temporelles élémentaires, comme nous l'avons fait dans la mission HotSpot [205], conduit à un problème formulable comme un RCPSP standard (*Resource Constrained Project Scheduling Problem*). Nous avons également rencontré dans le domaine spatial des ressources renouvelables dites *disjonctives*, c'est-à-dire ne pouvant pas être utilisées par deux tâches simultanément, par exemple un canal d'émission de données.

Ressources de type "énergie" Nous avons enfin eu à traiter des ressources moins classiques, en particulier pour gérer l'énergie disponible à bord d'un satellite. Basiquement, la ressource énergie est (1) consommée en permanence par l'ensemble de la plate-forme et par les instruments allumés ; (2) produite via les panneaux solaires lorsque le satellite est en situation d'éclairement (cas où les rayons du soleil atteignent effectivement les panneaux solaires). Dans certaines missions, la ressource énergie est traitée avec une approche conservative consistant à imposer un temps maximal d'acquisition et/ou de vidage par orbite. On retombe alors dans le cas d'une ressource consommable. Nous avons
* employé cette approche dans les missions Pflex [224] et Samson [215, 179]. D'autres missions requièrent de descendre à un niveau de granularité plus fin, et de prendre en compte la séquence des événements datés qui impactent la ressource. Dans la mission HotSpot
* notamment [205], où nous devions définir un planificateur pour satellite autonome, nous avions à considérer une ressource énergie évoluant de manière linéaire par morceaux, avec des pentes d'évolutions fonctions de la puissance courante consommée à bord du satellite. La ressource énergie ne devait pas descendre en dessous d'un certain niveau minimal $Emin$, mais surtout elle ne pouvait pas dépasser un niveau d'énergie maximal $Emax$ correspondant à un niveau de charge maximale des batteries. Ces hypothèses nous ont conduit à manipuler des profils d'évolution de l'énergie linéaires par morceaux avec une

saturation au niveau d'énergie $Emax$. Notons enfin que la ressource énergie est traitée de manière encore plus fine dans d'autres travaux, qui prennent en compte l'orientation des panneaux solaires à tout instant et qui réalisent une intégration continue pour estimer le profil d'énergie [101]. Cette finesse dans la représentation s'accompagne par contre d'un coût significatif en temps de calcul.

La ressource "température des instruments" est traitée de manière similaire à la ressource énergie, en remplaçant les puissances consommées et produites par des vitesses d'échauffement et de refroidissement, fonctions des activités en cours et de la situation d'éclairement du satellite. Nous avons été amenés à modéliser la température des instruments pour la mission Samson [215, 179]. La modélisation a conduit à des profils de * température linéaires par morceaux, avec une saturation à un niveau de température minimal imposé par un système de régulation de température actif à bord du satellite. Des modélisations plus complexes de l'évolution de la température sont présentées dans [101].

2.2.3 Gestion de l'état

Pour nombre de missions, il n'est pas nécessaire de suivre l'état du satellite à chaque instant pour déterminer si un plan est faisable ou non. Par exemple, pour assurer qu'une donnée n'est pas vidée avant son acquisition, une simple contrainte temporelle de précédence entre acquisition et vidage suffit, ou autrement dit il est inutile de maintenir explicitement un état donnant à chaque étape l'ensemble des acquisitions enregistrées à bord du satellite. Dans certains cas cependant, il est nécessaire de descendre à plus bas niveau pour suivre des évolutions complexes de l'état. Nous avons rencontré un tel cas dans la gestion de la mission Pflex [224]. Sur cette mission, on considérait que des clés de cryptage étaient utilisées * pour crypter les données lors du vidage. Des changements de clé de cryptage étaient requis entre deux vidages successifs d'acquisitions associées à des utilisateurs différents. Ces changements de clé étaient stockés dans une table de taille limitée, et lorsque le nombre de changements de clé réalisés atteignait la taille limite, il était nécessaire d'effectuer une opération de réinitialisation de la table pour pouvoir continuer à vider des acquisitions. Cette opération de réinitialisation prenait une certaine durée constante, pendant laquelle aucun vidage n'était possible. Nous avions ainsi à manipuler un état courant donné par le nombre de changements de clé depuis la dernière réinitialisation de la table, et la durée de transition entre deux vidages successifs dépendait de l'état de la ressource (durée de transition *state-dependent*). De telles évolutions ont pu être modélisées avec les techniques détaillées au chapitre 4, qui permettent de représenter des profils d'état définis par des équations d'évolution arbitraires. Notons que la distinction entre état et ressource est parfois assez ténue, puisque l'on peut par exemple considérer que la gestion des ressources mémoire et énergie correspond à un suivi de l'état de ces ressources.

2.2.4 Exigences utilisateurs et contraintes opérationnelles

Les contraintes présentées précédemment sont essentiellement des contraintes physiques décrivant ce qui est possible ou impossible pour le satellite. En plus de cela, il peut exister des exigences de la part des utilisateurs du système, qui ne se traduisent ni en contraintes sur le temps ni en contraintes sur les ressources. Par exemple, dans la mission Samson que nous avons considérée dans [215, 179], chaque satellite utilisé pouvait pointer soit sur la *

gauche, soit sur la droite de la trace au sol. Le pointage nominal était le pointage à droite. Un passage en pointage à gauche était possible, mais pour éviter des gaspillages en temps de manœuvre, il était imposé que tout passage en pointage à gauche devait commencer et finir par une acquisition prioritaire (pas de pointage à gauche sans raison valable). La
* mission Pflex [224] fournissait d'autres exemples d'exigences, avec une contrainte d'allocation spécifiant qu'une tâche de vidage associée à un utilisateur donné ne pouvait être réalisée que sur un sous-ensemble des stations de réception sol, et une contrainte spécifiant que tous les fichiers issus d'une même acquisition devaient tous être vidés dans une même fenêtre de visibilité station.

2.2.5 Critères d'optimisation

Pour juger de la qualité des plans d'acquisition et des plans de vidage, nous avons dû considérer dans nos travaux des critères d'optimisation différents des critères classiques utilisés en planification et en ordonnancement de type *reward* (maximisation d'une somme de récompenses collectées sur l'horizon), *makespan* (minimisation de la date de fin de réalisation des tâches), ou *tardiness* (minimisation du retard prévu par rapport à des dates de référence). Les méthodes classiques de planification et d'ordonnancement spécialisées pour optimiser ces quantités n'étaient pas directement pertinentes dans notre cas. En pratique, nous avons eu à considérer divers critères :

- *la priorité et le poids des tâches réalisées* : dans les missions à traiter, les acquisitions candidates sont souvent réparties suivant plusieurs niveaux de priorité étanches (toute amélioration au niveau de priorité k est préférée à une amélioration au niveau $k+1$), avec en plus de cela des poids permettant de représenter l'importance de chaque acquisition au sein d'un même niveau de priorité ; le critère le plus simple envisageable, que nous avons par exemple utilisé pour la mission HotS-
  * pot [205], consiste à optimiser de manière hiérarchique la somme des poids des tâches réalisées avec, dans le cas des tâches composites se décomposant en plusieurs tâches élémentaires, une pondération supplémentaire fonction du degré de réalisation de ces tâches élémentaires ;

- *la qualité de réalisation des acquisitions* : pour un satellite agile effectuant des observations dans le domaine optique, la qualité d'acquisition d'une zone à la surface du globe dépend de plusieurs facteurs, tels que la présence de nuages au-dessus de la zone et l'angle de visée du satellite au moment de l'acquisition ; la qualité de réalisation peut également être présente indépendamment de l'agilité ; par exemple,
  * dans la mission Integral [207], nous avions à considérer des acquisitions composites se décomposant en acquisitions élémentaires ; pour certaines acquisitions composites, le but était de minimiser l'étalement des acquisitions élémentaires sur l'horizon de planification ; pour d'autres, le but était de maximiser cet étalement ; pour d'autres encore, le but était de réaliser les acquisitions élémentaires avec la plus grande régularité possible ; dans notre traitement de la mission de surveillance des mers du globe Samson, la qualité d'observation d'une zone était fonction du nombre
  * total de revisites de la zone et de la fréquence de revisite [215, 179] ;

- *la qualité de réalisation des vidages* : des considérations liées à la qualité du vidage sont également devenues prépondérantes au cours des dernières années, le maître mot étant de minimiser l'âge de l'information, défini comme la durée séparant la date de réalisation d'une acquisition et la date de son vidage vers une station sol ; nous avons utilisé ce critère dans plusieurs missions, dont les missions Samson [215, 179] et Ecoute [255, 212] ; nous avons eu aussi à traiter des cas plus complexes, dans lesquels l'âge de l'information était défini comme la durée écoulée entre la date de réalisation d'une acquisition et la date de sa livraison à l'utilisateur final, en prenant en compte des phénomènes d'engorgement dans les transmissions sur le réseau sol [224] ; *

- *le partage équitable entre utilisateurs* : les satellites sont souvent utilisés par plusieurs entités (des scientifiques, des acteurs de la défense ou de la sécurité civile, des partenaires privés...), et nous avons dû sur certaines missions intégrer un critère de partage équitable des ressources ; dans les missions Ecoute [255, 212], AgataOne [197] et Pflex [224], nous nous sommes notamment appuyés sur des travaux de la théorie du choix social pour arriver à des compromis intermédiaires entre une approche utilitariste optimisant la somme des satisfactions des utilisateurs et une approche égalitariste maximisant la satisfaction de l'utilisateur le moins bien servi [35]. * *

A partir des critères précédents, nous avons toujours dérivé *in fine* un critère unique, en combinant éventuellement plusieurs sous-critères de manière hiérarchique et/ou en utilisant des sommes pondérées de critères. Une des raisons à cela est qu'au final un seul plan d'acquisition et un seul plan de vidage devaient être sélectionnés automatiquement.

2.2.6 Gestion des incertitudes

Un autre aspect présent dans les problèmes de décision du domaine spatial est que certains paramètres ne sont pas connus avec certitude au moment de la planification. Il peut par exemple y avoir des incertitudes sur la présence des nuages, des incertitudes sur le volume des données générées par les acquisitions, des incertitudes sur l'efficacité de la liaison descendante satellite-sol, des incertitudes sur l'état réel du satellite, des incertitudes sur l'occurrence de phénomènes, des incertitudes sur l'arrivée de demandes d'acquisitions urgentes... Pour traiter ces incertitudes, nous avons exploré dans nos travaux trois options qui diffèrent de par le lieu de la planification : une option correspondant à une planification au sol, une option correspondant à une planification à bord, et une option correspondant à une planification mixte bord-sol.

Planification sol La planification sol correspond à l'approche couramment utilisée dans le spatial. Elle consiste à définir au sol des plans d'acquisition et des plans de vidage entièrement datés, et à envoyer ces plans pour exécution à bord des satellites. Cette manière de faire est utilisée même pour des satellites d'observation très récents tels que SPOT6 ou SPOT7 lancés en 2012 et 2014, et Pléiades 1A et 1B lancés en 2011 et 2012. Dans nos travaux, nous avons contribué à la définition d'algorithmes de planification sol pour les missions Integral [207] et Samson [215, 179]. Pour que les plans produits soient applicables malgré les incertitudes, les plans synthétisés sont des plans robustes construits sur la base d'hypothèses pessimistes, afin d'être que sûr que les plans envoyés ne mettent pas en danger *

un système coûteux et difficilement réparable : hypothèses pessimistes sur les consommations de ressources, hypothèses pessimistes sur le volume des données générées, hypothèse d'absence de nouveaux phénomènes... Lorsque la planification sol est une planification long terme fournissant des plans gros grain non directement destinés à être exécutés, les hypothèses faites peuvent être des hypothèses moyennes. Nous n'avons par contre jamais opté même au sol pour une gestion de l'incertitude avec des approches de type synthèse de politique complète sur la base de MDPs (*Markov Decision Processes* [227]), essentiellement à cause de la taille des problèmes à traiter et de la complexité des contraintes temporelles, des ressources et des critères à prendre en compte.

Planification bord Pour quelques rares engins spatiaux, la planification peut être réalisée à bord. C'est le cas pour le satellite EO-1 de la NASA lancé en 2000 [46, 47] et pour le micro-satellite PROBA-1 de l'ESA lancé en 2001 [56, 111]. C'est aussi le cas pour des sondes spatiales telles que Deep Space One [166]. Plusieurs des études amonts que nous avons effectuées ont eu pour objectif de donner plus d'autonomie aux satellites dans leur prise de décision. Du point de vue des incertitudes, une planification à bord permet de raisonner avec une meilleure connaissance de l'état réel du satellite. Cette
* meilleure connaissance nous a permis sur une mission comme Pflex [224] d'accroître les performances du système de vidage de données, grâce à la connaissance à bord du volume réel des données générées par les acquisitions. L'autre intérêt de l'autonomie bord est de doter le satellite d'une plus grande réactivité par rapport à des phénomènes détectés à
* bord. Par exemple, dans les missions HotSpot [205] et AgataOne [197], la planification bord nous a permis de réaliser des observations de phénomènes détectés à la surface de la Terre directement dans la foulée de leur détection, plutôt que d'attendre la prochaine orbite du satellite ou plus (voir la figure 2.7). Les phénomènes en question étaient l'apparition de points chauds et l'apparition d'activités électromagnétiques. D'autres auteurs se sont aussi intéressés à la détection de nuages à bord [16]. Sur tous ces points, il est important de noter que les satellites d'observation et de surveillance en général ne sont pas en visibilité permanente depuis le sol, c'est-à-dire qu'ils ne peuvent ni envoyer en continu des informations sur leur état, ni recevoir en continu des plans à exécuter. Il est également important de noter que tout n'est pas forcément connu au moment de la planification bord, et dans ce cas nous avons à nouveau utilisé des hypothèses pessimistes de consommation
* des ressources [224]. Nous avons également traité les incertitudes en générant dans la mis-
* sion Pflex [224] des plans flexibles temporellement, adaptables en fonction du contexte rencontré, et représentés sous la forme de POS (*Partial Order Schedules* [192]).

Combinaison entre planification sol et planification bord L'inconvénient principal d'une gestion des incertitudes par une prise de décision à bord tient aux très faibles ressources de calcul disponibles à bord des satellites, avec des processeurs robustifiés pour le spatial qui sont jusqu'à 100 fois ou 1000 fois plus lents qu'un processeur standard. Ces limitations sont d'autant plus dimensionnantes que les décisions à bord doivent généralement être prises en temps très contraint, puisque le satellite bouge sur son orbite en permanence, et donc peut potentiellement manquer des opportunités de réalisation de tâches si les raisonnements sont trop longs. A l'inverse, les ressources de calcul disponibles au sol sont plus importantes. C'est pourquoi nous avons tenté plus récemment, à travers la thèse d'Adrien

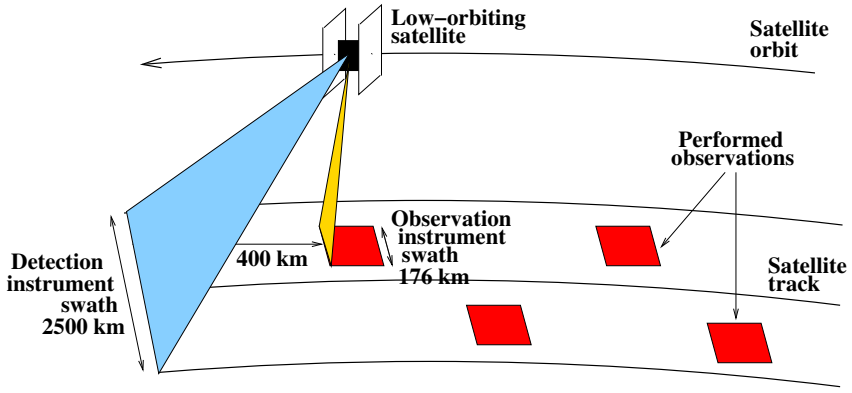


FIGURE 2.7 : Vue de la mission HotSpot [205] et de comment des détections à bord peuvent être utilisées pour améliorer la réactivité (observation directement après détection) *

Maillard, de réfléchir à des stratégies permettant de produire au sol des plans flexibles adaptables à bord pour des satellites agiles d'observation de la Terre. L'idée sous-jacente était de gérer les incertitudes en couplant une planification hors ligne (au sol) avec une planification en ligne (à bord), afin de bénéficier à la fois de la puissance de calcul disponible au niveau du centre de mission et de la connaissance disponible en ligne à bord. Un autre enjeu était d'apporter aux opérationnels plus de prévisibilité, et donc plus de garanties, sur le comportement du satellite par rapport à une planification complètement autonome. Diverses stratégies ont été proposées dans ce sens [149, 150, 151]. Ces stratégies vont d'une * approche dans laquelle les plans sont complètement définis au sol à une approche dans laquelle le bord est totalement autonome. Elles peuvent aller ou bien dans le sens d'une décomposition du problème de planification en plusieurs sous-problèmes, certains étant résolus au sol et d'autres à bord, ou bien dans le sens d'une synthèse au sol de plans de référence, contenant des décisions obligatoires et des conseils, ces plans de référence devant être suivis au mieux par le bord. Nous avons proposé de gérer les incertitudes en faisant des hypothèses pessimistes de consommation des ressources pour les acquisitions les plus prioritaires, et des hypothèses moyennes pour les autres. Nous avons enfin mené des travaux pour abstraire l'approche à des problèmes de décision plus généraux, faisant intervenir une phase de calcul hors ligne et une phase d'adaptation des plans en ligne. Des mécanismes de génération de plans conditionnels [186] auraient aussi pu être étudiés.

2.2.7 Décision en continu dans un environnement dynamique

Un dernier point est que comme pour beaucoup d'autres systèmes, la résolution d'un problème de planification n'est qu'une facette de la problématique du contrôle. L'autre facette est la mise en œuvre réelle de la décision au sein d'un système en activité permanente, sur un horizon de vie de plusieurs années. Comme illustré à la figure 2.8, la planification se fait dans le domaine spatial sur un horizon glissant, et l'un des principaux choix à faire concerne l'horizon sur lequel les plans sont construits ainsi que la fréquence des planifications.

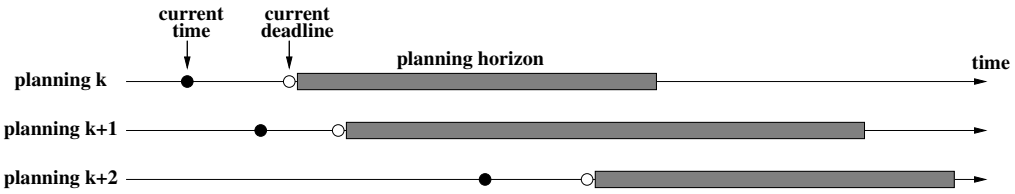


FIGURE 2.8 : Date limite pour la décision et horizon de planification

Pour la planification sol, une stratégie classique consiste à utiliser un horizon de pla-
* nification d'une journée, comme dans la mission Samson [215, 179]. Des horizons de pla-
nification des acquisitions à très long terme peuvent aussi être considérés, comme dans
* nos travaux sur le satellite d'observation de l'univers Integral [207]. Pour ce dernier, nous avions à construire un plan d'acquisition sur un horizon d'une année. L'exemple d'un tel plan est donné sur la partie gauche de la figure 2.9. Les techniques automatiques que nous avons définies pour cette mission ont permis de produire en quelques minutes des plans d'acquisition, là où l'approche précédente côté ESA, une approche manuelle, requérait des semaines de travail. En termes de fréquence de planification, le processus de construction d'un plan à envoyer au satellite peut être mis en œuvre ou bien de manière régulière (par exemple rafraîchissement toutes les 8 heures des plans construits sur un horizon d'une journée), ou bien suite à des événements tels que des demandes urgentes d'acquisition, comme décrit dans nos travaux sur la mission Samson de surveillance des mers du
* globe [179].

Pour la planification bord, nous avons considéré des horizons beaucoup plus courts que pour une planification sol mais des fréquences de planification beaucoup plus élevées. Sur les missions que nous avons traitées, cet horizon de planification variait de quelques minutes à quelques heures. Voir la partie droite de la figure 2.9 pour un exemple de
* plan d'acquisition construit à bord sur un horizon court pour la mission AgataOne [197]. Le processus de planification à bord était utilisé ou bien suite à l'arrivée d'une nouvelle information comme dans [45], par exemple suite à l'arrivée d'une nouvelle détection, ou bien juste avant une date à laquelle une activité candidate pouvait éventuellement être enclenchée, par exemple juste avant le début d'une fenêtre de visibilité d'une station de réception. Pour la planification mixte bord-sol, nous avons combiné les fréquences et les horizons utilisés au sol et à bord.

Dans le cas d'une décision bord, il est de plus nécessaire d'intégrer les algorithmes de planification au sein d'une architecture décisionnelle. L'architecture que nous avons utilisée, présentée à la figure 2.10, est composée de deux tâches : une tâche *réactive*, en charge de l'interaction avec l'environnement, et une tâche *délibérative*, en charge de produire des plans d'activités [140]. La tâche réactive lance la tâche délibérative en lui donnant un horizon de planification, l'état prévu au début de cet horizon, et une échéance de décision. L'horizon est calculé en fonction des actions non interruptibles en cours et de la prochaine échéance de décision. La ou les première(s) action(s) des plans produits par la tâche délibérative sont renvoyée(s) à la tâche réactive comme proposition de décision. La tâche délibérative est supposée avoir un comportement anytime, c'est-à-dire être capable de construire rapidement des plans d'activités de bonne qualité. Lorsque la date limite

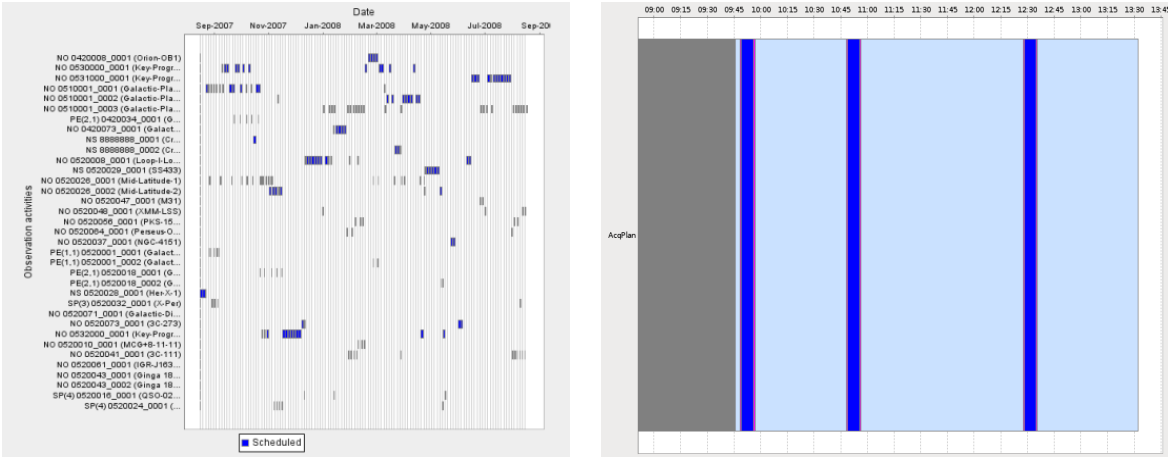


FIGURE 2.9 : Variété des horizons de planification ; partie gauche : plan produit par notre planificateur pour la mission Integral [207] sur un horizon long terme d'une année (chaque ligne correspond à une cible particulière à observer et les observations élémentaires sont représentées en bleu ; partie droite : plan d'acquisition produit par notre planificateur pour la mission AgataOne [197] sur un horizon court terme d'environ 4 heures (le grisé correspond au passé, le bleu ciel à du pointage géocentrique, le bleu foncé à des acquisitions, et le magenta à des manœuvres en attitude)

arrive, la tâche réactive prend une décision et s'engage dans la réalisation d'une certaine action. Lorsqu'aucune proposition de décision n'est disponible, par exemple dans le cas où la tâche délibérative n'a pas eu assez de temps pour construire un plan, la tâche réactive prend une décision réactive donnée par une règle de décision. Lorsqu'une proposition de décision est disponible, la tâche réactive vérifie la cohérence entre cette proposition et l'état courant de l'environnement avant d'engager l'action associée, pour ne pas mettre le système en danger. Cette architecture décisionnelle pour une planification bord a notamment été intégrée au sein d'un simulateur pour la mission AgataOne [196].

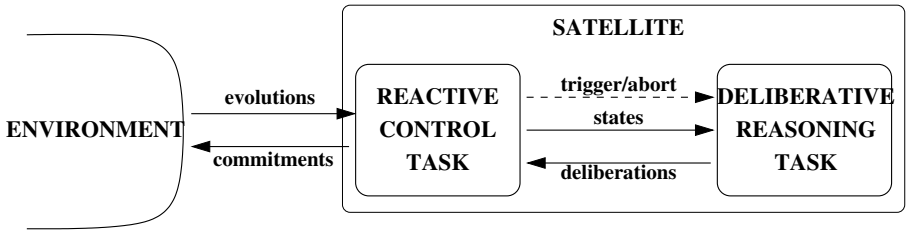


FIGURE 2.10 : Architecture décisionnelle combinant tâche réactive et tâche délibérative

2.3 Cadres de modélisation et techniques de résolution

Plusieurs approches sont utilisables pour aborder les problèmes de planification d'activités des satellites d'observation. Voir [48] pour des exemples de systèmes de planification automatiques utilisés pour diverses missions spatiales. Globalement, les approches utilisables

se répartissent en trois catégories : les approches tentant de réutiliser directement des outils d'optimisation existants, les approches consistant à développer des planificateurs spécifiques à chaque mission, et les approches consistant à introduire de nouvelles techniques génériques suffisamment souples et efficaces pour couvrir les missions dans toute leur diversité. Dans nos travaux de recherche, nous avons exploré ces trois approches, et nous avons peu à peu placé l'effort sur la dernière.

2.3.1 Réutilisation d'outils classiques d'optimisation combinatoire

Dans certains de nos travaux, nous avons tenté de réutiliser directement des outils classiques d'optimisation combinatoire. D'autres auteurs avaient déjà tenté le même type d'approche pour des problèmes du domaine spatial. Par exemple, des algorithmes de recherche de plus long chemin dans un graphe ont été étudiés dans [82, 83] pour traiter au sol des problèmes de sélection d'acquisitions pour un satellite non agile ; la programmation linéaire a été utilisée dans [178, 39, 231] pour planifier au sol les activités de vidage de la sonde Mars Express et dans [20, 2, 153, 144, 117] pour planifier au sol les acquisitions de divers satellites d'observation de la Terre ; la programmation par contraintes a été tentée pour planifier au sol les acquisitions de satellites [139, 141] ; la programmation dynamique a été expérimentée dans [104, 254] pour définir au sol des plans d'acquisition et des plans de vidage. Voir [96, 141] pour des états de l'art sur le sujet.

Dans notre expérience, la réutilisation directe d'outils existants s'est avérée fructueuse sur quelques missions offrant une structure suffisamment favorable. Voir l'utilisation de la programmation quadratique mixte que nous avons mise en œuvre pour traiter un problème de partage de fenêtres de communication avec le sol pour une constellation
* de satellites [216]. Pour beaucoup de missions cependant, la réutilisation directe d'outils d'optimisation classiques tels que IBM ILOG CPLEX/CpOptimizer [115], Choco [49] et COMET [107] nous a conduits à des impasses de diverses natures :

- des impasses de modélisation, qui pour être contournées ont nécessité d'épurer les problèmes à résoudre, par exemple d'épurer les contraintes temporelles *time-dependent* liées à l'agilité ou d'épurer le schéma exact d'évolution de certaines ressources ; pour traiter certains aspects, il aurait été envisageable de faire une modélisation avec le langage de planification PDDL+ [78] permettant de capturer des processus d'évolution complexes de l'état, mais les rares outils capables de traiter des modèles PDDL+ comme UPMurphi [185] n'étaient pas disponibles au moment de nos travaux, et leur application à des problèmes de grande taille reste encore un enjeu ;

- des impasses au niveau des temps de calcul et de l'espace mémoire consommé par les modèles, notamment pour les approches utilisant des recherches arborescentes en profondeur d'abord qui ne passent pas à l'échelle sur des problèmes impliquant plusieurs milliers ou dizaines de milliers de tâches à ordonnancer ; les problèmes de passage à l'échelle ont aussi été rencontrés lorsque nous avons cherché à ne pas épurer
* des problèmes aux spécifications complexes [205] ;

- des impasses pour une utilisation à bord d'un satellite autonome, à cause de contraintes d'implémentation sur les systèmes embarqués, liées notamment à la nécessité

d'un contrôle de la taille mémoire utilisée par le planificateur, et à la nécessité d'absence d'allocation dynamique de mémoire ; des contraintes d'implémentation prises en compte sur la mission EO-1 sont par exemple détaillées dans [250].

2.3.2 Définition de planificateurs complètement spécifiques

A l'opposé, nous avons également tenté dans certains de nos travaux une approche de définition directe de planificateurs dédiés tirant parti de toutes les spécificités des missions. Au sein de ces planificateurs, deux grands types de techniques ont été expérimentés :

- des techniques de recherche dites *gloutonnes*, qui prennent des décisions sans jamais revenir dessus, les avantages essentiels étant la rapidité de construction des plans, l'embarquabilité, et la simplicité des algorithmes, chose importante du point de vue opérationnel ; ces avantages viennent au prix de solutions potentiellement sous-optimales ; des approches gloutonnes spécifiques sont par exemple utilisées tous les jours pour planifier les activités des satellites SPOT ou Pléiades [134] : ces approches partent d'un plan vide et tentent, à chaque itération, d'insérer dans le plan une acquisition choisie parmi les acquisitions les plus prioritaires (glouton *hiérarchique*) ; nous avons testé l'utilisation d'un algorithme glouton dédié pour faire de la planification bord pour la mission HotSpot [205], plus précisément un algorithme glouton dit * *chronologique*, *stochastique*, et *itéré* ; cet algorithme consistait à effectuer plusieurs recherches gloutonnes successives selon des heuristiques de décision bruitées de plus en plus au fur et à mesure des itérations ; avec cet algorithme, les temps de calcul pour produire une première solution à chaque planification bord étaient largement inférieurs à la milli-seconde sur des scénarios réalistes, et le planificateur obtenu a été intégré au sein d'un démonstrateur CNES de satellite autonome ;

- des techniques de recherche dites *locales*, qui manipulent un plan courant et tentent de l'améliorer progressivement via des modifications locales (ajouts d'activités, retraits d'activités, échanges d'activités...) réalisées librement n'importe où sur l'horizon de planification ; dans ce contexte, tout l'arsenal des techniques de recherche locale peut être utilisé ; dans le domaine de l'observation de la Terre, voir [253, 24] pour une utilisation de la recherche tabou, [267, 95] pour une utilisation d'algorithmes évolutionnaires, [96] pour une utilisation d'un recuit simulé, et [97] pour une utilisation de la méthode SWO [121] ; de notre côté, nous avons proposé une méthode complètement spécifique de recherche locale pour traiter la mission Samson [179], avec un algorithme itérant des phases de construction et de destruction * d'un plan courant ; cet algorithme a été capable de passer à l'échelle sur un problème impliquant plusieurs dizaines de milliers d'acquisitions élémentaires.

Dans le cas des planificateurs spécifiques, un inconvénient est que le modèle du problème de planification et l'algorithme de résolution ad hoc sont entremêlés dans le code, ce qui rend l'approche moins facilement adaptable en cas de changement dans les spécifications du problème. L'approche impose de plus de tout redévelopper à chaque fois.

2.3.3 Définition de nouvelles techniques génériques

Les inconvénients rencontrés avec la stratégie de réutilisation d'outils d'optimisation combinatoire classiques et avec la stratégie de définition de planificateurs complètement spécifiques nous ont poussés à introduire de nouvelles techniques génériques plus à même de traiter les problèmes de planification du domaine spatial.

Cette recherche de nouvelles techniques génériques appropriées peut aussi être observée dans plusieurs agences spatiales :

- côté NASA Ames, le cadre CAIP (*Constraint-based Attribute and Interval Planning* [79]) mis en œuvre au sein de l'outil EUROPA est utilisé comme base pour traiter les problèmes de planification ; l'outil EUROPA fait appel à des techniques de programmation par contraintes pour raisonner sur les modèles, et il fonctionne en réparant des plans incomplets de manière itérative ;
- côté NASA JPL, les outils ASPEN (*Automated Scheduling and Planning ENvironment* [229]) et CASPER (*Continuous Activity Scheduling Planning Execution and Replanning* [45]) mettent à disposition des briques de base rencontrées dans plusieurs problèmes d'ordonnancement et de planification, comme la gestion du temps et la gestion des ressources ; ces outils utilisent des méthodes de recherche locale pour réparer un plan courant ;
- côté ESA, des moyens tels que APSI (*Advanced Planning and Scheduling Initiative* [41]) ont été définis, et suivent également une approche consistant à définir des briques de modélisation et de résolution de base réutilisables de mission à mission ;
- côté DLR/GSOC, des outils tels que Pinta/Plato [154, 90] ont été introduits.

Parmi les outils précédents, seul EUROPA est libre d'utilisation. Nous avons eu accès
* ponctuellement à la bibliothèque APSI pour traiter la mission Integral [207] ; pour notre besoin, cette bibliothèque présentait l'inconvénient de ne pas gérer les aspects critères et de ne pas être conçue à la base pour supporter la définition d'algorithmes de recherche locale. De notre côté, nous sommes partis sur des techniques utilisant le paradigme de la programmation par contraintes comme dans EUROPA, ce paradigme présentant un intérêt fort du point de vue de la modélisation, en offrant des modèles déclaratifs explicites à base de variables de décision, de contraintes et de critères. Nous avons défini successivement dans nos travaux trois nouveaux cadres de modélisation et de résolution :

* - le cadre CNT (*Constraint Network on Timelines* [260, 206, 261]), présenté au chapitre 3 ; ce cadre peut être vu comme une première tentative pour gérer des systèmes dynamiques à événements discrets avec des modèles à base de contraintes ;

* - le cadre CTA (*Constraint-based Timed Automata* [223]), présenté aussi au chapitre 3 ; ce cadre a permis de corriger certains défauts du cadre CNT et de mettre en œuvre facilement des algorithmes de recherche chronologique, prenant des décisions dans l'ordre du temps ;

* - le cadre InCELL (*Invariant-based Constraint EvaLuation Library* [212]), présenté au chapitre 4 ; l'idée initiale qui a motivé la définition d'InCELL était de définir

une bibliothèque de base sur laquelle s'appuyer pour définir des algorithmes dédiés de recherche gloutonne non chronologique ou de recherche locale, et non une bibliothèque hypothétique offrant des algorithmes complètement génériques et capables de résoudre tous les problèmes ; grossièrement, InCELL combine l'aspect contraintes d'EUROPA et l'aspect recherche locale d'ASPEN ; pour combiner ces aspects, nous nous sommes tournés vers des méthodes dites de *recherche locale à base de contraintes* (CBLS, *Constraint-Based Local Search* [107]).

Parmi ces cadres, le cadre InCELL est celui que nous avons le plus étudié et utilisé. Nous l'avons par exemple expérimenté sur la mission AgataOne [197] comme brique de * base pour mettre en œuvre un algorithme bord fonctionnant par ajouts successifs de tâches d'acquisition et de vidage au sein d'un plan, de la plus prioritaire à la moins prioritaire. Nous l'avons aussi expérimenté dans [224] sur la mission Pflex pour mettre en œuvre * rapidement des algorithmes de recherche locale. Des contraintes d'implémentation ont aussi été prises en compte dans l'implémentation de la bibliothèque InCELL, avec des efforts spécifiques pour définir des planificateurs bord capables de fonctionner à partir d'un modèle unique créé une fois pour toutes au début de la vie du satellite [212]. *

2.4 Au-delà du domaine spatial

L'utilisation d'une approche CBLS implémentée au-dessus de l'outil InCELL a été éprouvée sur plusieurs missions du domaine spatial. L'idée est ensuite venue d'appliquer l'approche à d'autres domaines. Cela a débouché sur la mise en œuvre d'une stratégie CBLS pour traiter des missions du domaine de la robotique, contributions que nous avons abordées dans [116] et [199]. Le but de la mission décrite dans [199] était de faire collaborer des robots au sol * et des drones pour gérer des situations de crise de type incident sur une centrale nucléaire, incident sur une usine chimique ou catastrophe naturelle. Plus précisément, l'objectif était d'utiliser des moyens robotiques pour réaliser, sur des zones à accès dangereux ou difficile, des opérations de collecte de données (flux vidéo, mesures de radioactivité...). Les données collectées devaient de plus être rapatriées en temps réel ou en quasi temps réel vers un centre d'opération, afin que les opérateurs puissent demander à la volée de nouvelles acquisitions en fonction des informations reçues. Les données devaient être transmises même en cas d'indisponibilité des moyens de communication traditionnels. Cela impliquait de déployer un réseau de communication sans fil ad hoc, faisant intervenir des points d'accès Wifi embarqués sur des drones jouant le rôle de relais de communication. La figure 2.11 donne un exemple de stratégie de déploiement obtenue sur une telle mission. Les ordonnancements produits devaient également être flexibles temporellement et robustes aux incertitudes sur les durées de déplacement des robots à l'exécution, ces durées n'étant pas connues à l'avance avec précision notamment pour les robots au sol censés se déplacer dans des zones potentiellement sinistrées.

