Systèmes multi-robots aériens : architecture pour la planification, la supervision et la coopération

Jérémi Gancet

Thèse

préparée au

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

en vue de l'obtention du

Doctorat de l'Institut National Polytechnique de Toulouse

Spécialité : Systèmes informatiques

par

Jérémi Gancet

Systèmes multi-robots aériens : architecture pour la planification, la supervision et la coopération

---

Soutenance le 29 Septembre 2005 devant le jury :

R.	**ALAMI**	**LAAS-CNRS, Toulouse**	Président
R.	**CHATILA**	**LAAS-CNRS, Toulouse**	Directeur de Thèse
P.	**LIMA**	**IST/UTL, Lisbon**	Rapporteur
A.	**SAFFIOTTI**	**Université d'Örebro**	Rapporteur
S.	**LACROIX**	**LAAS-CNRS, Toulouse**	Examinateur
C.	**TESSIER**	**ONERA, Toulouse**	Examinateur

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

Remerciements

Ce mémoire est la conclusion de quatre années de travail au LAAS-CNRS, à Toulouse, dans le groupe de robotique et d'intelligence artificielle. Je souhaite remercier les personnes qui m'ont permis de réaliser cette thèse dans d'aussi bonnes conditions.

Merci en premier lieu à Malik Ghallab, directeur du laboratoire, pour m'avoir accueilli au LAAS, et merci à Raja Chatila, directeur du groupe RIA et directeur de ma thèse, pour son accueil, son écoute, et aussi pour sa confiance.

Merci à Simon Lacroix, qui a dirigé ma thèse, pour son encadrement, sa disponibilité, son habileté, son savoir-faire qu'il partage sans compter.
Merci à Rachid Alami, qui a co-encadré ma thèse, pour la finesse de ses remarques et de son jugement, qui m'ont beaucoup aidé à prendre du recul lorsqu'il y en avait besoin.

Merci à mes rapporteurs Pedro Lima et Alessandro Saffiotti, pour avoir accepté de faire la revue de ma thèse, pour être venus de loin pour assister à la soutenance, et pour leurs remarques pertinentes.
Merci également à Catherine Tessier, qui a accepté de participer à mon jury en tant qu'examinateur, et dont les retours m'ont donné à réflechir à ce travail sous d'autres perspectives.

Merci aussi à tous mes collègues thésards, tout au long de ces quatres années de labeur : un clin d'oeil aux anciens, Solange, Fabien, Williams, Romain, Emmanuel (G.)... une tape dans le dos à mes "egos", Emmanuel (Z.) et Fred, qui sont eux-même également Docteurs au moment où j'écris ces lignes... et puis tous mes encouragements aux cadets, bien trop nombreux pour ne pas en oublier : une dédicace spéciale pour Seb, pour les bons moments et délires de COMETS, ainsi que pour Sylvain, pour la dure tâche de relecture de ma thèse.

Merci encore aux partenaires du projet COMETS, pour les bons moments passés dans le projet : une pensée particulière à Luis, Fernando, Iker, Fran, Volker, qui sont des gens dont on se souvient longtemps.

Merci enfin à toute ma famille : supporter un thésard pendant quatre ans n'est pas toujours une sinécure. La présence et la disponibilité de mes parents et de ma soeur Maud sont pour beaucoup dans la réussite de cette étape de ma vie.

Merci, Hân, pour ton attention, ta patience, ton amour.

Table des matières

Table des figures

Introduction

Si la robotique mobile est encore loin d'envahir notre quotidien à la façon des humanoïdes du film "I, robot"[1], ses progrès suivent, avec un léger décalage, l'essor fulgurant de l'informatique. L'accélération des avancées se produit encore principalement dans le domaine de la recherche, mais les prémices de la vulgarisation se font ressentir de jour en jour davantage : les applications ludiques se sont vues poussées sur le devant de la scène avec AIBO et ses descendants. Des applications scientifiques ont connu des succès retentissants, dans le domaine spatial en particulier avec les épopées des rovers martiens Spirit et Opportunity[2].

Alors que la robotique, en pleine ébullition, n'en est encore qu'à ses balbutiements en terme de réalisations, une de ses spécialités étudie et anticipe le déploiement conjoint de plusieurs robots : il s'agit de comprendre, de modéliser et éventuellement de bénéficier de la synergie qui naît de l'interaction entre les robots d'un tel système. Les systèmes multi-robots s'appuient sur des notions et des méthodes d'abord étudiées et formalisées dans les sciences humaines, les mathématiques et les sciences économiques au cours du XX^{eme} siècle, puis appliquées et étendues dans le cadre de l'informatique depuis les années 1980 avec les applications réparties et les systèmes multi-agents (intelligence artificielle distribuée). Le cadre de la robotique ajoute une dimension physique et tangible aux problématiques "agents" : les années 1990 ont connu un large développement de ce champ de recherche, très mobilisateur depuis lors.

Contexte et démarche

Le travail et la démarche rapportés dans ce manuscrit s'inscrivent dans cette optique multi-robots, en s'attachant à une branche de la robotique mobile qui connaît un essor tout particulier : celle de la robotique aérienne.

Les systèmes automatisés actuels permettent d'ores et déjà de disposer d'aéronefs démontrant une certaine autonomie opérationnelle : ainsi, de nombreux modèles d'avions de ligne sont potentiellement en mesure de réaliser un vol autonome, du décollage à l'atterrissage. Les motivations militaires ont également amené à mettre en oeuvre des véhicules aériens non-habités pour des missions de reconnaissance aérienne en zone dangereuse, et plus récemment, pour des missions offensives. Cependant, ces systèmes font uniquement preuve d'autonomie opérationnelle, essentiellement pour la navigation par points de passage : aucune autonomie décisionnelle (choix autonome d'actions afin de réaliser des buts) ne leur est déléguée.

Accroître l'autonomie décisionnelle des robots est un objectif omniprésent dans la communauté des chercheurs en robotique. D'abord parce qu'elle ouvre la voie à des applications où

[1] "I, Robot", film de Alex Proyas, 2004

[2] NASA, 2003, Mars exploration rover mission : http ://marsrover.jpl.nasa.gov/home/

l'humain ne peut techniquement pas assurer efficacement le contrôle du robot : la robotique planétaire, par exemple, bénéficierait considérablement de l'augmentation de l'autonomie des robots (une progression notable en terme d'autonomie opérationnelle a permis aux rovers Spirit et Opportunity de naviguer depuis 2003 de façon robuste sur des kilomètres, et non plus des centimètres, comme pour certaines missions précédentes totalement téléopérées[3]. Cependant, aucune autonomie décisionnelle n'est en vue pour le moment). Ensuite, parce qu'elle permet à l'humain de s'abstraire de tâches de contrôle répétitives, ou sujettes à des erreurs de manipulation (erreur humaine).

Cependant, dans le domaine de la robotique de terrain par exemple, le développement de fonctionnalités et capacités autonomes ne doit pas occulter l'importance et le besoin, lorsque c'est techniquement possible, de l'humain aux côtés du robot : d'abord parce que l'humain peut vouloir revendiquer, à tout moment, une certaine forme de contrôle vis à vis du robot. Ensuite, parce que aussi sophistiquée et robuste soit-elle, une machine n'est jamais totalement à l'abri de dysfonctionnements auquel l'humain peut vouloir directement remédier.

Dans les travaux présentés dans ce manuscrit, la place de l'humain est un leitmotiv.

Contributions

Les contributions du présent travail se situent à plusieurs niveaux :

– En premier lieu, **la définition d'une architecture** considérant explicitement une délégation incrémentale de capacités décisionnelles aux robots du système.

– Un **exécutif générique** est proposé afin de rendre transparents, du point de vue des niveaux inférieurs des robots, les changements de configuration de prise de décision.

– Dans les configurations où la prise de décision est distribuée, nous proposons une **couche délibérative** dotée d'outils de planification de tâche adaptés aux tâches des UAV, ainsi que d'un **gestionnaire d'interactions** exploitant des **modèles d'interaction** explicitement définis.

Une partie de ces contributions a été démontrée avec succès dans un système multi-UAV, lors des expérimentations du projet COMETS. L'autre partie a été testée dans des configurations de simulation, et pourra faire l'objet de futures expérimentations.

Aperçu du contenu

Le chapitre I, propose un tour d'horizon de la question multi-robots, ses composantes et problématiques. Nous y proposons également une introduction à la robotique aérienne, dont les spécificités méritent certaines précisions.

Dans le chapitre II, nous définissons un ensemble de notions autour de l'autonomie décisionnelle des robots, utilisées pour jeter les bases d'une architecture qui répond, selon nous, aux défis soulevés dans le premier chapitre. En particulier, deux schémas principaux sont dégagés et déclinés : le premier considère une prise de décision principalement centralisée, nous parlons alors de "bas degrés d'autonomie décisionnelle", pour les UAV. Le second considère une délégation plus poussée de la prise de décision au niveau des UAV. Nous parlons alors de "hauts

[3]NASA, 1997, Sojourner Rover Home Page : http ://mpfwww.jpl.nasa.gov/MPF/rovers/sojourner.html

degrés d'autonomie décisionnelle".

Le chapitre III introduit un exécutif générique, supportant la supervision d'exécution des tâches, proposant des moyens de coordination de bas niveau, et interfaçant la prise de décision (qu'elle soit centralisée ou distribuée) avec le niveau fonctionnel.

Nous présentons ensuite dans le chapitre IV une couche délibérative destinée à une prise de décision distribuée, déléguée aux robots du sytème, le cas échéant ("hauts degrés d'autonomie"). Nous y présentons les mécanismes délibératifs en introduisant les composantes de la couche délibérative : un couple planificateur symbolique / raffineurs géométriques, un gestionnaire d'interactions, et un superviseur de couche délibérative.

Dans le chapitre V, nous détaillons le gestionnaire d'interactions : ses formalismes, et ses traitements. Le gestionnaire d'interactions est destiné à modéliser et résoudre des sous-problèmes (tâches jointes) impliquant plusieurs UAV.

Le chapitre VI présente les résultats, avec les expérimentations concrètes réalisées dans le cadre du projet COMETS, et les tests réalisés en simulation.

Enfin, dans le chapitre VII, nous revenons sur les contributions, les résultats obtenus, les limites et les évolutions envisageables.

Chapitre I

Problématique et état de l'art

Ce chapitre parcourt l'état de l'art des deux grandes composantes de notre travail : d'un côté, les systèmes multi-robots, les problématiques sous-jacentes et les approches classiques. De l'autre côté, la robotique aérienne (ou les UAV, pour Unmanned Aerial Vehicles), son potentiel et ses contraintes. A la rencontre des deux, nous proposons en dernier lieu un tour d'horizon des travaux les plus significatifs liés aux systèmes multi-UAV.

I.1 Systèmes multi-robots : les problématiques

Elaborer un système multi-robots conduit nécessairement à s'interroger sur l'architecture individuelle des robots : dans quelle mesure des architectures de décision et de contrôle données peuvent-elles répondre aux exigences et contraintes posées par un tel système ? Suffit-il d'étendre une architecture mono-robot dans un cadre multi-robots, ou alors faut-il considérer la nature des problématiques multi-robots dans les fondements même de l'architecture ?

Dans un premier temps, nous présentons un bref tour d'horizon des idées qui s'articulent autour des notions d'architecture de décision et de contrôle, dans un robot.

Ensuite, afin de mieux en saisir les spécificités et enjeux, nous proposons trois approches pour appréhender les problématiques rencontrées dans les systèmes multi-robots.

I.1.1 Architecture et décision mono-robot : généralités

Architecture robotique

Les roboticiens s'emploient à doter de capacités autonomes de plus en plus évoluées les robots qu'ils conçoivent. Des robots très autonomes sont en mesure de prendre des décisions cohérentes dans un contexte d'exécution donné, sans nécessairement en référer aux compétences d'un opérateur.

Doter un robot de capacités fonctionnelles et/ou décisionnelles exige de considérer une architecture logicielle : celle-ci doit permettre la récupération et le traitement de données brutes en provenance de capteurs, et d'exploiter ces données pour synthétiser les directives de contrôle adéquates. De nombreuses architectures de contrôle ont vu le jour à la fin des années 1980. Citons à titre d'exemple la "sub-sumption architecture" de Brooks [Brooks 86], architecture réactive dans laquelle des comportements élémentaires sont combinés de façon à favoriser l'émergence de comportements plus complexes.

A l'autre extrême, les architectures délibératives (par exemple [Albus 89]) exploitent des modèles d'environnement et d'actions, pour générer de façon peu flexible (résultats bruts, non adaptables) les commandes sensori-motrices. Les données perçues sont utilisées pour mettre à jour les modèles, et les directives de contrôle sont successivement produites sur ces bases avant d'être effectivement exécutées ("Sense Model Plan Act", ou SMPA). Cette approche se trouve être mal adaptée à la robotique autonome, dans un monde réel et dynamique, car elle manque cruellement de réactivité et de flexibilité (pouvoir adapter, dans une certaine mesure, les actions aux aléas du monde).

Au croisement des architectures réactives et délibératives, les architectures hybrides représentent un compromis appréciable : la "3-layer architecture" de Gat (ATLANTIS) [Gat 91] est une des premières architectures hybrides, dans laquelle plusieurs niveaux de contrôle sont distinctement hiérarchisés.

3T (three-tiered) [Bonasso 97] est souvent considérée comme l'architecture hybride type : elle comprend un niveau délibératif, un niveau exécutif et un niveau fonctionnel. CLARAty [Volpe 01] revendique une distinction par rapport au modèle classique : le fait de rassembler au niveau délibératif les composantes délibératives et exécutives, en différenciant la granularité des données manipulées. Dans ce cadre, un modèle commun unifié est exploité. Cette architecture est donc "2-couches", l'autre couche concernant les aspects fonctionnels.

L'architecture LAAS [Chatila 92, Alami 93, Alami 98a] (voir la figure I.1) est un autre exemple d'architecture hybride : au plus près des aspects matériels, une couche fonctionnelle comprend les processus de contrôle des capteurs et des effecteurs, qui fonctionnent en temps réel, de façon très réactive. Au niveau intermédiaire, la couche de contrôle d'exécution est dédiée au contrôle des requêtes transmises à la couche fonctionnelle. Au plus haut niveau, la couche décisionnelle rassemble les composantes permettant au robot de planifier ses actions et de superviser l'exécution des tâches. Les fonctions décisionnelles ont des contraintes temps-réel moins prononcées que la fonctions de la couche fonctionnelle.

Ces différentes architectures hybrides sont habituellement considérées comme sensiblement équivalentes.

On pourra finalement noter une approche atypique : l'architecture IDEA, [Muscettola 02] dans laquelle chaque composant du système est doté de capacités délibératives et réactives, interagissant en tant qu'agents. Ceux-ci disposent de modèles de contrainte et de compatibilité, qui décrivent l'évolution possible de leurs états.

La prise de décision

Pour un robot, la prise de décision est l'activité consistant à anticiper le cours des actions à réaliser dans un futur plus ou moins proche. Dans le schéma habituel, un ensemble de buts

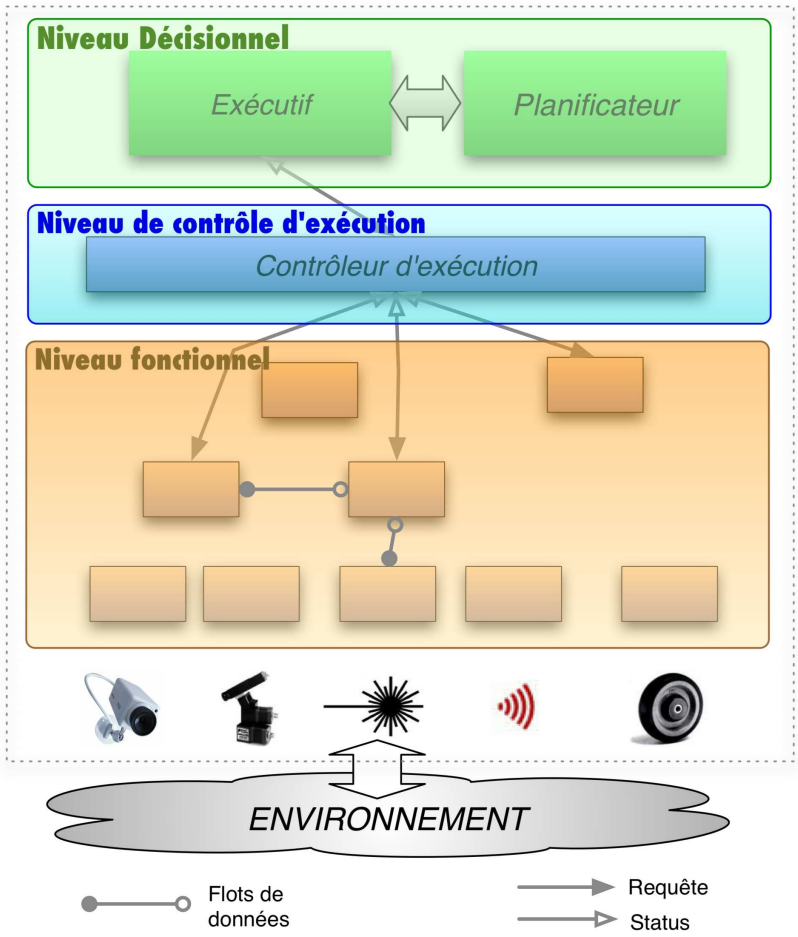


FIG. I.1 – L'architecture LAAS

doivent être atteints (explicitement demandés par un opérateur, ou automatiquement générés en réponse à un contexte donné).

La mise en œuvre de capacités autonomes de prise de décision en robotique est directement liée aux choix d'architecture : ainsi, dans les architectures de type réactives, la décision est vue en tant que résultante de la combinaison des composantes réactives élémentaires : aucune entité du système n'est dévolue à la prise de décision en tant que telle. C'est à la fois la force (simplicité architecturale, robustesse) et la faiblesse (limitations pour la réalisation de tâches complexes) de ce paradigme.

Dans les architectures délibératives et hybrides, au contraire, un (ou un ensemble de) composant(s) bien défini est en charge de raisonner sur des modèles afin de générer des plans, c'est à dire décomposer des buts en sous-ensembles de tâches élémentaires. Dans un schéma classique de planification, le planificateur construit un plan de tâches qui, appliquées depuis un état initial (généralemement l'état courant du robot), conduisent à satisfaire les buts.

De nombreux systèmes de planification de tâche existent, proposant différents paradigmes de représentation de plans et d'inférence. Pour un tour d'horizon complet des méthodes de planifi-

cation de tâche, nous renvoyons le lecteur à l'ouvrage de Ghallab, Nau et Traverso : [Ghallab 04].

Deux classes principales de planificateurs sont généralement identifiés : ceux (largement majoritaires) qui planifient dans l'espace d'états, et ceux qui planifient dans l'espace des plans partiels. Dans la première catégorie, on retrouve TLplan [Bacchus 01], TALplan [Kvarnström 01], qui exploitent des règles de contrôle dans la recherche de plan, en mettant en œuvre des logiques temporelles particulières. On y trouve aussi les planificateurs à base de réseau de tâches hiérarchiques (HTN), comme Shop2 [Nau 03] : les tâches sont hiérarchisées explicitement, selon un schéma de décomposition explicitement fourni dans le domaine de planification. Comme nous le verrons par la suite, c'est cette dernière approche que nous avons adoptée dans notre mise en œuvre de couche délibérative, pour les UAV.

Dans la seconde catégorie, Ixtet [Laborie 95] fait figure de référence : le principe de planification consiste à faire évoluer un plan partiellement construit en sélectionnant successivement des défauts de ce plan pour les résoudre. Ixtet est un planificateur raisonnant explicitement sur le temps, le monde étant décrit comme un ensemble de variables qui sont fonction du temps. Des "timelines" décrivent l'historique des évolutions de ces variables. Le planificateur Europa [Frank 00, Frank 01], développé à la NASA, est un autre exemple de planificateur de ce type.

Parmi ces différentes approches, très peu ont été réellement mises en œuvre sur un robot. La difficulté concerne d'une part la mise en correspondance des données symboliques manipulées dans le planificateur avec la réalité du monde où évolue le robot, et d'autre part, la prise en considération de contingences qui nécessite de réviser régulièrement les plans produits.

Au LAAS-CNRS, la délibération est assurée, dans l'architecture LAAS, par un exécutif temporel (Ixtex-Exec : [Lemai 04]) qui permet d'entremêler planification et exécution (on parle aussi de "planification continue"). Cette approche est expérimentée avec succès depuis plus d'un an sur un robot mobile type "rover" (on peut cependant noter que des travaux beaucoup plus anciens [Chatila 92] avaient déjà par le passé intégré des processus de délibération dans l'architecture de robots mobiles).

Parmi d'autres travaux ayant mis en œuvre un planificateur de tâches dans un robot mobile, nous citerons par exemple P-SA [Kortenkamp 98], et CASPER [Chien 00].

I.1.2 Trois points de vue pour appréhender les systèmes multi-robots

Les considérations précédentes, sur l'architecture et la prise de décision individuelles des robots, prennent une autre autre dimension dans un cadre multi-robots (MR). Si d'un point de vue individuel les questions de l'architecture et de la prise de décision sont toujours de mise, considérer un système multi-robots exige de mettre en œuvre des moyens d'interaction, qui peuvent s'avérer être plus ou moins couplés selon les systèmes MR.

Nous proposons dans cette partie de considérer les systèmes multi-robots selon différents angles, afin d'appréhender les tenants et aboutissants de la mise en œuvre de tels systèmes. La première approche considère la nature des interdépendances entre les robots d'un système, et s'apparente à un "pourquoi ?". La deuxième approche est une caractérisation des systèmes multi-robots selon des critères physiques, ce que nous pouvons rapprocher du "quoi ?". Le troisième axe répertorie les principaux moyens et paradigmes mis en œuvre, dans la littérature, pour coordonner l'activité des robots dans un système : il s'agit du "comment ?".

Organisations multi-robots : typologie d'interdépendances

Selon les champs d'application, les robots d'un système peuvent manifester différents niveau d'ouverture et de coopération vis-à-vis des autres robots : à un extrême un robot totalement égocentré ignorera les autres robots ou ne montrera aucun comportement coopératif à leur égard. Ce peut être le cas d'un robot qui n'est pas doté de capacités lui permettant de coopérer, ou bien ce peut être le cas dans certains cadres compétitifs. A l'autre extrême, un robot altruiste peut être amené à se sacrifier si l'effet est bénéfique pour le système. Ce peut être le cas d'un problème où un but commun justifie une aliénation absolue des robots.

Ferber introduit dans [Ferber 95] une typologie des interdépendances entre agents dans un système multi-agent, selon trois dimensions : compatibilité de buts, disponibilité des ressources, disponibilité des capacités. Si cette typologie est orientée "agents", elle s'applique également de façon satisfaisante aux systèmes multi-robots.

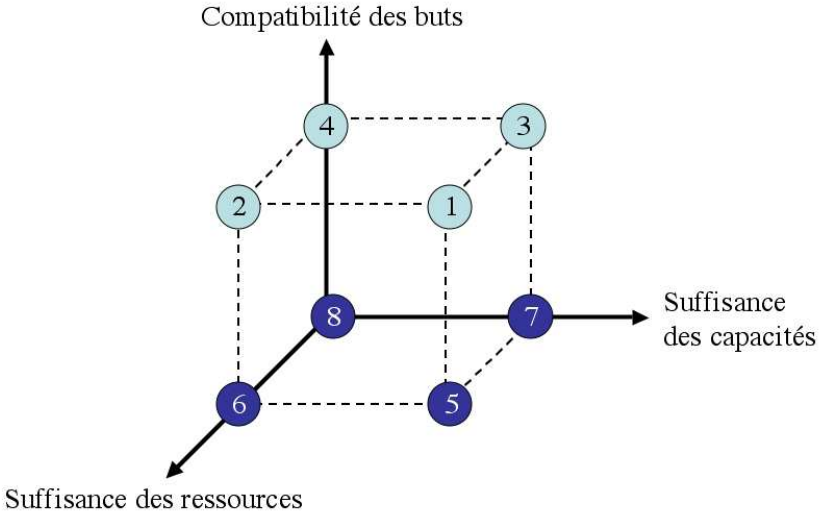


FIG. I.2 – 8 configurations d'interaction selon Ferber

La figure I.2 représente les 8 types de systèmes selon ces critères :

1. **Indépendance**. Les agents ont des buts compatibles, des ressources et des capacités suffisantes. Il n'y a pas interaction. Il s'agit en fait, pour chaque agent, d'un cadre mono-agent.
2. **Collaboration simple**. Les agents ont des buts compatibles, des ressources suffisantes, mais des capacités insuffisantes : les interactions consistent principalement en de l'allocation de tâche et du partage d'information.
3. **Encombrement**. Les agents ont des buts compatibles et des capacités suffisantes, mais les ressources sont insuffisantes : les agents doivent coordonner leur usage des ressources.
4. **Collaboration coordonnée**. Les agents ont des buts compatibles, mais ni les ressources suffisantes, ni les capacités suffisantes. Ce sont les systèmes les plus complexes : ils nécessitent à la fois une coordination des tâches des agents et de l'emploi des ressources.
5. **Compétition individuelle pure**. Les agents ont des buts incompatibles, mais des ressources et des capacités suffisantes. Il n'y a pas d'interaction entre les agents (cette configuration n'est pas courante dans les applications multi-agent).

6. **Compétition collective pure**. Les agents ont des buts incompatibles, des ressources suffisantes et des capacités insuffisantes. Dans cette situation, chaque agent a besoin du soutien des autres pour parvenir à son propre but. Mais pour chaque agent, soutenir les autres risque également de rendre ses propre buts irréalisables. Dans cette configuration peuvent se former des coalitions concurrentes, donnant lieu à de complexes protocoles de négociations argumentées.
7. **Conflits individuels pour des ressources**. Les agents ont des buts incompatibles, des ressources insuffisantes et des capacités suffisantes. Il s'agit typiquement de situations de luttes pour des ressources.
8. **Conflits collectifs pour des ressources**. Les agents ont des buts incompatibles, des ressources insuffisantes et des capacités insuffisantes. La réalisation des buts est assujettie à la disposition des ressources : soit les agents luttent individuellement pour l'obtention des ressources, soit ils se forment en coalitions concurrentes.

Les systèmes multi-robots pour lesquels les buts des robots ne sont pas compatibles ne sont pas très courants, du moins dans les applications actuelles. Citons toutefois la "robocup" [Robocup 05], qui se rapproche de la catégorie 8 (bien que les individus d'une même équipe aient des buts compatibles). Une grande majorité des applications multi-robots correspondent donc aux situations où les buts des robots du système sont compatibles : c'est à dire les situations 1 à 4.

En réalité, la situation 1 ne correspond pas à un système au sens que nous lui avons donné (aucun lien entre les robots). En effet, les robots s'ignorent totalement, et aucune sorte d'interaction existe.

Seules les situations 2, 3 et 4 sont effectivement représentées dans les applications multi-robots. Certains systèmes dits "à base d'économie de marché" (voir dans le paragraphe I.1.2) sont développés dans le cadre de la situation 2. Cependant, pour beaucoup d'applications multi-robots, l'interaction physique est primordiale. Elle est soit imposée par la proximité des robots (du fait des contraintes imposées par l'environnement), soit nécessaire pour réaliser certaines tâches comme réaliser une perception coordonnée d'un objectif, ou transporter conjointement un objet. Ainsi, la situation 3 est celle du projet Martha [Alami 98b], dans lequel les robots se coordonnent afin de prévenir les collisions (conflit de ressource "espace"). Les systèmes multi-robots les plus riches correspondent à la situation 4 : ils nécessitent la mise en œuvre de mécanismes de coordination pour profiter des capacités hétérogènes des robots, et pour organiser l'utilisation des ressources conflictuelles.

Dans l'approche que nous adoptons dans le présent travail, nous nous situons dans la situation 4 : nous mettons en œuvre une architecture permettant explicitement de considérer des activités jointes, compte tenu des capacités limitées (robots physiquement hétérogènes) et des ressources limitées (dans le cas des UAV, principalement l'espace).

Il faut noter que cette typologie est fortement connotée *agent*, et ne se trouve pas forcément être idéalement adaptée pour décrire des systèmes de robots bien réels (en particulier pour capturer le rapport des robots à la ressource espace).

Taxonomie physico-fonctionnelle

Dudek et al. [Dudek 02] introduisent une grille de lecture applicative des systèmes multi-robots, qui met l'accent sur les propriétés physiques et fonctionnelles du *collectif comme un tout* :

1. **Par taille de collectif** :
   - SIZE-ALONE : 1 robot. Le collectif minimal.
   - SIZE-PAIR : 2 robots. Le groupe le plus simple.
   - SIZE-LIM : n robots. "n" est petit devant la taille de la tâche ou de l'environnement.
   - SIZE-INF : $n \gg 1$ robots. Equivalent en pratique à une multitude de robots. Collectif de type "swarm" (essaim).

   Ce critère influence profondément les choix de conception d'un système multi-robots : il est en effet fréquent que des architectures bien adaptées à un petit nombre de robots (SIZE-PAIR ou SIZE-LIM) ne puissent être étendues à un grand nombre de robots. Inversement, les approches "swarm" favorisant l'emergence de capacités de groupes plus ou moins complexes sur la base de comportements individuels simples, se retrouvent vraisemblablement inefficaces dans un collectif de robots réduit, où la "masse critique" n'est pas atteinte. Les approches à base de rôles, dans lesquelles une hiérarchie entre des équipes structure le système, se démarquent quelque peu : elles apparaissent très polyvalentes vis à vis de ce critère de taille (dans le cas de SIZE-INF, il peut s'agir d'une hiérarchie d'équipes).
2. **Par portée de communication** :
   - COM-NONE. Pas de communication directe. Dudek autorise toutefois dans cette catégorie la communication *indirecte* (on dira aussi *implicite*), c'est à dire par observation.
   - COM-NEAR. Communication de courte portée : uniquement dans un voisinage (situation très réaliste dans le cadre de communications radio, par exemple, dont la portée est limitée).
   - COM-INF. Chaque robot peut communiquer avec chaque autre robot.