D'un point de vue technique, le problème à traiter était un problème d'ordonnancement dans lequel les activités des robots devaient être synchronisées dans le temps et dans l'espace, puisque pour communiquer deux robots avaient besoin de se trouver simultanément à des positions relativement proches. Par rapport aux problèmes du domaine spatial, la mission présentait un nombre plus important de degrés de liberté car (1) alors

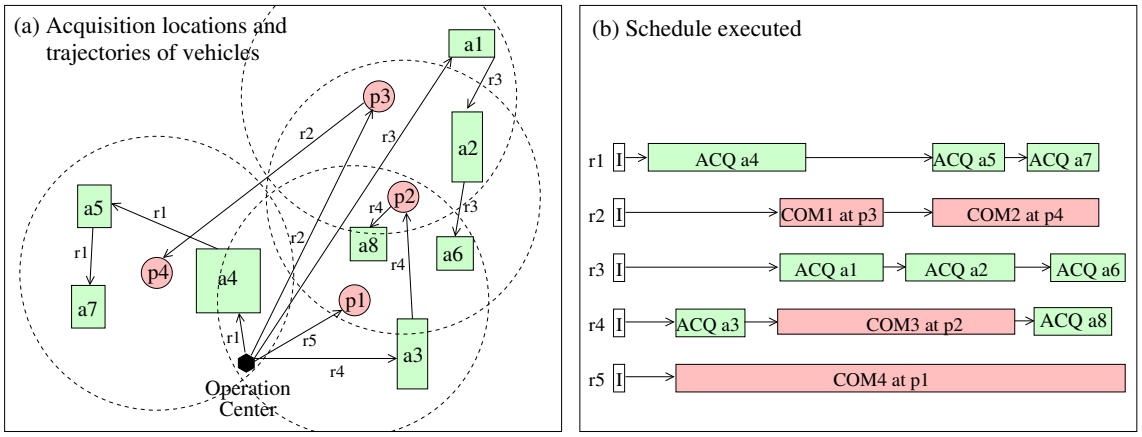


FIGURE 2.11 : Ordonnancement du déploiement d'une équipe de 5 robots (r_1 à r_5) pour réaliser 8 acquisitions (a_1 à a_8), et en positionnant des relais de communication sur l'une des 4 positions candidates (p_1 à p_4), la portée associée à chaque position étant représentée par les cercles en pointillés ; les actions de communication sont représentées en vert et les actions de relais de communication en rose

qu'un satellite reste sur son orbite, robots au sol et robots aériens peuvent bouger plus librement, et (2) alors qu'un satellite envoie des données vers des stations de réception sol généralement fixes, un robot collectant des données envoyait des informations vers des relais de communication mobiles. Par rapport aux solutions existantes dans le monde de la robotique [180, 163], nous avons contribué à définir une approche de type ordonnancement sur l'horizon de la mission, au lieu d'une approche complètement réactive et fortement sous-optimale. Sur ce problème, la bibliothèque InCELL a servi de base pour définir une stratégie de recherche locale utilisant la méta-heuristique GRASP (*Greedy Randomized Adaptive Search Procedure* [75, 228]). Des algorithmes spécifiques de théorie des graphes, inspirés de travaux sur les WSNs (*Wireless Sensor Networks* [3, 269, 138]), ont également été utilisés dans le planificateur pour construire de bons réseaux de robots
* relais. Enfin, comme dans la mission Pflex [224], les plans produits ont été représentés avec une approche POS (*Partial Order Schedule* [192]) permettant à chaque robot de respecter les différentes contraintes de synchronisation du problème.

Mise en œuvre de la décision Concernant les aspects architecture de décision, nous avons considéré dans nos travaux une planification hors ligne pour tous les engins au niveau du centre de mission, et une répartition des plans produits sur l'ensemble des engins. Chaque engin recevait la séquence des activités d'acquisition et de relais de communication dont il avait la charge, avec pour chacune des activités de cette séquence une description des activités d'autres robots avec lesquelles se synchroniser. La partie exécution a quant à elle été implémentée par d'autres sur la base de l'outil ProCoSa [14] utilisant le formalisme des réseaux de Petri.

Nous avons plus récemment tenté d'étendre le fonctionnement à une approche en ligne, au travers de la thèse de Guillaume Casanova. L'ambition était de gérer correctement les aspects temporels à l'exécution, dans un contexte où il peut être difficile d'anticiper les

durées exactes de déplacement d'un engin même dans une zone connue. Une première stratégie pour régler ce problème peut consister à réutiliser des travaux sur les STNU (*Simple Temporal Networks with Uncertainty* [263, 161]). Nous avons adopté une autre stratégie dans la thèse de Guillaume Casanova, dans laquelle le but est d'arriver à exécuter de manière distribuée des plans de mission en utilisant une approche de type planification/replanification en cas de déviations entre durée réelle des tâches à l'exécution et durée envisagée au moment de la planification. Ces déviations peuvent invalider les plans de mission ou les rendre fortement sous-optimaux, et l'un des objectifs est de détecter au mieux ces situations d'incohérence ou de sous-optimalité, pour ensuite réparer ou réoptimiser les plans ou bien localement au niveau d'un seul engin, ou bien plus globalement au niveau de plusieurs engins. Pour l'instant, la thèse s'est focalisée sur la gestion des contraintes temporelles dans un contexte distribué, et plusieurs stratégies ont été proposées, allant de stratégies plutôt centralisées dans lesquelles chaque engin envoie en permanence au centre d'opération ses connaissances sur les dates réelles de ses activités, à des stratégies plus distribuées dans lesquelles chaque engin envoie des informations uniquement aux engins directement concernés [37]. *

2.5 Bilan

Un enjeu dans nos travaux de recherche est de capturer des structures communes à plusieurs missions, de travailler spécifiquement sur ces structures, et de transférer les travaux réalisés sur de nouvelles applications. Pour remplir cet objectif, nous nous sommes peu à peu orientés vers des techniques génériques utilisant des principes de la programmation par contraintes pour obtenir une flexibilité en termes de modélisation, et vers des techniques dans lesquelles les algorithmes de résolution ne sont pas complètement génériques (recherche gloutonne spécifique ou recherche locale spécifique). Comme trouver une solution optimale est le plus souvent hors d'atteinte dans les problèmes que nous traitons, nous nous sommes concentrés plutôt sur le développement de techniques approchées, permettant de produire rapidement des solutions de bonne qualité. Enfin, nous avons poursuivi un travail de modélisation formelle des problèmes, qui a nécessité des interactions avec des spécialistes métiers. Ce travail sur la modélisation est aussi important que celui sur les algorithmes.

Chapitre 3

Contraintes et systèmes dynamiques à événements discrets

Dans ce chapitre, nous décrivons les premières contributions méthodologiques que nous avons réalisées pour traiter des problèmes de planification, en particulier les problèmes issus du domaine spatial présentés au chapitre précédent. Ces premières contributions avaient initialement pour objectif d'arriver à combiner au sein d'un même cadre de représentation à la fois des modèles dynamiques, décrivant les schémas d'évolution des systèmes considérés en fonction des décisions prises, et des modèles statiques, exprimant des connaissances plus globales sur les trajectoires suivies par ces systèmes sur l'horizon de raisonnement. Pour obtenir une telle combinaison entre modèles dynamiques et modèles statiques, nous avons successivement introduit deux nouveaux cadres de représentation et de résolution : le cadre CNT présenté à la section 3.1 et le cadre CTA présenté à la section 3.2.

3.1 Le cadre CNT : planification avec des CSP dynamiques

Classiquement, les problèmes de planification [92] sont représentés par des modèles "dynamiques" construits au-dessus des notions d'*états* et d'*actions*. Dans ces modèles, l'état du système à contrôler est décrit par un ensemble d'attributs, et les actions sont définies par leurs conditions d'application et leurs effets sur l'état. De telles descriptions ont été introduites historiquement avec le formalisme STRIPS [76] et ont été étendues ensuite avec le langage PDDL et ses variantes [155, 77, 67, 268, 89, 78]. Dans ces approches, un parti pris est souvent de définir des modèles dits *domain-independent*, dans lesquels seule la physique de base des actions est fournie au planificateur. Ce dernier peut alors avoir comme tâche de trouver en toute autonomie une séquence d'actions menant le système dans un état but. Des modèles dits *domain-dependent* ou *domain-configurable*, qui intègrent plus de connaissances, existent également, par exemple des modèles incluant des hiérarchies de tâches (modèle HTN [70]) et des modèles spécifiant des formules de logique temporelle à satisfaire par la séquence des états parcourus par le système [10, 132].

A l'opposé, les modèles de programmation par contraintes (PPC [234]) sont des modèles plutôt "statiques", construits sur les notions de *variables* et de *contraintes*. Plus précisément, la PPC repose sur le formalisme des CSPs (*Constraint Satisfaction Problems* [148]). Un

CSP est défini par un ensemble fini de variables V, chaque variable $x \in V$ ayant un domaine fini de valeurs possibles noté $\mathbf{d}(x)$, et par un ensemble fini de contraintes C, chaque contrainte $c \in C$ étant définie par l'ensemble $V(c) \subseteq V$ des variables sur lesquelles elle porte et par l'ensemble des combinaisons de valeurs qu'elle autorise. Une solution d'un CSP (V, C) se définit alors comme une instanciation de toutes les variables de V qui satisfait toutes les contraintes de C. A cela peut être ajoutée une variable objectif à optimiser pour départager différentes solutions. Les modèles PPC peuvent être qualifiés de *domain-dependent* dans le sens où dans un CSP, toute connaissance additionnelle permettant d'accélérer l'exploration de l'espace de recherche est bonne à prendre. En particulier peuvent être employés des choix de variables astucieux, des contraintes élaguant des solutions sous-optimales ou des solutions symétriques, ou encore des heuristiques spécifiques au problème permettant de guider le parcours de l'espace des instanciations possibles des variables.

La première piste explorée dans nos activités de recherche pour mêler modèles dynamiques et modèles statiques a consisté à travailler sur une hybridation entre planification et programmation par contraintes, et plus généralement sur une hybridation entre gestion de systèmes dynamiques à événements discrets et contraintes. Cette hybridation s'est appuyée sur le cadre des *CSPs dynamiques* [159], une extension des CSPs dans laquelle le nombre de variables et de contraintes n'est pas figé (possibilité d'ajouter ou de retirer des variables ou des contraintes en fonction de la valeur prise par certaines variables du modèle). L'idée de mélanger contraintes et planification n'était pas chose nouvelle au début de nos travaux. Deux grandes catégories de techniques existaient déjà dans cette veine, avec d'une part des travaux portant sur l'encodage automatique et efficace de modèles STRIPS/PDDL sous la forme de problèmes SAT / CSP / CSP dynamiques [125, 252, 64, 264], et d'autre part des travaux introduisant de nouveaux formalismes basés contraintes, tels que le formalisme CAIP (*Constraint-based Attribute and Interval Planning* [79]). Ces approches existantes conduisent à deux types de modèles de contraintes, avec d'un côté des modèles encodant le problème de planification sur un horizon fini et incrémentant cet horizon jusqu'à ce qu'un plan solution soit trouvé (approche *iterative deepening*), et de l'autre des modèles manipulant un plan courant incomplet et réparant ce plan par ajout de liens causaux ou de relations entre des éléments du plan (approche par *recherche dans l'espace des plans partiels*, utilisée notamment par les planificateurs CPT [264], EUROPA [79] et IxTeT [91, 133]).

Par rapport à ces approches existantes, l'objectif des travaux de recherche engagés était de revenir à un cadre le plus basique possible de type CSP dynamique, ou autrement dit de ne faire ni de l'encodage automatique de modèles de planification *domain-independent*, ni de la génération de contraintes à partir d'un formalisme *domain-dependent* de plus haut niveau. Les avantages perçus à l'origine dans cette démarche étaient la flexibilité du point de vue de la modélisation, la possibilité de construire directement des modèles *domain-dependent* optimisés vis-à-vis de la propagation de contraintes, la sémantique claire, et la capacité à couvrir, au-delà des problèmes de planification, des problèmes de gestion de systèmes dynamiques à événements discrets en général. Le cadre que nous avons introduit
* pour cela est le cadre CNT (*Constraint Networks on Timelines* [260, 206, 261]).

3.1.1 Le modèle CNT

Un CNT peut être vu comme une forme de CSP dynamique comportant deux parties : une partie statique et une partie dynamique. Une vision graphique d'un modèle CNT est donnée à la figure 3.1.

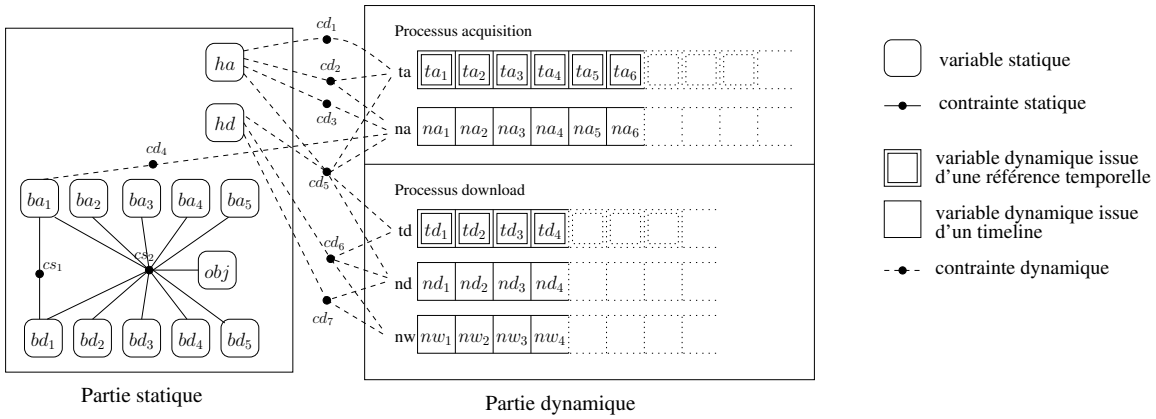


FIGURE 3.1 : Vue globale d'un modèle CNT sur un problème simplifié du domaine spatial.

Partie statique La partie statique d'un modèle CNT (partie gauche de la figure 3.1) correspond à un CSP classique, composé de variables dites *statiques* et de contraintes dites *statiques* portant sur ces variables. Les variables statiques permettent de représenter des paramètres indépendants du temps, par exemple la décision booléenne de réaliser ou non une acquisition sur l'horizon de planification. Une variable objectif décrivant la récompense totale obtenue pour un plan peut également être définie comme une variable statique.

Partie dynamique La partie dynamique (partie droite de la figure 3.1) sert à modéliser l'évolution des attributs décrivant l'état du système dynamique considéré. Ces attributs sont répartis sur un ensemble de processus, représentant chacun l'évolution d'une partie de l'état. Pour un satellite d'observation de la Terre, des exemples de processus sont le processus d'acquisition, le processus de vidage de données du satellite vers des stations sol, le processus d'éclairement des panneaux solaires par le soleil, ou encore le processus de gestion de la mémoire disponible à bord au fur et à mesure des collectes/vidages de données. Pour formaliser la partie dynamique d'un modèle CNT, nous avons associé trois éléments à chaque processus p :

- une variable h appelée *variable d'horizon*, faisant partie des variables statiques et représentant le nombre initialement inconnu d'étapes dans le processus p ; par exemple, pour le processus d'acquisition, le nombre de début/fin d'acquisitions dans un plan solution n'est pas connu a priori ;

- une *référence temporelle* t, représentant les dates auxquelles ont lieu les changements qui impactent le processus p ; pour le processus d'acquisition, il peut s'agir des dates de début et fin d'observation ; pour le processus d'éclairement des panneaux solaires,

il peut s'agir des dates de rentrée/sortie d'éclipse ; pour le processus mémoire, il peut s'agir des dates de réservation/libération de mémoire ; on note t_i la variable représentant la date associée à la ième étape du processus ;

- un ensemble de *timelines* Tl (*chronogrammes*), chaque timeline $tl \in Tl$ représentant l'évolution d'un attribut du processus p ; des exemples de timelines pour le processus de vidage de données peuvent être le numéro de l'acquisition en cours de vidage et l'indice de la station sol vers laquelle le vidage est réalisé ; on note tl_i la variable représentant la valeur prise par le timeline tl à la ième étape du processus.

Les variables t_i/tl_i sont dites *dynamiques* dans le sens où leur existence est conditionnée par la valeur des variables d'horizon des processus. L'évolution d'un timeline tl correspond à la séquence des couples (t_i, tl_i). Un processus est instancié lorsque (1) son horizon est instancié et (2) pour chaque indice i compris entre 1 et la valeur choisie pour l'horizon, toutes les variables dynamiques t_i/tl_i sont instanciées.

Afin de définir parmi les instanciations des processus celles qui correspondent à des évolutions acceptables, un CNT définit un ensemble de contraintes dites *dynamiques* portant sur les processus. Comme l'objectif était de définir un cadre de base, le choix a été fait d'autoriser ces contraintes à porter librement sur toutes les variables dynamiques t_i/tl_i. Le langage de contraintes obtenu ne se limite ainsi pas à des contraintes markoviennes faisant intervenir uniquement les étapes i et $i+1$ d'un processus, et il autorise notamment la définition de contraintes sur les trajectoires globales suivies par le système dynamique. Par exemple, une contrainte $allDifferent(tl_i \mid i \in [1..h])$ peut être définie pour indiquer que globalement, les valeurs successives prises par le timeline tl sur l'horizon variable h doivent être toutes distinctes. Les contraintes dynamiques peuvent porter ou bien sur un seul timeline, ou bien sur les timelines d'un même processus, ou bien sur des timelines de processus différents. Par exemple, étant donnés deux timelines tl, tl' appartenant à deux processus d'horizons respectifs h, h' et de références temporelles respectives t, t', la contrainte $\forall i \in [1..h], \exists j \in [1..h'], (tl_i = tl'_j) \wedge (t_i = t'_j)$ indique que toute valeur prise par le timeline tl doit se retrouver quelque part dans le timeline tl' à la même date. Les contraintes dynamiques peuvent également faire intervenir n'importe quelle variable statique, ce qui permet de lier les paramètres statiques des processus avec les processus eux-mêmes.

Notons enfin que le modèle de base n'impose rien sur la valeur prise par un timeline entre deux étapes de l'horizon. Si l'on considère que les timelines évoluent de manière constante par morceaux, alors des schémas d'évolution tels que celui donné à la figure 3.2 sont obtenus. Cette figure représente l'évolution d'un processus p instancié, composé d'un ensemble de deux timelines ($Tl = \{tl, tl'\}$) synchronisés sur une référence temporelle t. Le processus instancié contient 4 étapes (horizon $h = 4$), et les variables dynamiques *actives* sont $t_1, \ldots, t_4$, $tl_1, \ldots, tl_4$, $tl'_1, \ldots, tl'_4$.

Caractéristiques du modèle Un CNT est complètement instancié lorsque toutes les variables statiques et tous les processus de la partie dynamique sont instanciés. Nous avons défini une solution d'un CNT comme une instanciation complète qui satisfait toutes les contraintes statiques et dynamiques du modèle, et qui éventuellement optimise un certain critère. En toute généralité, le domaine des variables d'horizon n'est pas nécessairement fini. Cela permet d'englober des systèmes dynamiques pour lesquels le nombre d'étapes à

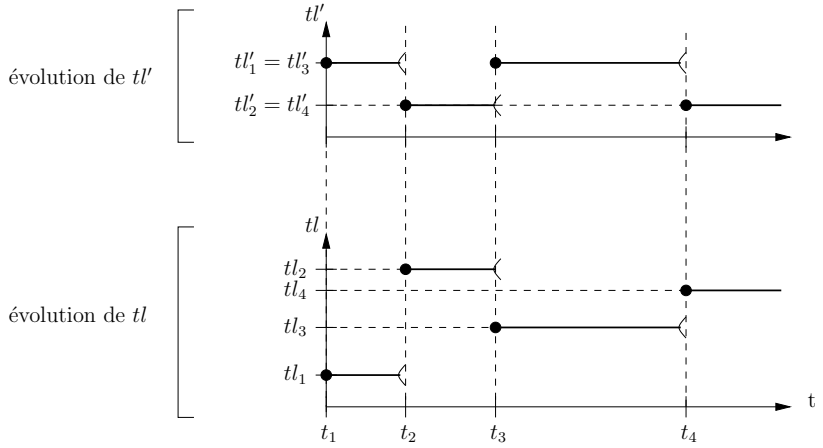


FIGURE 3.2 : Représentation graphique des timelines d'un même processus.

considérer n'est pas borné a priori. De même, le domaine d'évolution des timelines n'est pas nécessairement fini.

D'un point de vue théorique, il a été montré [260] que le problème de recherche d'une * solution d'un CNT couvrait les problèmes classiques d'atteignabilité associés aux automates, aux automates temporisés [4], aux réseaux de Petri [188, 164] et à la planification STRIPS [76], ou encore le problème RCPSP (*Resource-Constrained Project Scheduling Problem* [28, 13]). Il a par ailleurs été prouvé que le problème d'existence d'une solution d'un CNT était indécidable [260]. Ce problème devient semi-décidable dans le cas où toutes * les variables sauf les variables d'horizon ont un domaine de valeurs fini, et NP-complet lorsque tous les domaines de valeurs considérés sont finis [206]. *

3.1.2 Algorithmes

Une fois le cadre défini, nous avons proposé des techniques de résolution génériques pour rechercher une solution d'un CNT. Pour tous les travaux algorithmiques, il a été fait l'hypothèse que hormis pour les variables d'horizon, tous les domaines de valeurs considérés étaient finis. Cela permettait d'éviter l'utilisation de mécanismes de propagation de contraintes sur domaines continus [234].

Nous avons tout d'abord introduit dans [206] des algorithmes complets de recherche * de solutions optimales. La première idée a été de réutiliser une stratégie de recherche arborescente classique en programmation par contraintes : (1) choix à chaque étape d'une variable (statique ou dynamique) non instanciée, (2) réduction du domaine des valeurs autorisées pour cette variable, (3) raisonnements sur les contraintes suite à cette réduction, (4) *backtrack* (retour arrière) en cas d'incohérence, et (5) poursuite de la recherche jusqu'à obtenir une instanciation complète du CNT ou jusqu'à ce que tout l'espace de recherche ait été parcouru. Une adaptation à réaliser était que comme dans un CNT les horizons des processus ne sont pas figés initialement, le nombre de variables sur lesquelles réaliser des branchements n'était pas figé non plus. Ce point a été traité en choisissant de ne pas essayer d'instancier une variable dynamique dont l'existence n'était pas garantie. Plus formellement, pour une référence temporelle t ou un timeline tl associé à un processus

d'horizon h, il a été choisi de ne pas essayer d'instancier les variables dynamiques t_i/tl_i pour i strictement supérieur à la valeur minimale dans le domaine de valeurs de h, cette valeur minimale pouvant être par ailleurs actualisée par la propagation de contraintes au cours de la recherche. D'un point de vue CSP dynamique, cela signifie que nous avons fait le choix dans un CNT d'activer les variables et les contraintes en fonction de l'évolution des valeurs minimales dans les domaines des variables d'horizon.

Plusieurs heuristiques permettant de guider les choix de variable et de valeur à chaque étape de la recherche arborescente ont été étudiées : instanciation des variables d'horizon en premier, instanciation des variables autres que les variables d'horizon en premier, ins-
* tanciation des variables dans n'importe quel ordre [206]... Les travaux réalisés ont montré que fixer en premier les variables d'horizon comme dans les méthodes de type *iterative deepening* n'était pas toujours un bon choix. Ces méthodes peuvent en effet tarder à trouver une première solution fournissant une première borne sur le critère, borne à partir de laquelle des élagages de l'espace de recherche sont possibles. La réutilisation simple d'heuristiques classiques CSP, comme le choix d'instancier la variable ayant le plus petit domaine de valeurs courant, s'est souvent avérée plus efficace.

Des algorithmes *anytime*, permettant d'obtenir rapidement des solutions de bonne
* qualité, ont ensuite été spécifiés [208]. Pour définir ces algorithmes anytime, il a été choisi de parcourir l'espace de recherche de manière *chronologique*. Cela signifie qu'à chaque étape, les différents timelines ne peuvent être étendus que par la droite (avancée dans le temps, ou autrement dit les tableaux de variables du type de ceux de la figure 3.1 sont instanciés de gauche à droite). Afin de définir proprement cet aspect chronologique, nous avons introduit la notion de *temps courant* d'un CNT partiellement instancié, défini comme la plus petite date à laquelle une variable dynamique non instanciée peut être associée (plus petite date à laquelle un choix reste en suspens). Nous avons ensuite construit une heuristique de choix de variable consistant à sélectionner en priorité les variables dynamiques tl_i *ayant nécessairement lieu* à ce temps courant, puis les variables dynamiques temporelles t_i *pouvant avoir lieu* à ce temps courant, et *en dernier recours* les variables statiques. Des
* techniques de recherche avec *restarts* ont également été définies [208], afin de ne pas rester bloqué inutilement dans certaines zones de l'espace de recherche.

3.1.3 Implémentation

Les algorithmes génériques complets et anytime du cadre CNT ont été implémentés au-dessus de la librairie de programmation par contraintes Choco2 [49]. En pratique, seuls des CNTs avec domaines finis ont été considérés, et il a malheureusement été nécessaire de considérer que le domaine des variables d'horizon devait lui aussi être fini, les solveurs de contraintes ne supportant en général que très peu l'ajout de variables en cours de recherche. Une représentation de type CSP dynamique a été construite, utilisant notamment des contraintes *ifThen* pour simuler les activations de contraintes dynamiques.

Les algorithmes du cadre CNT ont été expérimentés sur des problèmes de gestion d'activités d'acquisition et de vidage pour des satellites d'observation de la Terre. Ils ont
* également été testés sur des benchmarks tirés des compétitions de planification [206, 208]. Sans trop de surprise, il a été montré que l'utilisation de modèles CNT encapsulant plus de connaissances sur le système à contrôler permettait d'obtenir des gains significatifs en

temps de calcul par rapport à des planificateurs classiques optimaux *domain-independent*. L'existence conjointe des parties statiques et dynamiques a permis en particulier à la fois d'utiliser des formulations de type CSP dans la partie statique, avec des contraintes bien choisies pour faire fonctionner la propagation de contraintes, et de définir des contraintes sur les évolutions du système à contrôler. Du point de vue planification anytime, il a été montré que les outils CNT produisaient des premières solutions potentiellement moins rapidement que des planificateurs heuristiques classiques, mais que les solutions produites devenaient assez vite de meilleure qualité grâce aux élagages induits par la propagation de contraintes.

En pratique, nous avons observé que les modèles CNT manipulés souffraient du même inconvénient que les approches dépliant le problème de planification sur un certain horizon, à savoir une explosion du nombre de variables et de contraintes. C'est pourquoi nous avons également développé des techniques particulières pour réduire l'augmentation du nombre de contraintes au fur et à mesure de l'allongement des horizons des processus : techniques de *slice-encoding* ou encodage par tranches [209]. La particularité des modèles de * contraintes obtenus avec ces nouvelles techniques est qu'ils gèrent de manière particulière les contraintes dynamiques dites *stationnaires*, qui ont la particularité de se répéter à toute étape de l'horizon. Par exemple, pour deux timelines tl et tl' appartenant à un même processus d'horizon h, la contrainte $\forall i \in [2..h]$, $tl_i \neq tl'_{i-1}$ est une contrainte dynamique stationnaire. Il est possible de l'encoder par une seule méta-contrainte $m : tl^{(0)} \neq tl'^{(-1)}$ portant sur deux variables, l'une représentant la valeur du timeline tl à l'étape courante (variable $tl^{(0)}$), l'autre représentant la valeur du timeline tl' à l'étape précédente (variable $tl'^{(-1)}$). Cette méta-contrainte m peut être utilisée au sein d'une seule contrainte globale modélisant de manière compacte l'expression $\forall i \in [2..h]$, $tl_i \neq tl'_{i-1}$. Pour les modèles faisant intervenir uniquement des contraintes dynamiques stationnaires, cette approche a permis de diminuer drastiquement le temps de création des modèles et leur taille en mémoire [209]. *

3.2 Le cadre CTA : hybridation contraintes/automates

Le cadre CNT présenté à la section précédente constituait une première tentative pour marier programmation par contraintes et gestion de systèmes dynamiques à événements discrets. Au final, un CNT est une forme de CSP dynamique dans lequel des éléments de base qui reviennent dans ces systèmes sont explicités, en particulier les notions d'horizons et de références temporelles. Le cadre est applicable aussi bien pour trouver une solution d'un problème de planification que pour valider des propriétés sur les évolutions possibles d'un système.

A l'issue de ces travaux, le bilan n'était cependant pas complètement positif à plusieurs titres. Du point de vue des modèles, certes l'expressivité du cadre CNT était intéressante, mais par contre le travail de modélisation n'était pas toujours aisé pour un utilisateur non expert de la programmation par contraintes. Pour traiter ce point, il aurait été possible d'ajouter des couches de modélisation de plus haut niveau. Du point de vue de l'efficacité algorithmique, des expérimentations supplémentaires ont révélé qu'il existait encore un ordre de grandeur en termes de performances entre les algorithmes génériques du cadre

CNT et un planificateur anytime spécifique que nous avions été amenés à construire pour
* le compte du CNES, pour traiter la mission spatiale HotSpot [205] déjà évoquée au chapitre 2. Une raison à cela est que malgré les efforts réalisés pour encoder de manière compacte toutes les contraintes dynamiques dites stationnaires, les modèles CNT contenaient toujours autant de variables dynamiques que le nombre maximal d'étapes dans les horizons considérés, avec pour chaque variable dynamique le besoin de maintenir un domaine de valeurs.

Plutôt que de s'enferrer dans la définition de techniques de compactage de modèles du cadre CNT, nous avons mis de côté le cadre CNT pour explorer une nouvelle piste, consistant à revenir à une représentation compacte de la partie dynamique des modèles. Dans la nouvelle proposition, cette partie dynamique est représentée plus simplement sous forme d'automates, ce qui revient à encoder uniquement la fonction de transition du système, et donc à revenir en quelque sorte à des modèles état-action (ou état-événement). En parallèle, un des objectifs était de conserver la partie statique des modèles CNT, utile en pratique pour exprimer des connaissances sur l'évolution globale des systèmes dynamiques. Le nouveau cadre qui a été introduit pour succéder au cadre CNT est le cadre CTA pour
* *Constraint-based Timed Automata* (automates à contraintes temporisés [223])

3.2.1 Le modèle CTA

A l'instar du cadre CNT, un modèle dans le cadre CTA se découpe en une partie décrivant la dynamique du système à contrôler et une partie définissant des connaissances globales (statiques) sur le système. Une illustration de l'organisation du cadre CTA, explicitée par la suite, est fournie à la figure 3.3. Pour aider à la modélisation, la notion unifiée de timeline du cadre CNT a été partitionnée dans le cadre CTA entre la notion de variable d'état, permettant de décrire l'état courant du système considéré, et la notion d'événement faisant évoluer l'état courant du système à des instants discrets. Nous parlons dans la suite plutôt d'événement que d'action pour utiliser un vocabulaire automates et pour insister sur le fait que certains événements ne sont pas forcément contrôlables. Pour un problème de gestion de satellites d'observations, les variables d'état à considérer peuvent être le temps courant, l'identifiant de l'observation en cours, la présence d'une acquisition en mémoire à bord, le fait qu'une station sol soit visible depuis le satellite... Des exemples d'événements sont les événements (contrôlables) de début et fin d'acquisition, avec en paramètre de chacun de ces événements le numéro de l'acquisition enclenchée, les événements (incontrôlables) de début et fin d'éclairement des panneaux solaires par le soleil...

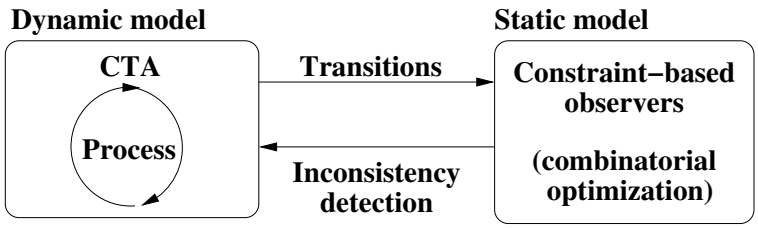


FIGURE 3.3 : Vue globale du cadre CTA et des observateurs.

Partie dynamique Nous avons défini la partie dynamique du cadre CTA de manière relativement classique, comme un tuple $(\mathbf{S}, \mathbf{t}, \mathbf{I}, \mathbf{E}, \mathbf{\Delta})$ avec S un ensemble de variables d'état, $t \in S$ une variable spéciale représentant le temps courant, I un ensemble d'instanciations initiales des variables de S, E un ensemble de types d'événements, et Δ un modèle d'évolution par passage du temps. Les éléments E et Δ sont décrits plus précisément ci-après.

Comme en planification classique, chaque type d'événement $e \in E$ est décrit par un triplet $(Param_e, Pre_e, Post_e)$ avec $Param_e$ un ensemble de variables correspondant aux paramètres des événements, Pre_e l'ensemble des préconditions de e (contraintes liant les paramètres de e et la valeur des variables d'état juste avant l'application de e), et $Post_e$ une fonction décrivant l'ensemble des effets de e (ensemble acyclique de contraintes unidirectionnelles spécifiant les changements d'instanciation des variables d'état juste après e, en fonction des paramètres de e et de la valeur des variables d'état juste avant e). Comme le cadre CNT, nous voulions que le cadre CTA reste un cadre ouvert en termes de modélisation, si bien que préconditions et effets des événements peuvent être décrits par des contraintes quelconques.

En plus des évolutions instantanées induites par des événements ponctuels, le cadre CTA modélise les évolutions induites par l'écoulement du temps, comme dans le langage PDDL+ [78] ou dans les automates hybrides [108]. Le cadre CTA modélise cet aspect non pas par une équation différentielle d'évolution mais plus directement par une fonction de transition Δ spécifiant l'état obtenu après une durée δ à partir de l'état courant à t, si aucun événement ne se produit entre t et $t + \delta$. Un exemple d'évolution d'état par écoulement du temps est le cas de l'évolution continue d'un niveau d'énergie en entre les dates t et $t+\delta$, en fonction d'une puissance courante p consommée à t. Une telle évolution est donnée par une équation de la forme $en(t+\delta) = en(t) + \delta \cdot p$. Dans le cas où le niveau d'énergie maximal stockable par le système est limité à une quantité $EnMax$, le modèle d'évolution avec le temps devient $en(t + \delta) = \min(en(t) + \delta \cdot p, EnMax)$.

Modèle d'arrivée des événements En plus des préconditions et des effets, un modèle CTA spécifie pour chaque événement e un ensemble **act** d'événements éventuellement activés par e dans le futur, après un certain délai. Des conditions peuvent être imposées pour restreindre les valeurs possibles des paramètres des événements activés, en fonction des paramètres de l'événement activateur et de l'état courant. En ce sens, le cadre CTA définit un modèle d'arrivée des événements. Par exemple, pour un satellite d'observation, chaque événement de début d'acquisition induit un événement de fin d'acquisition dans le futur, une fois écoulée la durée de l'acquisition, et chaque événement de rentrée en visibilité d'une station sol active de manière instantanée un événement d'enclenchement possible d'un nouveau vidage de données sur cette station. Dans un modèle CTA, des événements peuvent également être activés initialement, indépendamment de l'occurrence de tout autre événement. Par exemple, l'occurrence des événements de début et fin de créneaux de visibilité station est incontrôlable et indépendante des activités enclenchées par le satellite.

L'idée sous-jacente à l'introduction d'un modèle d'arrivée d'événements était double. Premièrement, certains cadres tels que les automates temporisés [4] permettent, de manière

détournée, de forcer l'arrivée de certains événements en jouant sur les gardes des transitions et sur des invariants d'état à satisfaire. De même, dans le langage PDDL, l'utilisation de *Timed Initial Literals* [67] ou de préconditions bien choisies permet de forcer indirectement l'occurrence d'événements. Nous souhaitions avoir une définition plus explicite dans les modèles. Deuxièmement, nous souhaitions pouvoir développer *in fine* des algorithmes rapides de recherche de plan dont le principe de base consiste à suivre le flot des événements activés et à prendre des décisions lorsque des choix se présentent concernant l'enclenchement effectif d'un événement et les paramètres de cet événement. Une telle approche rejoignait des implémentations de planificateurs spécifiques pour certaines missions spatiales, utilisant un principe mixte de *simulation-décision* : simulation chronologique du système à contrôler, passage de témoin à des heuristiques de décision lors des instants de décision, application des décisions prises et activations des événements ultérieurs induits par ces décisions... jusqu'à arriver à la fin de l'horizon. Ce principe de recherche en avant par *simulation-décision* est très efficace en pratique pour trouver rapidement des premiers plans solutions pour des problèmes sans blocage et sans état but, dans lesquels l'objectif est principalement d'optimiser les récompenses collectées sur l'horizon.

En réunissant tous ces éléments, nous avons défini dans le cadre CTA un plan comme une séquence d'événements datés telle que les événements de cette séquence sont d'une part applicables depuis l'état initial et d'autre part cohérents avec le modèle d'arrivée des événements. Plusieurs événements peuvent être associés à la même date, mais la séquence spécifie un ordre entre ces événements simultanés, afin que l'évolution du système soit définie de manière unique.