   Les hypothèses de portée de communication sont très variables selon les systèmes considérés. Dans des systèmes réactifs, où les capacités de délibération est très limitée, la première option peut suffire pour permettre une coordination des individus. Dans des systèmes plus complexes, où la délibération joue un rôle important, la communication directe est essentielle. La seconde option est dans ce cas la plus réaliste, dans un système situé dans le monde réel.
3. **Par topologie de communication** :
   - TOP-BROAD. Broadcast : chaque robot peut communiquer avec tous les autres robots à portée de communication, mais ne peut envoyer un message à un robot ciblé.
   - TOP-ADD. Adressage : chaque robot peut communiquer avec n'importe quel autre robot à portée de communication, connaissant son nom ou son adresse.
   - TOP-TREE. Les robots sont liés dans un "arbre" et ne peuvent communiquer qu'à travers cette hiérarchie.
   - TOP-GRAPH. Les robots sont liés dans un graphe plus général.

   Au contraire de TOP-BROAD et TOP-ADD, les topologies TOP-TREE et TOP-GRAPH correspondent à des restrictions qui ne sont pas d'ordre physique : il s'agit de restrictions organisationnelles.
4. **Par bande-passante de communication** :

- BAND-INF. Les communications sont libres : la bande-passante est suffisamment élevée pour que le coût associé à son usage soit négligeable.
- BAND-MOTION. Le coût de communication est du même ordre de grandeur que le coût de déplacement entre deux endroits.
- BAND-LOW. Coût très élevé : communiquer coûte beaucoup plus que de se déplacer entre deux endroits.
- BAND-ZERO. Pas de communication : les robots n'ont aucune faculté de se percevoir mutuellement.

Dans de nombreuses applications, l'hypothèse BAND-INF est de rigueur : cependant, des applications du domaine militaire ou spatial peuvent se décliner en BAND-MOTION, BAND-LOW, voire BAND-ZERO. Par ailleurs, BAND-MOTION correspond aussi à des applications où les communications sont indirectes (au sens de Dudek), à travers des déplacements ou mouvements.

5. **En fonction de la reconfigurabilité collective** :
   - ARR-STATIC. Configuration statique : la topologie est fixe.
   - ARR-COMM. Réarrangement coordonné : des modifications de topologie sont possibles parmi les individus qui sont à même de communiquer.
   - ARR-DYN. Réarrangement dynamique : les relations entre membres du collectif peuvent être modifiées arbitrairement.

   La reconfigurabilité est un gage de souplesse : si différentes fonctions sont identifiées dans le cadre d'une tâche collective (tâche jointe), il est souhaitable que l'allocation des fonctions puisse être révisée, dans un souci de robustesse (dysfonctionnement d'un des robots du système) et d'efficacité (mise à jour opportuniste de la configuration).
6. **En fonction de la nature des traitements des individus du collectif** :
   - PROC-SUM. Addition non-linéaire : unité de traitement pouvant être utilisé pour construire un réseau neuronal, selon Dudek.
   - PROC-FSA. Automate à état fini : modèle computationnel de référence pour les architectures de type "subsumption" [Brooks 86], correspondant aux approches dites "comportementales".
   - PROC-PDA. Automate à pile ("push-down").
   - PROC-TME. Machine de Turing : le modèle le plus répandu dans les systèmes robotiques.

   Sauf exception, seul le modèle PROC-TME est réellement employé dans les applications robotiques : d'autre modalités computationnelles peuvent éventuellement être ensuite émulées au dessus de ce modèle.
7. **En fonction de la composition du collectif** :
   - CMP-IDENT. Identique : le collectif est formé d'individus identiques aussi bien d'un point de vue matériel que logiciel.
   - CMP-HOM. Homogène : le collectif est formé d'individus essentiellement dotés des même caractéristiques physiques.
   - CMP-HET. Hétérogène : le collectif est formé d'individus qui ne sont pas physiquement uniformes.

   Cette caractéristique peut profondément influencer les mécanismes de coordination du système : dans le cadre de robots identiques ou homogènes, l'hypothèse selon laquelle

les robots du système sont bâtis sur le même modèle peut être exploitée par les robots : il n'y aura pas à considérer de possibles différences de capacités par rapport aux tâches à accomplir. Dans un collectif hétérogène, les robots n'ont généralement pas de modèles précis des robots différents d'eux-mêmes, et des mécanismes de négociations plus riches et plus complexes doivent être considérés dans la coordination des activités.

Cette taxonomie est bien connue (et reconnue) dans la communauté MR. Elle permet de regrouper les applications selon un certain nombre de critères objectifs, et donc de comparer des systèmes qui sont comparables. Vis à vis de cette taxonomie, nous situons notre système de la façon suivante :

SIZE-LIM : collectif de taille > 2, mais limitée (de l'ordre de la dizaine d'individus),

COM-NEAR : communication possible dans une certaine proximité,

TOP-ADD : communication de pair à pair,

BAND-INF : le coût de communication est négligeable devant les autres coûts,

ARR-DYN : le système est en mesure de se reconfigurer dynamiquement,

PROC-TME : machines de Turing,

CMP-HET : la composition du système est hétérogène.

Mécanismes de coordination

Dans un système multi-robots, les problématiques de la robotique mono-robot sont considérablement augmentées par les interactions entre robots. Ces interactions peuvent représenter une menace pour l'intégrité du système, ou tout au moins risquent-elles de dégrader ses performances. Cependant, les interactions peuvent également être mises à profit pour réaliser plus efficacement des tâches mono-robot (redondance, partage de tâches sécables), ou bien réaliser des tâches qui, par nature (opérations distantes simultanées...), ne pourraient être réalisées par un unique robot.

Mettre à profit les interactions dans un système multi-robots consiste à y introduire des mécanismes de coordination, afin de rendre cohérentes entre elles les actions des différents robots.

En étudiant les communautés humaines, Mintzberg [Mintzberg 79] a identifié trois processus fondamentaux de coordination :

1. **coordination par ajustements mutuels**. Forme de coordination où des individus s'accordent pour partager des ressources en vue d'atteindre un but commun : aucun individu n'a de contrôle sur les autres individus, et le processus de décision est conjoint.
2. **coordination par leadership / supervision directe**. Une relation hiérarchique existe entre des individus : certains individus exercent alors un contrôle sur d'autres.

3. **standardisation**. Des procédures sont pré-définies en vues de situations d'interaction particulières : par exemple des règles dont l'application peut limiter les conflits.

Les différents mécanismes de coordination exploités dans les systèmes multi-robots sont tous des manifestations d'un ou de plusieurs de ces processus fondamentaux. Nous proposons ci-après une liste des principales familles de mécanismes, illustrées par des exemples d'application dans la littérature.

– **Approches "market-based".**

Les approches "à base de marché" se fondent sur des système de contractants-contractés, généralement inspirés du "contract-net protocol" (CNP), introduit par Smith [Smith 80] en 1980. Dans le cadre le plus élémentaire du CNP, une tâche est mise aux enchères par un agent contractant, et les agents du système enchérissent. L'agent contracteur proposant la meilleure offre (du point de vue du contractant) remporte la tâche, et se retrouve engagé vis à vis du contractant dans la réalisation de cette tâche. De nombreuses variantes et extensions ont vu le jour, permettant par exemple de définir des pénalités de rupture de contrat, de mettre aux enchères des groupes ou lots de tâches, ou encore de former des coalitions pour enchérir sur certaines tâches en tant que groupe.
Les approches "à base de marché" sont de plus en plus exploitées en robotique, principalement pour répondre à la problématique de l'allocation de tâches dans un système multi-robots distribué. Elles confèrent beaucoup de souplesse à l'allocation de tâche, bien que sous-optimales. Elles supposent cependant que la décomposition d'une tâche "de haut niveau" en tâches "distribuables" est déjà réalisée, et que la décomposition de tâches en sous-tâches est indépendante de l'affectation. Elles supposent également que les tâches à distribuer peuvent systématiquement être évaluées par les contracteurs potentiels.
Ce type d'approche s'inscrit typiquement dans la catégorie 1 des mécanismes de coordination (coordination par ajustements mutuels). Parmi les approches "à base de marché", on trouve par exemple [Botelho 99] [Gerkey 02] [Dias 04] [Kalra 03, Kalra 05] et [Lemaire 04].

– **Approches à base d'"émergence".**

Les approches émergentes reposent sur l'émergence de comportements cohérents de groupe, à partir de primitives comportementales individuelles simples. Les approches à base d'émergences s'opposent aux approches délibératives, dans lesquelles les robots sont pourvus de modèles (du monde, d'eux-mêmes, et des autres individus) et raisonnent sur ces modèles. Les approches à base d'émergence ont pour principal avantage de permettre des exécutions flexibles et robustes : elles sont généralement employées dans un cadre réactif, pour des missions "unitaires". Elles atteignent cependant leurs limites dans les applications complexes, où l'interaction nécessite une forme de concertation.
Ce type d'approche se classe dans la catégorie 3 (coordination par standardisation) : les primitives comportementales sont en effet des formes de spécification de la conduite à tenir selon les situations rencontrées. Ces primitives peuvent être pré-définies dans le cadre d'un contexte particulier, ou bien apprises (les modalités d'apprentissage sont alors elles-mêmes des manifestations de la "standardisation").
On trouve dans cette catégorie les travaux de [Arkin 92], [Balch 95], [Parker 93], [Mataric 95], ou encore [B. Damas 04].

– **Planification centralisée.**
Les approches orientées ”planification centralisée” visent à synthétiser, pour les différents robots du système, un ensemble de plans dont l'union est un plan global pour une mission donnée. Les individus du système sont modélisés centralement, et la centralisation des connaissances permet de construire le plan global ”le mieux informé” (en principe), comparé à la plupart des alternatives distribuées. Ces approches permettent théoriquement d'obtenir des plans globaux optimaux, compte tenu des connaissances. Elles permettent aussi de faire face à des problèmes complexes, bien qu'elles se heurtent à des problèmes ”d'échelle” (scalability) : la centralisation des informations est en effet susceptible d'être à l'origine d'un goulot d'étranglement du transit des données. En pratique, seules, les approches centralisées s'avèrent peu adaptées à la dynamique des applications multi-robots. Par contre, alliées à d'autres paradigmes décentralisés, elles constituent des options envisageables.
L'approche ”planification centralisée” fait partie de la catégorie 2 (par supervision). Nous citerons dans cette catégorie [Caloud 90], [Chaimowicz 01] ou encore [Burgard 00].

– **Fusion de plans (ou planification distribuée incrémentale)**
La fusion de plans est un protocole de coordination de plans *a posteriori*, une fois que les différents agents du système ont individuellement acquis leur plan. Il s'agit de valider en un plan global les plans individuels des robots : pour cela, peu avant l'exécution effective d'une partie du plan, un robot réclame l'autorisation de manipuler le plan global. Une fois l'autorisation obtenue, il insère dans le plan global ses propres tâches, ainsi que des contraintes d'ordre destinées à prévenir les conflits possibles entre ses tâches et le plan global déjà existant. Les autres robots devront ensuite tenir compte de ces contraintes, avant d'insérer à leur tour leur propres tâches et contraintes dans le plan global, incrémentalement. Cette approche est particulièrement utile pour prévenir les conflits de ressources entre des robots.
L'approche par fusion de plans appartient à la catégorie 1 (ajustements mutuels), et nous y trouvons les travaux de [Ephrati 93], [Alami 95], [de Weerdt 03]...

– **Approches équipes/rôles.**
Dans cette approche, des sous-groupes d'agents (équipes) sont formés pour répondre aux besoins de tâches jointes. Pour une tâche jointe donnée, un certain nombre de rôles sont définis dans le cadre d'une équipe : les membres de l'équipe doivent alors endosser ces rôles. Les rôles permettent d'expliciter les asymétries parmi les tâches ou fonctions devant être assumées pour réaliser la tâche jointe.
Cette approche entre dans la catégorie 3 (coordination par standardisation), en considérant que les rôles sont prédéfinis. Cependant, on peut aussi considérer qu'elle fait appel à des mécanismes d'ajustement mutuels, pour le choix des rôles, et donc qu'elle s'inscrit dans aussi dans la catégorie 1. Parmi les travaux représentatifs de cette approche, citons [Tambe 97] [Nair 03] [Veloso 98] [Stone 99], ou encore [Lundh 04] (bien que la notion de rôle soit ici absente : les robots d'une équipe s'organisent en se proposant des services, en termes de fonctions).

– **Planification totalement distribuée, asynchrone.**
Il s'agit dans cette approche de réaliser la coordination de plans de concert avec la planification, c'est à dire en même temps que la décomposition des tâches. L'approche nécessite un flux

de communications élevé, afin de transmettre des requêtes informatives aux autres individus et de traiter les requêtes reçues. Le protocole peut inclure des notions d'intention et d'engagement, grâce auxquelles les robots peuvent prendre en considération dans leur propre plan les effets des activités des autres robots. Cette récente approche est ambitieuse, et son applicabilité sur robots réels n'est pas encore démontrée. Elle se situe dans le cadre de la catégorie 1 (coordination par ajustements mutuels). Cette approche est en particulier présentée dans [Brenner 03].

Ce tour d'horizon des mécanismes de coordination laisse entrevoir la diversité des paradigmes étudiés et exploités dans le champ du multi-robots. Cependant, il faut noter que les mécanismes énumérés sont rarement utilisés à l'exclusion de tous les autres.

Dans notre approche, nous nous situons dans une optique de coordination mixte, d'une part à base de modèles d'interaction (avec une mise en œuvre de rôles prédéfinis, catégorie 3), et d'autre part à base de négociations explicites (catégorie 1, par ajustements mutuels entre robots). Cette approche répond selon nous à un besoin, pour des systèmes que nous dirons "critiques", comme c'est le cas d'un système d'UAV, de spécifier explicitement le cadre souhaité des interactions. Comme nous le verrons dans la suite de ce manuscrit, nous mettons en œuvre un paradigme multi-robots dans lequel le champ d'interaction des individus se veut clairement codifié et explicité.

I.1.3 Conclusion

L'objectif de cette section était de proposer une vision d'ensemble des caractéristiques et techniques associées aux systèmes multi-robots. Le domaine est très riche, et de multiples paradigmes et conjectures sont proposés dans la littérature, parfois bien formalisés et éprouvés, et parfois aussi à l'état de théorie ou au contraire amenés sur des bases empiriques.

Notre approche s'inspire de techniques de coordination existantes, en considérant la nature des opérations jointes, des interactions envisageables dans un système de robots aériens. La section suivante présente les caractéristiques qui, selon nous, font des systèmes multi-UAV un contexte d'étude de système MR bien particulier.

I.2 Robotique aérienne : défis et spécificités

I.2.1 Le potentiel de la robotique aérienne

La robotique aérienne est une branche de la robotique mobile. Elle concerne l'étude, la conception et l'expérimentation de véhicules aériens non-habités (UAV pour "Unmanned Aerial Vehicles"). La robotique aérienne a connu un essor important dans les années 1990 : les motivations militaires ont conduit au développement de drones automatiques (autonomie opérationnelle), principalement utilisés pour des missions de reconnaissance (et plus récemment pour des missions offensives) au cours des conflits d'Irak, d'Afghanistan et de l'ex-Yougoslavie en particulier. Parallèlement aux applications militaires, des aéronefs téléopérés ont aussi été développés pour

des applications agricoles (hélicoptères R-MAX de Yamaha, utilisé dans ce but au Japon et aux Etats-Unis), et de surveillance civile, au cours de manifestations par exemple. D'autres applications telles que la production cinématographique, la cartographie, la surveillance ou reconnaissance de zones qui ont été l'objet d'un sinistre (catastrophes naturelles, catastrophes industrielles ou nucléaires...), sont autant d'applications potentielles qui soulèvent un intérêt majeur à la fois de la part des acteurs économiques et politiques, et sur la scène scientifique.

Augmenter l'autonomie d'un UAV à un niveau opérationnel permet de soulager l'opérateur de l'attention que nécessite le pilotage téléopéré d'un aéronef, et rend plus accessible la manipulation de l'UAV, qui requiert parfois une expertise de haut vol (pour les hélicoptères en particulier). Augmenter l'autonomie décisionnelle d'un UAV permet de lui déléguer certaines formes de prise de décision dans des situations qui s'y prêtent : soit parceque les tâches à réaliser sont pénibles pour un opérateur (répétitives...), soit parce que l'UAV dispose d'informations pertinentes dont ne dispose pas (ou tout au moins pas au moment pertinent) l'opérateur.

Parmi les travaux visant à accroître l'autonomie d'un UAV, nous pouvons citer [Doherty 00] pour le projet WITAS, dans lequel un hélicoptère est doté d'autonomie opérationnelle avancée (expérimentations dans le cadre de différents scenarii de reconnaissance et de surveillance, en particulier). Le projet MARVIN [Musial 01] est un autre exemple de mise en œuvre de capacités opérationnelles autonomes, éprouvé lors de compétitions de type "rescue" [AUVSI 05] et lors de mises en situation dans des scénarii du projet COMETS [Ollero 04a, Ollero 04b] (surveillance de feu de forêt). Il est à noter que des plates-formes aériennes ont rarement été dotées de capacités décisionnelles autonomes, jusqu'à présent.

Les contributions présentées dans ce manuscrit s'incrivent dans cette ligne : développer l'autonomie décisionnelle dans les systèmes multi-robots aériens (ou système **multi-UAV**).

I.2.2 Systèmes multi-UAV : tour d'horizon

Un système multi-UAV, à l'instar des systèmes multi-robots en général, étend les capacités d'un système mono-UAV en permettant de réaliser plus rapidement des tâches séquables ou en permettant de réaliser des tâches auxquelles un UAV seul ne peut prétendre. La coordination de robots aériens peut être bâtie sur des mécanismes tout à fait classiques de coordination ; on remarquera cependant quelques caractéristiques propres aux systèmes multi-UAV :

– **L'espace**. La ressource spatiale doit être considérée en trois dimensions, ce qui signifie concrètement qu'il est envisageable que des robots puissent se situer à la verticale les uns des autres. La coordination en terme d'occupation d'espace doit idéalement prendre en considération cette propriété.

– **Les tâches**. De façon réaliste, il paraît peu envisageable que des UAV entrent directement ou indirectement (manipulation simultanée d'un objet) en contact : plus généralement, l'hypothèse d'une distance de sécurité minimale entre les UAV du système est pratiquement toujours de mise. En second lieu, la manipulation en robotique aérienne n'est pas encore d'actualité dans les laboratoires de recherche : il en découle que les tâches de robotique aérienne sont généralement liées à des objectifs de déplacement et de perception. Nous nous intéressons donc principalement à la coordination des activités multi-UAV composées de *déplacements* et *perceptions*. (Remarque : on peut toutefois raisonnablement imaginer des problèmes de manipulation simples impliquant des activités d'épandage, de treuillage ou de largage, et donc

allant un petit peu plus loin que ces tâches de déplacement et de perception).

– **L'implication humaine**. L'implication humaine dans les systèmes multi-UAV ne se manifeste pas à travers l'interaction applicative homme-robot, comme ce peut être le cas en robotique terrestre (robots personnels...). Par essence, la robotique aérienne se place presque systématiquement dans le contexte de la robotique de terrain. L'humain est présent pour opérer le système, fournir des directives (missions...), récupérer et reprendre les échecs que le système n'est pas en mesure de traiter. Les mécanismes coopératifs mis en œuvre doivent, encore plus encore que pour un autre types de système multi-robots, prendre en considération l'opérabilité du système.

Des systèmes multi-UAV ont été conçus et expérimentés depuis la fin des années 1990, avec des travaux réalisés aux USA, tout d'abord dans le domaine militaire : il s'agissait principalement de coordonner des drones pour réaliser du vol en formation. C'est le cas par exemple à l'Air Force Research Laboratory [Schumacher 00].

Le projet AVATAR [Sukkhatme 02], à l'USC, étudie des systèmes de robots mixtes aériens / terrestres, dans le cadre d'applications de surveillance et reconnaissance. L'architecture de contrôle est de type comportementale, avec un faible couplage pour la coordination entre les robots (coordination passive, à travers une fusion externe des données de preception).

Le projet BEAR [Vidal 02], à Berkeley, considère un système d'hélicoptères physiquement hétérogènes (tailles différentes), dans le cadre de vol en formation et en particulier sur la thématique de l'évitement de collisions entre UAV ([Shim 03]).

Des travaux visant au contrôle d'un groupe d'UAV ont également été menés avec succès au MIT ([How 04]), avec un groupe de drones comprenant jusqu'à huit aéronefs. Des techniques stochastiques d'allocation de tâche sont employées pour répartir des cibles (type militaire) entre des UAV ([Alighanbari 05]).

Hors des Etats-Unis, des travaux très significatifs ont été réalisés à l'université de Sydney [Sukkarieh 02] : il est question de l'exploitation d'une flotille de drones pour des applications de perception coopérative (fusion distribuée de données). Si l'objectif de ces travaux ne portait pas sur les aspects d'architecture et de coordination multi-UAV, des travaux plus recents de ce laboratoire portent d'avantage sur ces problématiques ([Furukawa 04], par exemple).

En Europe, très peu de travaux ont déjà démontré la mise en œuvre de systèmes multi-UAV. COMETS [Ollero 04a] est un des premiers projets Européen à cibler cette thématique, dans un contexte applicatif de surveillance de feux de forêt. L'université de Linköping s'est récemment engagée dans la mise en œuvre de systèmes multi-UAV (principalement des hélicoptères), dans le cadre d'un laboratoire commun avec le groupe industriel Saab.

Si des projets et des travaux existent bel et bien dans le domaine des système multi-UAV, les efforts ne concernent pas encore le développement de capacités délibératives de haut niveau, telles qu'elles peuvent être étudiées sur les systèmes multi-robots terrestres : les aspects décisionnels introduits sont encore embryonnaires. En cela, le projet COMETS a fourni un cadre privilégié motivant pour concevoir et expérimenter notre architecture.

I.3 Conclusion

Nous avons présenté dans ce chapitre les différents champs de recherche en rapport avec le présent le travail. L'approche que nous proposons s'appuie sur des techniques de coordination d'UAV à base de rôles, dans le cadre d'une architecture délibérative destinée à augmenter la prise de décision (et la coordination) autonome dans un système d'UAV. Les travaux existant sur les systèmes multi-UAV ne considèrent que des primitives de coordination réactives, de bas niveau, et ne mettent pas en œuvre de démarches délibératives. C'est à ce niveau que se positionne principalement notre contribution : ce travail introduit une architecture de décision et des techniques de coordination de haut niveau visant à augmenter l'autonomie décisionnelle des robots dans un systèmes multi-UAV.

Chapitre II

Architecture décisionnelle pour systèmes multi-UAV hétérogènes

Dans ce chapitre, nous jetons les bases de notre architecture. Nous présentons d'abord, dans la première section, les caractéristiques que nous souhaitons y exhiber. Dans la deuxième section, nous proposons une notion d'autonomie décisionnelle des robots, donnant lieu à une classification en *degrés d'autonomie* articulée autour d'un exécutif générique que nous appelons Exécutif Multi-Degrés (EMD). Sur ces bases, la troisième partie introduit la couche délibérative qui vient compléter, pour les plus hauts degrés d'autonomie décisionnelle, l'EMD des robots. Nous y discutons finalement les choix réalisés.

Remarque : Ce chapitre, ainsi que les deux suivants, introduisent un nombre important de formalismes, souvent accompagnés d'acronymes : nous rappelons qu'un glossaire est à la disposition du lecteur, à la fin du mémoire.

II.1 Une architecture à géométrie variable

L'approche rapportée dans ce mémoire décrit une architecture décisionnelle et de contrôle pour systèmes multi-UAV, qui propose différentes configurations d'autonomie décisionnelle entre une entité centrale (centre de contrôle et interface opérateurs) et les UAV du système. A un extrême, l'intégralité des capacités décisionnelles est concentrée au niveau de l'entité centrale : chaque UAV reçoit une requête élémentaire d'exécution de tâche, et retourne les statuts d'exécution correspondants, avant de recevoir la tâche suivante, etc. A l'autre extrême, une mission de haut niveau est identifiée au niveau central par un opérateur. Une fois transmise aux UAV du

système, ceux-ci organisent d'eux-mêmes la décomposition de la mission en tâches, et l'allocation des tâches est gérée entre les UAV. Nous proposons une approche qui vise à couvrir tout le spectre des besoins, d'un extrême à l'autre.

II.1.1 Autonomie de décision et décision d'autonomie

On peut être amené à s'interroger sur le "besoin" de décision décentralisée, déléguée au niveau de UAV, en opposition à un contrôle décisionnel géré au niveau d'un centre de contrôle, très proche des opérateurs. L'autonomie de décision est certainement un atout dans l'absolu, mais qu'en est-il en pratique, du point de vue de l'utilisateur du système dans un contexte opérationnel ? Les possibilités qu'une autonomie décisionnelle accrue autorise, doivent être un support, un outil, mais certainement pas un fardeau ou une contrainte. Dans un cadre opérationnel nous pensons que l'autonomie décisionnelle des UAV, doit toujours pouvoir être "débrayée" : les utilisateurs doivent pouvoir obtenir à volonté la maîtrise sur les opérations. Ce peut être une nécessité dans une mission où des contingences dépassent les capacités autonomes des UAV. Ce peut être le cas aussi simplement parce que l'utilisateur désire arbitrairement obtenir la main sur le cours des opérations.

Au delà du choix binaire de déléguer - ou pas - le contrôle décisionnel aux UAV du système, nous proposons de décomposer graduellement, plus finement, les capacités décisionnelles que l'utilisateur peut souhaiter déléguer - ou pas - aux robots du système : nous parlerons alors de *degrés d'autonomie décisionnelle*. Cette notion, qui est capitale dans notre approche, est détaillée dans la section II.2.

II.1.2 Le cadre du projet COMETS

Le cadre applicatif du travail rapporté dans ce manuscrit est un projet d'envergure impliquant plusieurs partenaires académiques et industriels européens. Le projet COMETS vise à développer un système pour contrôler en temps réel une flottille d'UAV hétérogènes pouvant agir conjointement. Ainsi, un des objectifs de COMETS consiste à concevoir, expérimenter et évaluer une architecture de contrôle pour systèmes multi-UAV. Les objectifs de démonstration proposés à l'issue des trois années dévolues au projet concernent des applications de surveillance, de détection ainsi que de support opérationnel pour les incendies de forêt (comme la cartographie de zone avant / après sinistre).

Pour des informations plus ciblées sur le projet COMETS, le lecteur est invité à se référer au site internet du projet : [COMETS 05].

Objectifs scientifiques

Avant de nous pencher sur les questions d'architecture que la mise en œuvre de COMETS nous a conduit à considérer, il nous paraît souhaitable de donner une image générale des efforts de conception, de développement et d'intégration requis par le projet COMETS. Ceux-ci touchent un large horizon de la robotique :

– Automatique et contrôle : le projet COMETS a été l'occasion d'étudier ou d'améliorer des lois de commande et fonctions de contrôle pour deux types d'aéronefs : hélicoptère et ballon

dirigeable. Des lois de commandes pour un asservissement en vitesse, en cap ou en suivi de trajectoire sont les briques élémentaires de l'autonomie opérationnelle.

– Vision : une partie importante des efforts a concerné le développement de fonctions perceptuelles dédiées d'une part à la reconnaissance de feu de forêt (détection de foyer et de fumée, modélisation de front d'incendie...), et d'autre part à la cartographie (approches SLAM, stéréovision monoculaire...).

– Architecture : c'est la colonne vertébrale du système. L'architecture de COMETS rassemble les composantes suivantes :

1. Le segment sol : il s'agit des fonctions centralisées, rassemblant principalement des interfaces vers les opérateurs du système, le support à la planification et à la supervision de mission, et le système d'analyse et de traitement des données perceptuelles en provenance des UAV.
2. Le segment aérien : il rassemble les composants matériels et logiciels embarqués, ainsi que les extensions logicielles non-embarquées (ou "virtuellement" embarquées).
3. Le segment de communication : il concerne les média et les protocoles de communication entre les différentes entités du système : "sol $\leftrightarrow$ sol", "sol $\leftrightarrow$ aérien" et "aérien $\leftrightarrow$ aérien".

Une vue globale de l'architecture du système est proposée sur la figure II.1.

Choix et contraintes d'architecture dans COMETS

COMETS vise à intégrer un certain nombre d'UAV au sein d'un même système. Dans son cadre applicatif, les UAV de COMETS sont extrêmement hétérogènes, à un niveau physique (hélicoptères et ballon dirigeable), et aussi à un niveau décisionnel (téléopérés ou dotés d'une certaine autonomie). Ce cadre hétérogène est apparu comme une grande ligne du projet : il était en effet essentiel que l'architecture considère explicitement et concilie efficacement ces différences physiques, fonctionnelles et décisionnelles.

Le segment sol (figure II.1) comprend un système de planification de mission (SPM), des systèmes de monitoring et de contrôle (SMC) qui fournissent des interfaces vers les opérateurs humains, et un système de perception (SP) qui est en charge de traiter de façon centrale les données perçues individuellement par les UAV.

Si le segment sol s'apparente à un centre de contrôle assez conventionnel, le segment aérien comporte des particularités qu'il convient de préciser. Dans les parties "UAV" du segment aérien (figure II.1), un ensemble de fonctionnalités décisionnelles et opérationnelles sont réunies.

Cependant, certains UAV n'ont que peu de capacités de traitement à bord : ceux qui sont téléopérés par exemple. D'autres UAV peuvent avoir des capacités opérationnelles, mais pas assez de moyens pour des capacités décisionnelles. Enfin, certains des UAV peuvent embarquer l'intégralité des composants opérationnels et décisionnels.

Afin de contrôler (en terme de prise de décision) un tel système, il est souhaitable de fournir un cadre qui puisse s'adapter à ces différentes situations.