Partie statique Afin de conserver l'idée d'avoir une description statique du système en parallèle d'une description dynamique, un autre élément de modélisation a été ajouté au cadre CTA. Cet élément correspond à la notion d'*observateur*, qui permet d'imposer des exigences globales sur les trajectoires du système, i.e. sur la séquence des états parcourus en appliquant les événements du plan. La notion d'observateur est présente dans certains travaux sur les automates. Comme représenté à la figure 3.3, l'idée générale est qu'un observateur écoute à chaque étape les transitions réalisées par le système dynamique observé et assure qu'une certaine propriété est vérifiée, par exemple le fait qu'un instrument d'acquisition ne soit pas allumé et éteint plus de k fois sur la trajectoire. Les observateurs sont également utilisables pour imposer des contraintes limitant la combinatoire des choix à explorer. Par exemple, un observateur peut être utilisé pour calculer une borne supérieure ub sur la meilleure valeur d'un plan qui étend le plan courant. Si l'observateur montre que $ub < best$ avec $best$ la valeur du meilleur plan déjà connu, alors il renvoie une incohérence, indiquant ainsi à la partie dynamique que chercher à étendre le plan courant est inutile. Les observateurs du cadre CTA ont été formalisés en utilisant plusieurs types de modèles : modèles de type programmation par contraintes, modèles SAT, modèles de type programmation linéaire et programmation linéaire en nombres entiers. Pour produire la borne ub, l'observateur peut utiliser un modèle statique qui résulte d'une simplification du modèle de la dynamique complète : par exemple, il n'est pas rare qu'un problème de planification impliquant plusieurs engins puisse être relaxé comme un problème de tournée de véhicules formalisable en programmation linéaire en nombres entiers.

Techniquement, un observateur o correspond à un tuple $(S_o, I_o, E_o, \Delta_o, P_o)$. Le premier élément S_o est un ensemble de variables d'état permettant à l'observateur de maintenir un état interne. Le second élément I_o donne une valeur initiale à l'état interne de l'observateur. Les troisièmes et quatrièmes éléments E_o et Δ_o associent respectivement à toute transition par événement instantané et à toute transition par passage du temps effectuées sur la partie dynamique un nouvel état interne de l'observateur en fonction de son état interne précédent. Enfin, le dernier élément P_o, appelé la *fonction de cohérence* de l'observateur, est celui qui assure la fonction principale de l'observateur. En toute généralité, P_o : $\mathit{Trajs}(S_o) \rightarrow \{0,1\}$ est une fonction associant la valeur 0 ou 1 à toute séquence décrivant les états internes successifs de l'observateur : valeur 0 si l'observateur juge que la trajectoire courante du système est incohérente, valeur 1 sinon.

Dans le cas d'un observateur à base de contraintes, P_o est décrit par un ensemble de variables V_o, un ensemble C_o^S de contraintes de base portant sur V_o, et un ensemble C_o^D de contraintes portant sur S_o et V_o. A chaque fois qu'un nouvel état s_o de l'observateur est obtenu, on instancie les contraintes de C_o^D avec ce nouvel état et les contraintes obtenues sont ajoutées si besoin au modèle de contraintes courant de l'observateur. Ainsi, au fur et à mesure des modifications de l'état interne de l'observateur sur sa trajectoire, et donc au fur et à mesure de l'évolution du système dynamique, des contraintes sont accumulées dans le modèle de l'observateur qui peut donc faire des raisonnements de plus en plus avisés. L'observateur renvoie une incohérence lorsqu'il juge que son modèle de contraintes courant est incohérent.

3.2.2 Algorithmes

Les algorithmes développés au-dessus du cadre CTA reprennent le mécanisme de *simulation-décision* évoqué précédemment, réalisant une recherche en avant (chronologique). Cette recherche en avant se fait sur la base d'un élément appelé l'échéancier courant, qui contient tous les événements activés dans le futur et leurs dates d'arrivée. Cet échéancier contient au départ l'ensemble des événements activés depuis l'état initial indépendamment de tout autre événement. Si le temps courant est t et si le prochain temps associé à un événement dans l'échéancier est t', alors la recherche en avant effectue en premier lieu une transition par le temps d'une durée $t' - t$ si $t' > t$. Un événement e est ensuite choisi parmi les événements de l'échéancier datés à t'. Cet événement est retiré de l'échéancier. Une instanciation p des paramètres de e est sélectionnée parmi les instanciations satisfaisant les préconditions de e. L'événement instancié $e(p)$ est alors appliqué et l'état est modifié comme spécifié par les effets de e. On ajoute enfin à l'échéancier l'ensemble des événements activés par l'application de $e(p)$.

Au-dessus du mécanisme de base de recherche en avant, les observateurs tentent d'élaguer au plus tôt l'espace de recherche. Ainsi, chaque fois qu'une transition par événement ou par passage du temps se produit, chaque observateur actualise son état interne et détecte ou non une incohérence. Du fait de leur utilisation à chaque étape de la recherche, les observateurs effectuent des raisonnements rapides, non nécessairement complets, ce qui signifie que certaines situations d'incohérence peuvent éventuellement ne pas être détectées. Pour un observateur o, la fonction de cohérence P_o indiquant si la trajectoire courante du système est incohérente peut reposer sur de la propagation de contraintes

lorsque l'observateur est écrit en programmation par contraintes [234], sur des utilisations de la relaxation linéaire lorsque l'observateur est décrit par un programme linéaire en nombres entiers [168], ou sur le principe de résolution unitaire pour un observateur défini par des clauses SAT.

* Comme pour le cadre CNT, nous avons proposé deux types d'algorithmes [223], avec un algorithme complet utilisé pour trouver des solutions optimales, et un algorithme anytime utilisé pour trouver rapidement de bonnes solutions dans un contexte d'utilisation en ligne. La version complète réalise une recherche arborescente en profondeur d'abord avec backtrack ; cette version est applicable si les domaines de valeurs des variables définissant les paramètres des événements sont finis (les variables d'état peuvent par contre avoir un domaine contenant un nombre infini de valeurs). La version anytime est un glouton stochastique itéré, dans lequel les choix réalisés ne sont jamais remis en question (aspect *glouton*), dans lequel ces choix sont faits en suivant une heuristique bruitée (aspect *stochastique*), et dans lequel la recherche est relancée à partir d'un plan vide un certain nombre de fois (aspect *itéré*).

3.2.3 Implémentation

Dans l'outil que nous avons développé pour manipuler des modèles CTA, nous avons mis en place des interfaces pour pouvoir écrire des observateurs en programmation par contraintes sur la base des outils Choco [49] et IBM ILOG CpOptimizer [115], des observateurs en programmation linéaire et linéaire en nombres entiers sur la base des outils CPLEX [115] et LpSolve [147], et des observateurs SAT ou pseudo-booléens sur la base de l'outil SAT4J [235]. L'utilisation d'observateurs a été testée avec succès sur des benchmarks de planification classiques. Les expérimentations réalisées ont notamment montré que l'utilisation d'observateurs permettait d'obtenir des gains en temps de calcul intéressants par
* rapport aux meilleurs planificateurs optimaux existants [223], à nouveau grâce à la connaissance additionnelle exprimée dans la partie statique du modèle. Les algorithmes anytime du cadre CTA ont également permis d'obtenir sur la mission spécifique HotSpot évoquée précédemment des temps de production de premières solutions comparables avec les temps offerts par un planificateur spécifique optimisé développé pour le CNES.

3.3 Bilan

Deux cadres à base de contraintes ont été définis pour modéliser et résoudre des problèmes de décision associés aux systèmes dynamiques à événements discrets. Ces cadres marient des raisonnements sur des modèles dynamiques avec des raisonnements sur des modèles statiques. Les premiers travaux réalisés sur le cadre CNT, qui correspond à une forme de CSP dynamique, ont été laissés de côté pour laisser place à des développements sur le cadre CTA, qui utilise conjointement un modèle dynamique à base d'événements discrets pour décrire la partie dynamique des systèmes, et un modèle statique pour représenter des connaissances plus globales et calculer des bornes. Continuer des développements autour du cadre CTA correspond à une des perspectives de recherche (voir le chapitre 7).

Chapitre 4

Ordonnancement par recherche locale à base de contraintes

Les travaux présentés au chapitre précédent pour faire du contrôle de systèmes se situent dans la lignée d'une approche de type planification d'actions, dans laquelle le nombre d'actions à exécuter pour mener à bien une mission et le type de ces actions ne sont pas nécessairement connus a priori. Plus récemment, nous avons exploré une autre voie, partant du constat que pour beaucoup de problèmes que nous rencontrions en pratique dans le domaine spatial et dans le domaine robotique, l'ensemble des tâches candidates pour faire partie des plans pouvait facilement être délimité a priori.

Cette autre voie que nous avons explorée se situe dans la lignée d'une approche de type ordonnancement avec tâches optionnelles, dans laquelle l'objectif est de sélectionner des tâches à réaliser parmi un ensemble de tâches candidates et d'ordonnancer les tâches choisies de manière à satisfaire des contraintes sur le temps et sur les ressources. Pour explorer cette voie, nous nous sommes tournés vers l'utilisation de techniques de recherche locale [1], dont le principe consiste à réaliser des successions de mouvements locaux sur un ordonnancement courant, par exemple des mouvements d'ajouts et de retraits de tâches, dans le but d'améliorer progressivement le meilleur plan connu. Plusieurs raisons ont motivé ce choix, parmi lesquelles la capacité de ces techniques à fournir rapidement des plans de bonne qualité et leur capacité à passer à l'échelle sur des problèmes de grande taille. Nous souhaitions enfin conserver une approche à base de contraintes, afin de bénéficier à la fois de la souplesse de modélisation et de l'efficacité des techniques de raisonnement spécifiques à chaque contrainte.

L'ensemble des points précédents nous a progressivement conduits à travailler sur une approche utilisant le cadre théorique CBLS (*Constraint-Based Local Search* [107]), qui permet justement de mêler ordonnancement, recherche locale et programmation par contraintes. Les travaux que nous avons réalisés se sont traduits par le développement d'une nouvelle bibliothèque CBLS, la bibliothèque *InCELL* (*Invariant-based Constraint EvaLuation Library*), que nous avons introduite dans [212]. Cette bibliothèque a été appliquée à plusieurs problèmes, notamment trois problèmes de planification pour des satellites [212, 197, 224] et deux problèmes de planification multi-robots impliquant des robots terrestres et des robots aériens [116, 199]. Elle s'est enrichie au fur et à mesure du traitement de ces missions. Divers mécanismes de recherches locales avaient déjà été proposés

dans la communauté planification, avec des planificateurs comme ASPEN [229] fonctionnant par réparation itérative de plans, comme FF [110] utilisant du *enforced hill-climbing*, comme LPG [88] utilisant de la recherche locale stochastique, ou comme DAE [25] utilisant des algorithmes évolutionnaires. La différence avec ces planificateurs est que nous sommes partis sur une approche CBLS, avec un parfum plus ordonnancement que planification.

Dans ce chapitre, nous rappelons tout d'abord les fondements de l'approche CBLS (section 4.1). Nous décrivons ensuite le cadre InCELL que nous avons défini (section 4.2), les travaux effectués pour traiter des contraintes temporelles complexes (section 4.3), et enfin des travaux antérieurs sur la combinaison entre techniques de recherche locale et techniques de propagation de contraintes (section 4.4). Nous désignons par la suite sous le nom d'InCELL à la fois le cadre développé pour faire de la recherche locale à base de contraintes et l'implémentation de ce cadre sous la forme d'une bibliothèque.

4.1 Approche CBLS

InCELL est une nouvelle mise en œuvre du paradigme de la recherche locale à base de contraintes ou *CBLS* (*Constraint-Based Local Search* [107]), introduit dans les travaux fondateurs sur l'outil Localizer [158]. En CBLS, les modèles sont définis par des variables de décision, des contraintes et des critères, comme en programmation par contraintes classique. Une spécificité des modèles CBLS est qu'ils reposent sur l'utilisation d'éléments appelés des *invariants*. Ces derniers correspondent à des contraintes unidirectionnelles de la forme "$x \leftarrow exp$", avec x une variable appelée la sortie de l'invariant, et exp une expression fonction de variables appelées les entrées de l'invariant. Un invariant tel que "$x \leftarrow sum(i \in [1..N])\, y_i$" exprime par exemple que la valeur de la variable de sortie x doit être égale en permanence à la somme des valeurs des variables d'entrée y_i. L'ensemble des variables et des invariants d'un modèle CBLS définit une structure orientée dans laquelle les sorties d'un invariant peuvent jouer le rôle d'entrées pour d'autres invariants. Cette structure, qui doit rester acyclique afin qu'une variable ne soit pas une fonction d'elle-même, est appelée *le DAG des invariants* (DAG = *Directed Acyclic Graph*). Elle représente le graphe des expressions utilisées dans le modèle. La figure 4.1 donne un modèle CBLS ainsi que le DAG d'invariants qui lui est associé.

Variables de décision :
var$\{bool\}\, b$
var$\{int\}\, x \in [0..10]$
var$\{int\}\, y \in [2..5]$

Invariants :
var$\{int\}\ \ z \leftarrow ite(b, x, y)$
var$\{int\}\ \ t \leftarrow (x - y)$
var$\{bool\}\ \ u \leftarrow (z < t)$
var$\{int\}\ \ v \leftarrow (t + 2)$

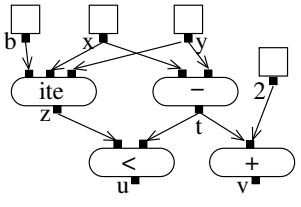


FIGURE 4.1 : Exemple de modèle CBLS ($ite(b, x, y)$ signifie "if b then x else y")

En CBLS, l'espace de recherche est exploré de manière beaucoup plus libre que dans une approche de recherche arborescente avec backtrack. Lors de la recherche d'une solution d'un modèle CBLS, toutes les variables du problème sont toujours instanciées, c'est-à-dire que l'approche manipule des instanciations complètes des variables. A chaque étape de la recherche, un mouvement local est réalisé en réinstanciant certaines des variables de

décision du problème. Ces variables correspondent à des racines du DAG d'invariants. Suite aux réinstanciations, les invariants impactés par les modifications sont réévalués dans un ordre qui doit être un ordre topologique du DAG d'invariants, de telle sorte qu'un invariant ne soit pas réévalué avant ses parents dans le DAG. L'originalité de l'approche est que, au même titre qu'il existe des algorithmes de propagation de contraintes spécifiques pour raisonner sur des contraintes spécifiques en programmation par contraintes classique, il existe des procédures spécifiques pour réévaluer très rapidement chaque type d'invariant. Sur l'exemple précédent "$x \leftarrow sum(i \in [1..N])\, y_i$", en cas de changement sur une entrée y_i pour un $i \in [1..N]$, x peut être réévalué incrémentalement en lui ajoutant la différence entre la nouvelle valeur de y_i et l'ancienne valeur de y_i. Cela permet de ne pas recalculer entièrement toute la somme et d'obtenir une réévaluation de la sortie x en temps constant. Plus généralement, les invariants permettent de manipuler efficacement et incrémentalement plusieurs types de contraintes et de critères d'optimisation. Dans la littérature CBLS, d'autres mécanismes ont également été proposés, par exemple des invariants maintenus au sein de contraintes dites *différentiables*, capables de fournir rapidement l'impact de la réinstanciation d'une variable sur le degré de violation d'une contrainte.

Au début de nos travaux et pendant nos travaux, plusieurs outils CBLS avaient déjà été ou ont été développés, depuis les travaux fondateurs sur Localizer [158]. Parmi ces outils, on trouve notamment COMET [107], iOpt [265], LocalSolver [19], Kangaroo [169] et OscaR.cbls [135]. Diverses raisons ont motivé la définition d'une nouvelle bibliothèque CBLS. Premièrement, parmi les outils cités précédemment, seul OscaR.cbls est accessible et libre d'utilisation. Deuxièmement, certains de ces outils, comme LocalSolver [19], sont dédiés au traitement de problèmes d'allocation, et non au traitement de problèmes d'ordonnancement. Troisièmement, nous souhaitions pouvoir maîtriser complètement une bibliothèque utilisée au cœur de nos planificateurs. Quatrièmement, nous voulions disposer d'un outil exploitable dans un contexte de décision embarquée, avec des contraintes d'implémentation que les outils CBLS existants ne garantissaient pas forcément. Cinquièmement, nous souhaitions pouvoir traiter des contraintes d'ordonnancement complexes sur le temps et les ressources, notamment des contraintes temporelles dites *time-dependent* non directement présentes dans les outils existants, ainsi que des contraintes sur des états évoluant continuement au cours du temps. Sixièmement, nous souhaitions pouvoir définir des modèles d'ordonnancement avec les notions d'intervalles, de séquences d'intervalles, d'intervalles avec plusieurs alternatives de réalisation... comme dans un outil tel que IBM ILOG CpOptimizer. Pour toutes ces raisons, nous avons introduit dans [212] une nouvelle mise en œuvre * des principes CBLS : le cadre InCELL (*Invariant-based Constraint EvaLuation Library*).

4.2 Cadre InCELL

InCELL est une instanciation des principes généraux de la recherche locale à base de contraintes. Plus spécifiquement, InCELL est une bibliothèque d'évaluation d'expressions, de contraintes et de critères, dédiée en particulier aux problèmes d'ordonnancement complexes. Cette bibliothèque n'est pas directement équipée d'algorithmes génériques de recherche locale, ce qui signifie que les mouvements locaux à utiliser pour faire évoluer une instanciation complète sont à définir au-dessus de la bibliothèque InCELL, de même

que les méta-heuristiques servant à guider l'enchaînement des mouvements locaux (recuit simulé [126], recherche tabou [98], algorithmes génétiques [99]...). Pour les parties mouvements locaux et méta-heuristiques, nous avons essentiellement utilisé lors du traitement d'applications des mécanismes ad hoc, comme illustré à la figure 4.2. Nous reviendrons sur ce point au chapitre 7 traitant des perspectives. Dans ce qui suit, nous décrivons les caractéristiques principales de la bibliothèque InCELL.

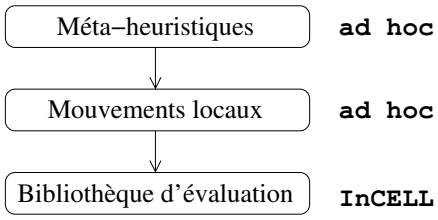


FIGURE 4.2 : Positionnement d'InCELL par rapport aux algorithmes de recherche locale, selon la méthodologie de décomposition décrite dans [71]

4.2.1 Variables InCELL

Les variables dans InCELL sont ou bien des variables dont la valeur peut être directement modifiée, ou bien des variables définies comme des expressions d'autres variables du problème. Les variables pouvant être directement modifiées correspondent aux racines du DAG d'invariants. Elles peuvent être ou bien des variables de décision ou bien des variables représentant des paramètres ajustables à la volée en fonction d'informations reçues. Comme dans les approches classiques de recherche locale, les variables du problème sont toutes instanciées en permanence. L'instanciation initiale des variables peut par exemple représenter un plan vide (aucune tâche sélectionnée pour faire partie du plan).

Nous avons conçu la bibliothèque InCELL pour qu'elle soit riche au niveau des types de variables manipulables. Les variables de base sont des variables dites atomiques, de type $\mathbf{var}\{bool\}$, $\mathbf{var}\{int\}$, $\mathbf{var}\{long\}$, $\mathbf{var}\{float\}$, $\mathbf{var}\{double\}$ et $\mathbf{var}\{T\}$, pouvant prendre respectivement des valeurs de type *bool*, *int*, *long*, *float*, *double* et T, avec T un type générique paramétrable ; une variable de type $\mathbf{var}\{T\}$ peut être vue comme une référence vers un objet de type T. Nous avons également introduit dans InCELL des variables dites composites, qui rassemblent plusieurs variables atomiques tout en offrant des fonctionnalités supplémentaires. Par exemple, nous avons défini dans InCELL un type permettant de décrire des séquences pouvant contenir des entiers compris entre 1 et N. Concrètement, ces séquences sont représentées par des variables atomiques $next[i]$ et $prev[i]$ décrivant respectivement les entiers qui suivent et qui précèdent chaque élément $i \in [1..N]$ dans la séquence. Ces séquences servent notamment à représenter un ordre d'enchaînement d'activités sur une ressource non partageable. Le type composite "séquence" offre en plus des fonctions d'ajouts et de retraits d'éléments dans la séquence. Nous avons aussi introduit des types de séquences plus généraux, avec notamment un type de variable composite permettant de décrire M séquences pouvant contenir des entiers compris entre 1 et N, et telles que chaque entier apparaît au plus une fois dans une des M séquences. Comme
* nous l'avons montré dans [199], de telles variables composites sont utiles pour décrire

de manière compacte des séquences de tâches effectuées sur un ensemble de machines. InCELL offre d'autres types composites prédéfinis, avec notamment un type composite pour décrire des *time-points*[1] (type utilisant deux variables atomiques pour représenter respectivement des dates d'occurrence au plus tôt et au plus tard), et un type composite pour décrire des *intervalles temporels* associés à la réalisation des tâches (type utilisant une variable booléenne pour décrire la présence de la tâche ainsi que deux *time-points* de début et de fin de la tâche).

Enfin, nous avons conçu la bibliothèque InCELL de telle sorte qu'elle soit extensible, dans le sens où elle permet de définir facilement de nouveaux types de variables. Cette conception modulaire nous a par exemple permis de traiter la mission décrite dans [197], * pour laquelle nous avons introduit de nouvelles variables atomiques permettant d'interfacer InCELL avec une bibliothèque externe de gestion de satellites agiles [146].

4.2.2 Invariants InCELL

Dans InCELL, nous avons formellement défini les invariants comme des triplets (I, O, f) avec I et O des séquences de variables correspondant respectivement aux entrées et aux sorties de l'invariant, et f une fonction $\mathbf{d}(I) \rightarrow \mathbf{d}(O)$ renvoyant une instanciation de O pour toute instanciation de I. Nous avons ainsi autorisé l'utilisation d'invariants à sorties multiples. Dès lors, le DAG des invariants dans InCELL est un DAG un peu particulier, qui contient pour chaque invariant du modèle un nœud possédant des ports d'entrée et des ports de sortie (autant de ports d'entrée que de variables dans I et autant de ports de sortie que de variables dans O). Ce genre de représentation a déjà été utilisé pour l'exemple de la figure 4.1.

Nous avons introduit dans InCELL un catalogue d'invariants prédéfinis. Ce catalogue comprend les invariants classiques en CBLS pour représenter des expressions arithmétiques (somme, produit, division, différence, somme pondérée, maximum, minimum...), des expressions logiques (et, ou, implication...) et des expressions ensemblistes (ensemble d'éléments triés, ensemble d'éléments satisfaisant une propriété donnée, ensemble d'éléments satisfaisant une propriété donnée et optimisant un critère donné...). Le catalogue contient également des invariants dits *combinatoires* en CBLS, l'exemple classique étant l'invariant *count*, qui possède comme entrées N variables pouvant prendre les valeurs 1 à M, et qui renvoie en sortie, pour tout $j \in [1..M]$, le nombre de variables qui prennent la valeur j. Un codage spécifique de cet invariant permet de réévaluer en temps constant la valeur des sorties en cas de modification sur une des entrées. Au fur et à mesure du traitement d'applications, nous avons introduit dans InCELL d'autres types d'invariants combinatoires, afin de gagner à la fois en temps de réévaluation suite à un mouvement local et en espace mémoire consommé par le modèle. On peut notamment citer :

- un nouvel invariant CBLS destiné à gérer l'ensemble des contraintes temporelles d'un modèle ; cet invariant, appelé l'invariant *TSTN* comme "*Time-dependent Simple Temporal Network*", prend en entrée des contraintes temporelles et renvoie en sortie les dates au plus tôt et au plus tard associées aux différents time-points du modèle ; l'invariant TSTN est capable de raisonner sur des contraintes temporelles

[1] Nous utilisons cet anglicisme dans la suite du document.

dites *simples*, de la forme $y - x \geq a$ avec x et y deux time-points et a une distance à respecter entre x et y ; la borne a peut être une expression elle-même définie par un invariant fonction d'autres variables, ce qui nous a par exemple permis de traiter
* dans [199] une mission multi-robots impliquant des temps de transition entre tâches fonctions des coordonnées géographiques choisies pour réaliser les tâches ; l'invariant TSTN est de plus capable de raisonner sur des contraintes temporelles plus complexes appelées des contraintes temporelles *time-dependent*, dans lesquelles la durée à respecter entre deux time-points est fonction de la date d'occurrence de ces time-points ; l'invariant TSTN a été optimisé pour effectuer le moins de recalculs possible lors d'une réévaluation, sur la base des techniques décrites à la section 4.3 ;

- un invariant CBLS appelé *StepProfile*, servant à maintenir des profils d'évolution résultants de l'application d'événements à effets instantanés ; voir la figure 4.3(a) pour une illustration ; cet invariant nous a par exemple permis de décrire des ressources cumulatives telles que la mémoire disponible à bord d'un satellite ; l'invariant prend en entrée un état initial, une séquence d'événements, les paramètres de ces événements et une description de l'effet instantané des événements ; il renvoie en sortie l'état juste après chacun des événements ;

- un invariant CBLS appelé *TimedProfile*, servant à maintenir des profils d'évolution résultants à la fois de l'effet instantané d'événements, et d'un processus d'évolution continue de l'état entre deux événements ; voir les figures 4.3(b) et 4.3(c) pour des illustrations ; cet invariant nous a par exemple été utile pour modéliser facilement les profils d'énergie évoqués à la section 2.2.2, évoluant de manière linéaire par morceaux avec saturation à un niveau d'énergie maximal ; l'invariant que nous avons défini prend en entrée un état initial, la date d'occurrence des événements, les paramètres de ces événements, une description de l'effet instantané des événements, et une description du processus d'évolution continue entre deux événements ; en sortie, il renvoie l'état obtenu juste avant et juste après chaque événement en appliquant les événements de manière chronologique, les événements associés à une même date étant départagés par des priorités ;

- un invariant servant à assurer une contrainte cumulative dite *robuste*, que nous ne
* détaillons pas ici mais que nous avons utilisée dans [199] pour une mission multi-robots dans laquelle la capacité d'une ressource cumulative devait être respectée quelles que soient les dates précises d'occurrence de tâches effectuées par des robots distincts ;

- un invariant à sortie graphique permettant de visualiser directement l'évolution des plans d'activités au cours d'une recherche locale.

Une originalité de nos travaux est que tous ces invariants peuvent être inclus dans un DAG d'invariants au même titre que n'importe quel autre invariant CBLS. En combinant un invariant StepProfile et un invariant TSTN, nous avons par exemple réussi à traiter
* la mission décrite dans [224] qui impliquait des interactions entre l'état d'un système, défini par l'invariant StepProfile, et le temps requis entre deux événements, décrit dans l'invariant TSTN.

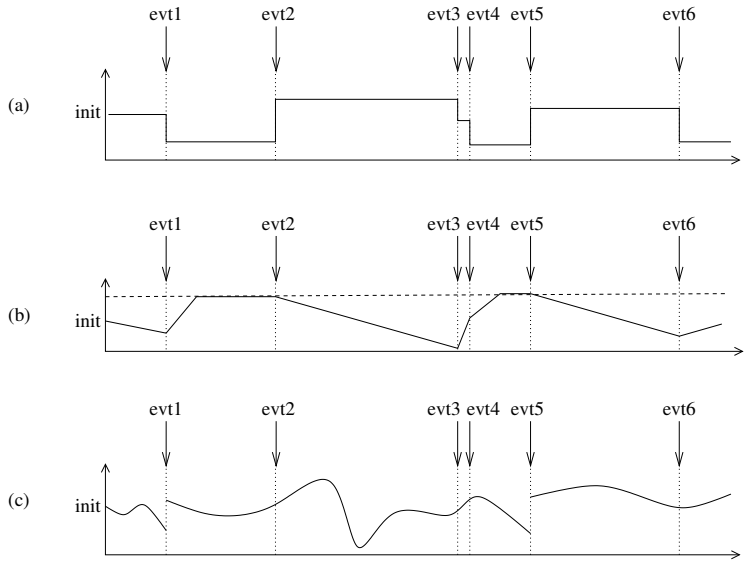


FIGURE 4.3 : Exemples de profils de ressources et d'états manipulables avec InCELL

A nouveau, nous avons conçu l'outil InCELL pour qu'il soit extensible, et des invariants supplémentaires peuvent facilement être ajoutés. Il suffit principalement de définir pour chaque nouvel invariant deux fonctions de base : une fonction d'initialisation, décrivant comment calculer initialement la valeur des sorties de l'invariant en fonction de la valeur de ses entrées, et une fonction de réévaluation, décrivant comment réévaluer de manière incrémentale la valeur des sorties suite à des modifications sur des entrées.

Sur les missions traitées, nous avons réussi à manipuler des modèles impliquant plusieurs dizaines de millions de variables et plusieurs millions d'invariants, ce sur une machine standard. Un extrait du DAG d'invariants que nous avons obtenu sur une mission du domaine spatial [212] est donné à la figure 4.4. *

4.2.3 Contraintes

Afin de dissocier les instanciations de variables qui sont des solutions de celles qui n'en sont pas, le cadre InCELL autorise chaque invariant à renvoyer un message d'incohérence, afin de stopper les réévaluations au plus tôt en cas de violation d'une contrainte. Par exemple, l'invariant TSTN renvoie directement une incohérence s'il n'existe aucune instanciation des time-points satisfaisant les contraintes temporelles du modèle. Il est également facile d'imposer des contraintes sur les profils d'évolution de l'état, par exemple des contraintes de respect de capacités maximales disponibles sur les ressources. Pour imposer des contraintes, InCELL contient également un invariant spécifique possédant une variable booléenne en entrée, pas de sortie, et renvoyant un message d'incohérence si l'entrée booléenne prend la valeur *false*. Notons que comme InCELL est juste une bibliothèque d'évaluation, elle ne fait pour l'instant pas d'ajout automatique de contraintes de précédence pour résoudre des conflits sur l'utilisation d'une ressource (pas de *precedence constraint posting* [193], pas d'utilisation de techniques de type *edge-finding* [156] ou *iterative flattening* [42]...).

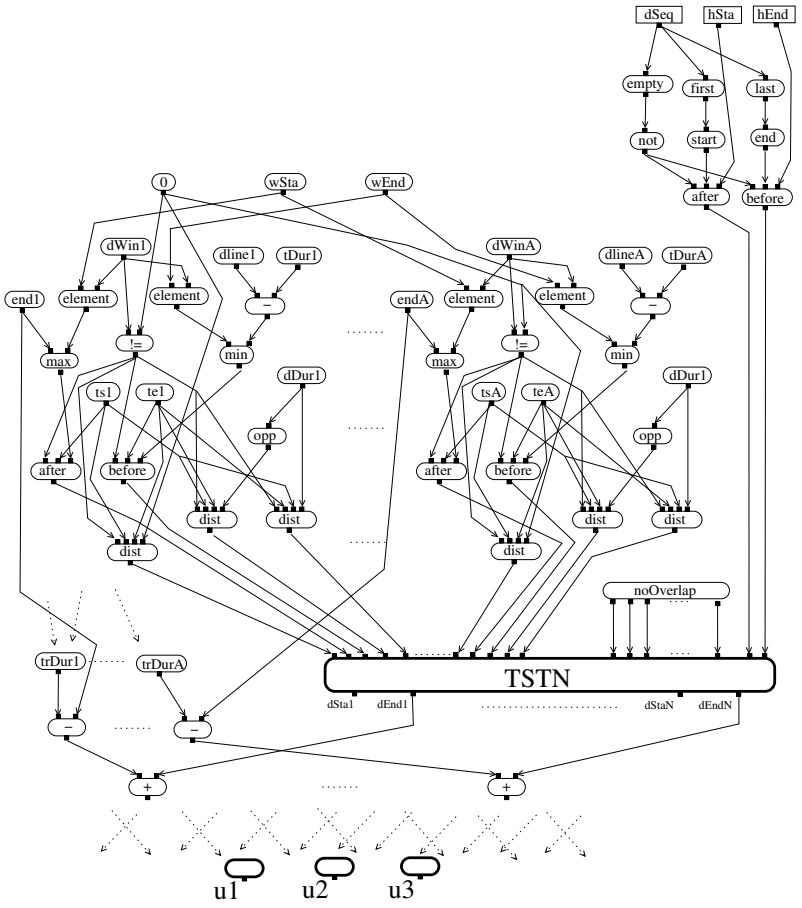


FIGURE 4.4 : Vue globale du DAG d'invariants associé au problème de planification décrit dans [212] ; les sorties des invariants feuilles représentent ici des critères d'optimisation ; les pointillés correspondent à des parties du DAG d'invariants qui ne sont pas présentées

*

4.2.4 Processus de réévaluation

La bibliothèque InCELL présente également des différences avec d'autres outils CBLS du point de vue du processus de réévaluation qu'elle utilise. Dans la littérature CBLS, deux grands types de stratégies sont utilisés pour réévaluer les invariants du modèle suite à une modification des variables de décision :

- une stratégie de réévaluation des invariants dans un ordre topologique du DAG d'invariants, utilisée par Localizer [158], COMET [107], LocalSolver [19] et OscaR.cbls [135] ;
- une stratégie de réévaluation appelée *mark-sweep* [114], utilisée par iOpt [265] et Kangaroo [169] ; cette stratégie marque d'abord l'ensemble des invariants potentiellement impactés par les modifications faites sur les variables racines du DAG d'invariants ; elle part ensuite des invariants feuilles devant être réévalués, et elle remonte dans le DAG si besoin pour demander des réévaluations d'invariants parents.

Dans InCELL, nous avons décidé d'utiliser la première de ces stratégies, mais contrairement à Localizer, COMET et OscaR.cbls, qui s'autorisent à avoir un DAG d'invariants

dynamique, nous avons décidé de manipuler uniquement des DAGs d'invariants statiques, comme dans LocalSolver [19], tout en étant capables de traiter des problèmes d'ordonnancement, chose que LocalSolver ne fait pas. Cela nous a permis de ne jamais avoir à payer un coût pour maintenir incrémentalement un ordre topologique du DAG d'invariants. Sans rentrer dans les détails techniques, nos invariants à sorties multiples nous ont permis de passer outre des raisons classiquement invoquées pour motiver l'utilisation d'un DAG d'invariants dynamique pour des problèmes d'ordonnancement.

4.2.5 Recherche locale avec InCELL

Pour faciliter la définition d'algorithmes de recherche locale au-dessus de la bibliothèque InCELL, cette dernière offre la possibilité après chaque réévaluation du DAG d'invariants ou bien d'accepter les dernières modifications réalisées (opération de *commit*), ou bien de revenir à l'état du DAG d'invariants avant cette modification (opération de *rollback*). Pour une mission donnée, la démarche d'utilisation de l'outil InCELL peut alors par exemple consister à tester des mouvements locaux pour explorer le voisinage de l'instanciation courante, avec une succession d'opérations d'évaluation et de *rollback*, puis à effectivement réaliser le meilleur de ces mouvements locaux avec une opération d'évaluation et de *commit*. Pour les cinq applications que nous avons traitées avec InCELL [212, 197, 224, 116, 199], * cinq algorithmes spécifiques de recherche locale ont été définis. Notre démarche a ainsi été proche d'une démarche de recherche opérationnelle dans laquelle les spécificités des problèmes rencontrés sont exploitées au maximum.

4.2.6 Insertion dans une architecture de décision embarquée

Le dernier point est que les développements réalisés ont pris en compte des contraintes d'implémentation relatives aux systèmes embarqués, par exemple des contraintes d'absence d'allocation dynamique de mémoire. Cela nous a permis dans [197] de réaliser effectivement * un prototype de planificateur bord reposant sur InCELL pour un satellite de surveillance de la Terre. Pour cette mission, un travail d'ingénierie a été effectué pour utiliser un unique modèle InCELL statique créé au début de la vie du satellite, et pour puiser au besoin dans les variables de ce modèle pour planifier en continu. Par exemple, le modèle InCELL illustré à la figure 4.4, de taille limitée, a pu être utilisé pour planifier en continu des activités de vidages de données sur un horizon glissant. Notons de plus qu'un des avantages offerts par l'approche CBLS est sa capacité à intégrer naturellement des perturbations sur l'état réel obtenu à l'exécution, puisqu'une perturbation des données d'entrée du problème de planification peut être vue comme un mouvement local comme un autre.

4.3 Gestion des contraintes temporelles *time-dependent*

Nous décrivons maintenant de manière plus précise les travaux que nous avons réalisés pour définir un invariant spécifique, à savoir l'invariant TSTN responsable de la gestion de toutes les contraintes temporelles d'un modèle InCELL. Les contraintes temporelles ont été traitées de manière spécifique car, à la différence des contraintes présentes de manière brute dans le DAG des invariants, elles ne sont pas nécessairement unidirectionnelles.

En interne, l'invariant TSTN introduit implique un processus de propagation permettant d'évaluer les dates au plus tôt et au plus tard associées aux activités.

4.3.1 Cadre des TSTNs

Pour gérer les aspects temporels dans InCELL, un point de départ était le cadre des STNs (*Simple Temporal Networks* [63]). Ce cadre considère des modèles définis par une conjonction de contraintes temporelles dites *simples*, qui correspondent à des contraintes de distance entre time-points de la forme $y - x \in [a, b]$ avec x, y deux time-points et a, b deux constantes. L'intérêt des STNs en pratique est lié au fait que plusieurs problèmes clés sur les STNs sont traitables en temps polynomial, par exemple le problème du calcul des dates au plus tôt et au plus tard associées à chaque time-point. Les STNs sont de plus souvent utilisés comme composant de base lors de la résolution de problèmes temporels plus complexes, comme des DTNs (*Disjunctive Temporal Networks* [246]).

Dans notre cas, nous souhaitions traiter des problèmes impliquant des contraintes temporelles *time-dependent*, déjà évoquées au chapitre 2. La première étape a donc été de proposer une extension des STNs que nous avons appelé TSTN pour *Time-dependent* * *STN* [213]. Cette extension couvre des contraintes temporelles pour lesquelles les distances minimales et maximales à respecter entre deux time-points x et y ne sont pas nécessairement constantes. Plus précisément, les contraintes manipulées dans les TSTNs prennent la forme $y - x \in [f(x, y), g(x, y)]$[2], c'est-à-dire que les bornes constantes utilisées dans le cas des STNs sont remplacées par des bornes variables dans le cas des TSTNs. De telles contraintes sont utiles pour modéliser des problèmes de *time-dependent scheduling* [44, 84], dans lesquels la durée d'une activité peut dépendre de sa date de début, ou des problèmes dans lesquels la durée de transition requise entre deux activités peut dépendre de la date à laquelle la transition est enclenchée. De telles contraintes apparaissent dans plusieurs domaines :

1. en logistique, lorsque la durée estimée pour aller d'un point A à un point B dépend de l'état de congestion du trafic, et ainsi de l'heure de début du transit de A à B :

2. en planification de production, où la durée requise pour réaliser une tâche à une date donnée dépend des ressources disponibles à cette date (main d'œuvre, machines...) ;

3. dans le domaine spatial, où la durée requise par un satellite pour se mouvoir d'un pointage vers un point A à un pointage vers un point B dépend de la date à laquelle la transition de A à B est réalisée (voir la figure 2.5 du chapitre 2).

Sur cette base, nous avons cherché à étendre les définitions classiques associées aux STNs. Comme dans les STNs, une solution d'un TSTN est une instanciation de tous les time-points qui satisfait toutes les contraintes temporelles du modèle, et un TSTN est dit cohérent si et seulement si il admet au moins une solution. De manière analogue à un STN, les contraintes $y - x \in [f(x, y), g(x, y)]$ manipulées dans un TSTN peuvent toujours être réécrites comme des ensembles de contraintes inégalité $y - x \geq dmin(x, y)$.

[2] Considérer des contraintes $y - x \in [f(x, y), g(x, y)]$ et non des contraintes $y - x \in [f(x), g(x)]$ a rendu plus facile la définition de méthodes de propagation pour les dates au plus tard associées aux time-points.

Informellement, $dmin(x, y)$ spécifie une distance temporelle minimale entre les événements associés aux time-points x et y. Nous avons appelé les contraintes time-dependent $y - x \geq dmin(x, y)$ des contraintes temporelles *t-simples*, par analogie avec les contraintes temporelles simples des STNs. De manière analogue à un STN toujours, un TSTN a pu être représenté graphiquement à l'aide de son *graphe de distance*, qui contient un arc $y \to x$ étiqueté par $-dmin(x, y)$ pour chaque contrainte temporelle t-simple $y - x \geq dmin(x, y)$. La seule différence avec un graphe de distance standard associé à un STN est que les étiquettes des arcs ne sont pas constantes dans le cas des TSTN.

Une notion importante que nous avons associée à une contrainte temporelle t-simple $y - x \geq dmin(x, y)$ est sa *fonction de retard*, définie par $delay(x, y) = x + dmin(x, y) - y$. Informellement, $delay(t, t')$ est le retard obtenu en y si le time-point x se réalise à la date t et si le time-point y se réalise à la date t'. Ce retard correspond à la différence entre la date minimale à laquelle y peut se réaliser $(t + dmin(t, t'))$ et la date à laquelle y se réalise (t'). Un retard strictement négatif correspond à une réalisation de y après la date minimale. Un retard strictement positif correspond à une réalisation de y avant la date minimale et donc à une violation de la contrainte temporelle. Un retard nul correspond à une réalisation de y pile à l'heure.

4.3.2 Propriétés des TSTNs

Cohérence locale d'une contrainte temporelle t-simple Partant de la définition des TSTNs, nous avons ensuite étudié à quel point les résultats disponibles sur les STNs pouvaient être étendus aux TSTNs, afin de les appliquer à la gestion de l'invariant TSTN. La première étape était de définir comment propager chaque contrainte temporelle t-simple prise individuellement, et plus précisément de définir l'impact de chaque contrainte sur les dates au plus tôt et au plus tard associées aux différents time-points du problème. Pour cela, nous avons généralisé les règles de réduction de domaine utilisées dans les STNs, qui pour une contrainte temporelle simple $y - x \geq c$ s'écrivent :

$$\mathbf{d}(y) \leftarrow \mathbf{d}(y) \cap [\min(\mathbf{d}(x)) + c, +\infty[\tag{4.1}$$

$$\mathbf{d}(x) \leftarrow \mathbf{d}(x) \cap]-\infty, \max(\mathbf{d}(y)) - c] \tag{4.2}$$

à des règles de réduction de domaine pour les TSTNs, qui pour une contrainte temporelle t-simple $y - x \geq dmin(x, y)$ s'écrivent :

$$\mathbf{d}(y) \leftarrow \mathbf{d}(y) \cap [earr(\min(\mathbf{d}(x))), +\infty[\tag{4.3}$$

$$\mathbf{d}(x) \leftarrow \mathbf{d}(x) \cap]-\infty, ldep(\max(\mathbf{d}(y)))] \tag{4.4}$$

avec *earr* et *ldep* deux fonctions définies par $earr(t) = \min\{t' \in \mathbf{d}(y) \mid t' - t \geq dmin(t, t')\}$ et $ldep(t') = \max\{t \in \mathbf{d}(x) \mid t' - t \geq dmin(t, t')\}$ respectivement. Informellement, si x et y sont vus respectivement comme les dates de début et de fin d'une transition, alors la règle 4.3 élague le domaine de y en calculant la date d'arrivée au plus tôt de la transition en y telle que la contrainte temporelle est satisfaite, en commençant cette transition au plus tôt en x (à la date $\min(\mathbf{d}(x))$), et la règle 4.4 élague le domaine de x en calculant la date de départ au plus tard depuis x telle que la contrainte temporelle est satisfaite en

arrivant en y aussi tard que possible (à la date $\max(\mathbf{d}(y))$).

Nous avons établi deux résultats principaux concernant ces règles de propagation :

1. elles établissent l'*arc-cohérence aux bornes*, ce qui signifie qu'après application des règles, les bornes temporelles associées à x et y participent à une instanciation qui satisfait la contrainte temporelle ;

2. pour des contraintes temporelles t-simples ayant une propriété de monotonie sur leur fonction de retard, n'importe quelle valeur restant dans le domaine de x (resp. y) après application des règles d'élagage possède un support dans le domaine de y (resp. x) ; la monotonie de la fonction de retard est une propriété naturelle qui exprime que plus tôt une activité commence, plus tôt elle finit ; cette propriété est vérifiée dans le cas des STNs classiques.

Concernant la façon dont *earr* et *ldep* sont calculés en pratique, plusieurs approches peuvent être considérées. Pour des contraintes temporelles simples $y - x \geq c$, une formulation analytique est possible (voir les règles 4.1 et 4.2). Des formulations analytiques peuvent aussi être obtenues pour plusieurs contraintes temporelles classiquement considérées en *time-dependent scheduling*. Dans le cas général, nous avons proposé l'utilisation d'une
* méthode itérative approchée reposant sur des techniques d'interpolation linéaire [213].

Cohérence globale d'un TSTN Pour les TSTNs, définis par un ensemble de contraintes
* temporelles t-simples, nous avons établi dans [213] trois résultats principaux :

1. comme pour les STNs, si toutes les contraintes d'un TSTN sont rendues arc-cohérentes aux bornes en utilisant les règles 4.3 et 4.4, alors l'ordonnancement qui instancie les time-points avec leurs dates au plus tôt est une solution du TSTN, de même que l'ordonnancement qui instancie les time-points avec leurs dates au plus tard ;

2. contrairement aux STNs, les valeurs non extrémales qui restent dans le domaine des time-points ne sont pas forcément globalement cohérentes, ce qui signifie que ces valeurs ne participent pas forcément à une solution du TSTN ; ce résultat tient même si la fonction de retard possède la propriété de monotonie mentionnée précédemment ;

3. si la fonction de retard est monotone et si les cycles du graphe de distance impliquent uniquement des contraintes temporelles simples (et non t-simples), alors le résultat classique des STNs redevient valide, *i.e.* toute valeur restante dans le domaine d'un time-point participe à une instanciation solution.

4.3.3 Résolution des TSTNs

Pour inclure des techniques de raisonnement sur les TSTNs dans InCELL, nous avons considéré un contexte dynamique dans lequel des contraintes temporelles peuvent être ajoutées ou retirées du problème. Cela se produit lorsque des activités sont ajoutées au plan courant ou retirées du plan courant lors d'une recherche locale. Nous nous sommes principalement intéressés à la manière de maintenir incrémentalement les dates au plus tôt et au plus tard associées à chaque time-point.

Propagation de contraintes Comme dans les travaux existants sur les STNs [40, 87, 239], nous avons utilisé des techniques de propagation de contraintes pour calculer les bornes min et max des time-points. Ces techniques appliquent les règles 4.3 et 4.4 pour réviser les contraintes temporelles jusqu'à parvenir à un point fixe.

Détection de cycles négatifs Nous avons ensuite cherché à généraliser des mécanismes utilisés sur les STNs pour détecter au plus vite l'incohérence d'un ensemble de contraintes temporelles. Dans un STN classique, l'incohérence est équivalente à l'existence d'un cycle de longueur négative dans le graphe de distance. Des techniques efficaces ont été proposées sur les STNs pour détecter de tels cycles [38, 40], en maintenant des éléments appelés des *chaînes de propagation*. Ces chaînes peuvent être vues comme des explications des bornes min et max courantes pour les différents time-points. L'intuition est que si un cycle de propagation $x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_n \rightarrow x_1$ est détecté pour les bornes min, alors cela signifie que la valeur minimale de x_1 a modifié la valeur minimale de x_2... qui a elle-même modifié la valeur minimale de x_n, qui a elle-même modifié la valeur minimale de x_1. En parcourant ce cycle de propagation un nombre suffisant de fois, le domaine de x_1 peut être entièrement élagué. Un résultat similaire est disponible pour les bornes max.

Nous avons montré dans [213] que pour un TSTN, la relation entre incohérence et * existence d'un cycle de propagation était plus complexe. En particulier, nous avons montré que l'existence d'un cycle de propagation dans un TSTN n'impliquait pas nécessairement une incohérence, une raison à cela étant que les réductions de domaine obtenues sur un TSTN en traversant un cycle de manière répétée peuvent devenir de plus en plus petites à cause de l'aspect *time-dependent*. Nous avons également proposé dans [213] plusieurs * conditions suffisantes permettant au résultat d'équivalence entre détection de cycle et incohérence d'être à nouveau valide.

Complexité Du point de vue de la complexité, nous avons étendu assez directement un résultat classique sur les STNs fournissant la complexité théorique de la propagation des contraintes temporelles sur un problème faisant intervenir n time-points et m contraintes. Nous avons montré que si l'existence d'un cycle de propagation impliquait une incohérence, alors il était possible d'établir l'arc-cohérence aux bornes en $O(nm)$ révisions de contraintes. Comme dans les STNs, la borne obtenue ne dépend pas de la taille du domaine des variables. Par contre, dans les TSTNs, même si le nombre de révisions de contraintes reste polynomial, la révision d'une contrainte ne se fait pas forcément en temps constant, à cause de l'aspect time-dependent.

Dépropagation des contraintes Les techniques de propagation de contraintes sont capables de gérer nativement les ajouts de contraintes et les renforcements de contraintes. Pour gérer les retraits de contraintes et les assouplissements de contraintes, nous avons réutilisé dans nos travaux des stratégies de dépropagation définies dans [239] pour les STNs. Ces stratégies permettent de recalculer les bornes minimales et maximales des variables de manière incrémentale. Plus précisément, elles évitent de réinitialiser les domaines de toutes les variables et de repropager l'ensemble des contraintes temporelles du modèle dès qu'une contrainte est supprimée ou assouplie. L'idée de base consiste à utiliser

les chaînes de propagation pour déterminer plus précisément les domaines de variables à réinitialiser et les contraintes à réviser en cas de relaxation du réseau temporel.

Ordre de révision des contraintes Nous avons proposé une dernière technique pour réduire le nombre de révisions de contraintes à réaliser. Cette réduction a été motivée par les TSTNs, pour lesquels une révision de contrainte peut être beaucoup plus coûteuse que pour les STNs, notamment lorsque les règles d'élagage 4.3 et 4.4 n'ont pas de formula-
* tion analytique. L'approche que nous avons proposée dans [211] a étendu une technique développée pour les STNs^{-}[87], une sous-classe des STNs dans laquelle toute contrainte temporelle doit pouvoir s'écrire sous la forme $y - x \geq c$ avec $c \geq 0$. La technique en question consiste à construire les composantes fortement connexes (SCCs) du graphe de distance associé au TSTN, et à réviser les contraintes selon un ordre topologique de ces SCCs. Nous avons maintenu un tel ordre topologique lors des ajouts et des retraits de contraintes temporelles en utilisant des algorithmes de calculs de SCCs disponibles pour des graphes dynamiques [103, 233].

Gestion des contraintes unaires Enfin, dans le schéma de propagation de contraintes utilisé, nous avons décidé de réviser les contraintes temporelles unaires $x \leq a$ et $x \geq b$ en premier car leur révision est rapide et car ces contraintes n'ont à être révisées qu'une seule fois. Un autre argument était que si les contraintes unaires avaient été traitées de manière classique comme des contraintes de distance par rapport à une position temporelle de référence x_0 (contraintes $x - x_0 \leq a$ et $x - x_0 \geq b$), alors le graphe de distance aurait contenu plusieurs arcs supplémentaires parasitant inutilement la définition des SCCs.

4.3.4 Inclusion dans InCELL

Toutes les techniques décrites précédemment (propagation de contraintes, détection de cycles de propagation, techniques de dépropagation de contraintes, propagation suivant les SCCs, traitement spécifique des contraintes unaires) ont été intégrées dans l'invariant TSTN de la bibliothèque InCELL. Cela nous a permis de représenter et de manipuler effectivement des TSTNs. Nous avons également interfacé des raisonnements génériques sur les TSTNs avec une bibliothèque de gestion de satellites agiles [146].

4.4 Recherche locale avec propagation de contraintes

Dans une toute autre direction, nous avons également proposé dans nos tous premiers
* travaux de recherche [203, 204], bien antérieurement au développement d'InCELL, une contribution sur la combinaison entre recherche locale et propagation de contraintes. Cette combinaison suivait une autre voie que la voie CBLS, en considérant des modèles de programmation par contraintes classiques, non limités à des modèles exprimés comme des DAGs d'invariants. L'idée de base était d'arriver à bénéficier à la fois de la liberté dans le parcours de l'espace de recherche offerte par la recherche locale et de l'efficacité de la propagation de contraintes. Un exemple d'algorithme de la littérature qui réalisait cela était l'algorithme DR (*Decision Repair* [122]), qui généralise des algorithmes classiques tels que CBJ (*Conflict-based BackJumping* [225]) et DBT (*Dynamic BackTracking* [94]).

Le fonctionnement de ces algorithmes est illustré à la figure 4.5. Ces algorithmes instancient progressivement les variables du problème afin de trouver une instanciation solution, et ils utilisent de la propagation de contraintes pour élaguer l'espace de recherche après chaque instanciation de variable. Mais alors qu'en cas de conflit une recherche arborescente classique repose sur du backtrack dit *chronologique* (remise en cause de la dernière instanciation de variable dans l'arbre de recherche), ces algorithmes exploitent une explication de l'incohérence courante pour déterminer sur quelles instanciations de variables revenir. L'algorithme DBT revient systématiquement sur l'instanciation de la dernière variable impliquée dans l'explication de l'incohérence courante, l'algorithme CBJ revient sur cette même instanciation mais en défaisant également les instanciations faites entre temps, et l'algorithme DR revient sur n'importe quelle variable impliquée dans l'explication d'incohérence courante. Ainsi, les variables ne sont pas forcément désinstanciées dans l'ordre de leur instanciation, ce qui implique par ailleurs de mettre en œuvre des techniques pour défaire intelligemment le travail fait par la propagation de contraintes. Au final, l'algorithme DR effectue une recherche locale dans l'espace des instanciations partielles guidée par les explications d'incohérence.

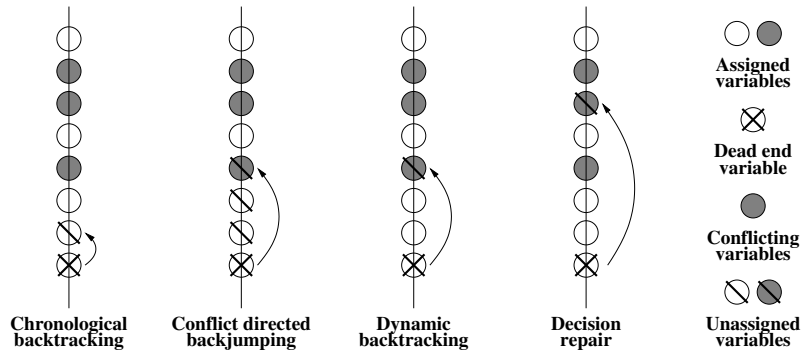


FIGURE 4.5 : Schémas d'instanciations et de désinstanciations de variables pour plusieurs algorithmes de recherche sur des instanciations partielles

Notre apport principal à l'algorithme DR a concerné l'étude d'heuristiques efficaces pour déterminer quelle variable désinstancier en cas de conflit. Dans [203, 204], nous avons * notamment étudié trois heuristiques de désinstanciation :

- désintanciation aléatoire, afin de diversifier la recherche ;
- déinstanciation de la variable qui efface le plus de valeurs dans les domaines des autres variables, afin de réviser d'abord les instanciations les plus contraignantes ; cette heuristique s'est avérée efficace sur des problèmes cohérents ;
- désinstanciation de la variable qui intervient le moins dans les justifications d'effacements de valeur, l'idée étant de conserver au maximum le cœur de l'incohérence ; cette heuristique s'est avérée efficace sur des problèmes incohérents ; nous avons également montré dans [204] que sous certaines conditions, cette heuristique de désinstanciation * était capable de prouver l'incohérence d'un problème, contrairement aux algorithmes classiques de recherche locale sur des instanciations complètes.

4.5 Bilan

Dans ce chapitre, nous avons présenté nos contributions concernant l'utilisation de techniques de recherche locale pour résoudre des problèmes de programmation par contraintes, en particulier des problèmes d'ordonnancement incluant des contraintes temporelles complexes, des contraintes sur des profils d'évolution, et des critères d'optimisation quelconques. Cette ligne de recherche s'est concrétisée par le développement de la bibliothèque InCELL, et elle correspond au point le plus actif dans nos développements en cours. Une des raisons à cela est que la bibliothèque InCELL permet de traiter effectivement des applications et de fédérer des travaux de recherche. Plusieurs pistes d'évolution sont envisagées pour l'approche CBLS InCELL. Nous décrivons ces pistes dans le chapitre 7 dédié aux perspectives.

Chapitre 5

Synthèse de contrôleur en contexte non déterministe

Les chapitres précédents ont présenté diverses réalisations associées à la problématique de la synthèse de plans et de la synthèse d'ordonnancements. Dans ce contexte, deux options ont été envisagées pour faire face aux incertitudes sur l'évolution réelle du système à contrôler et de son environnement :

- une option consistant à faire de la planification *réactive*, correspondant à calculer un plan à partir d'une version déterminisée de la réalité, et à réparer ce plan ou à replanifier en cas de changement du contexte ;
- une option consistant à produire un seul plan qui soit *robuste et flexible temporellement*, représenté sous la forme d'un POS (*Partial Order Schedule* [192]).

Nous présentons maintenant une autre catégorie de travaux, correspondant à des approches plus *proactives* dans lesquelles les incertitudes sur le système et son environnement sont traitées en amont. Dans ces approches, différentes situations rencontrables à l'exécution sont anticipées et une réponse adaptée à chacune de ces situations est précalculée. Les plans produits sont alors *conditionnels*, et l'ensemble des réponses à chaque situation est appelé une *politique de décision*. Dans ce sens, toujours avec comme ligne directrice l'ambition d'utiliser la Programmation Par Contraintes (PPC), nous avons initié des travaux en synthèse de politique pour traiter spécifiquement des problèmes de décision dans lesquels l'incertitude est modélisée par une relation de transition non déterministe. Cette dernière spécifie de manière booléenne les évolutions possibles du système à contrôler en fonction des décisions prises. Nous décrivons le contexte de ce travail (section 5.1), les modèles considérés (section 5.2), les méthodes de résolution introduites (sections 5.3 à 5.5) et un cas d'application traité (section 5.6).

5.1 Contexte et motivation

La motivation initiale de ce travail était de pouvoir construire des contrôleurs logiciels capables de gérer des sous-systèmes embarqués à bord d'un satellite ou d'un avion. Au sens large, un contrôleur logiciel est un automate qui se trouve dans un état particulier à

une étape donnée, et qui reçoit des signaux d'entrée déclenchant éventuellement l'émission de signaux de sortie ainsi qu'un changement d'état de l'automate. Voir la figure 5.1 pour une illustration.

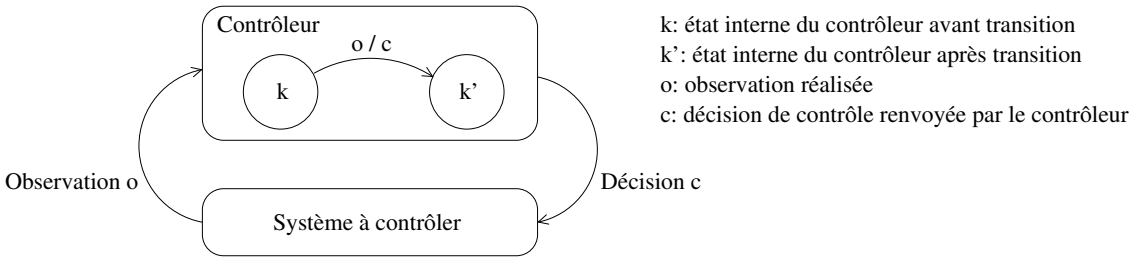


FIGURE 5.1 : Schéma classique d'interaction entre contrôleur et système à contrôler

La manière couramment utilisée dans l'industrie pour construire ces contrôleurs consiste à les spécifier directement sous la forme d'un ensemble de règles spécifiant les sorties à émettre en fonction des entrées reçues. Il est ensuite nécessaire de valider des propriétés garantes du bon fonctionnement du contrôleur, soit par simulation, soit par validation formelle, ou par ces deux moyens. Pour la validation formelle, la démarche consiste à confronter la spécification du contrôleur avec les propriétés à vérifier. Si l'une de ces propriétés n'est pas satisfaite, la spécification du contrôleur est revue, les propriétés sont vérifiées à nouveau... jusqu'à converger vers un contrôleur valide.

Dans les travaux que nous avons réalisés, nous avons cherché à proposer des méthodes permettant d'éviter ces aller-retours entre spécification du contrôleur et validation. Plus précisément, nous avons proposé des méthodes de synthèse de contrôleur [230], capables de synthétiser automatiquement des contrôleurs à partir des équations d'évolution du système à contrôler et des propriétés devant être satisfaites par ce système. Les contrôleurs obtenus sont alors valides par construction.

5.2 Modèles considérés et propriétés à satisfaire

Modèle du système à contrôler Dans l'approche de synthèse de contrôleur définie
* dans [214], nous avons considéré le cas général des systèmes partiellement observables, c'est-à-dire pour lesquels certaines parties de l'état peuvent rester cachées pour le contrôleur. Nous avons par ailleurs considéré uniquement des systèmes markoviens, c'est-à-dire tels que l'état du système à une étape donnée dépend uniquement du dernier événement appliqué et de l'état juste avant ce dernier événement, et non de tout l'historique des événements. Formellement, nous avons considéré des systèmes à contrôler définis par :

- un ensemble de variables d'état S décrivant l'état courant du système ; une instanciation s de S est appelée un état ;
- un sous-ensemble de variables $O \subseteq S$ correspondant aux variables d'état observables ; une instanciation o de O est appelée une observation ; l'observation associée à un état s, noté $obs(s)$, correspond à la restriction de s aux variables de O ;
- un ensemble de variables de contrôle C par l'intermédiaire desquelles le contrôleur

peut agir sur le système ; une instanciation c des variables de C est appelée une décision de contrôle ;

- une relation d'initialisation $I(S)$ décrivant parmi les instanciations s de S celles qui correspondent à des états initiaux possibles ;
- une relation de faisabilité $F(S, C)$ qui contient les couples (s, c) formés par un état s et une décision de contrôle c faisable dans l'état s ;
- une relation de transition $T(S, C, S')$ qui contient les triplets (s, c, s') formés par un état s, une décision de contrôle c et un état s', et tels que s' est un état successeur possible obtenu en appliquant la décision c dans l'état s.

Dans les travaux réalisés, nous avons représenté les différentes relations (initialisation, faisabilité, transition) sous la forme de CSPs définis par des contraintes portant sur $S \cup C \cup S'$. L'ensemble S' correspond à une copie de S utile pour représenter l'état du système après application du contrôle. Nous avons de plus fait une hypothèse de normalisation selon laquelle pour tout état s et pour toute décision de contrôle c faisable dans l'état s, il existe au moins un état successeur s' possible.

Propriétés à satisfaire Dans nos travaux, nous avons considéré deux types de propriétés à satisfaire par les contrôleurs :

- des propriétés d'*atteignabilité*, exprimant que le contrôleur construit doit mener le système dans un état but pour tout état initial envisageable ; ces propriétés sont définies par une relation $G(S)$ (G comme *goal*) qui contient les états s qui satisfont la condition à atteindre ;
- des propriétés de *sûreté*, exprimant des conditions à satisfaire à toute étape par le système contrôlé ; ces propriétés sont définies par une relation $P(S)$ qui contient les états s qui satisfont les propriétés de sûreté.

Nous voulions que ces propriétés d'atteignabilité et de sûreté puissent porter sur des attributs observables ou non de l'état. Dans le cas où seules des propriétés de sûreté étaient considérées, l'objectif était de faire en sorte que le système reste dans une enveloppe de fonctionnement sûr, sur un horizon infini. Dans le cas où des propriétés d'atteignabilité étaient définies, le problème s'apparentait à de la planification non déterministe, plus précisément à de la planification conformante (*conformant planning* [241]) ou à de la planification contingente (*contingent planning* [186, 226]). Dans le premier cas, aucune observation n'est réalisée sur le système et un contrôleur correspond à un plan menant au but partant de n'importe quel état initial possible. Dans le second cas, un contrôleur prend la forme d'un plan conditionnel dont les branches sont étiquetées par les résultats des observations collectées au fur et à mesure de l'application du plan. Le problème s'apparentait de plus à de la planification *forte* (*strong cyclic planning* [52]), ce qui signifie que nous voulions que les propriétés d'atteignabilité soient satisfaites sur toutes les trajectoires d'états potentiellement suivies par le système.

L'étape suivante aurait été de considérer des exigences plus générales décrites par des formules de logique temporelle type LTL (*Linear Temporal Logic*) ou CTL (*Computational*

Tree Logic) [69], ce qui aurait par exemple permis de considérer des propriétés dites de *vivacité*, exprimant qu'une condition doit toujours pouvoir être à nouveau satisfaite partant de n'importe quel état atteignable par le système. De telles propriétés sont considérées dans des outils de synthèse de contrôleur issus de la communauté *model-checking*, notamment les outils Anzu [120] et RATSY [29], reposant tous deux sur les idées algorithmiques définies dans [191]. Elles sont également considérées dans des travaux récents portant sur la planification non déterministe [183]. Cependant, toutes ces approches traitent uniquement le cas des systèmes complètement observables. Mixer l'observabilité partielle avec la généralité de LTL ou CTL reste un point encore ouvert.

Modèle du contrôleur En planification non déterministe, une manière classique de procéder pour contrôler des systèmes markoviens et partiellement observables consiste à rechercher des contrôleurs à *mémoire complète*. Ces contrôleurs renvoient une décision à prendre en fonction de l'état de croyance courant [30], l'état de croyance étant défini comme l'ensemble de tous les états courants possibles étant donné l'ensemble de toutes les observations passées. Dans nos travaux sur la synthèse de contrôleur, un des objectifs était de définir des contrôleurs qui soient à la fois réactifs et embarquables. L'utilisation de contrôleurs à mémoire complète n'était dès lors pas complètement adaptée, puisque d'une part maintenir en ligne un état de croyance pouvait compromettre la réactivité, et d'autre part le nombre d'états de croyance à considérer, au pire exponentiel en le nombre de variables d'état, pouvait compromettre l'embarquabilité.

C'est pourquoi nous nous sommes tournés vers une autre approche, inspirée de travaux définis dans [32] pour faire de la planification non déterministe, et de travaux définis dans [157, 194] pour traiter des MDPs partiellement observables (POMDP [123]). Les travaux en question portent sur la synthèse de *contrôleurs à états finis*, qui utilisent une mémoire limitée pour enregistrer de l'information sur le passé. Plus précisément, un contrôleur à états finis maintient un état interne défini par un entier k compris entre 1 et N, avec N un paramètre correspondant à la taille mémoire du contrôleur. Cet entier est potentiellement beaucoup plus petit que le nombre total d'états de croyance possibles. Un contrôleur à états finis est alors décrit par une politique de décision π qui étant donné une observation o et la valeur courante k de l'état interne du contrôleur renvoie une décision de contrôle $c = \pi(o, k)$ à appliquer ainsi qu'un nouvel état interne k' du contrôleur. Dans notre cas, nous avons recherché des politiques π pouvant être partielles, c'est-à-dire pouvant ne pas être définies pour des observations o non atteignables. Nous avons de plus recherché des politiques déterministes, c'est-à-dire définissant au plus une décision par couple observation-état interne.

Notons qu'avec les approches classiques de type planification conformante ou planification contingente, il est toujours possible de construire un contrôleur à états finis. En planification conformante, il suffit d'utiliser pour N le nombre d'actions dans le plan trouvé, et de prendre comme état interne k du contrôleur le numéro de la prochaine action du plan à enclencher. En planification contingente, il suffit d'utiliser pour N le nombre total de nœuds dans le plan conditionnel solution et de prendre comme état interne k le numéro du nœud associé à la prochaine action à exécuter. Cependant, dans ces cas le nombre d'états internes du contrôleur est défini a posteriori, après calcul d'une solution,

et l'approche ne permet pas de contrôler le système sur un horizon potentiellement infini.

Une autre remarque est qu'il est possible de se ramener à la recherche de contrôleurs *sans mémoire*, définis par une politique π renvoyant une décision de contrôle $c = \pi(o)$ à appliquer en fonction de la dernière observation o réalisée. Pour cela, il suffit de faire passer artificiellement la partie état interne du contrôleur dans la partie système, en ajoutant une variable d'état observable x dans O pour représenter l'état interne du contrôleur et une variable de contrôle y dans C pour contrôler directement la valeur de x.

5.3 Formulation complète en PPC

5.3.1 Formulation en PPC pure : équations de satisfiabilité

Pour résoudre le problème de la synthèse de contrôleur à états finis, nous avons tout d'abord tenté de réutiliser directement des outils de programmation par contraintes existants. Cela a conduit aux travaux présentés dans [258], dans lesquels le problème de la synthèse est formulé comme un problème de programmation par contraintes pur. *

La formulation proposée dans [258] s'applique aux problèmes impliquant uniquement des propriétés de sûreté (pas de propriétés d'atteignabilité). Elle s'inspire de la forme des équations d'optimalité de Bellman utilisées dans un contexte probabiliste pour calculer une politique optimale d'un MDP [227]. Ces équations de Bellman peuvent être traitées avec des outils de programmation linéaire, et de manière analogue les équations de satisfiabilité que nous avons proposées sont traitables avec des outils de programmation par contraintes. Une vue générale du modèle de contraintes défini est fournie à la figure 5.2[1]. Le modèle introduit permet de ne raisonner que sur une seule tranche d'évolution du système, ou autrement dit il ne nécessite pas de dérouler explicitement les trajectoires du système à contrôler. Au niveau des variables, le modèle contient : *

- pour chaque observation o, une variable $\pi(o)$ représentant le contrôle à renvoyer lorsque l'observation o est réalisée ;
- pour chaque état s, une variable booléenne $r(s)$ indiquant si l'état s est considéré comme atteignable ou non avec la politique π utilisée.

La principale difficulté dans la définition du modèle concernait la manière de définir les états considérés comme atteignables. Dans l'absolu, calculer l'ensemble des états atteignables pour une politique donnée nécessite de dérouler les évolutions possibles du système. Plutôt que de faire cela, nous avons choisi d'introduire des contraintes pour juger de l'atteignabilité d'un état. Ces contraintes spécifient que tout état initial doit être considéré comme atteignable et que pour tout état s considéré comme atteignable et associé à une observation o, les états successeurs s' obtenus en appliquant $\pi(o)$ dans l'état s doivent tous être considérés comme atteignables ($r(s') = \mathit{true}$). Comme montré dans [258], le modèle obtenu en procédant de la sorte définit une forme d'atteignabilité dite *faible*, qui peut surestimer l'ensemble des états réellement atteignables en appliquant la politique, mais il a été prouvé que malgré cela l'approche restait à la fois *correcte* et *complète*. Cela signifie que d'une part les politiques produites avec le modèle de contraintes sont bien des *

[1] En toute rigueur, on considère dans [258] des propriétés de sûreté $P(S, C, S')$ portant sur les transitions. *

Conditions sur la politique
$\forall s \in \mathbf{d}(S),\ r(s) \rightarrow P(s)$ // respect des exigences
$\forall s \in \mathbf{d}(S),\ r(s) \rightarrow (\pi(obs(s)) \neq \bot)$ // existence d'une décision ($\bot \equiv$ politique indéfinie)
$\forall s \in \mathbf{d}(S),\ r(s) \rightarrow F(s, \pi(obs(s)))$ // applicabilité des décisions
Conditions sur l'atteignabilité faible
$\forall s \in \mathbf{d}(S),\ I(s) \rightarrow r(s)$
$\forall s \in \mathbf{d}(S), \forall s' \in \mathbf{d}(S),\ (r(s) \wedge T(s, \pi(obs(s)), s')) \rightarrow r(s')$

* FIGURE 5.2 : Problème de synthèse de contrôleur formulé directement en PPC [258]

solutions, et que d'autre part s'il existe une politique satisfaisant les exigences, alors elle est trouvée par la méthode, et s'il n'en existe pas alors le modèle de contraintes n'a pas de solution.

* Dans [255], le modèle de contraintes obtenu a permis de synthétiser un contrôleur d'un sous-équipement embarqué à bord d'un satellite. L'approche, qui a utilisé IBM ILOG CpOptimizer [115] comme solveur de contraintes, a fonctionné sur cette instance de taille raisonnable. Le passage à l'échelle sur des instances de plus grande taille a été plus problématique. En effet, le modèle de contraintes introduit contient une variable $r(s)$ par état s et une variable $\pi(o)$ par observation o, même pour les états et les observations qui ne sont pas atteignables. Le modèle contient ainsi un nombre de variables exponentiel en le nombre de variables dans S. Le modèle contient de plus un nombre de contraintes au moins quadratique en le nombre d'états, si bien que même créer le modèle peut devenir
* compliqué. Une autre tentative a également été faite dans [142] pour produire des politiques en raisonnant uniquement sur une seule étape d'évolution du système, mais avec les mêmes problèmes de passage à l'échelle.

Même si nous ne l'avons pas proposé concrètement, les problèmes impliquant des propriétés d'atteignabilité auraient aussi pu être modélisés tout en conservant un raisonnement uniquement sur une tranche d'évolution du système. Une différence essentielle est que pour traiter les propriétés d'atteignabilité, il aurait été utile d'introduire des variables entières $n(s)$ représentant le nombre d'étapes restantes avant d'atteindre la propriété à satisfaire lorsque la politique est appliquée en partant de l'état s. Le problème de passage à l'échelle des modèles aurait alors été encore plus critique.

5.3.2 Formulation en PPC quantifiée : raisonnement sur horizon fini

Pour traiter les problèmes faisant intervenir des propriétés d'atteignabilité, une autre stratégie aurait été de créer des modèles de contraintes dépliant les évolutions du système sur un horizon fini. Cette approche aurait été l'analogue de l'approche résolvant des problèmes de planification non déterministe via un encodage en formules booléennes quantifiées (QBFPLAN [232]).

Pour faire la même chose avec des contraintes, il aurait suffit de construire un planificateur raisonnant avec des CSPs quantifiées (QCSP [33]) ou avec des CSPs à quantification restreinte (QCSP+ [18]), et d'utiliser les diverses techniques disponibles sur ces types de CSPs : arc-cohérence quantifiée [33, 152, 170], transformation vers des formules booléennes quantifiées [85], *conflict-based backjumping* [86], réparation de solutions [247], parcours de la séquence de quantification de droite à gauche [262]... Formellement, un QCSP est

défini par deux éléments : un ensemble de contraintes C et une séquence de quantification $Q = Q_1x_1 \dots Q_nx_n$ où chaque Q_i correspond ou bien au quantificateur existentiel ($\exists$) ou bien au quantificateur universel ($\forall$). Un QCSP défini par $Q = \exists x_1 \forall x_2 \exists x_3 \forall x_4$ et $C = \{c_1, c_2\}$ doit être interprété comme "Existe-t-il une valeur pour x_1 telle que quelle que soit la valeur prise par x_2, il existe une valeur pour x_3 telle que pour toute valeur de x_4 les contraintes c_1 et c_2 sont satisfaites ?". Résoudre un QCSP signifie répondre oui ou non à la question précédente, et produire une stratégie gagnante en cas de réponse positive. En notant A_x l'ensemble des variables quantifiées universellement qui précèdent x dans la séquence de quantification, une telle stratégie gagnante est généralement définie comme un ensemble de fonctions $f_x : \prod_{y \in A_x} \mathbf{d}(y) \rightarrow \mathbf{d}(x)$ (une fonction par variable x quantifiée existentiellement). L'ensemble de ces fonctions peut être représenté par un élément classiquement appelé un arbre de politique (*policy tree*).