C'est dans cette optique que les composants "EMD" (Exécutif Multi Degrés) et "CD" (Couche Décisionnelle) ont été définis : au même type que les autres composants, il est envisageable de les embarquer, ou bien de les laisser au sol si les capacités de l'UAV ne l'autorisent pas. Ils permettent d'unifier, de façon distribuée, les formalismes et traitements décisionnels dont le système doit être doté.

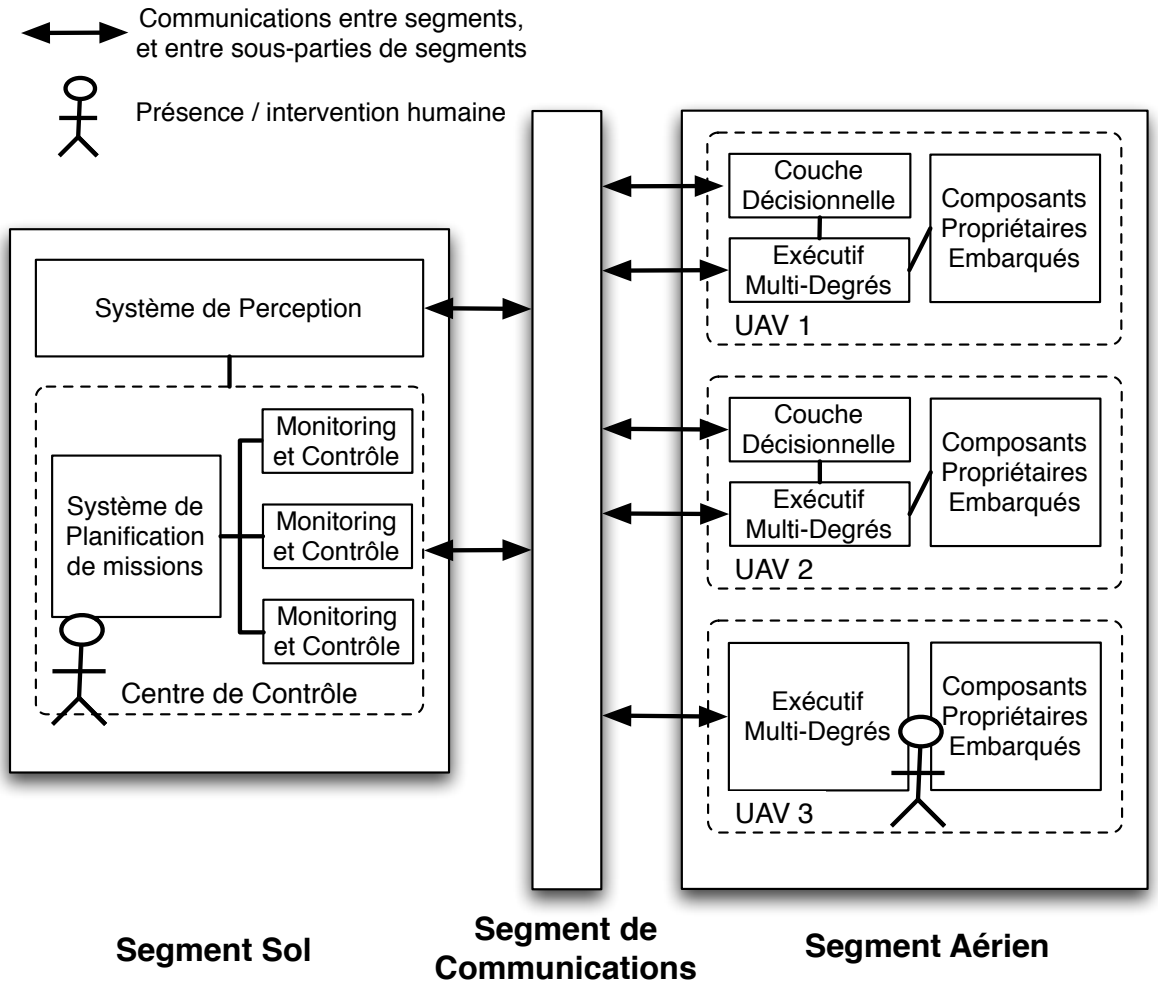


FIG. II.1 – Schéma général de l'architecture du système COMETS

La possibilité de pouvoir embarquer ou pas ces composants permet de proposer une architecture ouverte à de nouveaux UAV, sans contrainte ni présupposés sur les capacités de traitement et l'architecture embarquée des UAV.

Du point de vue du roboticien, en considérant des versions expérimentales du système, il est vraisemblablement préférable d'embarquer autant que possible l'intégralité de ces composants : c'est un gage d'autonomie et de robustesse. En pratique, les contraintes énergétiques et de poids (cette dernière n'est généralement pas un problème pour les robots terrestres) conduisent à limiter significativement, ou tout du moins à chercher le compromis dans ce qui doit être embarqué et ce qui peut être laissé au sol. Les fonctions décisionnelles étant *les moins* exigeantes en termes de réactivité / temps réel, et également *les plus* exigeantes en terme de puissance de calcul, c'est naturellement celles-ci qui sont choisies en priorité pour être traitées au sol. Le risque encouru est de perdre la communication entre les parties embarquées et les parties au sol.

Nous émettons par conséquence l'hypothèse suivante : soit les UAV du système peuvent,

de façon robuste, perdre la communication avec le sol et la retrouver après un certain temps sans conséquence sur leur intégrité logique et matérielle, soit la communication entre le sol et l'UAV est fiable. Pour un UAV téléopéré (par exemple l'UAV 3 sur la figure II.1), le pilote est à l'interface entre l'EMD et les composantes fonctionnelles composantes propriétaires embarquées (CPE, sur la figure) : c'est en particulier dans cette configuration que l'on suppose une communication fiable entre pilote et UAV. Si un UAV est doté d'un minimum de capacités opérationnelles embarquées, on peut supposer que des routines de sécurité permettent de gérer temporairement une perte de contact avec le sol.

L'architecture présentée sur la figure II.1 est celle du système multi-UAV. Au niveau de chaque UAV, on peut considérer, en référence à la section I.1.1, que l'architecture est de type hybride : chaque EMD, éventuellement accompagné d'une CD, représente le niveau délibératif (soit l'exécutif fait partie de cette couche délibérative, comme dans l'architecture LAAS [Alami 98a], soit il correspond à une couche de l'architecture à part entière, à la façon de 3T [Bonasso 97]) : nous regrouperons ces composants délibératifs sous l'appelation *noeud décisionel distribué* (NDD), en opposition aux capacités de prise de décision centralisées, dans le CC, qui est donc considéré comme le *noeud décisionel centralisé* (NDC).

En poursuivant le parallèle avec les architectures hybrides, les Composants Propriétaires Embarqués (CPE) représentent alors le niveau fonctionnel, réactif, proche des aspects matériels.

II.1.3 Conclusion

La nécessité de considérer les facteurs humains d'opération d'un système multi-UAV, et la mise en œuvre effective d'un tel système dans le projet COMETS, muni d'aéronefs extrêmement hétérogènes, sont les points de départ et les lignes directrices du développement de notre architecture. Les deux prochaines sections s'attachent à décrire les principes et le contenu proprement dit d'une architecture qui répond aux besoins énoncés.

II.2 Architecture et autonomie décisionnelle

Dans le paradigme que nous défendons, l'humain est à même de décider que la prise de décision soit déléguée aux robots du système ou bien, au contraire, que la décision soit de son propre ressort, au niveau d'un centre de contrôle en charge du système.

Nous proposons explicitement une classification de la "charge d'autonomie" qu'un opérateur peut déléguer aux robots du système.

II.2.1 Une classification en degrés d'autonomie décisionnelle

Dans la classification proposée, les configurations de répartition des capacités décisionnelles sont établies selon cinq degrés d'autonomie décisionnelle. A chaque degré est associé un ensemble de fonctionnalités décisionnelles qui sont déléguées au robot. Les critères choisis représentent des capacités décisionnelles dans un sens large, c'est à dire mettant en œuvre des mécanismes qui manipulent et infèrent sur des modèles de l'environnement, des robots et des interactions. Nous y opposerons les capacités *opérationnelles*, regroupant les mécanismes servant à synthétiser

ces modèles à partir des données perçues, et les mécanismes de génération de consignes pour les actionneurs (il s'agit principalement des fonctions sensori-motrices).

Les capacités décisionnelles que nous exhibons dans cette classification sont les suivantes :

1. Supervision : l'aptitude à gérer le déroulement d'un plan de tâches donné et à traiter de façon adéquate les statuts d'exécution retournés.
2. Coordination exécutive : l'aptitude à gérer des mécanismes simples de coordination entre robots, sur la base de messages de synchronisation par exemple, au moment de l'exécution.
3. Planification et coordination décisionnelle : l'aptitude à synthétiser un plan de tâches exécutables, à partir de l'énoncé d'une mission de haut niveau, et à mettre en œuvre des mécanismes de coordination pertinents afin de rendre les plans de tâches cohérents parmi les robots du système (ou autrement dit un "plan global" ou "plan joint" cohérent).
4. Ré-allocation dynamique de tâche : aptitude à remettre en cause en cours de mission, lorsque cela est pertinent, l'allocation des tâches parmi les robots du système.

Ces quatres capacités (ou ensembles de capacités) reflètent des caractéristiques de la *prise de décision*, au sens large, ou de fonctionnalités nécessaires à la mise en œuvre de cette prise de décision en contexte multi-robot. Par ailleurs, la césure proposée entre ces capacités permet d'isoler des fonctionnalités décisionnelles qui ont une signification tangible pour un opérateur : le premier point concerne la supervision du déroulement d'un plan donné, pour un robot. Le deuxième point s'attache au déroulement d'un ensemble de plans parmi plusieurs robots, y compris lors de synchronisations inscrites dans ces plans. Le troisième point concerne la planification et la coordination des tâches, de façon à répondre à une mission de haut niveau donnée. Le quatrième point permet finalement de réviser de façon opportuniste la distribution des tâches parmi les robots.

Chacune de ces capacités peut être gérée par l'entité centrale NDC (Nœud Décisionnel Centralisé) et liée aux opérateurs humains, ou bien déléguée aux robots du système, chaque robot étant alors doté d'un NDD (Nœud Décisionnel Distribué).

Dans le premier cas, chacune de ces capacités peut être indifférement réalisée par des systèmes automatiques ou par un opérateur humain , et plus vraisemblablement par une combinaison des deux, comme cela a été le cas dans le projet COMETS - voir le chapitre VI. Par contre, dans le second cas, la délégation (que nous attacherons toujours, dans la suite de ce manuscrit, à la *distribution* des capacités décisionnelles) repose nécessairement sur des traitement automatiques.

Il est important de noter que, si les capacités décisionnelles présentées peuvent théoriquement être indépendamment centralisées ou distribuées auprès des robots du système, nous ne considérons qu'un sous-ensemble de configurations qui nous paraissent avoir du sens dans un système multi-robot. Ces configurations sont exprimées du point de vue d'un robot, par rapport aux capacités dont il est muni : les capacités sont hiérarchisées de telle sorte qu'un robot ne puisse être doté (sous-entendu, qu'on puisse lui déléguer) des capacités étiquetées N que si et seulement si il est déjà doté des capacités étiquetées N-1. Cette hiérarchie reflète, à notre à sens, plusieurs considérations :

– **Une complexité croissante des capacités décisionnelles** proposées, sous la forme d'un découpage incrémental intuitif en "paquets décisionnels", auxquels un opérateur humain peut attacher une signification tangible : déléguer la supervision est un dégagement de responsabilités minimales, accompagné de peu de risques. Autoriser une coordination exécutive autonome revient à laisser aux robots du système le soin de dérouler le plan global (fourni par

ailleurs dans ses détails), jusqu'aux synchronisations d'exécutions de tâche. La planification et la coordination décisionnelle représentent une rupture importante : il s'agit de déléguer les capacités de décomposition de missions en plans exécutables dans le contexte multi-robot, en exploitant des fonctionnalités autonomes complexes de coordination . L'allocation dynamique de tâches laisse finalement carte blanche aux robots du système pour réaliser une mission donnée, exprimée à un haut niveau d'abstraction par un opérateur : on peut parler d'auto-organisation.

– **Une dépendance fonctionnelle des capacités supérieures** vis à vis des capacités inférieures : l'allocation dynamique de tâches (point 4) suppose la disponibilité de fonctions d'évaluation de plans dans un contexte multi-robot, ce qui s'apparente aux fonctionnalités de planification et coordination décisionnelle (point 3). Celles là même supposent des fonctionnalités de coordination élémentaires, matérialisées par la coordination exécutive (point 2). Enfin la coordination exécutive nécessite des mécanismes de supervision de l'exécution (point 1).

– **Des exigences décroissantes en terme de réactivité** (ou inversement, une flexibilité croissante par rapport aux contraintes "temps-réel"). Les capacités de plus bas niveau ont en effet des contraintes temps-réel plus prononcées que celles de haut niveau : il paraît alors souhaitable de déléguer en priorité aux robots les capacités appelées à fonctionner avec de plus fortes contraintes temporelles, et sous cette perspective, il paraît peu pertinent que des capacités supérieures soient distribuées alors que les capacités inférieures ne le sont pas.

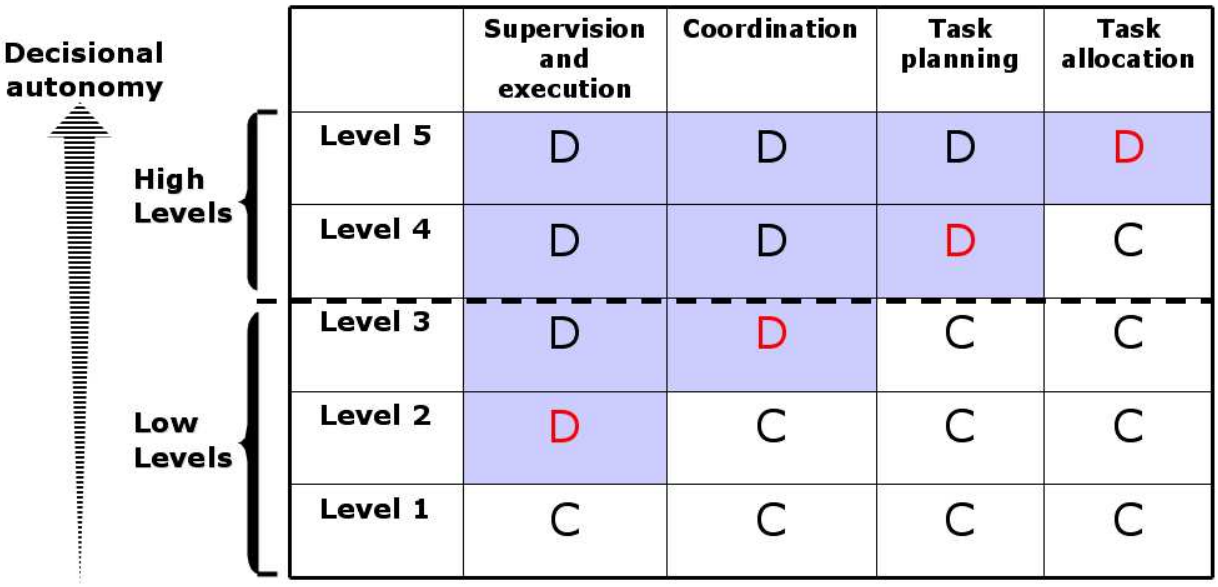


FIG. II.2 – 5 degrés d'autonomie décisionnelle (C signifie *Centralisé*, et D signifie *distribué*)

Suite à ces considérations, nous proposons la classification en degrés d'autonomie illustrée sur la figure II.2. Les degrés 1 à 3 sont regroupés en tant que *faibles degrés d'autonomie* : dans ces configurations, le NDC est en charge de la prise de décision en tant que telle et du maintien de la cohérence du plan global. Les degrés 4 et 5 représentent quant à eux des degrés élevés d'autonomie décisionnelle : les capacités majeures de prise de décision dans le contexte multi-

robots sont alors déléguées au NDD. La section suivante apporte des précisions sur les degrés faibles et élevés d'autonomie décisionnelle.