A partir des arbres de politique, il aurait été possible de se ramener à un contrôleur à états finis contenant autant d'états internes que de nœuds dans l'arbre de politique, mais le nombre d'états internes à considérer aurait de fait été fixé a posteriori. Un autre inconvénient est que l'approche aurait eu du mal à passer à l'échelle sur des problèmes de grande taille car en dépliant le problème, le nombre de variables et de contraintes devient linéaire en la taille de l'horizon. Enfin, l'approche aurait eu du mal à traiter facilement des propriétés de sûreté sur un horizon infini. Pour toutes ces raisons, nous avons préféré placer l'effort sur autre approche présentée dans la section suivante.

5.4 PPC incluse dans une synthèse de contrôleur en avant

Nous avons introduit dans [214] et [210] deux nouvelles méthodes pour la synthèse de contrôleur, avec une version générale applicable aux systèmes partiellement observables, et une version optimisée pour les systèmes complètement observables. Ces méthodes sont présentées respectivement dans les sections 5.4.1 et 5.4.2. Elles utilisent toutes deux un mécanisme dédié de recherche en avant dans l'espace des états atteignables, et elles délèguent les raisonnements sur les relations de base à un solveur de contraintes. *

5.4.1 Cas de l'observabilité partielle

Au moment de nos travaux, des techniques de synthèse de politique en avant avaient déjà été définies dans [23] pour des domaines de planification non déterministes avec observabilité partielle. L'approche, implémentée dans le planificateur MBP qui utilise des techniques de *model-checking*, produisait cependant des plans conditionnels et non des contrôleurs à états finis. Elle n'était dès lors pas adaptée à notre besoin.

D'autres travaux avaient été introduits dans [32] pour construire des contrôleurs à états finis, menant à un planificateur que nous appellerons par la suite BPG en référence aux noms des auteurs. Du point de vue des modèles, BPG utilisait une formalisation PDDL pour décrire les actions du contrôleur et leurs effets, et faisait certaines hypothèses restrictives du type : (1) pas de préconditions sur les actions, (2) actions avec effets déterministes uniquement (seul l'état initial était imparfaitement connu), et (3) objectif d'atteindre un état but mais pas forcément de s'arrêter dans un état but. Du point de vue des algorithmes,

ces travaux réalisaient une transformation vers un problème de planification déterministe classique.

Par rapport au planificateur BPG, l'introduction d'une nouvelle méthode était motivée par trois impératifs : premièrement relâcher toutes les hypothèses restrictives faites par BPG au niveau du modèle ; deuxièmement tenter une approche de traitement direct sans traduction coûteuse vers un modèle déterministe ; et troisièmement pouvoir résoudre des problèmes de contrôle sur un horizon infini avec uniquement des propriétés de sûreté. Ces points se sont concrétisés avec l'algorithme *SB* (*Simulate and Branch*) que nous avons
* proposé dans [214]. Cet algorithme utilise une recherche en avant, c'est-à-dire en partant des états initiaux possibles et en raisonnant dans l'espace des états atteignables, et non dans l'espace de tous les états possibles du système. L'algorithme part d'une politique vide π et réalise une exploration en profondeur d'abord de deux arbres :

- un arbre T_B, appelé l'*arbre de branchement sur la politique*, qui décrit les décisions possibles concernant la politique ; chaque nœud de T_B correspond à une politique partielle définie par l'ensemble des décisions prises depuis la racine de T_B ;

- un arbre T_S, appelé l'*arbre de simulation du système*, qui sert à vérifier si les évolutions du système induites par la politique courante π satisfont bien les exigences ; chaque nœud de T_S correspond à un état s atteignable en appliquant la politique courante.

L'arbre de recherche principal est T_B. A chaque nœud de cet arbre, une exploration de T_S est lancée pour simuler l'application de la politique courante sur le système. Dans T_S, dès qu'un état s associé à une observation o non couverte par la politique courante π est rencontré, une décision est prise dans T_B concernant la commande $\pi(o)$ renvoyée par le contrôleur pour l'observation o. Avec cette nouvelle commande, la politique augmentée est testée sur le reste des états atteignables jusqu'à ou bien tomber à nouveau sur une observation non couverte, ou bien couvrir l'ensemble des états atteignables, ou bien tomber sur un état ne satisfaisant pas les propriétés requises. Dans ce dernier cas, un backtrack a lieu sur les choix faits dans l'arbre T_B de branchement sur la politique. La recherche s'arrête soit lorsqu'une politique solution est trouvée, soit lorsqu'un backtrack a lieu au niveau du nœud racine de l'arbre T_B, auquel cas toutes les politiques potentielles ont été étudiées. Une illustration du fonctionnement de la méthode est donnée à la figure 5.3.

Diverses optimisations ont été mises en œuvre pour accélérer l'exploration des deux arbres T_B et T_S. Nous avons notamment utilisé des techniques de marquage reprises de [23] permettant de ne pas explorer à nouveau des états déjà analysés et de ne pas parcourir indéfiniment des cycles d'états, des mécanismes d'explication d'incohérence permettant de faire du *conflict-based backjumping* [225], et des mécanismes permettant de ne pas considérer des contrôleurs équivalents qui diffèrent uniquement sur la numérotation de leurs états internes.

Ces techniques ont été implémentées dans l'outil *DynCode* que nous avons construit au-dessus de la librairie de programmation par contraintes *Gecode*. Dans DynCode, toutes les relations manipulées ($I(S)$, $F(S,C)$, $T(S,C,S')$, $P(S)$, $G(S)$) sont représentées sous la forme d'ensembles de contraintes. A chaque étape de la recherche en avant, il est demandé au solveur de contraintes Gecode de fournir l'ensemble des solutions de CSPs particuliers.

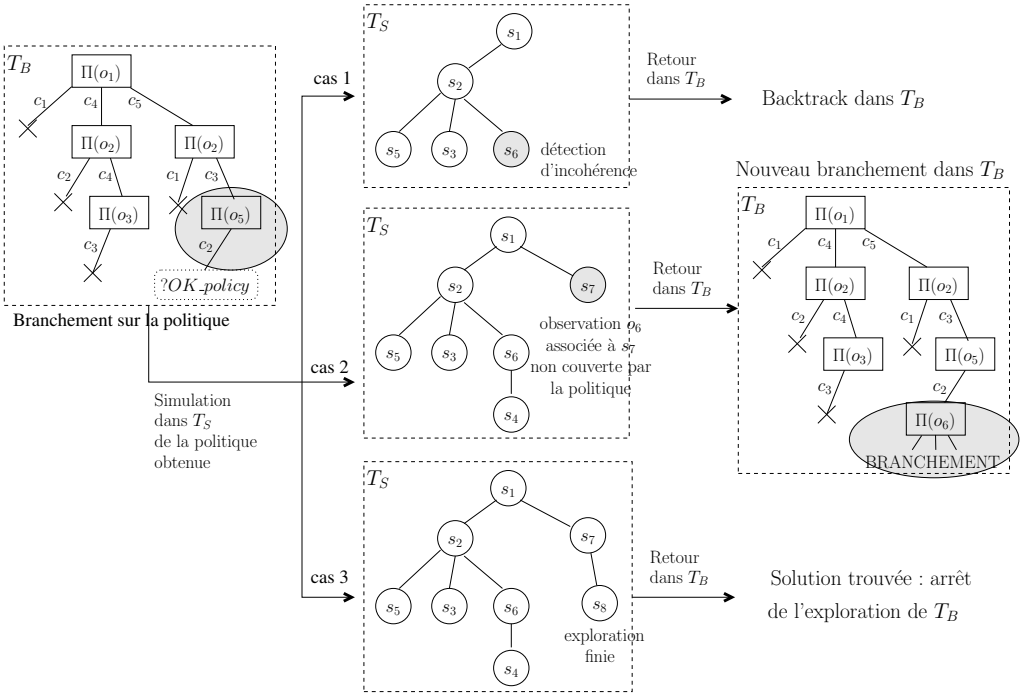


FIGURE 5.3 : Fonctionnement de l'algorithme SB (Simulate and Branch [214]) *

Par exemple, pour déterminer l'ensemble des états suivants possibles si une décision de contrôle c est appliquée dans un état s, il suffit de considérer le CSP décrivant $T(S, C, S')$, d'instancier dans ce CSP les variables de S et C comme dans s et c respectivement, de demander l'ensemble des solutions du CSP obtenu, et de projeter ces solutions sur S'.

Dans [214], nous avons montré que sur des problèmes de construction de contrôleurs à états finis pour la planification non déterministe, l'approche directe utilisée par l'outil DynCode était plus performante que l'approche par traduction vers un problème de planification classique considérée dans BPG [32]. Nous avons aussi réalisé des comparaisons avec le planificateur MBP (*Model-Based Planner* [23]), qui produit des plans conditionnels pour des problèmes de planification non déterministe avec observabilité partielle. Le résultat de la comparaison a été que d'un côté MBP fournissait des temps de calcul plus courts que DynCode, mais que de l'autre DynCode était capable de produire des contrôleurs à états finis dont la taille pouvait être de plusieurs ordres de grandeur inférieure à celle des plans conditionnels fournis par MBP. Pour nos besoins d'embarquabilité, le critère de taille des contrôleurs était plus critique que le problème de temps du production du contrôleur, à payer une seule fois et hors ligne. *

5.4.2 Cas de l'observabilité complète

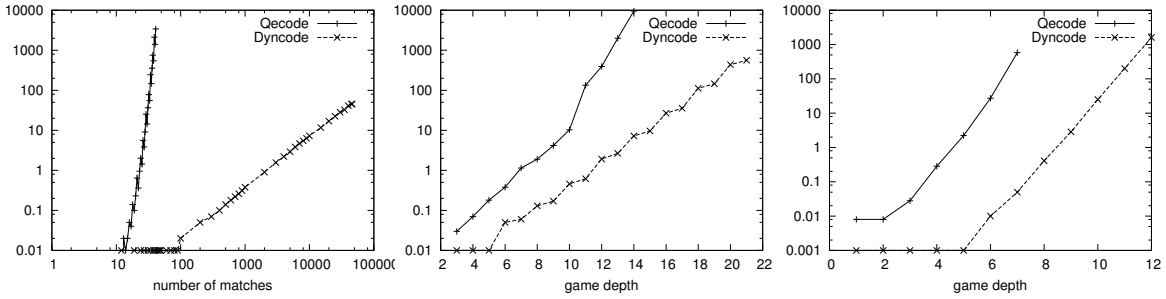
Au même titre que dans les modèles stochastiques il existe des algorithmes distincts pour traiter MDPs et POMDPs, il nous a été possible de définir dans [210] des méthodes spécifiques aux systèmes complètement observables. Dans [210], nous nous sommes concentrés sur des problèmes d'atteignabilité d'un but. Dans ce contexte, comme pour les MDPs, dès lors que le système est markovien et que les propriétés à vérifier portent uniquement * *

sur l'état courant, rechercher un contrôleur sans mémoire stationnaire associant à l'état courant observé s une décision de contrôle $\pi(s)$ est suffisant et optimal.

Dans nos travaux sur l'observabilité complète, nous avons considéré, par commodité de modélisation, un modèle du système à contrôler légèrement différent de celui utilisé pour le cas de l'observabilité partielle. Dans ce nouveau modèle, que nous avons baptisé MGCSP pour *Markovian Game CSP*, nous avons utilisé une représentation de la forme jeu à deux joueurs, impliquant d'un côté des variables de décision contrôlables dans C, et de l'autre des variables de décision non contrôlables dans un nouvel ensemble U. La relation de transition non déterministe $T(S, C, S')$ utilisée dans le cas de l'observabilité partielle a alors été "éclatée" en une *fonction* de transition $T_c(S, C, S')$ spécifiant l'effet déterministe des décisions contrôlables et une *fonction* de transition $T_u(S, U, S')$ spécifiant l'effet déterministe des décisions non contrôlables. Avec cette formulation, le non déterminisme a été reporté dans l'instanciation des variables non contrôlables de U. De même, la relation de faisabilité $F(S, C)$ a été séparée en deux relations, une relation $F_c(S, C)$ concernant la faisabilité des décisions contrôlables, et une relation $F_u(S, U)$ concernant la faisabilité sur la partie incontrôlable. D'un point de vue pratique, cette formulation nous a permis de faciliter la modélisation de systèmes dont l'état est régi par les actions enclenchées par le contrôleur (description dans T_c et F_c) et par l'occurrence de pannes sur certaines parties du système (description dans T_u et F_u).

* Nous avons introduit dans [210] un algorithme de synthèse de contrôleur qui raisonne en avant, en partant des état initiaux possibles et en déroulant les trajectoires du système induites par la politique courante. Une différence avec l'algorithme SB décrit à la section 5.4.1 est que grâce à l'hypothèse d'observabilité complète, la construction d'une politique de décision a pu être faite en explorant un seul arbre en profondeur d'abord, et non deux arbres comme dans l'algorithme SB. L'algorithme est noté par la suite MGDFS comme *Markovian Game Depth-First Search*. Cet algorithme reprend les principes de la programmation dynamique en avant (*forward dynamic programming* [136]). Plus précisément, il réalise une exploration en profondeur d'abord d'un arbre ET/OU qui alterne des niveaux OU correspondant à l'instanciation des variables contrôlables et des niveaux ET correspondant à l'instanciation des variables non contrôlables. Initialement, la politique de décision est vide et cette politique s'enrichit au fur et à mesure du parcours de l'arbre de recherche.

Comme pour l'algorithme SB, plusieurs techniques ont été mises en place dans [210]
* pour accélérer l'exploration, avec un marquage des états pour éviter d'analyser à nouveau des états déjà analysés, et avec des techniques de raisonnement sur les cycles pour détecter des états à partir desquels il est impossible d'atteindre les propriétés requises. Pour l'exploration des niveaux associés aux variables contrôlables, nous avons mis en place dans l'algorithme des raisonnements sur le CSP représentant $F_c \wedge T_c$, afin d'extraire toutes les décisions faisables sur C et tous les états suivants possibles associés pour un état de départ s particulier. Nous avons également mis en place des raisonnements sur des CSPs plus généraux, par exemple le CSP $F_c \wedge T_c \wedge G$ pour déterminer si le contrôleur possède un moyen d'atteindre le but G en un coup. Considérer des CSPs permettant des raisonnements sur plusieurs coups aurait été envisageable, mais plus coûteux en temps de calcul à chaque étape. Nous avons enfin mis en place des mécanismes de *restart* à partir d'une



(a) NimFibo (b) Connect4 (c) MatrixGame

FIGURE 5.4 : Comparaison des temps de calcul en secondes (sur l'axe des ordonnées) obtenus avec Dyncode et avec le solveur de QCSP+ Qecode [18], pour plusieurs jeux [210] *

politique vide, avec des heuristiques de choix de valeur aléatoires permettant de diversifier la recherche. Côté implémentation, nous avons enrichi l'outil *DynCode* mentionné précédemment avec l'algorithme MGDFS obtenu.

Expérimentalement, un des objectifs de nos travaux sur l'observabilité complète était de montrer qu'une approche utilisant la programmation par contraintes au sein d'une recherche dans l'espace des états atteignables était meilleure qu'une approche basée sur l'utilisation de modèles de contraintes quantifiées comme les QCSP/QCSP+. Nous avons donc montré que sur tous les problèmes de jeux habituellement considérés comme cas tests dans la littérature QCSP/QCSP+, l'algorithme MGDFS implémenté dans DynCode permettait d'obtenir des améliorations de plusieurs ordres de grandeur par rapport à des solveurs QCSP/QCSP+ performants, comme illustré à la figure 5.4 tirée de [210]. La justification à ces résultats est que dans un QCSP/QCSP+, la notion d'état du système n'est pas prise en compte explicitement, les problèmes sont modélisés sur un nombre d'étapes borné à définir a priori, et enfin les solutions recherchées sont des arbres de politique qui d'une certaine manière utilisent une mémoire complète du passé, et qui peuvent être exponentiellement moins compacts que des politiques définies plus directement comme des contrôleurs sans mémoire associant une décision à l'état courant. Il est à noter que par contre, l'utilisation de QCSP/QCSP+ peut tout à fait rester justifiée pour des problèmes impliquant seulement quelques alternances de quantificateurs comme dans l'ordonnancement avec adversaires [18], ou pour les problèmes impliquant de l'observabilité partielle ou des contraintes non markoviennes. *

5.5 Compilation de connaissances

Dans une autre direction, pour faire face aux problèmes liés à la taille de l'espace d'état du système à contrôler, nous avons étudié l'applicabilité de techniques dites de *compilation de connaissances* [58], qui servent à reformuler des connaissances sous une forme compacte et efficace. Cette étude a été réalisée essentiellement au cours de la thèse d'Alexandre Niveau [172]. Pour ces travaux, les connaissances à compiler pouvaient être la fonction de transition du système ($T(S, C, S')$), ou encore la politique solution elle-même ($\pi(S)$ dans le cas de l'observabilité complète, $\pi(O)$ dans le cas de l'observabilité partielle). L'intérêt d'une compilation hors ligne de la politique était de rendre cette dernière plus facilement

embarquable et plus efficacement utilisable en ligne.

Pour réaliser le travail de compilation, un aspect important résidait dans le choix d'un bon langage de représentation. Par exemple, pour compiler des formules de logique propositionnelle, il est possible de choisir, parmi d'autres alternatives, entre le langage des BDDs (*Binary Decision Diagrams* [36]) et le langage des NNFs (*Negation Normal Forms* [58]). Ces deux langages sont illustrés respectivement aux figures 5.5(b) et 5.5(c), où sont données des compilations de la connaissance exprimée dans la table de vérité de la figure 5.5(a). Deux propriétés particulièrement importantes pour guider le choix d'un langage sont les performances en espace (taille mémoire requise pour encoder des formules) et les performances en temps (durée requise pour répondre à une requête ou pour effectuer une opération de transformation de la forme compilée). Au final, choisir un bon langage répondant au besoin en termes d'opérations et de requêtes à traiter peut être fait en s'appuyant sur la *carte de compilation* [58], un objet qui regroupe l'ensemble des propriétés connues sur des langages de compilation existants.

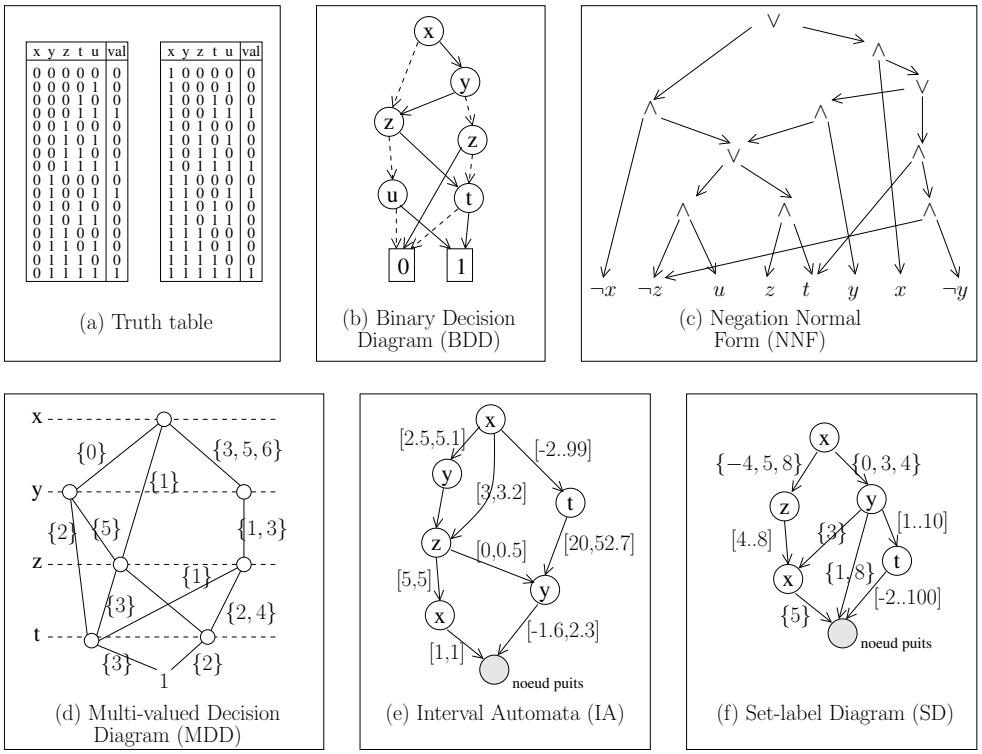


FIGURE 5.5 : Exemples de langages de compilation ; les langages (e) et (f) sont ceux que nous avons étudiés spécifiquement pour la planification non déterministe avec variables à grand domaine énuméré ou à domaine continu

Définition de nouveaux langages Plusieurs travaux avaient déjà été proposés dans la littérature pour mixer planification et compilation de connaissances. Parmi ces travaux, on peut citer l'approche *planning as model-checking* [53, 54, 55] utilisable pour résoudre des problèmes de planification non déterministe à l'aide de BDDs, ou l'algorithme SPUDD [109] utilisé pour résoudre des MDPs à l'aide d'ADDs (*Algebraic Decision Diagrams* [11]). On trouvait également diverses utilisations de d-DNNF (*Deterministic*

Decomposable NNF [58]) en planification déterministe [15, 31] et en planification conformante [181]. Dans notre cas, la nouveauté était que nous souhaitions traiter des problèmes de contrôle faisant intervenir des variables à grand domaine de valeurs, par exemple des variables décrivant un niveau de mémoire disponible, ou des variables à domaine continu, par exemple des variables décrivant le niveau d'énergie courant. En analysant la carte de compilation, il est cependant apparu qu'aucun langage n'avait été étudié en profondeur sur les domaines continus, et que les langages disponibles sur les domaines discrets n'étaient pas légion, des exceptions notables étant les MDDs (*Multi-valued Decision Diagrams* [124]), dont une illustration est fournie à la figure 5.5(d), et les *tree-driven automata* [74]. Par rapport à la carte de compilation, les MDDs étaient cependant des structures déjà bien particulières, qui n'avaient pas la généralité des BDDs ou des NNFs et de leurs sous-langages respectifs.

C'est pourquoi nous avons proposé lors de la thèse d'Alexandre Niveau deux nouveaux langages de représentation généraux pour étendre la carte de compilation existante :

- le langage des IAs (*Interval Automata* [176, 175, 177]), pour représenter des fonctions à valeur booléenne (des contraintes) portant sur des variables à domaine continu ; *
- le langage des SDs (*Set-label Diagrams* [173, 174, 7, 8]), pour représenter des fonctions à valeur booléenne (des contraintes) portant sur des variables à domaine énuméré. *

Un exemple d'IA et un exemple de SD sont fournis aux figures 5.5(e) et 5.5(f) respectivement. Globalement, IAs et SDs sont des graphes acycliques orientés comportant des nœuds étiquetés par des variables et des arcs étiquetés par des ensembles de valeurs. Pour les IAs, ces ensembles de valeurs sont des intervalles fermés de réels, et pour les SDs ils sont des ensembles de valeurs discrètes. IAs et SDs contiennent également un nœud puits. La sémantique associée est que tous les chemins dans le graphe acyclique qui mènent au nœud puits correspondent à des modèles de la formule manipulée.

Au niveau applicatif, les opérations qui nous intéressaient particulièrement étaient les opérations de conditionnement, d'extraction de modèles, et de comptage de modèles. Par exemple, pour utiliser en ligne une politique π (compilée hors ligne), nous avions besoin du conditionnement pour pouvoir conditionner la politique par l'état courant, et de l'extraction de modèle pour extraire une décision de contrôle adaptée à cet état. Pour pouvoir reproduire des méthodes de planification de type *planning as model-checking*, nous étions également intéressés par les opérations de conjonction de formules, de disjonction de formules, de négation d'une formule, d'oubli existentiel d'une variable dans une formule ($\exists x$) et éventuellement d'oubli universel ($\forall x$).

Propriétés des langages Pour étudier les propriétés des IAs et des SDs, nous avons tout d'abord défini des algorithmes de réduction fonctionnant en temps polynomial et permettant de compacter les représentations [177, 173]. Nous avons aussi étudié la compacité de ces langages et leur complexité temporelle. Les résultats obtenus ont été insérés dans une carte de compilation étendue [8]. Nous avons également introduit des sous-langages potentiellement un peu moins compacts que les IAs et les SDs, mais permettant de traiter plus d'opérations et de requêtes en temps polynomial. Certains sous-langages ont été construits en reprenant des principes déjà utilisés pour d'autres langages. Par exemple, les OSDs et

* *

les OIAs (O comme *ordered*) ont été introduits pour représenter les sous-langages dans lesquels les variables qui étiquettent les nœuds apparaissent toujours dans le même ordre sur tous les chemins menant au nœud puits. D'autres sous-langages ont été construits en considérant des types de restriction nouveaux, notamment les FIAs et les FSDs (F comme *focusing*), dans lesquels l'ensemble des valeurs possibles associées à une variable ne peut que se réduire au fur et à mesure du parcours d'un chemin menant au puits. Il a d'ailleurs été prouvé que cette propriété de *focusing* permettait de faire de l'extraction de modèle en temps polynomial, chose que les IAs et les SDs ne permettent pas. Le lien entre SDs et MDDs a aussi été explicité, et il a été montré que des représentations sous forme de SDs pouvaient être exponentiellement plus succinctes que des représentations sous forme de MDDs.

Construction d'IAs et de SDs Pour construire concrètement des IAs et des SDs, nous sommes partis de problèmes exprimés respectivement comme des CSPs continus et comme des CSPs discrets. Pour produire une forme compilée de l'ensemble des solutions de ces CSPs, deux techniques ont été proposées :

- une première technique qualifiable de *bottom-up* ; dans cette technique, l'ensemble des solutions du problème considéré est calculé avec un outil de résolution de contraintes : *RealPaver* [100][2] dans le cas des CSP continus, et *Choco* [49] dans le cas des CSP discrets, et ces solutions sont ajoutées une à une à la forme compilée ;

- une deuxième technique de type *compilation par trace* ; cette technique est inspirée de la méthode *DPLL with a trace* [113], utilisée pour construire des NNFs représentant des formules de logique propositionnelle ; elle consiste à suivre l'arbre de recherche exploré par un outil d'énumération de solutions et à créer au fur et à mesure du parcours de cet arbre de nouveaux nœuds à ajouter à la forme compilée ; en suivant la trace des arbres de recherche explorés par RealPaver et par Choco, nous avons
* obtenu les méthodes de compilation *RealPaver with a trace* [175] et *Choco with a*
* *trace* [174] ; cette technique consistant à compiler en utilisant la trace d'un solveur de contraintes avait aussi été proposée dans la littérature pour produire des MDDs [102].

L'avantage de la seconde approche est qu'elle est moins gourmande en mémoire car elle garantit que la structure manipulée à chaque étape n'est jamais plus grande que la forme compilée finale, alors que la première technique peut avoir à manipuler des structures intermédiaires exponentiellement plus volumineuses que le résultat. Un autre avantage de la seconde méthode est qu'elle peut permettre de réduire le travail à faire pour fouiller l'arbre de recherche lui-même, car la structure compilée construite pas à pas peut jouer le rôle de mémoire servant à détecter des sous-problèmes déjà résolus. Nous avons utilisé ce
* mécanisme dans *Choco with a trace* [174], en reprenant des techniques définies dans [137] pour accélérer la détection de sous-problèmes déjà traités. Nous avons aussi adapté des techniques proposées dans [5] pour définir des heuristiques de branchement sur les variables permettant d'arriver à des structures compilées les plus petites possibles.

[2]Formellement, RealPaver calcule un sur-ensemble des solutions et un sous-ensemble ne contenant que des solutions.

5.6 Exemple d'application de la synthèse de contrôleur

Nous avons décrit dans [198] une application de nos travaux à la synthèse de contrôleurs de sous-équipements embarqués à bord de satellites. Nous avons considéré plusieurs sous-systèmes, dont le sous-système *ObsToMem* responsable de la gestion des connexions entre un instrument d'observation et la mémoire de masse d'un satellite. Une vue globale de l'architecture physique que le contrôleur recherché avait à gérer est donnée à la figure 5.6. Cette architecture fait intervenir les barrettes de capteurs de l'instrument d'observation (à gauche), la mémoire de masse du satellite découpée en un ensemble de banques mémoire (à droite), des compresseurs mémoires ou COMs (au centre) chargés de compresser les données produites par les barrettes de capteurs, et enfin des connexions entre tous ces éléments. La tâche du contrôleur était de maintenir les connexions à utiliser pour que chaque barrette de capteurs puisse effectivement écrire des données sur une tranche mémoire, même en cas de panne ou d'indisponibilité d'un compresseur mémoire ou d'une tranche mémoire. Le modèle développé (variables d'état, variables de contrôle, fonction de transition...) a permis de synthétiser un contrôleur valide par construction. Les résultats obtenus sont décrits dans [198]. A noter enfin que nous avons utilisé dans [198] une méthodologie de modélisation similaire à celle utilisée dans le langage graphique SCADE[3], basé sur les langages synchrones [21]. Enfin, des essais de compilation de la fonction de transition du système et de la politique de décision du contrôleur ont été réalisés [172].

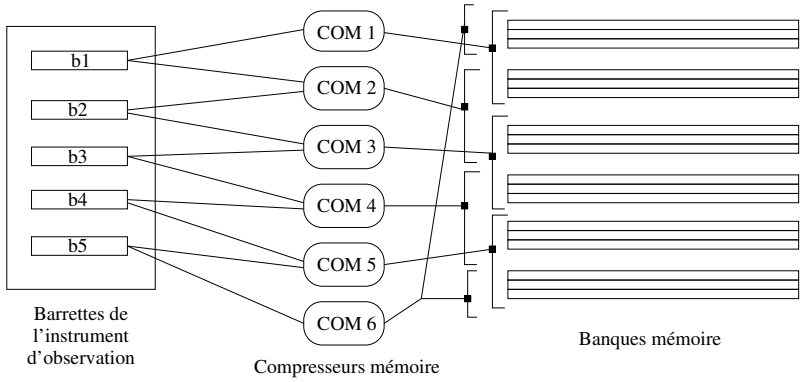


FIGURE 5.6 : Architecture physique à contrôler par le sous-système *ObsToMem* [198]

5.7 Synthèse d'allocations robustes aux pannes

Nous avons également eu à synthétiser des solutions robustes aux incertitudes pour des problèmes issus du monde aéronautique. Comme dans une planification conformante [241], ces solutions devaient être adaptées à tous les scénarios rencontrables. Plus précisément, nous avons eu à modéliser et résoudre des problèmes d'allocation dans le cadre d'une étude réalisée en partenariat avec Airbus et l'IRIT. Le problème considéré, dont une version préliminaire a été décrite dans [26], consistait à allouer des fonctions et des flux de données entre fonctions sur une architecture avionique composée de calculateurs et de chemins réseau entre calculateurs. Les contraintes à satisfaire correspondaient ou bien à

[3]http ://www.esterel-technologies.com/

des exigences métiers, ou bien à des contraintes de sûreté de fonctionnement imposant que les allocations produites devaient pouvoir assurer la disponibilité d'une fonction globale dans un certain nombre de scénarios de pannes portant sur les calculateurs et le réseau. Des travaux ont également été menés concernant la définition d'un outil d'allocation interactif, avec utilisation d'outils de programmation par contraintes classiques et d'outils de recherche locale à base de contraintes.

5.8 Bilan

Les travaux effectués autour de la synthèse de contrôleur valide ont été plus prospectifs que les travaux réalisés en synthèse de plan, essentiellement car l'approche consistant à écrire à la main des contrôleurs puis à les valider est encore très ancrée dans les pratiques actuelles. Sur certains systèmes, cette tendance est assez naturelle car spécifier directement un contrôleur peut être plus rapide que construire un modèle pour faire de la synthèse. C'est sur des systèmes plus complexes ou plus combinatoires que la synthèse de contrôleur peut exprimer son potentiel. Pour traiter ces systèmes combinatoires, nous avons tenté d'utiliser des techniques de programmation par contraintes, soit en utilisant directement un encodage CSP pur, soit en utilisant un outil de résolution de contraintes au sein d'un algorithme dédié de recherche en avant dans l'espace des états atteignables. Cette dernière approche s'est avérée être la plus performante.

Chapitre 6

Le cadre PFU, un cadre générique pour la décision dans l'incertain

La dernière catégorie de travaux que nous avons réalisés dans le domaine de la décision séquentielle dans l'incertain a porté sur la définition d'un cadre générique appelé le cadre "*PFU*" pour "*Plausibilité, Faisabilité, Utilité*". Ces travaux correspondent essentiellement aux travaux de thèse réalisés de 2003 à 2006. Du point de vue de la modélisation, la motivation initiale était d'arriver à regrouper au sein d'un cadre commun divers formalismes proposés depuis plusieurs décennies par les chercheurs en intelligence artificielle pour modéliser et résoudre des problèmes de décision. Un objectif était aussi de mieux comprendre les liens entre les formalismes existants, et éventuellement d'en introduire de nouveaux. La liste est longue, mais on peut citer parmi ces formalismes : la satisfiabilité d'une formule de logique propositionnelle (SAT), les formules booléennes quantifiées (QBFs), les formules booléennes stochastiques (*stochastic SAT* [145]) ; les problèmes de satisfaction de contraintes [148, 234] et leurs extensions permettant de prendre en compte des préférences (*VCSP* et *Semiring-CSP* [27]), du non déterminisme (*Quantified CSP* [33], *Mixed CSP* [73]) ou des aspects stochastiques (*Stochastic CSP* [72, 266]) ; les réseaux bayésiens [184] et leurs extensions à la décision séquentielle (*influence diagrams* [112], *valuation networks* [238] réseaux bayésiens dynamiques [59]) ; les MDPs (*Markov Decision Processes* [227]) et leurs extensions permettant de prendre en compte de l'observabilité partielle (*Partially Observable MDPs* [160]) ou la structure des états (*factored MDPs* [34]) ; le cadre de la planification STRIPS [76, 92] et ses extensions intégrant des incertitudes sur l'état initial et sur l'effet des actions (*conformant planning* [241], *contingent planning* [186, 226], *probabilistic planning* [131]).

Du point de vue de la résolution, la motivation était d'arriver à développer une algorithmique générique efficace, permettant de regrouper des avancées développées dans chacun des formalismes couverts. La démarche d'unification tentée était proche de celle qui avait conduit à des cadres graphiques et algébriques tels que les *Valuation algebras* [237, 130], les *Valued* ou *Semiring-based CSPs* [27], ou les *Algebraic MDPs* [187]. L'unification à définir était simplement un peu plus ambitieuse en termes de cadres couverts.

La variété des cadres visés aurait pu faire penser que le résultat aurait été un monstre ingérable. Les travaux de thèse se sont attachés à prouver le contraire en tirant parti des points communs existant entre les différents formalismes, qui mettent tous en jeu : (1) des

variables représentant soit l'état de l'environnement soit les décisions prises ; (2) des relations entre ces variables, qui représentent divers types de connaissances ; ces relations sont appelées par la suite des *fonctions locales* pour insister sur le fait que chaque relation porte seulement sur un sous-ensemble des variables ; (3) des opérateurs pour *agréger* et *synthétiser* les connaissances exprimées dans les fonctions locales ; on entend ici par agrégation le fait de joindre des connaissances, et par synthèse le fait de projeter des connaissances sur un plus petit nombre de variables.

Ce chapitre donne une vue d'ensemble du cadre PFU que nous avons introduit et de
* ses algorithmes. Voir [195, 222] pour une vue exhaustive.

6.1 Définition du cadre

Le cadre PFU tire son nom des trois types de connaissances exprimées par les fonctions locales considérées : des *plausibilités* représentant les incertitudes sur l'état de l'environnement, des *faisabilités* exprimant des contraintes fortes sur les décisions, et des *utilités* exprimant des préférences sur les instanciations des variables. Basiquement le cadre PFU s'appuie sur trois éléments : (1) une *structure algébrique générique* décrivant les opérateurs utilisés pour manipuler les connaissances ; (2) une *structure graphique générique* décrivant les variables et les fonctions locales liant ces variables ; (3) une *structure générique des requêtes* qu'il est possible de formuler sur le modèle graphique ainsi obtenu. Ces trois
* éléments, que nous avons introduit initialement dans [218, 219], après quelques essais
* antérieurs [221, 217, 257], sont décrits ci-après.

6.1.1 Une structure algébrique générique

La structure algébrique générique que nous avons considérée pour le cadre PFU avait pour but de définir de manière abstraite les opérateurs utilisés pour agréger et synthétiser des connaissances, ainsi que les propriétés algébriques de ces opérateurs. Nous souhaitions pouvoir englober aussi bien des approches de décision dans l'incertain reposant sur des utilités espérées probabilistes que des approches reposant sur des utilités espérées possibilistes [66], des utilités espérées avec fonctions de Spohn [93] ou encore des approches utilisant du non déterminisme booléen. Pour parvenir à cet objectif, nous avons repris et adapté des travaux décrits dans [80, 105, 51], qui définissaient une généralisation axiomatique et algébrique des notions classiques de probabilité et d'utilité espérée sous les dénominations de *plausibilité* et d'*utilité espérée généralisée*. Cela nous a amenés à considérer une structure algébrique définie par sept éléments paramétrables $(E_p, E_u, \oplus_p, \otimes_p, \otimes_u, \oplus_u, \otimes_{pu})$. Dans cette structure,

- E_p et E_u sont deux ensembles appelés respectivement l'ensemble des degrés de plausibilités et l'ensemble des degrés d'utilité ; les éléments de E_p (resp. E_u) permettent de représenter à quel point une situation est certaine (resp. préférée) ;
- $\oplus_p : E_p \times E_p \to E_p$ et $\otimes_p : E_p \times E_p \to E_p$ sont les deux opérateurs qui décrivent comment synthétiser et combiner des plausibilités : la plausibilité d'occurrence d'un événement couvrant deux événements disjoints de plausibilités respectives p_1 et p_2 est $p_1 \oplus_p p_2$, et la plausibilité d'occurrence simultanée de deux événements indépendants

de plausibilités respectives p_1 et p_2 est $p_1 \otimes_p p_2$; dans le cas où l'incertain est décrit par des probabilités, $(E_p, \oplus_p, \otimes_p) = (\mathbb{R}, +, \times)$: utilisation de la somme pour marginaliser et du produit pour combiner des probabilités d'événements indépendants ;

- $\otimes_u : E_u \times E_u \to E_u$ désigne l'opérateur utilisé pour agréger des utilités locales : si u_1 et u_2 désignent deux degrés d'utilité locaux, alors l'utilité globale obtenue est $u_1 \otimes_u u_2$; par exemple, la structure à utiliser pour manipuler des utilités additives est $(E_u, \otimes_u) = (\mathbb{R}, +)$;

- $\oplus_u : E_u \times E_u \to E_u$ et $\otimes_{pu} : E_p \times E_u \to E_u$ sont les deux opérateurs utilisés pour calculer l'utilité espérée associée à un ensemble de n situations ayant chacune une plausibilité p_i et une utilité u_i, via la formule d'utilité espérée généralisée $\oplus_{u\, i \in [1..n]} p_i \otimes_{pu} u_i$; dans un contexte probabiliste, les opérateurs utilisés sont $\oplus_u = +$ et $\otimes_{pu} = \times$, ou autrement dit l'utilité espérée est donnée par $\sum_{i \in [1..n]} p_i \times u_i$.

Mathématiquement, nous avons supposé que la structure $(E_p, \oplus_p, \otimes_p)$ était un *semi-anneau commutatif*, que la structure $(E_u, \otimes_u)$ était un *monoïde commutatif* et que la structure $(E_p, E_u, \oplus_u, \otimes_{pu})$ était un *semi-module* sur $(E_p, \oplus_p, \otimes_p)$. Cela implique notamment des propriétés d'associativité et de commutativité traduisant le fait qu'une plausibilité globale ou une utilité globale ne dépend pas de l'ordre dans lequel les plausibilités locales ou les utilités locales sont agrégées. Cela implique également des hypothèses de distributivité entre certains opérateurs. Cela implique aussi pour $\oplus_p$ et $\otimes_p$ l'existence d'éléments neutres respectifs 0_p et 1_p. Informellement, 0_p correspond à la plausibilité d'une situation impossible, et 1_p à la plausibilité d'une situation certaine. Dans le cas probabiliste, $0_p = 0$ et $1_p = 1$. Nous avons également supposé que E_p et E_u étaient équipés d'ordres $\preceq_p$ et $\preceq_u$ permettant de comparer des degrés de plausibilité et d'utilité, avec des hypothèses de monotonie sur les opérateurs.

Les hypothèses faites sur ces structures ont été le fruit de compromis toujours discutables entre des considérations sémantiques et algorithmiques : pouvoir englober le plus grand nombre possible d'approches, et garder à l'esprit que les problèmes concrets doivent pouvoir être représentés de façon suffisamment compacte et résolus de façon raisonnablement efficace. La table 6.1 donne des exemples de structures rencontrées dans la littérature qui sont couvertes par la structure algébrique du cadre PFU. Ces exemples vont de contextes dans lesquels les incertitudes sont des probabilités à des contextes dans lesquels les incertitudes correspondent à des possibilités [65], des κ-rankings [242], ou du non déterminisme booléen. Dans tous ces cadres, les plausibilités ont un caractère distributionnel : la plausibilité d'un ensemble d'événements disjoints E est complètement déterminée par la plausibilité individuelle de chacun des événements couverts par E. Des cadres non distributionnels comme les fonctions de croyance de Dempster-Shafer [236] ne sont pas couverts.

Dans ce qui suit, pour un opérateur $\oplus$ et une fonction f portant sur deux ensembles de variables disjoints S et S', on note abusivement $\oplus_S f$ la fonction g définie par : $\forall A' \in \mathbf{d}(S')$, $g(A') = \oplus_{A \in \mathbf{d}(S)} f(A, A')$. On appelle dans ce cas $\oplus$ un opérateur d'*élimination*, dans le sens où il permet de projeter des connaissances en éliminant des variables : passage de la fonction f portant sur $S \cup S'$ à la fonction g portant sur S' uniquement.

	E_p	$\oplus_p$	$\otimes_p$	E_u	$\otimes_u$	$\oplus_u$	$\otimes_{pu}$
1	$\mathbb{R}^+$	$+$	$\times$	$\mathbb{R} \cup \{-\infty\}$	$+$	$+$	$\times$
2	$\mathbb{R}^+$	$+$	$\times$	$\mathbb{R}^+$	$\times$	$+$	$\times$
3	$[0,1]$	max	min	$[0,1]$	min	max	min
4	$[0,1]$	max	min	$[0,1]$	min	min	$\max(1-p,u)$
5	$\mathbb{N} \cup \{\infty\}$	min	$+$	$\mathbb{N} \cup \{\infty\}$	$+$	min	$+$
6	$\{t,f\}$	$\vee$	$\wedge$	$\{t,f\}$	$\wedge$	$\vee$	$\wedge$
7	$\{t,f\}$	$\vee$	$\wedge$	$\{t,f\}$	$\wedge$	$\wedge$	$\rightarrow$
8	$\{t,f\}$	$\vee$	$\wedge$	$\{t,f\}$	$\vee$	$\vee$	$\wedge$
9	$\{t,f\}$	$\vee$	$\wedge$	$\{t,f\}$	$\vee$	$\wedge$	$\rightarrow$

TABLE 6.1 : Ensembles et opérateurs utilisés dans neuf cadres : (1) utilité espérée probabiliste avec utilités additives (permet de calculer l'espérance d'un gain ou d'un coût), (2) utilité espérée probabiliste avec utilités multiplicatives (pour calculer la probabilité que des contraintes soient satisfaites), (3) utilité espérée possibiliste optimiste, (4) utilité espérée possibiliste pessimiste, (5) utilité espérée avec fonctions de Spohn et utilités uniquement positives, (6) utilité espérée booléenne optimiste avec utilités conjonctives (pour savoir s'il existe un monde possible dans lequel tous les buts d'un ensemble de buts B sont satisfaits), (7) utilité espérée booléenne pessimiste avec utilités conjonctives (pour savoir si dans tous les mondes possibles tous les buts sont satisfaits), (8) utilité espérée booléenne optimiste avec utilités disjonctives (pour savoir s'il existe un monde possible satisfaisant un des buts), (9) utilité espérée booléenne pessimiste avec utilités disjonctives (pour savoir si tous les mondes possibles satisfont un des buts)

6.1.2 Une structure graphique générique

Le deuxième ingrédient que nous avons introduit dans le cadre PFU correspond à un modèle graphique définissant d'une part des variables, qui peuvent être des *variables de décision* (contrôlables par un agent) ou des *variables d'environnement* (non contrôlables), et d'autre part des fonctions locales exprimant des connaissances sur ces variables. Ces fonctions locales peuvent être de trois types :

- des fonctions locales de *plausibilité* qui représentent des incertitudes, par exemple une distribution de probabilité conditionnelle $P_{z \mid x,y}$ portant sur trois variables d'environnement x, y, z ;
- des fonctions locales de *faisabilité* qui représentent des contraintes sur les décisions, par exemple une contrainte $F_{x,y}$ sur deux variables de décision x et y ;
- des fonctions locales d'*utilité* qui représentent des préférences, par exemple une fonction de récompense U_x portant sur une variable x.

L'ensemble de toutes ces fonctions locales est représentable sous la forme d'un modèle graphique composite tel que celui fourni à la figure 6.1(a), contenant plusieurs types de nœuds et plusieurs types d'arêtes reliant ces nœuds.

Une question naturelle qu'il était possible de se poser concernait la façon de produire les fonctions locales pour représenter correctement les connaissances. La manière de définir les variables et les fonctions d'utilité était relativement claire. Le point dur concernait la marche à suivre pour définir les bonnes fonctions locales de plausibilité et de faisabilité.

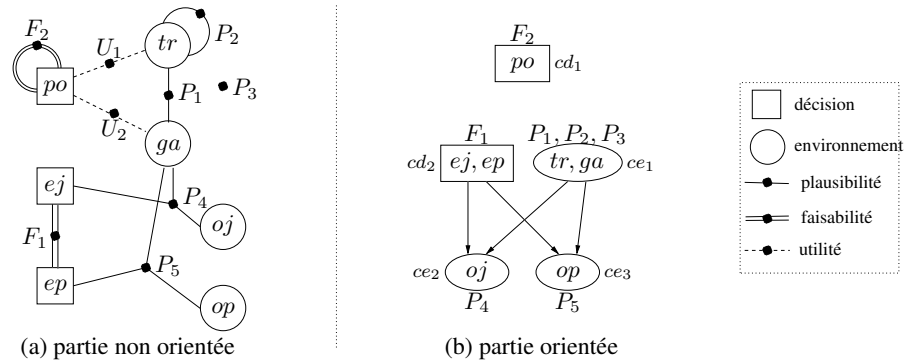


FIGURE 6.1 : Représentation graphique d'un réseau PFU : (a) hypergraphe des fonctions locales entre variables ; (b) graphe orienté acyclique entre composantes

Pour répondre à cette question, nous nous sommes tout d'abord appuyés sur les travaux de [80, 105] pour étendre les notions classiques de distribution de probabilité et de distribution de probabilité conditionnelle au cas des plausibilités : nous avons défini formellement des notions de *distribution de plausibilité* sur un ensemble de variables V et de *distribution de plausibilité conditionnelle* $\mathcal{P}_{S|S'}$ sur des variables dans un ensemble S sachant des variables dans un autre ensemble S'. Nous avons ensuite étendu un résultat central disponible dans un contexte probabiliste pour les *chain graphs* [81]. Ces derniers généralisent à la fois la représentation de l'incertain avec des modèles graphiques orientés comme les réseaux bayésiens [184] et avec des modèles graphiques non orientés comme les champs de Markov [43]. Dans un *chain graph*, l'ensemble V des variables aléatoires du problème est partitionné en un ensemble d'éléments $C = \{C_1, \ldots C_m\}$ appelés des *composantes*, l'idée étant que les variables appartenant à une même composante $c \in C$ sont corrélées. Dans le cas des réseaux bayésiens, il y a une composante par variable aléatoire, et dans le cas des champs de Markov, il y a une seule composante contenant toutes les variables. Un exemple de DAG de composantes est donné à la figure 6.1(b). Pour un *chain graph*, il est établi que si G est un DAG sur l'ensemble des composantes C et si chaque composante $c \in C$ est conditionnellement indépendante de ses non-descendants sachant ses parents, alors la distribution de probabilité $\mathcal{P}_V$ se factorise sous la forme :

$$\mathcal{P}_V = \prod_{c \in C} \mathcal{P}_{c|pa(c)}$$

chaque terme $\mathcal{P}_{c|pa(c)}$ associé à chaque composante c pouvant lui-même être factorisé en un produit de facteurs Φ_c : $\mathcal{P}_{c|pa(c)} = \prod_{\{P_i \in \Phi_c\}} P_i$.

Sur cette base, nous avons établi pour le cadre PFU un théorème généralisé faisant le lien entre indépendances conditionnelles représentées par le DAG et factorisation, permettant dans le cas des plausibilités d'obtenir un premier niveau de factorisation de la forme :

$$\mathcal{P}_V = \otimes_{p\, c \in C} \mathcal{P}_{c|pa(c)}$$

et un deuxième niveau de factorisation par composante de la forme $\mathcal{P}_{c|pa(c)} = \otimes_{p\{P_i \in \Phi_c\}} P_i$. On retrouve ici l'ensemble des fonctions de plausibilité P_i à considérer. Le même genre

de résultat a pu être établi pour définir la manière d'obtenir une représentation factorisée d'une distribution de faisabilité globale. Ces résultats nous ont permis d'exhiber une façon naturelle de construire un réseau PFU correspondant à un problème concret : identifier les variables de décision et d'environnement, identifier les composantes et construire un DAG représentant des indépendances conditionnelles, exprimer les distributions de plausibilité ou de faisabilité conditionnelles associées à chaque composante, en utilisant des fonctions aussi locales que possible, et enfin exprimer les fonctions locales d'utilité.

Au final, nous avons défini un *réseau PFU* comme un tuple (V, G, P, F, U) tel que :

- V est un ensemble fini de variables à domaines finis, partitionné en un ensemble V_D de variables de décision et un ensemble V_E de variables d'environnement ;
- G est un DAG dont les sommets, appelés composantes, sont une partition C de V, plus précisément l'union d'une partition C_D de V_D et d'une partition C_E de V_E ; pour tout $c \in C$, on note $pa(c)$ l'ensemble des variables présentes dans les composantes parentes de c dans le DAG ;
- $P = \{P_1, P_2, \ldots\}$ est un ensemble fini de fonctions locales de plausibilité ; toute fonction locale de plausibilité P_i est associée à une composante d'environnement unique $c \in C_E$ et l'ensemble Φ_c des fonctions P_i associées à une composante $c \in C_E$ satisfait la condition de normalisation : $\oplus_{p_c} (\otimes_{p\{P_i \in \Phi_c\}} P_i) = 1_p$, ce qui traduit le fait que la disjonction de toutes les situations possibles est certaine ;
- $F = \{F_1, F_2, \ldots\}$ est un ensemble de fonctions locales de faisabilité ; toute fonction locale de faisabilité F_i est associée à une composante de décision unique $c \in C_D$ et l'ensemble Φ_c des fonctions F_i associées à une composante $c \in C_D$ doit satisfaire la condition de normalisation : $\vee_c (\wedge_{\{F_i \in \Phi_c\}} F_i) = \mathit{true}$, ce qui traduit le fait qu'au moins une décision est toujours faisable ;
- $U = \{U_1, U_2, \ldots\}$ est un ensemble de fonctions locales d'utilité.

6.1.3 Une structure générique des requêtes

Le troisième et dernier ingrédient que nous avons introduit dans le cadre PFU nous a permis de formuler des tâches de raisonnement. Cet ingrédient correspond à la notion de *requête* sur un réseau PFU. Syntaxiquement, nous avons défini une requête sur un réseau PFU comme un couple $(Sov, \mathcal{N})$ avec

- $\mathcal{N} = (V, G, P, F, U)$ un réseau PFU ;
- Sov une séquence de paires opérateur-variables (op, S) telles que ou bien $S \subseteq V_D$ (variables de décision) et $op = \min$ ou $\max$, ou bien $S \subseteq V_E$ (variables d'environnement) et $op = \oplus_u$.

Sémantiquement, l'idée était d'utiliser la séquence Sov pour exprimer un scénario de décision séquentielle. Plus précisément, l'idée était d'utiliser la séquence Sov pour exprimer un ordre dans lequel les décisions étaient prises par les différents agents et dans lequel les observations étaient réalisées. Pour les variables de décision dans V_D, la flexibilité sur l'opérateur utilisé (min ou max) était présente pour permettre de considérer des

scénarios où les décisions pouvaient être prises par un agent coopératif ou non. La valeur d'une requête devait alors correspondre à l'utilité espérée associée au scénario décrit par la requête, avec l'hypothèse que quand il prend une décision, un agent est informé de toutes les observations passées. Nous supposions de plus avec cette définition que tous les agents cherchaient à maximiser ou à minimiser la même utilité espérée.

Les seules conditions imposées sur les séquences Sov définissant les requêtes étaient que premièrement chaque variable pouvait apparaître au plus une fois dans la séquence, et deuxièmement pour toute paire (x, y) de variables de natures différentes (l'une de décision, l'autre d'environnement), si la composante à laquelle appartenait y était dans les descendantes de la composante à laquelle appartenait x, alors y ne pouvait pas apparaître à gauche de x dans la séquence. Cette dernière condition visait à exclure des requêtes incohérentes du point de vue de la causalité.

Valeur d'une requête Nous avons défini la valeur d'une requête en utilisant le concept d'arbre de décision. Dans l'arbre de décision utilisé, les variables sont considérées dans un ordre compatible avec leur ordre d'apparition dans Sov, afin que l'arbre reflète bien le scénario de décision séquentielle exprimé par la requête. Tout nœud n de l'arbre de décision correspond à une variable x et toute arête vers un nœud fils correspond à une instanciation $(x = a)$. Si x est une variable d'environnement, cette arête est de plus pondérée par une plausibilité conditionnelle $\mathcal{P}(x = a|A)$, où A est l'instanciation associée au chemin de la racine à n. Si x est une variable de décision, les branches issues de n correspondent aux instanciations a de x telles que la faisabilité conditionnelle $\mathcal{F}(x = a|A)$ est vraie, où A est l'instanciation associée au chemin de la racine à n. Toute feuille de l'arbre de décision correspond à une instanciation de l'ensemble des variables. L'utilité d'un nœud feuille est donnée par la combinaison des fonctions locales d'utilité U_i, l'utilité d'un nœud interne associé à une variable d'environnement est donnée par l'utilité espérée de la valeur de ses fils, et l'utilité d'un nœud interne associé à une variable décision est donnée par l'utilité optimale parmi ses fils. La valeur de la requête, notée $Ans(Q)$, correspond alors à la valeur du nœud racine de l'arbre de décision. Des règles de décision spécifiant quelles décisions prendre en fonction de ce qui a été observé peuvent également être retenues lors du parcours de l'arbre.

Cette définition, sémantiquement justifiée par les concepts d'arbre de décision et de loterie, posait cependant des problèmes au niveau algorithmique. En effet, le calcul de la valeur $Ans(Q)$ d'une requête nécessitait un appel répété au calcul de quantités telles que $\mathcal{F}(x = a|A)$ et $\mathcal{P}(x = a|A)$ non présentes telles quelles dans la définition du réseau PFU, et dont le calcul peut être de complexité exponentielle. Ce constat nous a amené à établir un théorème montrant que la valeur $Ans(Q)$ d'une requête $Q = (Sov, \mathcal{N})$ sur un réseau PFU $\mathcal{N} = (V, G, P, F, U)$ pouvait être définie de manière équivalente par la formule :

$$Ans(Q) = Sov(((\underset{F_i \in F}{\wedge} F_i) \star ((\underset{P_i \in P}{\otimes_p} P_i) \otimes_{pu} (\underset{U_i \in U}{\otimes_u} U_i)))) \tag{6.1}$$

Du point de vue de la résolution, cette formule nous a permis d'adopter une vision complètement algébrique de la décision séquentielle dans l'incertain, puisqu'elle ramène

un problème de décision à un problème de calcul d'une séquence d'éliminations de variables donnée par *Sov* sur la combinaison de toutes les fonctions locales du réseau PFU, avec les opérateurs adéquats. L'opérateur $\star$ utilisé dans la formule est un opérateur que nous avons spécialement introduit pour faire en sorte que tout ce qui correspond à des décisions infaisables soit ignoré par tous les opérateurs d'élimination. La formule 6.1 est également algorithmiquement attrayante dans la mesure où elle fait appel uniquement à des quantités déjà présentes dans la définition d'un réseau PFU. Nous avons de plus montré que les règles de décision obtenues à partir de l'équation 6.1 en mémorisant pour chaque décision la(les) valeur(s) optimale(s) (l'*argmax* ou l'*argmin*) étaient identiques aux règles de décision optimales obtenues avec l'approche par arbre de décision.

6.1.4 Classe de complexité et cadres couverts

* Nous avons montré dans [219, 220] que le problème de décision associé à une requête PFU était *PSPACE-complet*, et que les requêtes sur des réseaux PFU permettaient de modéliser de manière très directe de nombreux problèmes issus d'autres formalismes :

- dans le cadre de la satisfiabilité : SAT, QBF, stochastic SAT (SSAT) et extended-SSAT [145] ;
- dans le cadre des CSPs : recherche et comptage de solutions ; résolution d'un CSP valué ou d'un CSP quantifié [33] ; recherche de règles de décision conditionnelles ou inconditionnelles pour un CSP mixte ou mixte probabiliste [73] ; recherche de politique(s) optimale(s) ou de valeur supérieure à un certain seuil pour un CSP stochastique ou un problème d'optimisation stochastique [266] ;
- recherche d'un plan solution de longueur bornée en planification classique (planification STRIPS [76, 92]) ;
- résolution des requêtes classiques sur un réseau bayésien [184], un champ de Markov [43] ou un graphe chaîné [81], avec des plausibilités exprimées comme des probabilités, des possibilités ou des κ-rankings : calcul d'une distribution de plausibilité, résolution de problèmes MAP (*Maximum A Posteriori hypothesis*) et MPE (*Most Probable Explanation*), calcul de la plausibilité associée à une observation ou à une clause logique dans un réseau hybride [61] ;
- résolution d'un diagramme d'influence [112] ;
- avec un horizon fini, résolution d'un MDP probabiliste [227], possibiliste, ou basé sur les κ-rankings, complètement ou partiellement observable [160], factorisé ou non [34].

6.2 Algorithmes génériques

* Nous avons proposé plusieurs techniques [202, 195] pour calculer la réponse à une requête PFU telle que donnée par l'équation 6.1 : des techniques basiques de recherche arborescente et d'élimination de variables, décrites à la section 6.2.1, des techniques de structuration des requêtes, décrites à la section 6.2.2, et enfin des techniques travaillant à partir de cette structuration, décrites à la section 6.2.3.

6.2.1 Premiers algorithmes

Recherche arborescente en profondeur d'abord Le premier algorithme que nous avons introduit est un algorithme de recherche arborescente en profondeur d'abord, qui explore l'espace des instanciations possibles des variables du problème en utilisant comme ordre d'instanciation des variables un ordre compatible avec la séquence Sov de la requête. Cet algorithme nous a essentiellement servi à exhiber une méthode de calcul de la réponse à une requête PFU fonctionnant en espace polynomial. Le résultat de PSPACE-complétude est venu de là, allié avec le fait que le cadre PFU couvrait les formules booléennes quantifiées, elles-mêmes PSPACE-complètes.

Elimination de variables Pour calculer la valeur d'une requête, il était assez naturel de tenter l'utilisation d'un algorithme dit d'élimination de variables [22, 60], qui cette fois parcourt la séquence Sov de la droite vers la gauche. Ce genre d'algorithme exploite la factorisation d'une fonction globale en fonctions locales pour accélérer les calculs à réaliser. Pour donner un exemple, supposons que l'on cherche à calculer la quantité suivante :

$$\max_{x_1,x_2,x_3,x_4} (\varphi_{12} + \varphi_{13} + \varphi_{23} + \varphi_{14})$$

avec φ_{ij} désignant une fonction locale à valeurs réelles portant sur les variables x_i et x_j. A chaque étape d'un algorithme classique d'élimination de variables, une variable x à éliminer est sélectionnée. L'élimination de x est alors réalisée en prenant en compte uniquement la partie du problème qui dépend de x. Par exemple, l'élimination de x_4 se fait en calculant $\varphi'_1 = \max_{x_4}(\varphi_{14})$. La fonction résultante φ'_1 dépend de x_1, et la quantité restant à calculer devient $\max_{x_1,x_2,x_3}(\varphi_{12} + \varphi_{13} + \varphi_{23} + \varphi'_1)$. Si l'on choisit ensuite d'éliminer x_3, alors on calcule $\varphi'_{12} = \max_{x_3}(\varphi_{13} + \varphi_{23})$, qui est une fonction portant sur x_1 et x_2. La quantité à évaluer devient $\max_{x_1,x_2}(\varphi_{12} + \varphi'_{12} + \varphi'_1)$. Après élimination de toutes les variables, la valeur de l'expression initiale est obtenue. Il est également possible de mémoriser des argmax pour produire une instanciation optimale de x_1, x_2, x_3, x_4.

Un des intérêts des algorithmes d'élimination de variables pour le cadre PFU est qu'ils offraient une borne de complexité temporelle potentiellement bien inférieure à celle d'une recherche arborescente classique. Plus précisément, leur complexité théorique temporelle et spatiale est exponentielle non pas en le nombre de variables mais en $w+1$, où w est un paramètre appelé *largeur induite* (*induced-width* [9, 60]), qui reflète la complexité du plus dur des calculs locaux à réaliser. Ce paramètre correspond au nombre maximal de variables dont dépendent les fonctions créées au fur et à mesure des éliminations. La largeur induite w peut varier suivant l'ordre d'élimination choisi et rechercher un ordre d'élimination minimisant la largeur induite est un problème NP-complet [9], mais heureusement de bonnes heuristiques existent pour trouver de bons ordres d'élimination.

Pour adapter ces techniques au cadre PFU, la première idée a été de partir de travaux existants [237, 130] qui proposaient une version généralisée des algorithmes d'élimination de variables, utilisant des opérateurs abstraits définis par leurs propriétés algébriques de base. L'adaptation de ces travaux posait néanmoins plusieurs difficultés :

- premièrement, ces travaux avaient été définis dans un contexte où un seul opérateur de combinaison était utilisé, alors que nous en avions plusieurs dans le cadre PFU

$(\otimes_p, \otimes_u$ et $\otimes_{pu})$;

- deuxièmement, ces travaux avaient été définis dans un contexte où un seul opérateur d'élimination était utilisé, alors que nous en avions à nouveau plusieurs dans le cadre PFU (min, max et $\oplus_u$) ; dans la littérature, des algorithmes d'élimination de variables avaient été définis pour traiter des problèmes multi-opérateurs [167, 182], mais jamais dans un cadre algébrique général ;

- troisièmement, la non distributivité de certains opérateurs entre eux posait quelques problèmes, notamment la non distributivité de l'opérateur d'élimination $\oplus_u$ par rapport à l'opération de combinaison $\otimes_u$ qui nous empêchait d'écrire des transformations de la forme $\oplus_{ux}(f_x \otimes_u g_y) = (\oplus_{ux} f_x) \otimes_u g_y$, ou autrement dit de ne considérer que les fonctions locales dépendantes de x pour éliminer x.

Les trois problèmes précédents ont pu être résolus : le premier par l'intermédiaire de transformations algébriques permettant de se ramener à un seul opérateur de combinaison, le deuxième en imposant simplement des contraintes sur l'ordre d'élimination des variables, et le troisième en imposant des propriétés algébriques supplémentaires sur les opérateurs du cadre PFU. Plus précisément, nous avons introduit pour traiter le dernier point deux axiomes supplémentaires, correspondant chacun à deux sous-structures algébriques distinctes couvrant à elles deux les instanciations du cadre PFU rencontrées en pratique, et notamment toutes les instanciations décrites à la table 6.1. Pour des raisons non explicitées ici, le premier axiome a été appelé *l'axiome du cas semi-groupe*, et le second *l'axiome du cas semi-anneau*[1]. Sous l'hypothèse de satisfaction de l'un de ces deux axiomes, nous
* sommes parvenus à définir dans [202] l'algorithme MVE (*Multi-operator Variable Elimination*), qui fait de l'élimination de variables dans un contexte multi-opérateurs. Au lieu d'être exponentielle en la largeur induite, la complexité de l'algorithme MVE est exponentielle en un paramètre appelé la *largeur induite contrainte* [182], qui prend en compte les contraintes sur l'ordre d'élimination engendrées par la séquence *Sov*.

Suite à ces travaux, d'un côté nous avions réussi à passer d'un algorithme classique d'élimination de variables à MVE, mais de l'autre pour y parvenir nous avions usé de transformations qui masquaient la structure réelle des requêtes. Nous arrivions par exemple à trouver sur le papier des décompositions que la méthode MVE n'arrivait pas à exploiter. L'idée est donc venue d'essayer d'introduire des mécanismes plus sophistiqués capables d'exploiter la structure réelle des requêtes.

6.2.2 Structuration des requêtes multi-opérateurs

Pour analyser la structure réelle des requêtes, nous sommes repartis de la définition de base donnée à l'équation 6.1. Moyennant quelques hypothèses assez faibles, satisfaites par tous les cadres du tableau 6.1, nous avons réussi à nous ramener à une structure algébrique plus simple utilisant seulement deux opérateurs abstraits, notés $\oplus$ et $\otimes$, au lieu de cinq $(\oplus_p, \oplus_u, \otimes_{pu}, \otimes_p, \otimes_u)$. Les opérateurs $\oplus$ et $\otimes$ introduits ont été définis sur un ensemble

* [1] Voir [195, 222] pour les détails. L'axiome du cas semi-groupe fait l'hypothèse que $\oplus_u = \otimes_u$ (vrai par exemple dans un cadre probabiliste avec utilités additives), et l'axiome du cas semi-anneau fait notamment une hypothèse de distributivité de $\otimes_u$ par rapport à $\oplus_u$ (vrai par exemple dans un contexte probabiliste avec utilités multiplicatives).

E et il a été fait l'hypothèse que $(E, \oplus, \otimes)$ formait un semi-anneau commutatif muni d'un ordre total pour lequel $\oplus$ et $\otimes$ étaient monotones. Nous avons alors montré que la réponse $Ans(Q)$ à une requête $Q = (Sov, (V, G, P, F, U))$ se réécrivait sous la forme :

$$Ans(Q) = Sov((\underset{F_i \in F}{\wedge} F_i) \star ((\underset{P_i \in P}{\otimes} P_i) \otimes (\underset{U_i \in U}{\otimes} U_i))) \text{ dans le cas semi-anneau}$$
$$Ans(Q) = Sov((\underset{F_i \in F}{\wedge} F_i) \star ((\underset{P_i \in P}{\otimes} P_i) \otimes (\underset{U_i \in U}{\oplus} U_i))) \text{ dans le cas semi-groupe}$$

avec Sov pouvant contenir les opérateurs d'élimination min, max, et $\oplus$. Cette diminution du nombre d'opérateurs a simplifié les développements algorithmiques.

Nous avons ensuite introduit dans [200] et [201] des techniques pour structurer automatiquement les calculs à réaliser, avec une structuration fonctionnant en deux étapes : *

- dans la première étape, appelée l'étape de *macrostructuration*, un système de règles de réécriture est utilisé pour exprimer la réponse à une requête sous une forme structurée ; plusieurs types de règles ont été proposées, avec des règles qui décomposent les calculs, des règles qui les recomposent pour faire apparaître des libertés dans l'ordre d'élimination des variables, et enfin des règles de simplification qui exploitent les conditions de normalisation sur les plausibilités et sur les faisabilités ; nous avons introduit dans [201] un système de règles de réécriture pour le cas semi-groupe, * et dans [200] un autre système de règles pour le cas semi-anneau ; un exemple de * macrostructure obtenue est donné à la figure 6.2(a) ;

- la seconde étape, plus fine que la précédente, a consisté à exploiter au mieux les libertés d'ordre d'élimination révélées par la macrostructuration ; sur ce point, nous avons fait appel à des techniques de *décomposition arborescente* [9], qui exploitent les propriétés topologiques des modèles graphiques pour structurer un problème donné en un arbre de sous-problèmes plus simples à résoudre ; elles s'appliquent très bien lorsqu'un seul opérateur d'élimination est utilisé, ce qui était le cas à l'intérieur de chaque nœud de la macrostructure.

La structure obtenue à l'issue de ce processus est un DAG que nous avons baptisé *MCDAG* pour "Multi-operator Cluster DAG" [200, 201]. Un exemple est fourni à la figure 6.2(b). Les nœuds d'un MCDAG, appelés des clusters, représentent des calculs à réaliser. Formellement, chaque cluster est défini par trois éléments : un ensemble de variables $S(c)$ à éliminer, un ensemble $\Phi(c)$ de fonctions locales associées au cluster, et un couple $(\oplus^c, \otimes^c)$ d'opérateurs qui forment un semi-anneau commutatif. La valeur $val(c)$ d'un cluster c correspond à l'élimination des variables de $S(c)$ avec l'opérateur $\oplus^c$, sur la combinaison avec $\otimes^c$ des fonctions locales dans $\Phi(c)$ et de la valeur des clusters fils de c : * *

$$val(c) = \underset{S(c)}{\oplus^c} ((\underset{\varphi \in \Phi(c)}{\otimes^c} \varphi) \otimes^c (\underset{s \in Fils(c)}{\otimes^c} val(s))) \tag{6.2}$$

Enfin, la valeur d'un MCDAG est la valeur de son nœud racine. En pratique, pour $(\oplus^c, \otimes^c)$, les clusters obtenus utilisaient les opérateurs $(\min, \oplus)$, $(\max, \oplus)$, $(\min, \otimes)$, $(\max, \otimes)$, $(\oplus, \otimes)$.

Nous avons établi plusieurs résultats théoriques sur le processus de structuration : unicité de la macrostructure obtenue après application des règles de réécriture, correction du processus de structuration (valeur du MCDAG égale à la réponse à la requête PFU

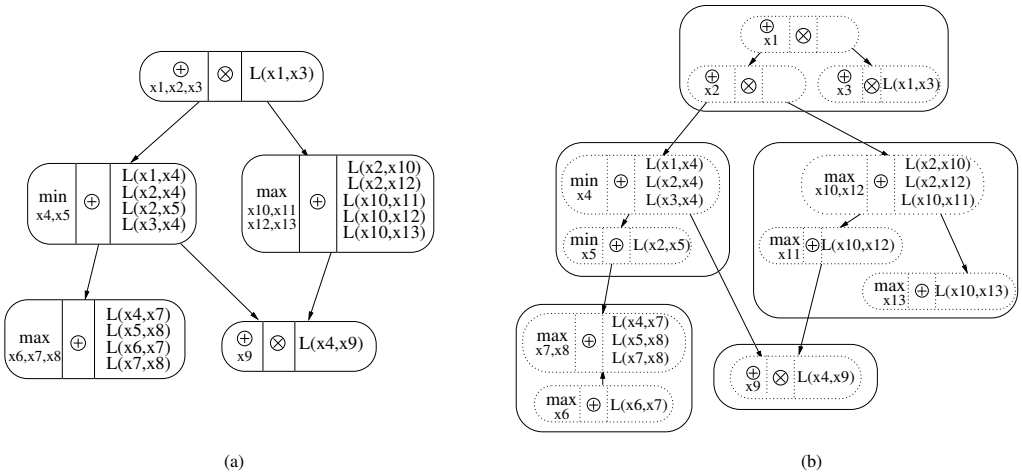


FIGURE 6.2 : Structuration d'une requête PFU : (a) macrostructure obtenue par règles de réécriture ; (b) MCDAG obtenu par techniques de décomposition en arbre ; pour chaque cluster de calcul on indique l'opérateur d'élimination à utiliser et les variables à éliminer (à gauche), les fonctions locales associées au cluster (à droite), et l'opérateur à utiliser pour combiner ces fonctions locales et les valeurs des clusters fils (au centre)

de départ), complexité polynomiale de la réécriture. Nous avons montré qu'en termes de largeur, l'approche structurée était toujours au moins aussi bonne qu'une approche non structurée du type de celle utilisée dans l'algorithme *MVE*, et qu'elle pouvait engendrer des gains exponentiels sur la complexité théorique. Par rapport aux processus de structuration utilisés par exemple pour les QBFs [17] ou les diagrammes d'influence [119], les MCDAGs sont d'une part plus génériques car définis dans un cadre algébrique général, et d'autre part potentiellement exponentiellement meilleurs car ils utilisent plus de mécanismes. L'originalité de l'architecture MCDAG est qu'elle est la seule à utiliser à la fois plusieurs opérateurs d'élimination et plusieurs opérateurs de combinaison.

Globalement, en utilisant le cheminement décrit à la figure 6.3, nous sommes parvenus à transformer le problème du calcul de la réponse à une requête PFU "plate", exprimée par une séquence d'élimination *Sov*, en un problème de calcul de la valeur d'un MCDAG. L'exploration des deux branches axiomatiques "cas semi-anneau" et "cas semi-groupe" n'aura été que passagère, dans le sens où l'architecture MCDAG a permis de retrouver une vision algorithmique unifiée.

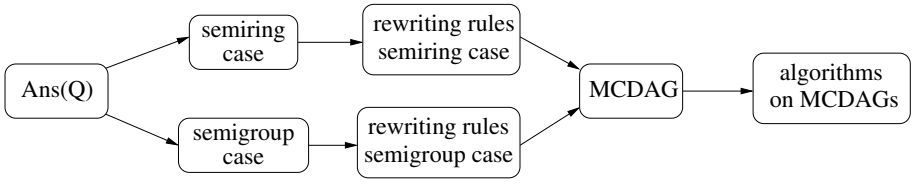


FIGURE 6.3 : Vers une architecture de calcul unique

6.2.3 Algorithmes exploitant la structure des requêtes

Algorithme bottom-up Une fois l'architecture MCDAG obtenue, il a été assez direct de définir un algorithme générique d'élimination de variables. La procédure utilisée consiste simplement à dire que dès qu'un cluster c du MCDAG a reçu la valeur $val(s)$ de chacun de ses fils s, il calcule sa propre valeur $val(c)$ puis la transmet à ses parents. A l'issue de ce processus, qui propage les informations des feuilles du MCDAG vers la racine, on obtient la valeur du cluster racine, et donc la réponse à la requête. Pour calculer la valeur d'un cluster selon l'équation 6.2, on peut soit éliminer les variables de $S(c)$ une à une, soit les éliminer simultanément. La première approche est appelée VE-MCDAG (VE pour *Variable Elimination*) et la seconde est appelée CTE-MCDAG (CTE en référence à l'algorithme *Cluster-Tree Elimination* [22] dont elle s'inspire). Comme indiqué à la table 6.2, VE-MCDAG et CTE-MCDAG ont la même complexité temporelle mais une complexité spatiale légèrement différente. CTE-MCDAG est un peu moins gourmand en espace, grâce à une mémorisation réalisée sur les séparateurs entre clusters du MCDAG (on appelle *séparateur* entre deux clusters c et c' l'ensemble des variables communes à c et c').