II.2.2 Degrés faibles

Degré 1 :

Dans cette configuration, l'intégralité des capacités décisionnelles est gérée au niveau du NDC, et le robot est directement aux ordres du NDC. Le NDD se comporte alors comme une interface transparente entre NDC et CPE. Cette configuration est celle de la figure II.3.

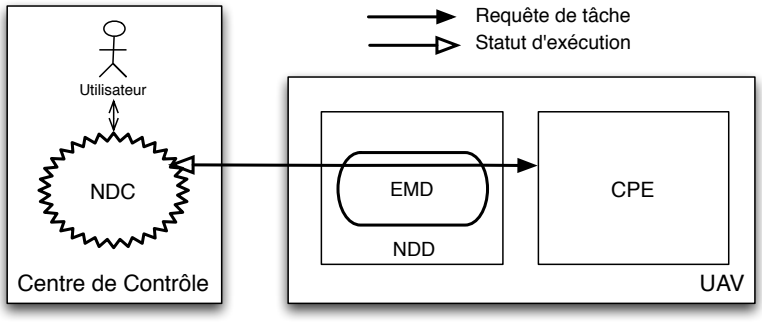


FIG. II.3 – Degré 1 : pas de capacité décisionnelle autonome

Degré 2 :

Dans cette configuration, l'EMD (Exécutif Multi-Degrés) devient actif : il gère directement les séquences de tâches, et supervise leur exécution par le CPE. Il s'agit donc de déléguer au robot la *supervision de tâche* (figure II.4).

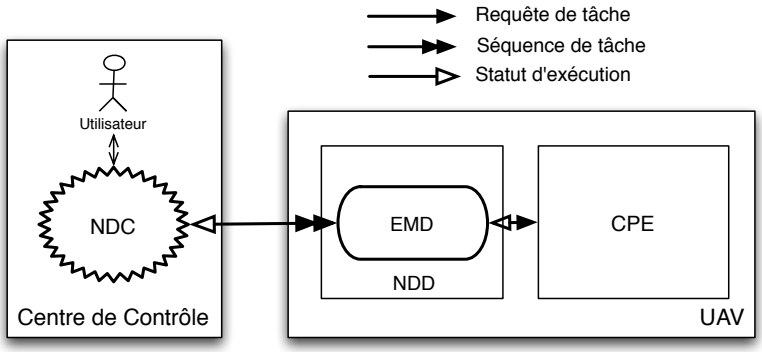


FIG. II.4 – Degré 2 : supervision autonome

Degré 3 :

Cette configuration permet une coordination exécutive par l'envoi de messages directs entre robots (de EMD à EMD). Elle met en œuvre peu de changements dans les mécanismes de supervision, par rapport aux degrés inférieurs, mais correspond à une avancée importante en terme d'autonomie du système multi-robots : c'est le premier pas vers des coordinations directes entre UAV (figure II.5).

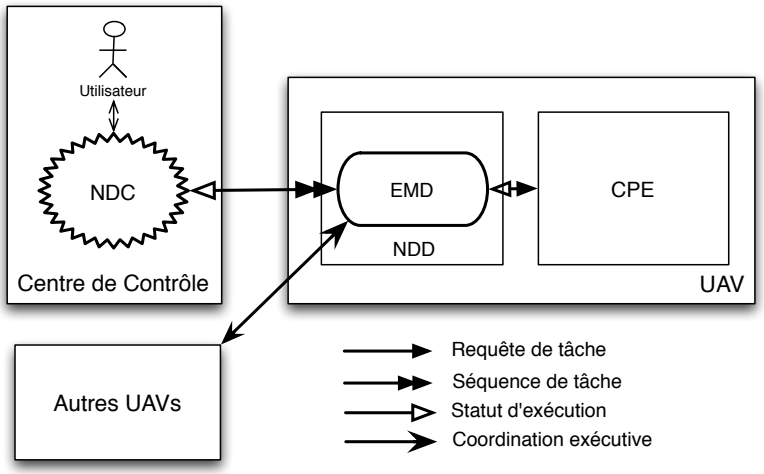


FIG. II.5 – Degré 3 : coordination exécutive autonome

II.2.3 Degrés élevés

Degré 4 :

Un saut important existe entre le degré 3 et le degré 4 : les fonctions de prises de décision proprement dites sont en effet dorénavant déléguées au robot, ainsi que les moyens de s'assurer de la cohérence des plans globaux (union des plans des UAV). Une couche décisionnelle dans le NDD se substitue alors au NDC pour générer les plans : ceux-ci, après traitements (coordination décisionnelle en particulier), sont transmis sous forme de séquences de tâches à l'EMD (figure II.6). Du point de vue de l'EMD, le fait que les tâches proviennent du NDC ou du NDD est transparent.

Degré 5 :

Le degré 5 autorise finalement le robot à remettre en question son allocation de tâches, et par conséquent à négocier avec les autres robots la ré-allocation des tâches dynamiquement, c'est-à-dire en cours de mission (figure II.7). Un ou des protocoles d'allocation de tâches doivent être employés dans cette optique.

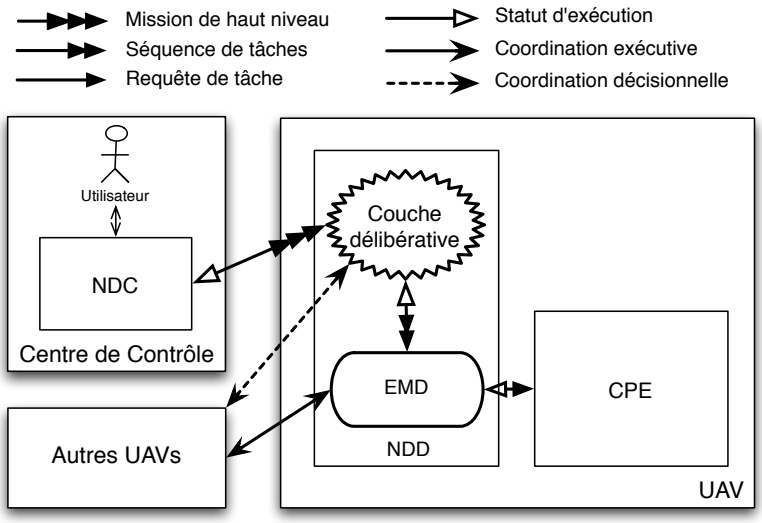


FIG. II.6 – Degré 4 : planification et coordination exécutive autonomes

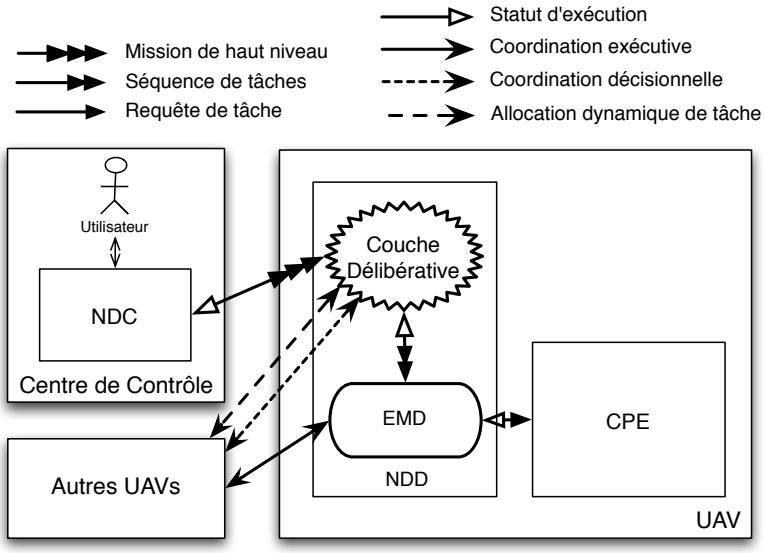


FIG. II.7 – Degré 5 : allocation dynamique autonome

II.2.4 Conclusion

Les degrés d'autonomie présentés dans cette section tracent de façon assez naturelle et incrémentale les besoins en terme de capacités de décision. Au niveau du NDD, l'EMD rend possible un interfaçage alternatif entre prise de décision par NDC et prise de décision par NDD (la mise en œuvre de cet exécutif fait l'objet de la section III), tandis que la CD regroupe les composants destinés à la prise de décision décentralisée, au niveau des robots du système. Nous présentons la couche délibérative dans la section II.3 ci-après.

II.3 Proposition d'une couche délibérative orientée interactions multi-robots

La couche délibérative introduite dans cette section regroupe un ensemble de composantes qui sont associées pour fournir différentes capacités décisionnelles autonomes (donc au niveau de l'UAV). Nous présentons en particulier le cadre et les moyens de planification de tâches, et abordons la mécanique délibérative générale, dans un contexte multi-robots. Les chapitres IV et V décrivent par la suite beaucoup plus en détail les composants de la couche délibérative et les mécanismes sous-jacents.

II.3.1 Architecture délibérative : tour d'horizon

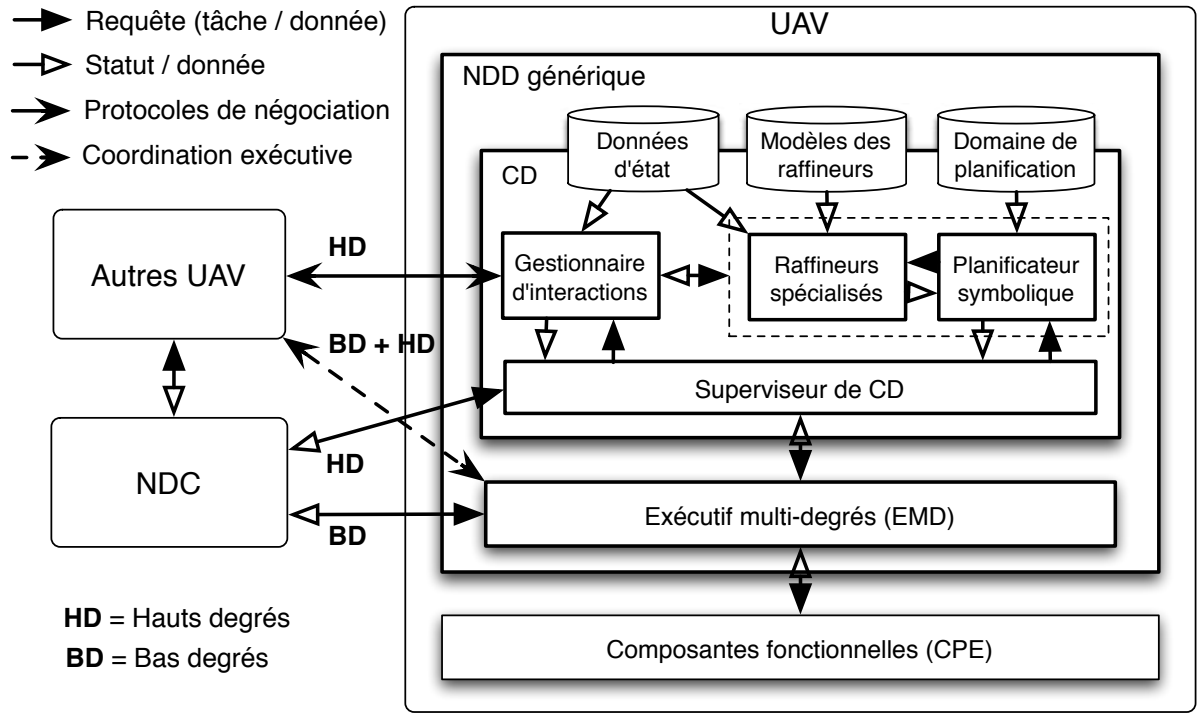


FIG. II.8 – Architecture décisionnelle : schéma général

Au cœur de l'architecture délibérative, la couche délibérative proprement dite regroupe 4 entités, comme illustré sur la figure II.8 :

1. **Le planificateur symbolique (PS) :** il s'agit du "moteur" de planification, utilisé pour générer des plans de tâches à partir de missions de haut niveau. Il décompose et ordonne partiellement dans un plan les tâches à réaliser, à partir d'une formulation hiérarchique (HTN) du domaine de planification. Le paragraphe II.3.2 ci-après présente plus en avant cette approche.

2. **Les raffineurs spécialisés (RS) :** ils peuvent être vus comme des planificateurs géométriques spécialisés qui exploitent les modèles disponibles au niveau de l'UAV : le modèle d'UAV lui-même, le modèle d'environnement, ou encore les modèles de communication. Nous les présentons dans le paragraphe II.3.3.
3. **Le gestionnaire d'interactions (GI) :** il regroupe les mécanismes de coordination de haut niveau, notamment à travers l'utilisation de modèles d'interaction. Ces mécanismes permettent de raffiner des tâches jointes, dont il sera question dans le chapitre IV. Le GI joue un rôle crucial dans les interactions entre robots.
4. **Le superviseur de la couche délibérative (SCD) :** il organise les activités et rythme les flux de données dans la couche délibérative, en employant notamment des opérateurs de traitements sur les plans.

Chacune de ces composantes est brièvement présentée ci-après.

II.3.2 Le planificateur symbolique

Le planificateur symbolique est employé pour élaborer des plans de tâches, sous la forme de plans partiellement ordonnés : des dépendances temporelles sont en effet définies entre certaines tâches. Le formalisme adopté pour la description de tâches et missions est directement lié au paradigme de planification symbolique que nous avons choisi : il s'agit en l'occurence de Shop2, système de planification développé à l'université du Maryland, USA, par l'équipe du Pr. Dana Nau [Nau 03]. Ce planificateur a remporté plusieurs prix lors de la compétition internationale de planification en 2002.

Shop2 est un système de planification indépendant du domaine, appartenant à la famille des "HTN" (Hierarchical Task Network).

Les planificateurs HTN exploitent une définition préalable de moyens de décomposer des tâches de haut niveau (dites *méthodes*) en sous-tâches plus élémentaires, jusqu'à parvenir aux tâches les plus élémentaires (appelées *opérateurs*). Un plan est alors donné par un ensemble d'opérateurs partiellement ordonnés.