Algorithme top-down : recherche arborescente structurée Afin de bénéficier à la fois de l'efficacité pratique de la recherche arborescente et de la bonne complexité temporelle théorique des algorithmes d'élimination de variables, nous avons défini un algorithme générique de recherche arborescente qui tire parti des décompositions structurelles exprimées par l'architecture MCDAG. Dans son principe, une telle idée avait déjà été exploitée dans les algorithmes *AND/OR search* [62], *recursive conditioning* [57] et *BTD* [118]. La nouveauté dans les algorithmes à définir pour le cadre PFU résidait dans le côté multi-opérateurs des MCDAGs. La méthode que nous avons proposée consiste à effectuer une recherche arborescente en profondeur d'abord dans le MCDAG à la manière de l'algorithme BTD (*Backtrack bounded by Tree Decomposition* [118]). Le MCDAG est parcouru de la racine vers les feuilles et les valeurs associées à chaque cluster sont calculées au fur et à mesure du parcours de l'arbre. Par rapport à une recherche arborescente classique, la seule différence est liée au fait que certains branchements dans l'arbre correspondent à des branchements sur les clusters fils du cluster courant, et pas forcément à l'instanciation d'une variable. Il suffit d'utiliser les bons opérateurs pour synthétiser la valeur des nœuds associés à de tels branchements pour que l'algorithme fonctionne. L'algorithme obtenu a été appelé BTD-MCDAG.

Au-dessus de ce mécanisme de base, nous avons proposé deux ajouts principaux :

- l'ajout de techniques de *mémorisation* comme dans BTD : pour éviter des calculs redondants, il est possible d'utiliser une structure de mémorisation portant sur les séparateurs entre clusters, comme dans CTE-MCDAG, afin d'éviter de réexplorer un cluster plusieurs fois pour la même instanciation du séparateur avec son cluster parent ; on troque ainsi de la complexité spatiale contre de la complexité temporelle ;

- l'ajout de techniques de *branch-and-bound*, pour élaguer l'espace de recherche ; sur ce point, nous avons adapté des techniques venant de l'algorithme alpha-beta [127, 12] utilisé en théorie des jeux ; dans ces techniques, des bornes inférieures et supérieures à ne pas dépasser sont maintenues lors du parcours de l'arbre de recherche.

Nous avons ainsi proposé trois versions de l'algorithme de recherche arborescente structurée : une version de base, une version avec mémorisation, et une version avec mémorisation et élagage. Nous avons montré que toutes ces méthodes étaient correctes
* et complètes, et nous avons analysé leurs complexités spatiales et temporelles [195, 222]. Les résultats obtenus sont résumés dans la table 6.2. Par rapport à ces résultats, il faut également avoir à l'esprit la différence entre complexité théorique et complexité pratique.

Méthode	complexité temporelle	complexité spatiale
DFS	$O(d^n)$	linéaire
MVE	$O(d^{w(Sov)+1})$	$O(d^{w(Sov)})$
VE-MCDAG	$O(d^{w(MCDAG)+1})$	$O(d^{w(MCDAG)})$
CTE-MCDAG	$O(d^{w(MCDAG)+1})$	$O(d^{s(MCDAG)})$
DFS-MCDAG	$O(d^{h(MCDAG)})$	linéaire
DFS-MCDAG + recording	$O(d^{w(MCDAG)+1})$	$O(d^{s(MCDAG)})$
DFS-MCDAG + recording + bounds	$O(d^{h(MCDAG)})$	$O(d^{s(MCDAG)})$

TABLE 6.2 : Complexités théoriques des algorithmes génériques du cadre PFU, avec d : taille du plus grand domaine d'une variable ; n : nombre de variables dans le réseau PFU ; $w(Sov)$: largeur induite contrainte [182] imposée par la séquence Sov de la requête PFU ; $w(MCDAG)$: largeur du MCDAG, définie comme le nombre maximal de variables dans un cluster moins un ; $s(MCDAG)$: nombre maximal de variables dans un séparateur du MCDAG ; $h(MCDAG)$: hauteur du MCDAG, définie comme le nombre maximal de variables présentes sur un chemin de la racine à une feuille ; remarque : on a toujours $s(MCDAG) \leq w(MCDAG) + 1 \leq w(Sov) + 1$ et $w(MCDAG) + 1 \leq h(MCDAG)$.

6.3 Bilan

Le cadre PFU développé au cours des travaux de thèse a permis d'aboutir à une vision transverse de plusieurs formalismes utilisés en intelligence artificielle. Ce cadre est le résultat d'une conception prenant en compte à la fois des enjeux d'expressivité et des enjeux algorithmiques. Comparé aux approches génériques existantes, son originalité est qu'il manipule explicitement plusieurs types de variables (décision et environnement), plusieurs types de fonctions locales (plausibilités, faisabilités et utilités) et plusieurs opérateurs de combinaison et d'élimination. D'un point de vue opérationnel, nous avons défini des algorithmes génériques à base de recherche arborescente ou d'élimination de variables pour répondre à des requêtes PFU. Nous avons notamment travaillé sur la structuration des requêtes, ce qui nous a mené à l'architecture de calcul MCDAG. Les techniques introduites peuvent contribuer à la résolution des QBFs, des problèmes de satisfiabilité stochastique, des CSPs quantifiés ou stochastiques, ou encore des diagrammes d'influence probabilistes
* ou possibilistes. Notons enfin qu'un outil générique de résolution a été développé [195]. En pratique, dans le contexte de travail Onera, il a été cependant utile pour résoudre des problèmes de contrôle issus du monde aéronautique et spatial de repartir de cadres de représentation un peu moins ambitieux en termes d'expressivité mais plus directement utilisables à court et moyen terme, tels que ceux considérés aux chapitres précédents.

Chapitre 7

Perspectives

Ce dernier chapitre décrit des perspectives de recherche envisagées à plus ou moins long terme. Ces perspectives s'organisent selon trois axes :

- axe 1 : recherche locale à base de contraintes ;
- axe 2 : décision en continu et dans l'incertain ;
- axe 3 : programmation par contraintes pour le traitement d'applications.

Le contenu de ces trois axes est décrit plus précisément ci-après.

7.1 Recherche locale à base de contraintes

Le premier axe de recherche visé consiste à approfondir et étendre les méthodes de type CBLS (*Constraint-Based Local Search*) présentées au chapitre 4 et mises en œuvre au sein de la bibliothèque InCELL. Par rapport à des méthodes complètes, la plus-value essentielle des méthodes CBLS réside dans leur capacité à raisonner sur des systèmes dynamiques relativement complexes et dans leur capacité à traiter rapidement des problèmes de grande taille. Plusieurs pistes sont envisagées pour poursuivre le développement de ces méthodes.

Améliorations sur l'existant : incrémentalité, reformulation et parallélisation Une première piste envisagée consiste à définir de nouvelles techniques pour améliorer les méthodes au niveau des temps de calcul et de l'espace mémoire consommé. Cette piste essentiellement algorithmique peut se matérialiser de diverses façons : (1) en recherchant de nouvelles méthodes de réévaluation incrémentale des invariants ; de nouvelles méthodes sont notamment envisagées pour les invariants gérant les aspects temps et ressources ; (2) en travaillant sur la reformulation des modèles, en réutilisant ou bien des techniques de reformulation, par exemple pour factoriser des sous-expressions utilisées à plusieurs endroits dans les modèles, ou bien directement des techniques de compilation de modèles telles que celles présentées à la section 5.5 ; (3) en essayant de paralléliser les calculs à réaliser au sein d'une recherche locale ou au sein d'une batterie d'algorithmes de recherche locale, sur des architectures de calcul multi-cœurs.

Recherche locale automatique ou semi-automatique Une autre piste envisagée consiste à étendre les travaux CBLS déjà réalisés pour pouvoir mettre en œuvre plus facilement des algorithmes de recherche locale au-dessus de la bibliothèque InCELL. Relativement à la figure 4.2 du chapitre 4 (page 44), cela signifie introduire des techniques pour d'une part définir facilement des mouvements locaux et des voisinages sur un problème donné, et d'autre part définir facilement des méta-heuristiques de recherche (recherche locale sur grands voisinages, recherche tabou, recuit simulé, algorithmes évolutionnaires...). L'ambition serait de couvrir un continuum allant d'une recherche locale complètement ad hoc à une recherche locale complètement automatique, en passant par une recherche locale implémentée rapidement à l'aide de primitives génériques de base. Pour la recherche complètement automatique, nous pouvons envisager réutiliser des techniques employées dans des outils comme LocalSolver [19], reposant sur la définition d'un catalogue de mouvements locaux efficaces et sur des techniques d'apprentissage des bons mouvements locaux à effectuer. La difficulté dans notre cas est que les problèmes traités sont beaucoup moins cadrés que dans le cas de LocalSolver, qui s'intéresse uniquement à des problèmes d'allocation sous contraintes formalisés à l'aide d'un nombre d'invariants relativement restreint. Pour la définition de primitives de base, un point de départ correspond aux techniques utilisées dans l'outil CBLS COMET [107], avec notamment la présence d'objets dits différentiables permettant d'estimer rapidement l'impact de la réinstanciation d'une variable sur le modèle, et la présence d'éléments pour identifier les causes d'une incohérence et pour résoudre des conflits (voir les méthodes de type *iterative flattening* [42] fonctionnant sur des ressources cumulatives). Des notions de voisinages complexes pourraient aussi être regardées [189], de même que l'adaptation de mécanismes classiques de propagation de contraintes pour circonscrire les mouvements locaux qu'il est utile de tester.

Extension de l'expressivité des modèles Un point essentiel qui participe à l'efficacité de la programmation par contraintes classique est la disponibilité d'un catalogue de contraintes globales utilisables dans les modèles, et sur lesquelles des techniques de propagation de contraintes spécifiques ont été étudiées. L'équivalent en CBLS est le catalogue des invariants offerts au modélisateur. Nous pensons que ce catalogue nécessite d'être enrichi pour arriver à traiter des problèmes plus complexes. Parmi les éléments de représentation que nous souhaiterions étudier se trouvent :

- des invariants dédiés au traitement de systèmes dynamiques à événements discrets, dans lesquels des événements peuvent être déclenchés en cas de dépassement de certains seuils sur les ressources ; de tels aspects sont modélisables avec le langage de planification PDDL+ [78], mais pour l'instant les méthodes CBLS ne les traitent pas ; un objectif à long terme serait de pouvoir modéliser dans une approche CBLS des comportements continus définis par des automates hybrides [108], afin de représenter des interactions complexes entre état, temps et ressources ;

- des invariants dédiés au traitement des aspects optimisation, notamment concernant les interactions entre aspects temporels et critères, typiquement pour des applications dans lesquelles les dates de réalisation des tâches influent sur la qualité du plan ; cela se produit par exemple dans le domaine spatial où se posent des problèmes de recentrage des observations, consistant à calculer des dates de réalisation des

acquisitions permettant d'effectuer les prises de vue le plus possible à la verticale des zones à observer ; sur ce point, les travaux combinant STN et critère d'optimisation linéaire [162] sont un bon point de départ ; toujours sur les aspects optimisation, nous voudrions aussi définir des invariants pour encoder des paradigmes de décision multi-critères ;

- pour la gestion de mission, des invariants dédiés au traitement d'applications dans lesquelles interviennent des aspects géométriques utiles pour modéliser le déplacement des engins ; traiter ce point signifierait intégrer des algorithmes de planification de trajectoire incrémentaux [245, 129] au sein d'une approche CBLS.

Langage de modélisation CBLS pour des problèmes d'ordonnancement complexes Enfin, pour accélérer la phase de spécification des modèles, nous pensons qu'il serait utile de définir un nouveau langage dédié à la représentation des problèmes d'ordonnancement complexes avec des modèles CBLS. L'idée serait de combiner au sein d'un même langage des briques de modélisation utilisées en programmation par contraintes, par exemple dans des langages tels que MiniZinc (`http://www.minizinc.org/`), avec des briques de modélisation utilisées pour décrire des systèmes dynamiques, par exemple dans des langages tels que PDDL [155] et ANML [240]. Le cadre TECK introduit dans [259] est aussi un point de départ possible. *

7.2 Décision en continu et dans l'incertain

Dans la plupart des applications que nous traitons, résoudre de manière optimale un problème d'ordonnancement ou un problème de planification sur un horizon donné a peu de valeur en soi. Le critère déterminant pour juger de la qualité de nos méthodes est le résultat obtenu à l'issue de la durée de vie du système, ou tout du moins après une période de fonctionnement suffisamment significative. Nous avons ainsi à prendre en compte le fait que nous souhaitons synthétiser des décisions et engager ces décisions en continu. Ce travail doit de plus être fait dans un contexte incertain et dynamique, dans lequel de nouvelles informations arrivent au fil de l'eau, et dans lequel de nouvelles requêtes à satisfaire peuvent aussi apparaître progressivement. Toutes ces problématiques posent deux questions principales, avec d'une part la question d'une planification avec anticipation des conséquences à long terme des décisions prises, et d'autre part la question du suivi de la situation du système au cours de l'exécution, pour pouvoir adapter au mieux les actions prévues.

Sur ces points, nous avons pour l'instant utilisé deux types d'approches, avec :

- d'un côté une approche réactive consistant à produire des plans, éventuellement robustes, et à replanifier en cas de déviation trop importante entre le modèle et la réalité, avec les techniques présentées aux chapitres 3 et 4,

- de l'autre une approche proactive de synthèse de contrôleur consistant à essayer d'anticiper toutes les situations possibles pour ne plus avoir à refaire de tâches de raisonnement combinatoires en ligne, avec les techniques présentées au chapitre 5.

Dans nos perspectives, nous envisageons d'utiliser une approche intermédiaire consistant à prendre en compte les incertitudes sur l'évolution du système [165] et à générer des plans suffisamment couvrants par rapport à ces incertitudes. Notre ambition est de fouiller davantage l'utilisation de plans conditionnels, capables de s'adapter au contexte rencontré et de fournir des comportements de meilleure qualité que les plans robustes. Nous souhaiterions aussi maintenir ces plans conditionnels incomplets sur un horizon de décision glissant, et les faire évoluer dynamiquement au fur et à mesure de l'avancée du temps. Tout cela implique une réflexion sur la forme des plans eux-mêmes, point sur lequel il pourrait être intéressant de partir de techniques de représentation classiques [251], ainsi que des techniques de synthèse de contrôleurs à états finis présentées au chapitre 5. Nous pourrions aussi nous inspirer d'un algorithme comme RFF [249] développé au-dessus du formalisme des MDPs pour produire des faisceaux de plans. Un autre aspect est la nécessité de développer des techniques applicables dans un contexte multi-agents, contexte qui devient de plus en plus présent dans plusieurs types d'applications envisagées. Dans ce contexte, la présence d'autres agents avec lesquels se coordonner augmente parfois drastiquement le degré d'incertitude.

Nous envisageons également d'explorer plus finement des mécanismes de délibération dite *hiérarchique*, combinant une planification court terme et une planification long terme, avec des modèles précis pour le court terme et des modèles plus grossiers pour le long terme. Des exemples de cadres existants utilisant ce genre de philosophie sont les cadres OSCO (*Online Combinatorial Stochastic Optimization* [106]) et HOP (*Hindsight Optimization* [50]), qui permettent, dans un contexte stochastique, de déterminer la prochaine action à effectuer en fonction de simulations sur l'effet des décisions à long terme. Une autre source d'inspiration possible est la méthode UCT (*Upper Confidence Tree* [128]), qui propose sur le court terme une combinaison intéressante entre exploration des branches de décision encore peu fouillées et exploitation des branches de décision les plus prometteuses, et sur le long terme l'application d'une stratégie de décision par défaut. Sur le très court terme, ou autrement dit du côté de l'exécution, nous pensons qu'il est important de creuser le suivi de situation pendant l'application des plans : suivi des écarts entre le plan prévu et la réalité, suivi de l'état de panne des composants utilisés pour réaliser la mission, tout cela afin de pouvoir replanifier au mieux ou de pouvoir reconfigurer le système considéré. Une des difficultés dans ce suivi de situation est que l'état des composants réalisant la mission n'est pas forcément accessible en permanence, d'autant plus sur les missions impliquant des communications intermittentes entre les acteurs impliqués.

Enfin, décider et agir en continu soulève la problématique de la décision en temps contraint. Les algorithmes de recherche locale à base de contraintes peuvent être adaptés dans ce contexte. Il ne le sont cependant pas toujours, car ils imposent par exemple de représenter de manière explicite un réseau de contraintes temporelles portant sur tous les éléments de l'horizon de planification, ce qui peut être coûteux en temps et en espace mémoire. A l'inverse, des méthodes de recherche en avant telles que celles du cadre CTA présenté au chapitre 3 sont moins gourmandes en temps et en espace, car elles nécessitent seulement de maintenir un état courant du système lors des raisonnements (en plus du plan en cours de construction). Pour des systèmes complexes, il serait intéressant de faire du contrôle de système en combinant ces approches en avant avec des outils de modélisation

de systèmes dynamiques à événements discrets comme Ptolemy [68] ou Lustre, qui sont plus riches que de "simples" modèles PDDL. Pour arriver à marier décision en avant et simulation de systèmes complexes, nous pensons donc poursuivre l'étude de techniques de simulation-décision, avec des heuristiques de décision utilisées à chaque étape.

Le dernier point est que si une décision en continu est réellement mise en œuvre au sein d'un système autonome, il est nécessaire de disposer de techniques de validation (par méthodes formelles ou par simulation) pour apporter des garanties sur les modèles utilisés pour planifier, sur les résultats produits, sur les temps de réponse pire cas obtenus, et plus généralement sur l'interaction avec le reste de l'architecture décisionnelle. Pour assurer des propriétés de contrôlabilité malgré les incertitudes, des travaux sur les STNU (*Simple Temporal Networks with Uncertainty* [263, 161, 243, 171]) pourraient aussi être considérés. Pour la décision réellement embarquée, nous voudrions enfin poursuivre les travaux initiés lors de la thèse d'Adrien Maillard, portant sur la combinaison entre une planification hors ligne (non embarquée) et une replanification en ligne (embarquée) utilisant des mécanismes rapides de réparation de plans.

7.3 Programmation par contraintes pour le traitement d'applications

La troisième et dernière perspective que nous souhaitons poursuivre correspond au traitement d'applications soumises par des partenaires industriels ou envisagées pour des démonstrateurs propres. De nouvelles catégories d'applications sont déjà en discussion, notamment des problèmes de surveillance de débris spatiaux, des problèmes de planification de maintenance en aéronautique, ou encore des problèmes d'ordonnancement de tâches sur processeurs temps réel. Notre objectif est de transférer les approches formelles que nous développons ou celles disponibles dans la littérature, d'apporter des procédures de résolution automatiques et rapides, et d'améliorer la qualité des solutions utilisées à l'heure actuelle. A cet objectif de transfert s'ajoute pour nous l'enjeu de voir de nouveaux contextes, de nouvelles contraintes, de nouveaux critères, qui nous donnent matière à raisonnement.

Pour aller plus avant dans le transfert, une piste intéressante serait d'étudier plus intensément les méthodes de résolution interactive, pour que les utilisateurs apprivoisent les techniques proposées ou tout simplement parce que lors des premières étapes de la conception des systèmes, certaines contraintes et certains critères métiers ne sont pas présents dans les modèles. Pour conduire de telles études, les méthodes de configuration interactive sont un bon point de départ [6]. A cela s'associe un travail d'interfaçage qui pose de vraies questions techniques pour rendre disponible en quasi temps réel des informations sur des modèles ou sur les solutions proposées.

Bibliographie

[1] E. Aarts and J. Lenstra, editors. *Local Search in Combinatorial Optimization*. John Wiley & Sons, 1997.

[2] M. Abramson, D. Carter, S. Kolitz, M. Ricard, and P. Scheidler. Real-time optimized Earth observation planning. In *Proc. of the Earth Science Technology Conference (ESTC-02)*, 2002.

[3] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks : a survey. *Computer Networks*, 38 :393–422, 2002.

[4] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126 :183–235, 1994.

[5] J. Amilhastre. *Représentation par automate d'ensemble de solutions de problèmes de satisfaction de contraintes*. Thèse de doctorat, Université Montpellier II, 1999.

[6] J. Amilhastre, H. Fargier, and P. Marquis. Consistency Restoration and Explanations in Dynamic CSP : Application to Configuration. *Artificial Intelligence*, 135(1) :199–234, 2002.

[7] J. Amilhastre, H. Fargier, A. Niveau, and C. Pralet. Compiling CSPs : A complexity map of (non-deterministic) multivalued decision diagrams. In *Proc. of the 24th IEEE International Conference on Tools with Artificial Intelligence (ICTAI-12)*, pages 1–8, 2012. *

[8] J. Amilhastre, H. Fargier, A. Niveau, and C. Pralet. Compiling CSPs : A complexity map of (non-deterministic) multivalued decision diagrams. *International Journal of Artificial Intelligence Tools*, 23(4) :1460015, 2014. *

[9] S. Arnborg. Efficient Algorithms for Combinatorial Problems on Graphs with Bounded Decomposability - A Survey. *BIT*, 25 :2–23, 1985.

[10] F. Bacchus and F. Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, 16 :123–191, 2000.

[11] R. Bahar, E. Frohm, C. Gaona, G. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Algebraic Decision Diagrams and Their Applications. In *IEEE /ACM International Conference on CAD*, pages 188–191, 1993.

[12] B. Ballard. The *-Minimax Search Procedure for Trees Containing Chance Nodes. *Artificial Intelligence*, 21(3) :327–350, 1983.

[13] P. Baptiste, C. L. Pape, and W. Nuijten. *Constraint-based Scheduling : Applying Constraint Programming to Scheduling Problems*. Kluwer Academic Publishers, 2001.

[14] M. Barbier, C. Barrouil, J.-F. Gabard, and G. Zanon. ProCoSA : a Petri Net based software package for autonomous system supervision. In *ATPN'06 Application and Theory of Petri Nets and Other Models of Concurrency*, 2006.

[15] A. Barrett. Domain compilation for embedded real-time planning. In *Proc. of the ICAPS-03 Workshop on Plan Execution*, 2003.

[16] G. Beaumet, G. Verfaillie, and M.-C. Charmeau. Feasibility of autonomous decision making on board an agile Earth-observing satellite. *Computational Intelligence*, 27(1) :123–139, 2011.

[17] M. Benedetti. Quantifier Trees for QBF. In *Proc. of the 8th International Conference on Theory and Applications of Satisfiability Testing (SAT-05)*, 2005.

[18] M. Benedetti, A. Lallouet, and J. Vautard. QCSP made practical by virtue of restricted quantification. In *Proc. of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 38–43, 2007.

[19] T. Benoist, B. Estellon, F. Gardi, R. Megel, and K. Nouioua. Localsolver 1.x : a black-box local-search solver for 0-1 programming. *4OR : A Quarterly Journal of Operations Research*, 9(3) :299–316, 2011.

[20] E. Bensana, G. Verfaillie, J. Agnèse, N. Bataille, and D. Blumstein. Exact and Approximate Methods for the Daily Management of an Earth Observation Satellite. In *Proc. of the 4th International Symposium on Space Mission Operations and Ground Data Systems (SpaceOps-96)*, 1996.

[21] G. Berry and G. Gonthier. The Esterel Synchronous Programming Language : Design, Semantics, Implementation. *Science of Computer Programming*, 19(2) :87–152, 1992.

[22] U. Bertelé and F. Brioschi. *Nonserial Dynamic Programming*. Academic Press, 1972.

[23] P. Bertoli, A. Cimatti, M. Roveri, and P. Traverso. Planning in Nondeterministic Domains under Partial Observability via Symbolic Model Checking. In *Proc. of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 473–478, 2001.

[24] N. Bianchessi, J. Cordeau, J. Desrosiers, G. Laporte, and V. Raymond. A heuristic for the multi-satellite, multi-orbit and multi-user management of Earth observation satellites. *European Journal of Operational Research*, 177(2) :750–762, 2007.

[25] J. Bibaï, P. Savéant, M. Schoenauer, and V. Vidal. An evolutionary metaheuristic based on state decomposition for domain-independent satisficing planning. In *Proc. of the 20th International Conference on Automated Planning and Scheduling (ICAPS'10)*, pages 18–25, 2010.

* [26] P. Bieber, J.-P. Bodeveix, C. Castel, D. Doose, M. Filali, F. Minot, and C. Pralet. Constraint-based design of avionics platform - preliminary design exploration. In *Proc. of 4th European Congress Embedded Real Time Software (ERTS-08)*, 2008.

[27] S. Bistarelli, U. Montanari, F. Rossi, T. Schiex, G. Verfaillie, and H. Fargier. Semiring-Based CSPs and Valued CSPs : Frameworks, Properties and Comparison. *Constraints*, 4(3) :199–240, 1999.

[28] J. Blazewicz, J. Lenstra, and A. Kan. Scheduling subject to resource constraints : classification and complexity. *Discrete Applied Mathematics*, 5(1) :11–24, 1983.

[29] R. Bloem, A. Cimatti, K. Greimel, R. Könighofer, M. Roveri, V. Schuppan, and R. Seeber. RATSY - A new requirements analysis tool with synthesis. In *Proc. of the 22nd International Conference on Computer Aided Verification (CAV-10)*, pages 425–429, 2010.

[30] B. Bonet and H. Geffner. Planning with incomplete information as heuristic search in belief space. In *Proc. of the 5th International Conference on Artificial Intelligence Planning and Scheduling (AIPS-00)*, pages 52–61, 2000.

[31] B. Bonet and H. Geffner. Heuristics for planning with penalties and rewards using compiled knowledge. In *Proc. of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR-06)*, pages 452–462, 2006.

[32] B. Bonet, H. Palacios, and H. Geffner. Automatic Derivation of Memoryless Policies and Finite-State Controllers Using Classical Planners. In *Proc. of the 19th International Conference on Automated Planning and Scheduling (ICAPS-09)*, 2009.

[33] L. Bordeaux and E. Monfroy. Beyond NP : Arc-consistency for Quantified Constraints. In *Proc. of the 8th International Conference on Principles and Practice of Constraint Programming (CP-02)*, pages 371–386, 2002.

[34] C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic Dynamic Programming with Factored Representations. *Artificial Intelligence*, 121(1-2) :49–107, 2000.

[35] S. Bouveret. *Fair allocation of indivisible items : modeling, computational complexity and algorithmics*. Thèse de doctorat, ISAE, Toulouse, France, 2007.

[36] R. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8) :677–691, 1986.

[37] G. Casanova, C. Pralet, and C. Lesire. Managing dynamic multi-agent simple temporal network. Submitted to *the 15th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-15)*, 2015. *

[38] R. Cervoni, A. Cesta, and A. Oddi. Managing dynamic temporal constraint networks. In *Proc. of the 2nd International Conference on Artificial Intelligence Planning Systems (AIPS-94)*, pages 13–18, 1994.

[39] A. Cesta, G. Cortellessa, S. Fratini, A. Oddi, and N. Policella. An Innovative Product for Space Mission Planning : An A Posteriori Evaluation. In *Proc. of the 17th International Conference on Automated Planning and Scheduling (ICAPS-07)*, 2007.

[40] A. Cesta and A. Oddi. Gaining efficiency and flexibility in the simple temporal problem. In *Proc. of the 3rd International Workshop on Temporal Representation and Reasoning (TIME-96)*, pages 45–50, 1996.

[41] A. Cesta and S. Fratini. The timeline representation framework as a planning and scheduling software development environment. In *Proc. of the 27th PlanSIG-08 Workshop*, 2008.

[42] A. Cesta, A. Oddi, and S. F. Smith. Iterative flattening : A scalable method for solving multi-capacity scheduling problems. In *Proc. of the 17th National Conference on Artificial Intelligence (AAAI-00)*, pages 742–747, 2000.

[43] R. Chellappa and A. Jain. *Markov Random Fields : Theory and Applications*. Academic Press, 1993.

[44] T. Cheng, Q. Ding, and B. Lin. A concise survey of scheduling with time-dependent processing times. *European Journal of Operational Research*, 152 :1–13, 2004.

[45] S. Chien, R. Knight, A. Stechert, R.Sherwood, and G. Rabideau. Using Iterative Repair to Improve the Responsiveness of Planning and Scheduling. In *Proc. of the 5th International Conference on Artificial Intelligence Planning and Scheduling (AIPS-00)*, 2000.

[46] S. Chien, R. Sherwood, D. Tran, B. Cichy, G. Rabideau, R. Castano, A. Davies, R. Lee, D. Mandl, S. Frye, B. Trout, J. Hengemihle, J. D'Agostino, S. Shulman, S. Ungar, T. Brakke, D. Boyer, J. Van-Gaasbeck, R. Greeley, T. Doggett, V. Baker, J. Dohm, and F. Ip. The EO-1 Autonomous Science Agent. In *Proc. of the 3rd Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-04)*, 2004.

[47] S. Chien, R. Sherwood, D. Tran, B. Cichy, G. Rabideau, R. Castano, A. Davies, D. Mandl, S. Frye, B. Trout, S. Shulman, and D. Boyer. Using autonomy flight software to improve science return on Earth Observing One. *Journal of Aerospace Computing, Information, and Communication*, 2 :196–216, 2005.

[48] S. A. Chien, M. Johnston, J. Frank, M. Giuliano, A. Kavelaars, C. Lenzen, and N. Policella. A generalized timeline representation, services, and interface for automating space mission operations. In *Proc. of the 12th International Conference on Space Operations (SpaceOps-12)*, 2012.

[49] CHOCO development team. CHOCO. http ://www.emn.fr/z-info/choco-solver/.

[50] E. K. P. Chong, R. L. Givan, and H. S. Chang. A framework for simulation-based network control via hindsight optimization. In *Proceedings of the 39th IEEE Conference on Decision and Control (CDC-00)*, pages 1433–1438, 2000.

[51] F. Chu and J. Halpern. Great Expectations. Part I : On the Customizability of Generalized Expected Utility. In *Proc. of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, 2003.

[52] A. Cimatti, M. Pistore, M. Roveri, and P. Traverso. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence*, 147(1-2) :35–84, 2003.

[53] A. Cimatti, F. Giunchiglia, E. Giunchiglia, and P. Traverso. Planning via model checking : A decision procedure for AR. In *Proc. of the 4th European Conference on Planning (ECP-97)*, pages 130–142, 1997.

[54] A. Cimatti, M. Roveri, and P. Traverso. Automatic OBDD-based generation of universal plans in non-deterministic domains. In *Proc. of the 15th National Conference on Artificial Intelligence (AAAI-98)*, pages 875–881, 1998.

[55] A. Cimatti, M. Roveri, and P. Traverso. Strong planning in non-deterministic domains via model checking. In *Proc. of the 4th Conference on Artificial Intelligence Planning Systems (AIPS-98)*, pages 36–43, 1998.

[56] R. Creasey and F. Teston. Project for Onboard Autonomy : PROBA. In *Proc. of the ESA Workshop on On-Board Autonomy*, 2001.

[57] A. Darwiche. Recursive Conditioning. *Artificial Intelligence*, 126(1-2) :5–41, 2001.

[58] A. Darwiche and P. Marquis. A Knowledge Compilation Map. *Artificial Intelligence*, 17 :229–264, 2002.

[59] T. Dean and K. Kanazawa. A Model for Reasoning about Persistence and Causation. *Computational Intelligence*, 5(3) :142–150, 1989.

[60] R. Dechter. Bucket Elimination : a Unifying Framework for Reasoning. *Artificial Intelligence*, 113(1-2) :41–85, 1999.

[61] R. Dechter and D. Larkin. Hybrid Processing of Beliefs and Constraints. In *Proc. of the 17th International Conference on Uncertainty in Artificial Intelligence (UAI-01)*, pages 112–119, 2001.

[62] R. Dechter and R. Mateescu. AND/OR Search Spaces for Graphical Models. *Artificial Intelligence*, 171(2-3) :73–106, 2007.

[63] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49 :61–95, 1991.

[64] M. Do and S. Kambhampati. Planning as Constraint Satisfaction : Solving the Planning-Graph by Compiling it into CSP. *Artificial Intelligence*, 132(2) :151–182, 2001.

[65] D. Dubois and H. Prade. Possibility Theory : An Approach to Computerized Processing of Uncertainty. Plenum Press, 1988.

[66] D. Dubois and H. Prade. Possibility Theory as a Basis for Qualitative Decision Theory. In *Proc. of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1925–1930, 1995.

[67] S. Edelkamp and J. Hoffman. PDDL2.2 : The language for the classical part of the 4th international planning competition. Technical Report No. 195, 2004.

[68] J. Eker, J. Janneck, E. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong. Taming Heterogeneity : the Ptolemy Approach. *Proceedings of the IEEE*, 91(1) :127–144, 2003.

[69] E. Emerson. Temporal and Modal Logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B :Formal Models and Semantics*, pages 995–1072. Elsevier, 1990.

[70] K. Erol, J. Hendler, and D. S. Nau. HTN planning : Complexity and expressivity. In *Proc. of the 12th National Conference on Artificial Intelligence (AAAI-94)*, pages 1123–1128, 1994.

[71] B. Estellon, F. Gardi, and K. Nouioua. High-performance local search for task scheduling with human resource allocation. In *Proc. of the 2nd International Workshop on Engineering Stochastic Local Search Algorithms (SLS-09)*, pages 1–15, 2009.

[72] H. Fargier, J. Lang, R. Martin-Clouaire, and T. Schiex. A Constraint Satisfaction Framework for Decision under Uncertainty. In *Proc. of the 11th International Conference on Uncertainty in Artificial Intelligence (UAI-95)*, 1995.

[73] H. Fargier, J. Lang, and T. Schiex. Mixed Constraint Satisfaction : a Framework for Decision Problems under Incomplete Knowledge. In *Proc. of the 13th National Conference on Artificial Intelligence (AAAI-96)*, pages 175–180, 1996.

[74] H. Fargier and M.-C. Vilarem. Compiling CSPs into tree-driven automata for interactive solving. *Constraints*, 9(4) :263–287, 2004.

[75] T. Feo and M. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6 :109–133, 1995.

[76] R. Fikes and N. Nilsson. STRIPS : a new approach to the application of theorem proving. *Artificial Intelligence*, 2(3-4) :189–208, 1971.

[77] M. Fox and D. Long. PDDL2.1 : An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20 :61–124, 2003.

[78] M. Fox and D. Long. Modelling mixed discrete-continuous domains for planning. *Journal of Artificial Intelligence Research*, 27 :235–297, 2006.

[79] J. Frank and A. Jónsson. Constraint-based attribute and interval planning. *Constraints*, 8(4) :339–364, 2003.

[80] N. Friedman and J. Halpern. Plausibility Measures : A User's Guide. In *Proc. of the 11th International Conference on Uncertainty in Artificial Intelligence (UAI-95)*, pages 175–184, 1995.

[81] M. Frydenberg. The Chain Graph Markov Property. *Scandinavian Journal of Statistics*, 17 :333–353, 1990.

[82] V. Gabrel, A. Moulet, C. Murat, and V. Paschos. A New Single Model and Derived Algorithms for the Satellite Shot Planning Problem Using Graph Theory Concepts. *Annals of Operations Research*, 69 :115–134, 1997.

[83] V. Gabrel and D. Vanderpooten. Enumeration and interactive selection of efficient paths in a multiple criteria graph for scheduling an Earth observing satellite. *European Journal of Operational Research*, 139 :533–542, 2002.

[84] S. Gawiejnowicz. *Time-Dependent Scheduling*. Springer, 2008.

[85] I. P. Gent, P. Nightingale, and A. Rowley. Encoding quantified CSPs as quantified boolean formulae. In *Proc. of the 16th European Conference on Artificial Intelligence (ECAI-04)*, pages 176–180, 2004.

[86] I. P. Gent, P. Nightingale, and K. Stergiou. QCSP-Solve : A solver for quantified constraint satisfaction problems. In *Proc. of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 138–143, 2005.

[87] A. Gerevini, A. Perini, and F. Ricci. Incremental algorithms for managing temporal constraints. In *Proceedings of the 8th IEEE International Conference on Tools with Artificial Intelligence (ICTAI-96)*, pages 360–365, 1996.

[88] A. Gerevini, A. Saetti, and I. Serina. Planning through stochastic local search and temporal action graphs in LPG. *Journal of Artificial Intelligence Research*, 20(1) :239–290, 2003.

[89] A. Gerevini and D. Long. Preferences and soft constraints in PDDL3. In *Proc. of the ICAPS-2006 Workshop on Preferences and Soft Constraints in Planning*, pages 46–54, 2006.

[90] M. Geyer, F. Mrowka, and C. Lenzen. TerraSAR-X/TanDEM-X mission planning - handling satellites in close formation. In *Proc. of the 11th International Conference on Space Operations (SpaceOps-10)*, 2010.

[91] M. Ghallab and H. Laruelle. Representation and control in ixtet, a temporal planner. In *Proc. of the 12th National Conference on Artificial Intelligence (AAAI-94)*, pages 61–67, 1994.

[92] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning : Theory and Practice*. Morgan Kaufmann, 2004.