Les planificateurs HTN s'opposent généralement aux planificateurs dans lesquels la structure du domaine et les dépendances entre tâches ne sont pas données *a priori* au système : le planificateur exploite alors des opérateurs disponibles pour évoluer dans l'élaboration d'un plan, en effectuant la recherche soit dans l'espace des plans partiels (comme par exemple Ixtet [Laborie 95]), soit dans l'espace des états (par exemple TALplanner [Doherty 99]).

Shop2 permet une inférence axiomatique, autorise le mélange d'expressions numériques et symboliques, et permet des appels à des programmes extérieurs.

L'utilisation de Shop2, comme pour la plupart des planificateurs, suppose de définir d'un côté un *domaine*, décrivant les méthodes et opérateurs propres à la structure du domaine auquel doit s'appliquer le planificateur, et de l'autre côté un *problème*, dans lequel sont mentionnés l'état "initial", sous forme de faits, ainsi que des "buts" : dans les planificateurs HTN comme Shop2, les buts sont exprimés sous forme d'appels explicites à des méthodes du domaine. Un problème P se définit toujours dans le cadre d'un domaine D particulier.

II.3.3 Les raffineurs spécialisés

Les raffineurs fournissent un ensemble de services spécialisés pour la planification géométrique : ils exploitent différents types de modèles (UAV, environnement, communication) pour décomposer des tâches dans un contexte donné.

Le développement des raffineurs a été l'objet du travail de DEA de Gautier Hattenberger [Hattenberger 04, Gancet 05].

Les raffineurs spécialisés sont utilisés par le planificateur symbolique. En effet, le mécanisme de recherche du planificateur symbolique permet de faire appel à des routines extérieures au planificateur : lors de la décomposition de méthodes en opérateurs, il est possible d'effectuer, en cours de recherche, un appel vers les raffineurs spécialisés. La requête nécessaire est formulée dans une routine annexe du planificateur, et transmise aux raffineurs. Ceux-ci traitent la requête et retournent les informations attendues (dans une situation nominale). Le planificateur symbolique exploite alors les informations fournies pour détailler les paramètres des opérateurs ou évaluer un coût de réalisation d'une opération, avant de poursuivre la recherche (voir chapitre IV).

II.3.4 Le gestionnaire d'interactions

Le *Gestionnaire d'Interactions* (GI) joue un rôle prépondérant dans les interactions entre UAV. Il est au cœur du paradigme choisi pour toutes les opérations coordonnées non-triviales (i.e. ne pouvant reposer uniquement sur des synchronisations approximatives occasionnelles entre les UAV). Nous nommerons ces activités *Tâches Jointes* (TJ) dans la suite de ce manuscrit. Le GI repose sur l'utilisation de *Modèles d'Interaction* (MI). Un modèle d'interaction rassemble des *rôles* qui doivent être endossés par des UAV, pour qu'une tâche jointe donnée puisse être réalisée. A un type de tâche jointe donné sera donc associé un modèle d'interaction particulier. Les rôles contiennent des "plans de haut niveau", dont le raffinement se produit en ligne, dans le contexte d'exécution courant.

Tous les détails relatifs aux différents modèles utilisés (rôle, modèle de contraintes spatiales, table de coordination) et à leur mise en œuvre proprement dite sont décrits dans le chapitre V.

Les rôles définissent donc les actions à entreprendre dans le cadre "joint", c'est à dire durant la période où les UAV sont supposés opérer conjointement. Pour ce faire, les rôles peuvent en particulier faire appel à des *modalités de négociation* (MN), qui définissent des cadres de communications dédiés à la résolution de conflits de ressources (coordination) ou à de l'allocation de tâches, par exemple. Certaines de ces modalités peuvent être définies dans un cadre générique, sans restriction à un domaine particulier, et au contraire d'autres modalités peuvent être conçues explicitement pour un cadre applicatif particulier.

II.3.5 Le superviseur de CD

La couche délibérative est munie d'un superviseur, que nous pouvons qualifier de *superviseur délibératif*, en opposition à la notion de superviseur d'exécution ou *exécutif*, qui désigne dans notre cas l'EMD.

Le superviseur de la couche délibérative trouve une utilité du fait de la multiplicité des composantes délibératives de haut niveau : en particulier, la présence du gestionnaire d'interactions, en plus du couple planificateur symbolique / raffineurs, induit une complexité nouvelle dans la

construction, l'évaluation ou la remise en cause des plans. L'introduction du superviseur de CD permet de déléguer des traitements ou manipulations de plans à une entité tiers, un "chef d'orchestre" qui, au centre de la couche délibérative, assure la cohérence de la construction de plans.

Ce superviseur est par ailleurs l'interface de la couche délibérative, vu par exemple de l'exécutif multi-degrés (EMD). A ce titre, il doit assurer une réactivité cohérente avec les besoins des entités de plus bas niveau (exécutif et couche fonctionnelle), donc assumer des niveaux de contraintes proches du temps réel qu'il est difficile de garantir directement aux niveau des autres composantes décisionnelles.

En terme de proximité du temps-réel, nous pouvons donc établir l'ordre suivant :

Composantes délibératives $\prec$ Superviseur CD $\prec$ EMD $\prec$ Composantes fonctionnelles (CPE)

Le superviseur de CD manipule les plans en utilisant un certain nombre d'opérateurs : ceux-ci concernent différents types de raffinement, d'évaluation, de traitements sur les plans, qui seront opérés par les différentes composantes de l'architecture (CPE ou EMD).

II.3.6 Remarques et réflexions sur cette architecture

Nous venons de présenter brièvement les éléments qui composent notre architecture : l'EMD y joue un rôle pivot, articulant la prise de décision avec le niveau fonctionnel pour les bas degrés comme pour les hauts degrés d'autonomie décisionnelle.

A notre connaissance, aucune autre architecture de système robotique considère explicitement, de la sorte, une délégation incrémentale de capacités décisionnelles. Cette approche est d'autant plus pertinente que nous nous situons dans un contexte multi-robots, (et plus précisément *multi-UAV*, dans lequel une délégation d'autonomie est très appréciable compte tenu de la quantité des opérations à réaliser dans un tel système. Mais paradoxalement, la présence de l'humain est aussi hautement souhaitable, pour reprendre la main dans des situations critiques par exemple (ou pour apporter ses connaissances, son expertise au système, dans des situations données ; nous n'aborderons cependant pas ici la problématique de l'initiative mixte).

Dans les configurations d'autonomie décisionnelle où la prise de décision est laissée principalement aux robots (degrés 4 et 5), notre couche décisionnelle est à l'œuvre : abstraction faite du reste de l'architecture du système, cette couche correspond aux fonctionnalités de planification et de coordination avec les autres robots du système. En nous penchant sur la littérature, les approches multi-robots réellement implémentées et testées sur des robots sont peu nombreuses. Notons que la thèse de B. Dias [Dias 04] couvre et compare assez exhaustivement les approches existantes. L'accent est mis en particulier sur les approches employées et les niveaux d'implémentations et de validation sur des systèmes bien concrèts. L'approche TraderBots que Dias propose, comme plusieurs autres approches ([Gerkey 02], [Laengle 98]...) dites "Market based" (paradigme d'économie de marché), considèrent un mécanisme unique de négociation, valable pour des tâches élémentaires dont le coût peut être obtenu lorsque nécessaire. Si ces approches ont effectivement fait leur preuve dans des systèmes où l'interaction entre robot correspond, le plus souvent, à de l'allocation de tâches de déplacement vers des lieux qui doivent être visités, elles atteignent rapidement des limites lorsque les tâches se complexifient, ou nécessitent une coordination plus étroite ("Tight Coordination").

Nous avons souhaité introduire, dans notre approche, la possibilité de coordonner les tâches à exécuter à travers de multiples modalités de négociations, dans le cadre de rôles attachés à la réalisation de tâches jointes. La formulation de rôle pour expliciter le cadre d'interactions entre agents est une approche popularisée par Tambe [Tambe 97], dans laquelle STEAM, un cadre automatisé d'organisation en équipe, est appliqué à chaque "intervenant" (agent, robot, humain...) modélisé dans le système. Des "opérateurs d'équipe", basés sur la notion "d'intentions jointes" [Levesque 90], définissent des activités à réaliser en équipe. Dans l'approche de Tambe, l'apprentissage est introduit pour adapter le cadre de coordination défini dans STEAM à une application donnée. Implémentée avec succès dans des systèmes multi-agents, l'approche de Tambe n'a pas encore été démontrée sur un système de robots réels. Notre approche peut être mise en parallèle avec celle de Tambe dans la mesure ou le Gestionnaire d'Interaction que nous introduisons dans la couche délibérative peut s'identifier au cadre de STEAM, lui aussi à base de rôles. Les modèles d'interaction que nous introduisons peuvent être rapprochés des opérateurs d'équipe de Tambe. Cependant, nous ajoutons à notre couche délibérative des moyens de raisonnement (planificateur symbolique) dont ne dispose pas l'architecture de Tambe. Par ailleurs, nous ne nous intéressons pas, dans ce travail, aux capacités d'apprentissage. Dans notre démarche, nous considérons explicitement les problématiques et domaines multi-UAV, avec les particularités applicatives que nous avons introduites dans le chapitre I. Nous considérons en particulier des modèles de contraintes spatiales explicites : les tâches que les UAV doivent accomplir reposent en effet en grande partie sur leur positions et déplacements, et ce plus encore que pour les robots mobiles terrestres (une erreur de déplacement est susceptible d'être fatale pour un UAV, beaucoup moins pour un rover, par exemple). De plus, la criticité de déploiement et d'opérations de plusieurs aéronefs non-habités requiert, à notre sens, de pouvoir définir explicitement le cadre spatial des opérations jointes. Comme il en est question dans le chapitre V, le GI propose la définition explicite d'un cadre spatial de ce type, dans la définition des interactions.

On peut encore s'interroger sur la pertinence de découpler la planification de tâche à proprement parler (couple PS + RS) de la coordination des plans (GI). A notre connaissance, aucune approche de planification et coordination entremélés n'a jamais été démontrée dans un système multi-robots réels. Des expérimentations en simulation distribuées dans le cadre de problématiques agents ont d'ores et déjà donné des résultats prometteurs (par exemple l'équipe de Fiorino [Pellier 05], où un plan commun joint est dynamiquement coordonné et révisé le cas échéant). Mais à notre sens, la mise en œuvre d'un tel système est encore hors de portée, du fait des contraintes bien réelles rencontrées dans les systèmes robotiques (communication, incertitudes / contingences, considérations temporelles, etc.).

Parallèlement, nous avons choisi d'opter pour un système de planification efficace (rapidité de production de plan), qui puisse profiter des connaissances du domaine (rappelons-le, ce travail s'attache en premier lieu aux systèmes multi-UAV), ce à quoi la représentation HTN du domaine répond parfaitement. Le choix d'une coordination différée de plans, dans notre système, repose sur la constatation suivante : à supposer que des moyens de coordination temps réel en cours de planification existent, deux questions se posent :

– celle de la qualité et du débit de communication requis,
– et celle de la remise en cause de plans.

S'il est possible de surmonter la question de la communication en simulation, la mise en œuvre effective d'un système de plusieurs UAV montre combien la question de la communication continue de données est complexe à mettre en œuvre de façon fiable. Nous rejetons donc

une approche où le maintien permanent de la communication est indispensable à la génération individuelle de plans.

De plus, la génération (et a fortiori la révision éventuelle) de plans doit pouvoir être réalisée dans des temps compatibles avec l'horizon d'exécution d'un UAV. Sans être en mesure de le quantifier, il ne nous paraît pas concevable, dans l'état actuel des techniques disponibles, de considérer un système dans lequel la planification et la coordination sont entremélés en "temps réel" (ou compatibles avec du temps réel) dans un contexte de robotique aérienne distribuée.

En choisissant une approche de coordination de plan *a posteriori*, nous réduisons le potentiel de coordination "étroite" (tight coordination), mais en contrepartie, nous disposons de moyens de réaction plus rapides lors d'aléas : un plan (au moins partiellement) exécutable peut être produit rapidement, à tout moment, sans être soumis pour cela à un protocole de validation ou d'assentiment collectif.

L'approche délibérative que nous présentons est dans les faits plus exactement hybride : si la coordination est de façon générale réalisée *a posteriori*, nous pouvons considérer que le traitement proprement dît des tâches jointes mêle dans une certaine mesure coordinations (sur une base de négociations) et exécution.

Ces travaux ont fait l'objet de publications ([Gancet 04, Gancet 05]), et nous invitons le lecteur à s'y référer en complément de ce manuscrit.

Chapitre III

L'EMD : un exécutif générique

Nous présentons dans cette section les caractéristiques de l'exécutif générique employé dans notre architecture. En premier lieu, nous précisons le formalisme de représentation de tâche et des liens de dépendances entre tâches. Ensuite, les principaux mécanismes de gestion des tâches sont détaillés. Enfin, quelques exemples illustratifs sont proposés.

III.1 Considérations générales et formalisme adopté

L'exécutif générique que nous proposons est construit avec OpenPRS : il s'agit d'un système d'outils et de méthodes pour représenter et exécuter des procédures, qui satisfait en particulier aux besoins de la supervision d'exécution, dans les systèmes robotiques.

PRS, dans ses premières versions, a été développé au SRI en 1992 par F. Ingrand, M. P. Georgeff et A. S. Rao [Ingrand 92]. Défini comme un "exécutif procédural" (PRS signifie "Procedural Reasoning System"), il est basé sur la notion d'agent rationnel qui peut raisonner et planifier sous fortes contraintes temporelles. Un agent PRS peut être vu comme une architecture BDI ("Belief, Desires, Intentions"), comprenant :

– **une base de faits :** Les croyances, représentant les connaissances supposées sur le monde. Elles sont maintenues dans une base de faits, interagissant en permanence avec les procédures de l'agent.

– **une bibliothèque de procédures :** Pouvant être considérées comme partie des croyances, les procédures décrivent les moyens d'action de l'agent. Elles sont dépendantes du contexte. Les procédures comprennent un champ d'invocation, pouvant être activé soit par la présence (ou l'arrivée) d'un fait dans la base de fait, soit par une activation explicite depuis une autre procédure. Un second champ permet de définir un contexte qui doit être satisfait pour que la procédure puisse être activée (on peut parler de préconditions). Un troisième champ est le corps de la procédure, il contient un "programme" définissant des tests, conditions, boucles

et appels à procédures, ou manipulant des faits. Un quatrième champ permet finalement de définir des "effets de sorties", ou post-conditions, sous formes de manipulation de faits de la base de faits.

– **un graphe d'intentions :** Les Intentions, représentant l'activité en cours de l'agent, et conservant l'historique de l'application des procédures, au cas où des retours ("backtrack") seraient nécessaires.

– **un ensemble de buts :** A la façon des planificateurs HTN (voir chapitre II), les buts, lorsqu'ils sont explicitement formulés, correspondent à des appels explicites à des procédures PRS.

PRS a été employé dans de nombreux projets au SRI, à la NASA, et au LAAS-CNRS où il continue à être développé et maintenu. Des interfaces dédiées permettent aux exécutifs développés avec PRS de s'intégrer avec facilité dans l'architecture LAAS [Alami 98a].