[93] P. Giang and P. Shenoy. A Qualitative Linear Utility Theory for Spohn's Theory of Epistemic Beliefs. In *Proc. of the 16th International Conference on Uncertainty in Artificial Intelligence (UAI-00)*, pages 220–229, 2000.

[94] M. Ginsberg. Dynamic Backtracking. *Journal of Artificial Intelligence Research*, 1 :25–46, 1993.

[95] A. Globus, J. Crawford, J. Lohn, and R. Morris. Scheduling Earth Observing Fleets Using Evolutionary Algorithms : Problem Description and Approach. In *Proc. of the 3rd NASA International Workshop on Planning and Scheduling for Space*, 2002.

[96] A. Globus, J. Crawford, J. Lohn, and R. Morris. A comparison of techniques for scheduling Earth observing satellites. In *Proc. of the 16th Conference on Innovative Applications of Articial Intelligence (IAAI-04)*, 2004.

[97] A. Globus, J. Crawford, J. Lohn, and A. Pryor. Scheduling Earth observing satellites with evolutionary algorithms. In *Proc. of the 1st International Conference on Space Mission Chalenges for Information Technology (SMC-IT-03)*, 2003.

[98] F. Glover and M. Laguna. Tabu Search. In *Modern Heuristic Techniques for Combinatorial Problems*, pages 70–141. Blackwell Scientific Publishing, 1993.

[99] D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley Publishing Company, 1989.

[100] L. Granvilliers and F. Benhamou. Algorithm 852 : Realpaver : an interval solver using constraint satisfaction techniques. *ACM Transactions on Mathematical Software*, 32(1) :138–156, 2006.

[101] R. Grasset-Bourdel, G. Verfaillie, and A. Flipo. Building a really executable plan for a constellation of agile Earth observation satellites. In *Proc. of the 7th International Workshop on Planning and Scheduling for Space (IWPSS-11)*, 2011.

[102] T. Hadzic, J. N. Hooker, B. O. Sullivan, and P. Tiedemann. Approximate compilation of constraints into multivalued decision diagrams. In *Proc. of the 14th International Conference on Principles and Practice of Constraint Programming (CP-08)*, pages 448–462, 2008.

[103] B. Haeupler, T. Kavitha, R. Mathew, S. Sen, and R. E. Tarjan. Incremental cycle detection, topological ordering, and strong component maintenance. *ACM Transactions on Algorithms*, 8(1) :Article 3, 2012.

[104] N. Hall and M. Magazine. Maximizing the Value of a Space Mission. *European Journal of Operational Research*, 78 :224–241, 1994.

[105] J. Halpern. Conditional Plausibility Measures and Bayesian Networks. *Journal of Artificial Intelligence Research*, 14 :359–389, 2001.

[106] P. V. Hentenryck and R. Bent. *Online Stochastic Combinatorial Optimization.* MIT Press, 2006.

[107] P. V. Hentenryck and L. Michel. *Constraint-Based Local Search.* The MIT Press, 2005.

[108] T. Henzinger. The theory of hybrid automata. In *Proc. of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS-96)*, pages 278–292, 1996.

[109] J. Hoey, R. St-Aubin, A. J. Hu, and C. Boutilier. SPUDD : Stochastic planning using decision diagrams. In *Proc. of the 15th Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 279–288, 1999.

[110] J. Hoffmann and B. Nebel. The ff planning system : Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14 :253–302, 2001.

[111] C. Honvault, C. Simon, P. David, and E. Bornschlegl. An Autonomous On-board Mission Manager for LEO Satellite Powered by the ERC32SC. In *Proc. of the ESA Workshop on On-Board Autonomy*, 2001.

[112] R. Howard and J. Matheson. Influence Diagrams. In *Readings on the Principles and Applications of Decision Analysis*, pages 721–762. Strategic Decisions Group, Menlo Park, CA, USA, 1984.

[113] J. Huang and A. Darwiche. DPLL with a trace : From SAT to knowledge compilation. In *Proc. of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 156–162, 2005.

[114] S. E. Hudson. Incremental attribute evaluation : A flexible algorithm for lazy update. *ACM Trans. Program. Lang. Syst.*, 13(3) :315–341, 1991.

[115] Ilog. IBM ILOG CPLEX and CpOptimizer, http ://www-03.ibm.com/software/products/.

* [116] G. Infantes, C. Lesire, and C. Pralet. Multi-robot planning and execution for an exploration mission : a case study. In *ICAPS-14 Workshop on "Planning and Robotics" (PlanRob-14)*, pages 49–58, 2014.

[117] J. Jang, J. Choib, H.-J. Bae, and I.-C. Choi. Image collection planning for KOrea Multi-Purpose SATellite-2. *European Journal of Operational Research*, 230(1) :190–199, 2013.

[118] P. Jégou and C. Terrioux. Hybrid Backtracking bounded by Tree-decomposition of Constraint Networks. *Artificial Intelligence*, 146(1) :43–75, 2003.

[119] F. Jensen, F. Jensen, and S. Dittmer. From Influence Diagrams to Junction Trees. In *Proc. of the 10th International Conference on Uncertainty in Artificial Intelligence (UAI-94)*, pages 367–373, 1994.

[120] B. Jobstmann, S. Galler, M. Weiglhofer, and R. Bloem. Anzu : A Tool for Property Synthesis. In *Proc. of the 19th International Conference on Computer Aided Verification (CAV-07)*, pages 258–262, 2007.

[121] D. E. Joslin and D. P. Clements. "squeaky wheel" optimization. *Journal of Artificial Intelligence Research*, 10(1) :353–373, 1999.

[122] N. Jussien and O. Lhomme. Local Search with Constraint Propagation and Conflict-based Heuristics. *Artificial Intelligence*, 139 :21–45, 2002.

[123] L. Kaelbling, M. Littman, and A. Cassandra. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, 101(1-2) :99–134, 1998.

[124] T. Kam, T. Villa, R. Brayton, and A. Sangiovanni-Vincentelli. Multi-valued decision diagrams : Theory and applications. *International Journal on Multiple-Valued Logic*, 4 :9–62, 1998.

[125] H. Kautz and B. Selman. Planning as satisfiability. In *Proc. of the 10th European Conference on Artificial Intelligence (ECAI-92)*, pages 359–363, 1992.

[126] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by Simulated Annealing. *Science*, 220, 1983.

[127] D. Knuth and R. Moore. An Analysis of Alpha-Beta Pruning. *Artificial Intelligence*, 8(4) :293–326, 1975.

[128] L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. In *Proc. of the 17th European Conference on Machine Learning (ECML-06)*, pages 282–293, 2006.

[129] S. Koenig, M. Likhachev, and D. Furcy. Lifelong planning A*. *Artificial Intelligence*, 155(1-2) :93–146, 2004.

[130] J. Kolhas. *Information Algebras : Generic Structures for Inference*. Springer, 2003.

[131] N. Kushmerick, S. Hanks, and D. Weld. An Algorithm for Probabilistic Planning. *Artificial Intelligence*, 76(1-2) :239–286, 1995.

[132] J. Kvarnström and P. Doherty. TALplanner : A Temporal Logic Based Forward Chaining Planner. *Annals of Mathematics and Artificial Intelligence*, 30 :119–169, 2001.

[133] P. Laborie and M. Ghallab. Planning with sharable resource constraints. In *Proc. of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1643–1649, 1995.

[134] J.-M. Lachiver. Pléiades : operational programming first results. In *Proc. of the 12th International Conference on Space Operations (SpaceOps-12)*, 2012.

[135] R. D. Landtsheer. *OscaR.cbls : a Constraint-Based Local Search Engine*, 2012.

[136] R. Larson and J. Casti. *Principles of Dynamic Programming*. Marcel Dekker Inc., 1978.

[137] C. Lecoutre, L. Saïs, S. Tabary, and V. Vidal. Transposition tables for constraint satisfaction. In *Proc. of the 22nd AAAI Conference on Artificial Intelligence (AAAI-07)*, pages 243–248, 2007.

[138] S. Lee and M. F. Younis. EQAR : Effective QoS-aware relay node placement algorithm for connecting disjoint wireless sensor subnetworks. *IEEE Transactions on Computers*, 60(12) :1772–1787, 2011.

[139] M. Lemaître and G. Verfaillie. Daily management of an Earth observation satellite : comparison of ILOG solver with dedicated algorithms for valued constraint satisfaction problems. In *Proc. of the 3rd ILOG International Users Meeting*, 1997.

[140] M. Lemaître and G. Verfaillie. Interaction between reactive and deliberative tasks for on-line decision-making. In *CP/ICAPS Workshop on Planning and Plan Execution for Real-World Systems*, 2007.

[141] M. Lemaître, G. Verfaillie, F. Jouhaud, J.-M. Lachiver, and N. Bataille. Selecting and scheduling observations of agile satellites. *Aerospace Science and Technology*, 6 :367–381, 2002.

[142] M. Lemaître, G. Verfaillie, C. Pralet, and G. Infantes. Synthèse de contrôleur simplement valide dans le cadre de la programmation par contraintes. In *5èmes Journées Francophones de Planification, Décision et Apprentissage pour la conduite de systèmes (JFPDA-10)*, 2010. *

[143] C. Lenzen, M. T. Wörle, F. Mrowka, and M. P. Geyer. Automated scheduling for the TerraSAR-X/TanDEM-X. In *Proc. of the 7th International Workshop on Planning and Scheduling for Space (IWPSS-11)*, 2011.

[144] W. Lin, D. Liao, C. Liu, and Y. Lee. Daily imaging scheduling of an Earth observation satellite. *IEEE Transactions on Systems, Man, and Cybernetics*, 35(2) :213–223, 2005.

[145] M. Littman, S. Majercik, and T. Pitassi. Stochastic Boolean Satisfiability. *Journal of Automated Reasoning*, 27(3) :251–296, 2001.

[146] M. Llibre. Programme commun ONERA/CNES AGATA Autonomie des sytèmes spatiaux - Tâche 3.2 : Constitution d'une bibliothèque de calculs spatiaux. Rapport technique 8/14013 DCSD, ONERA, 2009.

[147] LpSolve development team. LpSolve. http ://lpsolve.sourceforge.net/5.5/.

[148] A. Mackworth. Consistency in Networks of Relations. *Artificial Intelligence*, 8(1) :99–118, 1977.

* [149] A. Maillard, C. Pralet, G. Verfaillie, J. Jaubert, and T. Desmousceaux. Flexible planning for agile Earth-observing satellites. In *Proc. of the 12th International Symposium on Artificial Intelligence, Robotics, and Automation in Space (i-SAIRAS-14)*, 2014.

* [150] A. Maillard, G. Verfaillie, C. Pralet, J. Jaubert, and T. Desmousceaux. Building flexible data download schedules for agile Earth-observing satellites. In *15ème congrès annuel de la Société française de recherche opérationnelle et d aide à la décision (ROADEF-14)*, 2014.

* [151] A. Maillard, C. Pralet, J. Jaubert, I. Sebbag, F. Fontanari, and J. L'Hermitte. Ground and board decision-making on data downloads. To appear in *25th International Conference on Automated Planning and Scheduling (ICAPS-15)*, 2015.

[152] N. Mamoulis and K. Stergiou. Algorithms for quantified constraint satisfaction problems. In *Proc. of the 10th International Conference on Principles and Practice of Constraint Programming (CP-04)*, pages 752–756, 2004.

[153] C. Mancel. A column generation approach for Earth observing satellites. In *Proc. of the 2nd Operational Research Peripatetic Postgraduate Programme (ORP3-03)*, 2003.

[154] E. Maurer, F. Mrowka, A. Braun, M. Geyer, C. Lenzen, Y. Wasser, and M. Wickler. TerraSAR-X mission planning system : Automated command generation for spacecraft operations. *IEEE Transactions on Geoscience and Remote Sensing*, 48(2) :642–648, 2010.

[155] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL, the Planning Domain Definition Language. Technical report, Yale Center for Computational Vision and Control, 1998.

[156] L. Mercier and P. V. Hentenryck. Edge finding for cumulative scheduling. *INFORMS Journal on Computing*, 20(1) :143–153, 2008.

[157] N. Meuleau, L. Peshkin, and L. Kaelbling. Learning Finite-State Controllers for Partially Observable Environments. In *Proc. of the 15th International Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 427–436, 1999.

[158] L. Michel and P. V. Hentenryck. Localizer. *Constraints*, 5(1-2) :43–84, 2000.

[159] S. Mittal and B. Falkenhainer. Dynamic Constraint Satisfaction Problems. In *Proc. of the 8th National Conference on Artificial Intelligence (AAAI-90)*, pages 25–32, 1990.

[160] G. Monahan. A Survey of Partially Observable Markov Decision Processes : Theory, Models, and Algorithms. *Management Science*, 28(1) :1–16, 1982.

[161] P. Morris, N. Muscettola, and T. Vidal. Dynamic Control of Plans with Temporal Uncertainty. In *Proc. of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 494–499, 2001.

[162] P. Morris, R. Morris, L. Khatib, S. Ramakrishnan, and A. Bachmann. Strategies for global optimization of temporal preferences. In *Proc. of the 10th International Conference on Principles and Practices of Constraint Programming (CP-04)*, pages 408–422. Springer, 2004.

[163] P. Mukhija, K. M. Krishna, and V. Krishna. A two phase recursive tree propagation based multi-robotic exploration framework with fixed base station constraint. In *Proc. of IROS-10*, 2010.

[164] T. Murata. Petri Nets : Properties, Analysis and Applications. *Proc. of the IEEE*, 77(4) :541–580, 1989.

[165] N. Muscettola. Computing the Envelope for Stepwise-Constant Resource Allocation. In *Proc. of the 8th International Conference on Principles and Practice of Constraint Programming (CP-02)*, pages 139–154, 2002.

[166] N. Muscettola, P. Nayak, B. Pell, and B. Williams. Remote Agent : To Boldly Go Where No AI System Has Gone Before. *Artificial Intelligence*, 103(1-2) :5–48, 1998.

[167] P. Ndilikilikesha. Potential Influence Diagrams. *International Journal of Approximated Reasoning*, 10 :251–285, 1994.

[168] G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization.* John Wiley & Sons, 1988.

[169] M. H. Newton, D. Pham, A. Sattar, and M. Maher. Kangaroo : An efficient constraint-based local search system using lazy propagation. In *Proc. of CP-11*, pages 645–659, 2011.

[170] P. Nightingale. Non-binary quantified CSP : Algorithms and modelling. *Constraints*, 14(4) :539–581, 2009.

[171] M. Nilsson, J. Kvarnström, and P. Doherty. EfficientIDC : A Faster Incremental Dynamic Controllability Algorithm. In *24th International Conference on Automated Planning and Scheduling (ICAPS-14)*, pages 199–207, 2014.

[172] A. Niveau. *Knowledge Compilation for Online Decision-Making : Application to the Control of Autonomous Systems.* Thèse de doctorat, Université Paul Sabatier, Toulouse, France, 2012.

[173] A. Niveau, H. Fargier, and C. Pralet. Representing CSPs with set-labeled diagrams : A compilation * map. In *Proc. of the 2nd International Workshop on Graph Structures for Knowledge Representation and Reasoning (GKR-11)*, pages 137–171, 2011.

[174] A. Niveau, H. Fargier, and C. Pralet. Set-labeled diagrams for CSP compilation. In *Proc. of the 6th* * *Starting AI Researcher's Symposium (STAIRS-12)*, pages 216–227, 2012.

[175] A. Niveau, H. Fargier, C. Pralet, and G. Verfaillie. Handling the ouput of interval-based constraint * solvers by interval automata compilation. In *Proc. of the CP-09 Workshop on Interval Analysis, Constraint Propagation, Applications (intCP-09)*, 2009.

[176] A. Niveau, H. Fargier, C. Pralet, and G. Verfaillie. Using interval automata to represent decision * policies with continuous variables. In *Proc. of the Doctoral Consortium of the 19th International Conference on Automated Planning and Scheduling (ICAPS-09)*, pages 13–16, 2009.

[177] A. Niveau, H. Fargier, C. Pralet, and G. Verfaillie. Knowledge compilation using interval automata * and applications to planning. In *Proc. of the 19th European Conference on Artificial Intelligence (ECAI-10)*, pages 459–464, 2010.

[178] A. Oddi and N. Policella. A max-flow approach for improving robust- ness in a spacecraft downlink schedule. In *Proc. of the 4th International Workshop on Planning and Scheduling for Space (IWPSS-04)*, pages 151–158, 2004.

[179] X. Olive, G. Verfaillie, C. Pralet, S. Rainjonneau, and I. Sebbag. Planning Acquisitions for an Ocean * Global Surveillance Mission. In *Proc. of the 11th International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS-12)*, 2012.

[180] A. Pal, R. Tiwari, and A. Shukla. Communication constraints multi-agent territory exploration task. *Applied Intelligence*, 38(3) :357–383, 2013.

[181] H. Palacios, B. Bonet, A. Darwiche, and H. Geffner. Pruning conformant plans by counting models on compiled d-DNNF representations. In *Proc. of the 15th International Conference on Automated Planning and Scheduling (ICAPS-05)*, pages 141–150, 2005.

[182] J. Park and A. Darwiche. Complexity Results and Approximation Strategies for MAP Explanations. *Journal of Artificial Intelligence Research*, 21 :101–133, 2004.

[183] F. Patrizi, N. Lipovetzky, and H. Geffner. Fair LTL synthesis for non-deterministic systems using strong cyclic planners. In *Proc. of the 23rd international Joint Conference on Artificial Intelligence (IJCAI-13)*, pages 2343–2349, 2013.

[184] J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[185] G. D. Penna, D. Magazzeni, F. Mercorio, and B. Intrigila. UPMurphi : A tool for universal planning on PDDL+ problems. In *Proc. of the 19th International Conference on Automated Planning and Scheduling (ICAPS-09)*, pages 106–113, 2009.

[186] M. A. Peot and D. E. Smith. Conditional nonlinear planning. In *Proc. of the 1st International conference on Artificial Intelligence Planning Systems (AIPS-92)*, pages 189–197, 1992.

[187] P. Perny, O. Spanjaard, and P. Weng. Algebraic Markov Decision Processes. In *Proc. of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*, 2005.

[188] C. A. Petri. *Kommunikation mit Automaten.* Phd thesis, Darmstadt University of Technology, Germany, 1962.

[189] Q. D. Pham, Y. Deville, and P. V. Hentenryck. LS(Graph) : a constraint-based local search for constraint optimization on trees and paths. *Constraints*, 17(4) :357–408, 2012.

[190] M. Pinedo. *Scheduling : Theory, Algorithms, and Systems.* Springer, 2012.

[191] N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of reactive(1) designs. In *Proc. of the 7th International Conference on Verification, Model Checking and Abstract Interpretation*, pages 364–380, 2006.

[192] N. Policella, S. Smith, A. Cesta, and A. Oddi. Generating Robust Schedules through Temporal Flexibility. In *Proc. of the 14th International Conference on Automated Planning and Scheduling (ICAPS-04)*, pages 209–218, 2004.

[193] N. Policella, A. Cesta, A. Oddi, and S. F. Smith. From precedence constraint posting to partial order schedules : A CSP approach to robust scheduling. *AI Communications*, 20(3) :163–180, 2007.

[194] P. Poupart and C. Boutilier. Bounded Finite State Controllers. In *Proc. of the 17th Conference on Neural Information Processing Systems (NIPS-03)*, 2003.

* [195] C. Pralet. *A Generic Algebraic Framework for Representing and Solving Sequential Decision Making Problems with Uncertainties, Feasibilities, and Utilities.* PhD thesis, Ecole Nationale Supérieure de l'Aéronautique et de l'Espace, Toulouse, France, 2006.

* [196] C. Pralet, G. Infantes, and G. Verfaillie. Application showcase : a generic constraint-based local search library for the management of an electromagnetic surveillance space mission. In *Proc. of the ICAPS-13 Application showcase*, 2013.

* [197] C. Pralet, G. Infantes, and G. Verfaillie. A generic constraint-based local search library for the management of an electromagnetic surveillance space mission. In *ICAPS-13 Workshop on "Scheduling and Planning Applications" (SPARK-13)*, 2013.

* [198] C. Pralet, M. Lemaître, G. Verfaillie, and G. Infantes. Controller synthesis for autonomous systems : a constraint-based approach. In *Proc. of the 10th International Symposium on Artificial Intelligence, Robotics, and Automation in Space (i-SAIRAS-10)*, 2010.

* [199] C. Pralet and C. Lesire. Deployment of mobile wireless sensor networks for crisis management : a constraint-based local search approach. In *20th International Conference on Principles and Practice of Constraint Programming (CP-14)*, pages 870–885, 2014.

* [200] C. Pralet, T. Schiex, and G. Verfaillie. Decomposition of Multi-Operator Queries on Semiring-based Graphical Models. In *Proc. of the 12th International Conference on Principles and Practice of Constraint Programming (CP-06)*, pages 437–452, 2006.

* [201] C. Pralet, T. Schiex, and G. Verfaillie. From Influence Diagrams to Multioperator Cluster DAGs. In *Proc. of the 22nd International Conference on Uncertainty in Artificial Intelligence (UAI-06)*, 2006.

* [202] C. Pralet, T. Schiex, and G. Verfaillie. Algorithmes et Complexités Génériques pour Différents Cadres de Décision Séquentielle dans l'Incertain. *Revue d'Intelligence Artificielle*, 21(4) :459–488, 2007.

* [203] C. Pralet and G. Verfaillie. Travelling in the World of Local Searches in the Space of Partial Assignments. In *Proc. of the International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming for Combinatorial Optimisation Problems (CP-AI-OR-04)*, pages 240–255, 2004.

[204] C. Pralet and G. Verfaillie. About the Choice of the Variable to Unassign in a Decision Repair Algorithm. *RAIRO Operations Research*, 39 :55–74, 2005. *

[205] C. Pralet and G. Verfaillie. Decision upon observations and data downloads by an autonomous Earth surveillance satellite. In *Proc. of the 9th International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS-08)*, 2008. *

[206] C. Pralet and G. Verfaillie. Using constraint networks on timelines to model and solve planning and scheduling problems. In *Proc. of the 18th International Conference on Automated Planning and Scheduling (ICAPS-08)*, pages 272–279, 2008. *

[207] C. Pralet and G. Verfaillie. AIMS : A tool for long-term planning of the ESA INTEGRAL mission. In *Proc. of the 6th International Workshop on Planning and Scheduling for Space (IWPSS-09)*, 2009. *

[208] C. Pralet and G. Verfaillie. Forward constraint-based algorithms for anytime planning. In *Proc. of the 19th International Conference on Automated Planning and Scheduling (ICAPS-09)*, 2009. *

[209] C. Pralet and G. Verfaillie. Slice Encoding for Constraint-Based Planning. In *Proc. of the 15th International Conference on Principles and Practice of Constraint Programming (CP-09)*, 2009. *

[210] C. Pralet and G. Verfaillie. Beyond QCSP for solving control problems. In *Proc. of the 17th International Conference on Principles and Practice of Constraint Programming (CP-11)*, pages 744–758, 2011. *

[211] C. Pralet and G. Verfaillie. Time-dependent simple temporal networks. In *Proc. of the 18th International Conference on Principles and Practice of Constraint Programming (CP-12)*, pages 608–623, 2012. *

[212] C. Pralet and G. Verfaillie. Dynamic online planning and scheduling using a static invariant-based evaluation model. In *Proc. of the 23th International Conference on Automated Planning and Scheduling (ICAPS-13)*, 2013. *

[213] C. Pralet and G. Verfaillie. Time-dependent simple temporal networks : Properties and algorithms. *RAIRO Operations Research*, 2013. *

[214] C. Pralet, G. Verfaillie, M. Lemaître, and G. Infantes. Constraint-Based Controller Synthesis in Non-Deterministic and Partially Observable Domains. In *Proc. of the 19th European Conference on Artificial Intelligence (ECAI-10)*, pages 681–686, 2010. *

[215] C. Pralet, G. Verfaillie, X. Olive, S. Rainjonneau, and I. Sebbag. Planning for an ocean global surveillance mission. In *7th International Workshop on Planning and Scheduling for Space (IWPSS-11)*, 2011. *

[216] C. Pralet, G. Verfaillie, X. Olive, S. Rainjonneau, and I. Sebbag. Allocation of downlink windows for a constellation of satellites. In *11th International Symposium on Artificial Intelligence, Robotics, and Automation in Space (i-SAIRAS-12)*, 2012. *

[217] C. Pralet, G. Verfaillie, and T. Schiex. Composite Graphical Models for Reasoning about Uncertainties, Feasibilities, and Utilities. In *Proc. of the CP-05 International Workshop on "Preferences and Soft Constraints"*, 2005. *

[218] C. Pralet, G. Verfaillie, and T. Schiex. Decision with Uncertainties, Feasibilities, and Utilities : Towards a Unified Algebraic Framework. In *Proc. of the 17th European Conference on Artificial Intelligence (ECAI-06)*, pages 427–431, 2006. *

[219] C. Pralet, G. Verfaillie, and T. Schiex. An Algebraic Graphical Model for Decision with Uncertainties, Feasibilities, and Utilities. *Journal of Artificial Intelligence Research*, 29 :421–489, 2007. *

[220] C. Pralet, G. Verfaillie, and T. Schiex. Un Cadre Graphique et Algébrique pour les Problèmes de Décision incluant Incertitudes, Faisabilités et Utilités. *Revue d'Intelligence Artificielle*, 21(3) :419–448, 2007. *

[221] C. Pralet, G. Verfaillllie, and T. Schiex. Belief and Desire Networks for Answering Complex Queries. In *Proc. of the CP-04 Workshop on "Constraint Solving under Change and Uncertainty"*, 2004. *

[222] C. Pralet, T. Schiex, and G. Verfaillie. *Sequential Decision-Making Problems : Representation and Solution*. Wiley-ISTE, 2009. *

* [223] C. Pralet and G. Verfaillie. Combining static and dynamic models for boosting forward planning. In *Proc. of the 9th International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming for Combinatorial Optimisation Problems (CPAIOR-12)*, pages 322–338, 2012.

* [224] C. Pralet, G. Verfaillie, A. Maillard, E. Hébrard, N. Jozefowiez, M.-J. Huguet, T. Desmousceaux, P. Blanc-Paques, and J. Jaubert. Satellite data download management with uncertainty about the generated volumes. In *24th International Conference on Automated Planning and Scheduling (ICAPS-14)*, pages 430–438, 2014.

[225] P. Prosser. Hybrid Algorithms for the Constraint Satisfaction Problems. *Computational Intelligence*, 9(3) :268–299, 1993.

[226] L. Pryor and G. Collins. Planning for contingencies : A decision-based approach. *Journal of Artificial Intelligence Research*, 4 :287–339, 1996.

[227] M. Puterman. *Markov Decision Processes, Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

[228] L. Quesada, K. Brown, B. O'Sullivan, L. Sitanayah, and C. Sreenan. A constraint programming approach to the additional relay placement problem in wireless sensor networks. In *Proc. of ICTAI-13*, pages 1052 – 1059, 2013.

[229] G. Rabideau, R. Knight, S. Chien, A. Fukunaga, and A. Govindjee. Iterative Repair Planning for Spacecraft Operations Using the Aspen System. In *Proc. of the 5th International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS-99)*, pages 99–106, 1999.

[230] P. Ramadge and W. Wonham. The Control of Discrete Event Systems. *Proc. of the IEEE*, 77(1) :81–98, 1989.

[231] G. Righini and E. Tresoldi. A mathematical programming solution to the Mars Express memory dumping problem. *IEEE Transaction on Systems, Man, and Cybernetics, Part C*, 40(3) :268–277, 2010.

[232] J. Rintanen. Constructing conditional plans by a theorem-prover. *Journal of Artificial Intelligence Research*, 10 :323–352, 1999.

[233] L. Roditty and U. Zwick. Improved dynamic reachability algorithms for directed graphs. *SIAM Journal on Computing*, 37(5) :1455–1471, 2008.

[234] R. Rossi, P. V. Beek, and T. Walsh, editors. *Handbook of Constraint Programming*. Elsevier, 2006.

[235] SAT4J development team. SAT4J. http ://www.sat4j.org/.

[236] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[237] P. Shenoy. Valuation-based Systems for Discrete Optimization. In *Proc. of the 6th International Conference on Uncertainty in Artificial Intelligence (UAI-90)*, pages 385–400, 1990.

[238] P. Shenoy. Valuation-based Systems for Bayesian Decision Analysis. *Operations Research*, 40(3) :463–484, 1992.

[239] I. Shu, R. Effinger, and B. Williams. Enabling fast flexible planning through incremental temporal reasoning with conflict extraction. In *Proc. of the 15th International Conference on Automated Planning and Scheduling (ICAPS-05)*, pages 252–261, 2005.

[240] D. Smith, J. Frank, and W. Cushing. The ANML language. In *Proc. of the 18th International Conference on Automated Planning and Scheduling (ICAPS-08)*, 2008.

[241] D. Smith and D. Weld. Conformant Graphplan. In *Proc. of the 15th National Conference on Artificial Intelligence (AAAI-98)*, pages 889–896, 1998.

[242] W. Spohn. A General Non-Probabilistic Theory of Inductive Reasoning. In *Proc. of the 6th International Conference on Uncertainty in Artificial Intelligence (UAI-90)*, pages 149–158, 1990.

[243] J. Stedl and B. Williams. A fast incremental dynamic controllability algorithm. In *Proceedings of the ICAPS Workshop on Plan Execution*, 2005.

[244] R. Steel, M. Niezette, A. Cesta, S. Fratini, A. Oddi, G. Cortellessa, R. Rasconi, G. Verfaillie, C. Pralet, M. Lavagna, A. Brambilla, F. Castellini, A. Donati, and N. Policella. Advanced planning and scheduling initiative : MrSpock AIMS for XMAS in the space domain. In *Proc. of the IJCAI-09 Workshop on Artificial Intelligence in Space*, 2009. *

[245] A. Stentz. Optimal and efficient path planning for partially-known environments. In *Proc. of the International Conference on Robotics and Automation (ICRA-94)*, pages 3310–3317, 1994.

[246] K. Stergiou and M. Koubarakis. Backtracking algorithms for disjunctions of temporal constraints. *Artificial Intelligence*, 120 :81–117, 2000.

[247] K. Stergiou. Repair-based methods for quantified CSPs. In *Proc. of the 11th International Conference on Principles and Practice of Constraint Programming (CP-05)*, pages 652–666, 2005.

[248] F. Teichteil, C. Seguin, and C. Pralet. Validation of decision models for an autonomous Earth surveillance satellite. In *Proc. of the 9th International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS-08)*, 2008. *

[249] F. Teichteil-Königsbuch, U. Kuter, and G. Infantes. Incremental plan aggregation for generating policies in MDPs. In *Proc. of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-10)*, pages 1231–1238, 2010.

[250] D. Tran, S. Chien, G. Rabideau, and B. Cichy. Flight software issues in onboard automated planning : Lessons learned on EO-1. In *Proc. of the 4th International Workshop on Planning and Scheduling for Space (IWPSS-04)*, 2004.

[251] I. Tsamardinos, T. Vidal, and M. Pollack. CTP : A New Constraint-Based Formalism for Conditional, Temporal Planning. *Constraints*, 8(4) :365–388, 2003.

[252] P. van Beek and X. Chen. CPlan : A Constraint Programming Approach to Planning. In *Proc. of the 16th National Conference on Artificial Intelligence (AAAI-99)*, pages 585–590, 1999.

[253] M. Vasquez and J. Hao. A Logic-constrained Knapsack Formulation and a Tabu Algorithm for the Daily Photograph Scheduling of an Earth Observation Satellite. *Journal of Computational Optimization and Applications*, 20(2) :137–157, 2001.

[254] G. Verfaillie and E. Bornschlegl. Designing and Evaluating an On-line On-board Autonomous Earth Observation Satellite Scheduling System. In *Proc. of the 2nd NASA International Workshop on Planning and Scheduling for Space*, pages 122–127, 2000.

[255] G. Verfaillie, G. Infantes, M. Lemaître, N. Théret, and T. Natolot. On-board decision-making on data downloads. In *Proc. of the 7th International Workshop on Planning and Scheduling for Space (IWPSS-11)*, 2011.

[256] G. Verfaillie, M. Lemaître, N. Bataille, and J.-M. Lachiver. 2003 ROADEF challenge : Informal and formal problem description, 2003.

[257] G. Verfaillie and C. Pralet. The Basic Ingredients of a Constraint-based Framework for Decision-making under Uncertainty. In *Proc. of the CP-05 International Workshop on "Constraint solving under Change and Uncertainty"*, 2005. *

[258] G. Verfaillie and C. Pralet. Constraint programming for controller synthesis. In *Proc. of the 17th International Conference on Principles and Practice of Constraint Programming (CP-11)*, pages 100–114, 2011. *

[259] G. Verfaillie and C. Pralet. A timeline, event, and constraint-based modeling framework for planning and scheduling problems. In *Proc. of the 4th Workshop on Knowledge Engineering for Planning and Scheduling (KEPS-13)*, pages 61–68, 2013. *

[260] G. Verfaillie, C. Pralet, and M. Lemaître. Constraint-based modeling of discrete event dynamic systems. *Journal of Intelligent Manufacturing (JIM), Special Issue on Planning, Scheduling, and Constraint Satisfaction*, 2008. *

[261] G. Verfaillie, C. Pralet, and M. Lemaître. How to Model Planning and Scheduling Problems using Timelines. *The Knowledge Engineering Review*, 25(3) :319–336, 2010. *

[262] G. Verger and C. Bessiere. Blocksolve : a bottom-up approach for solving quantified CSPs. In *Proc. of the 12th International Conference on Principles and Practice of Constraint Programming (CP-06)*, pages 635–649, 2006.

[263] T. Vidal and H. Fargier. Handling Contingency in Temporal Constraint Networks : From Consistency to Controllabilities. *Journal of Experimental and Theoretical Artificial Intelligence*, 11(1) :23–45, 1999.

[264] V. Vidal and H. Geffner. Branching and pruning : An optimal temporal POCL planner based on constraint programming. *Artificial Intelligence*, 170 :298–335, 2006.

[265] C. Voudouris, R. Dorne, D. Lesaint, and A. Liret. iOpt : A software toolkit for heuristic search methods. In *Proc. of CP-01*, pages 716–719, 2001.

[266] T. Walsh. Stochastic Constraint Programming. In *Proc. of the 15th European Conference on Artificial Intelligence (ECAI-02)*, pages 111–115, 2002.

[267] W. Wolfe and S. Sorensen. Three Scheduling Algorithms applied to the Earth Observing Systems Domain. *Management Science*, 46(1) :148–168, 2000.

[268] H. Younes and M. Littman. PPDDL : An Extension to PDDL for Expressing Planning Domains with Probabilistic Effects. Technical Report CMU-CS-04-167, Carnegie Mellon University, Pittsburgh, PA, 2004.

[269] M. Younis and K. Akkaya. Strategies and techniques for node placement in wireless sensor networks : A survey. *Ad Hoc Networks*, 6(4) :621–655, 2008.

Approches par contraintes pour la planification et l'ordonnancement: méthodes, outils et applications

Les travaux présentés dans cette HDR relèvent du contrôle de systèmes à l'aide de techniques d'optimisation combinatoire. L'objectif visé d'un point de vue applicatif consiste à produire automatiquement des décisions pour des systèmes tels que des satellites, en prenant en compte les degrés de liberté disponibles, les contraintes à respecter (contraintes sur le temps, contraintes sur les ressources, contraintes sur l'état du système, contraintes opérationnelles...) et les objectifs à optimiser. D'un point de vue technique, les travaux effectués s'appuient sur le cadre de la programmation par contraintes pour définir de nouvelles méthodes de planification et d'ordonnancement capables de fournir des plans d'action pour des engins, dans des contextes pouvant faire intervenir des incertitudes sur certains paramètres de l'environnement, des exigences de décision en temps contraint, ou encore des aspects multi-agents. Les contributions réalisées ont successivement porté sur des modèles de type problèmes de satisfaction de contraintes dynamiques, sur des modèles de type automates basés contraintes, et enfin sur des techniques dites de recherche locale à base de contraintes. Les approches fouillées ont par ailleurs été ou bien des approches réactives, dans lesquelles un plan d'action est produit et réparé/reconstruit en cours de mission en fonction des conditions réelles rencontrées, ou bien des approches proactives, dans lesquelles une politique de décision directement utilisable dans plusieurs situations rencontrables à l'exécution est fournie, ou bien des approches mixtes, dans lesquelles des plans avec un certain degré de flexibilité sont générés. Des études plus fondamentales ont également été conduites en lien avec la problématique générale de la décision séquentielle en intelligence artificielle, avec des modélisations de l'incertitude de diverses natures (probabilistes, possibilistes, ensemblistes...) et avec éventuellement une utilisation de techniques dites de compilation de connaissances. Globalement, la démarche employée est une démarche qui part des applications rencontrées, ces dernières faisant souvent intervenir des spécifications complexes comme des aspects "time-dependent" ou des contraintes sur l'état continu d'un système. Pour ces applications, les algorithmes définis sont souvent des algorithmes de recherche gloutonne ou de recherche locale, aptes à fonctionner sur des problèmes de grande taille. Les solutions développées sont ensuite étendues pour aboutir à des méthodes et outils génériques applicables à une plus grande classe de problèmes.

Mots-clés : OPTIMISATION COMBINATOIRE ; ORDONNANCEMENT ; PLANIFICATION ; RECHERCHE LOCALE ; APPLICATION

Office National d'Études et de Recherches Aérospatiales
2, avenue Edouard Belin - BP 74025
31055 TOULOUSE Cedex 4
Tél. : +33 5 62 25 25 25
http://www.onera.fr