Le système PRS, dans sa version la plus récente (OpenPRS), est disponible (open) sur le site [Ingrand 05a]. Des informations supplémentaires sont disponibles sur la page LAAS de Felix Ingrand [Ingrand 05b].

La première motivation de l'EMD est de gérer des séquences de tâches partiellement ordonnées de façon cohérente : l'ordre de déclenchement des tâches est celui décrit par l'ordre partiel. L'ordre est formé par des dépendances entre tâches, sur une base événementielle. Avant de décrire les mécanismes sous-jacents, nous présentons maintenant le formalisme employé pour représenter les tâches et leurs interdépendances.

III.1.1 Formalisme de tâche

Une tâche est définie dans une optique événementielle : elle est délimitée par un événement `started`, possède une certaine étendue temporelle, et se termine par un événement `ended` (dans un déroulement nominal). Elle peut générer d'autres événements en cours d'exécution : `running`, et `aborting`, `aborted`, en cas d'interruption (sortie du contexte nominal : l'état `aborted` est incompatible avec un futur état `ended`).

Voici la définition formelle d'une tâche T dans l'EMD :

Définition III.1

$\mathsf{T} = (t_{type}, t_{id}, \Pi, \Xi, \ params)$, où :
– t_{type} est le nom de la tâche,
– t_{id} est un identifiant unique de tâche,
– Π est un ensemble de préconditions dont la satisfaction déclenche l'exécution de T,
– Ξ est un ensemble de conditions de sortie (eXIt conditions) dont la satisfaction déclenche l'interruption de T si T est en cours d'exécution, et son annulation si T est planifiée mais pas encore en cours d'exécution,
– *params* est un ensemble de paramètres d'exécution pour T.

III.1.2 Formalisme de condition

Par ailleurs, les conditions peuvent être de trois types :

1. Condition CE sur un état E, d'une tâche T : pour que CE soit satisfaite, il faut que la tâche T soit dans l'état E. C'est ce type de précondition qui permet de décrire un ordre partiel sur les tâches.
2. Condition CM sur la réception d'un message : satisfaite si un fait portant le message attendu est établi.
3. Condition temporelle CT sur un instant I (CT ayant une des modalités *déjà passé* ou *pas encore passé*). Satisfaite si CT/I est vérifié.

Π et Ξ sont des formules propositionnelles : nous les exprimons sous forme normale conjonctive, ou FNC (conjonction de clauses), dont chaque littéral correspond à une condition (ou sa négation), comme décrite précédemment.

Exemple : $(A \vee B \vee C) \wedge (\neg A \vee D) \wedge F$,
où $\{A, B, C, D, F\}$ est un ensemble de conditions.

FNC(F) existe quelque soit F : il s'agit du théorème de complétude fonctionnelle par FNC. Toute formule propositionnelle peut donc théoriquement être transformée en FNC.

La principale motivation pour cette représentation est de l'ordre de l'implémentation : d'une part, l'expression d'une formule sous forme de FNC est propice à une représentation concise (matrice à 2 dimensions). D'autre part, le traitement d'une formule en tant que FNC est beaucoup plus direct que l'analyse d'une formule propositionnelle quelconque.

On dira que Π ou Ξ est satisfait, à un instant donné, si la formule propositionnelle sous-jacente est vraie à cet instant.

III.1.3 Modalités sur les conditions

Enfin, les conditions peuvent être l'objet de deux modalités :
– $\Box$: Obligatoire.
– $\Diamond$: Optionnelle.
Ces modalités concernent l'effet sur les littéraux (les conditions) lorsque leur satisfiabilité est remise en cause. S'il s'avère, pour une raison arbitraire, qu'une condition C, dotée de la modalité M ne peut plus être satisfaite, alors cette modalité donnera lieu à l'effet suivant :
– M = $\Box \Rightarrow$ C devient définitivement faux.
– M = $\Diamond \Rightarrow$: C devient vraie.

Cette notion de satisfiabilité est intéressante à considérer dans les dépendances entre tâches : la mise en œuvre est précisée dans la section suivante, et un exemples de traitement de tâches dans l'EMD est ensuite proposé dans la section III.3.

III.2 Principales fonctionnalités

Nous décrivons dans cette section les principaux mécanismes de l'EMD : il s'agit de l'insertion et l'annulation dynamique de tâche, du déroulement de tâches planifiées et de la gestion des synchronisations.

III.2.1 Gestion des requêtes

L'EMD reçoit des requêtes sur les tâches, que ce soit en provenance d'un centre de contrôle (NDC) ou de la couche délibérative du NDD (voir section II.2).

Les requêtes suivantes sont acceptées par l'EMD :

– **START** : requête permettant l'insertion d'une tâche dans le plan courant de l'EMD,

– **ABORT** : requête permettant l'annulation d'une tâche du plan courant,

– **PROCESS-MSG** : requête permettant la transmission d'un message à l'EMD,

– **UPDATE-COND** : requête permettant la mise à jour des conditions (préconditions / conditions de sortie) d'une tâche,

– **SUPPRESS** : requête de suppression d'une tâche parvenue dans un état terminal (`ended` ou `aborted`).

Lors d'une requête de type START, nous distinguons plusieurs modes possibles d'insertion de la tâche associée dans le plan courant de l'EMD : ce mode d'insertion conditionne la façon dont les dépendances vont être établies entre cette tâche et les tâches déjà planifiées.

Les modes d'insertion de tâche sont :

– SEQ (SEQuential task) : c'est le mode d'insertion principal. Il permet de définir explicitement les conditions souhaitées (Π et Ξ) pour la tâche à insérer.

– VUT (Very Urgent Task) : ce mode permet de déclencher immédiatement la tâche à insérer, en lui accordant une priorité supérieure à toutes les tâches d'ores et déjà planifiées. Si des tâches incompatibles sont en cours d'exécution, l'insertion de la nouvelle tâche déclenche l'interruption de ces tâches incompatibles.

– DEP (DEPendant task) : ce mode permet d'insérer une tâche en demandant implicitement que ses préconditions Π regroupent en conjonction, pour chacune des tâches déjà planifiées ou en cours d'exécution, une condition dont la modalité est **obligatoire** sur l'état `ended` sur cette tâche. Par conséquent, si une des tâches planifiée ou en cours d'exécution se retrouve dans un état `aborted`, alors la tâche à insérer se retrouve également dans l'état `aborted`.

– NUT (Non Urgent Task) : ce mode permet d'insérer une tâche en demandant implicitement que ses préconditions Π regroupent en conjonction, pour chacune des tâches déjà planifiées ou en cours d'exécution, une condition dont la modalité est **optionnelle** sur l'état `ended` sur cette tâche.

La figure III.1 illustre le mécanisme d'insertion de tâche, en considérant en particulier les requêtes START, ABORT et PROCESS-MSG.

Lors d'une requête d'insertion de tâche (requête START), l'EMD vérifie la cohérence des préconditions et des conditions de sortie. La cohérence porte sur les conditions sur états de tâche. Pour les préconditions, vérifier la cohérence consiste à s'assurer de la satisfiabilité des conditions d'états au moment de l'insertion. La satisfiabilité est vérifiée si les conditions ne portent que sur des tâches préalablement insérées, dans des états étant - ou pouvant conduire - aux états

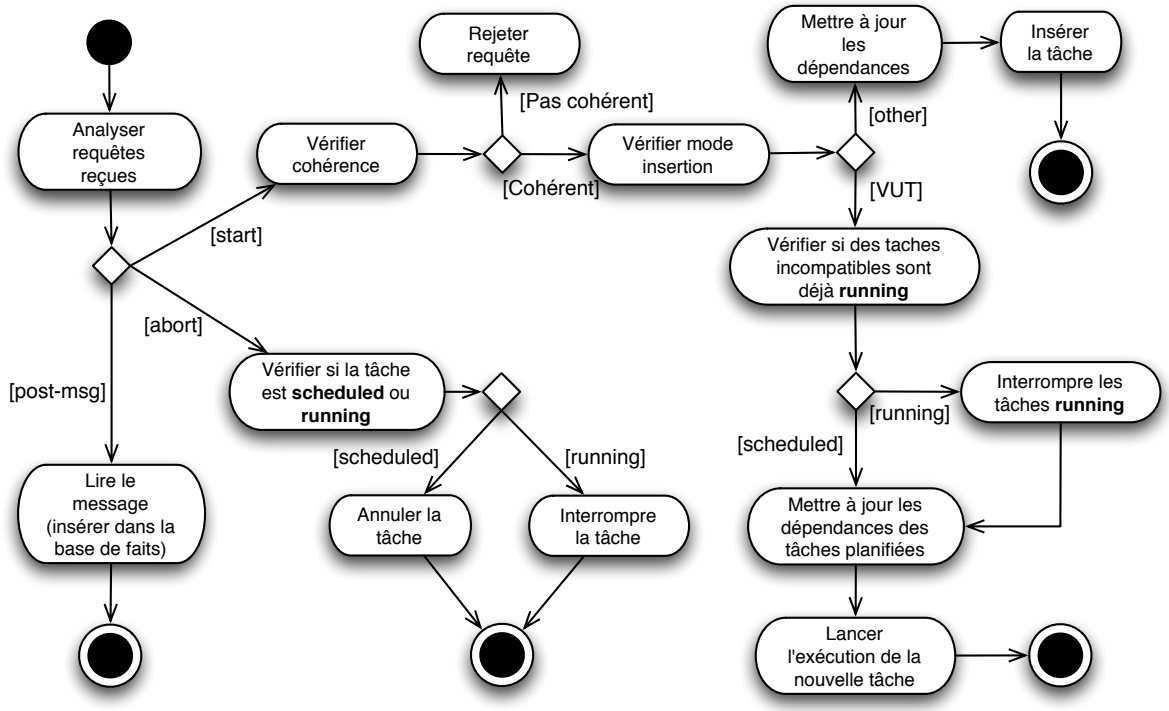


FIG. III.1 – Traitement des requêtes : graphe de transitions

décrits dans les conditions. Dans le cas contraire, les préconditions sont considérées comme non cohérentes, et la nouvelle tâche arrive immédiatement dans l'état `aborted`. Pour les conditions de sortie, vérifier la cohérence consiste au contraire à s'assurer que celles ci ne sont pas satisfaites. Dans le cas contraire, la nouvelle tâche arrive également dans l'état `aborted`. Ensuite, si les conditions de la nouvelle tâche sont cohérentes, celle-ci est insérée dans le plan courant de l'EMD : l'insertion produit des dépendances entre les tâches, selon la modalité d'insertion associée à la requête d'insertion (conformément aux modalités d'insertion décrites ci-avant).

Pour une requête du type ABORT, l'EMD vérifie que la tâche est dans le plan, et procède à l'annulation immédiate de la tâche si celle-ci est dans un état `scheduled`, ou à son interruption si celle-ci est dans un état `started` ou `running`. Dans tous les autres états, cette requête est sans effet.

La dernière requête (PROCESS-MSG) n'est pas directement liée aux tâches : elle permet de demander explicitement à l'EMD d'insérer dans sa base de fait un message, quelle que soit sa nature et sa provenance.

III.2.2 Gestion des événements

Le fonctionnement événementiel de l'EMD est illustré sur la figure III.2. Lorsqu'un nouvel événement se produit au niveau l'EMD (sous la forme d'un "fait" inséré dans la base de faits), les préconditions des tâches en état `scheduled` ainsi que les conditions de sortie des tâches en état `running` sont vérifiées : les préconditions et conditions de sortie sont mises à jour en

conséquence. Il peut alors arriver que les préconditions ou conditions de sortie soient satisfaites : dans ce cas, la tâche concernée est respectivement déclenchée ou interrompue (générant à son tour l'évenement correspondant). Des déclenchements et annulations en cascade peuvent subvenir en conséquence, comme cela est illustré dans la sous-section III.3.

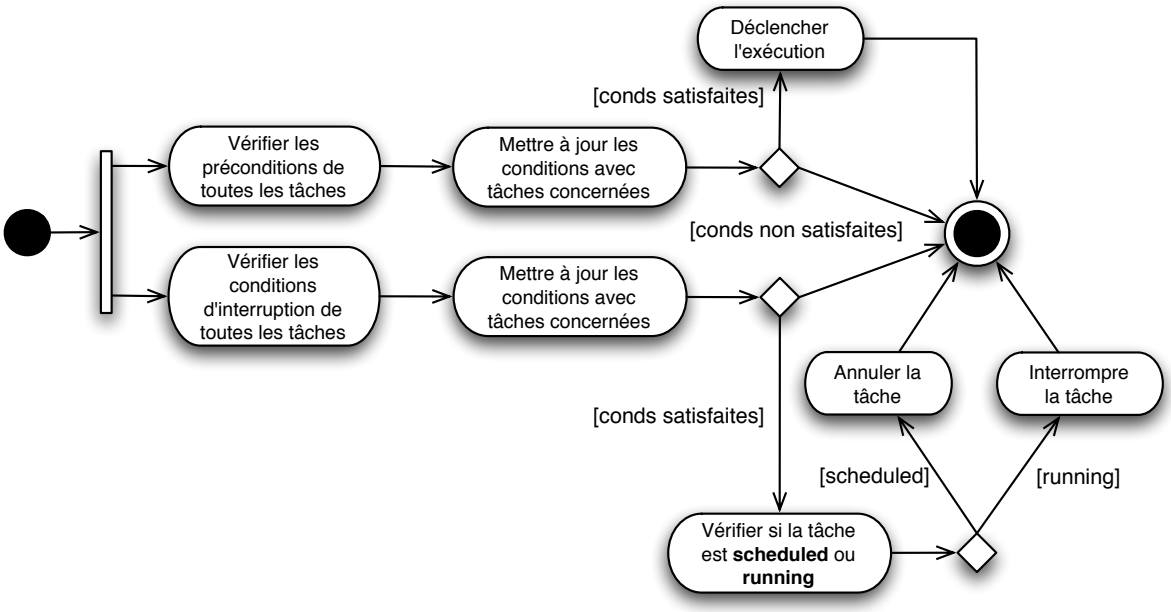


FIG. III.2 – Traitement des événements : graphe de transitions

III.2.3 Exécution de tâche et synchronisation

L'exécution d'une tâche peut être de deux types différents : pour les tâches exécutables par la couche fonctionnelle, l'exécution d'une tâche au niveau de l'EMD consiste à envoyer une requête d'exécution de tâche vers la couche fonctionnelle de l'UAV.

Nous introduisons cependant une tâche particulière : la tâche de synchronisation. Celle-ci est traitée au niveau de l'EMD, en étant absolument invisible du point de vue de la couche fonctionnelle. Son traitement consiste à achever la synchronisation, qui se compose de deux activités :

1. Envoyer un message de synchronisation aux UAVs concernés.
2. Attendre la réception des messages de synchronisation en provenance de chacun des UAVs concernés.

De façon plus formelle, une synchronisation se définit par deux ensembles : un ensemble S (senders) d'UAVs devant transmettre un message de synchronisation, et un ensemble R (Receivers) en attente de réception. Si S = R, alors la synchronisation est un rendez-vous. Différentes combinaisons de S et R permettent de spécifier différents cadres de synchronisations, à "simple" ou "double" sens.

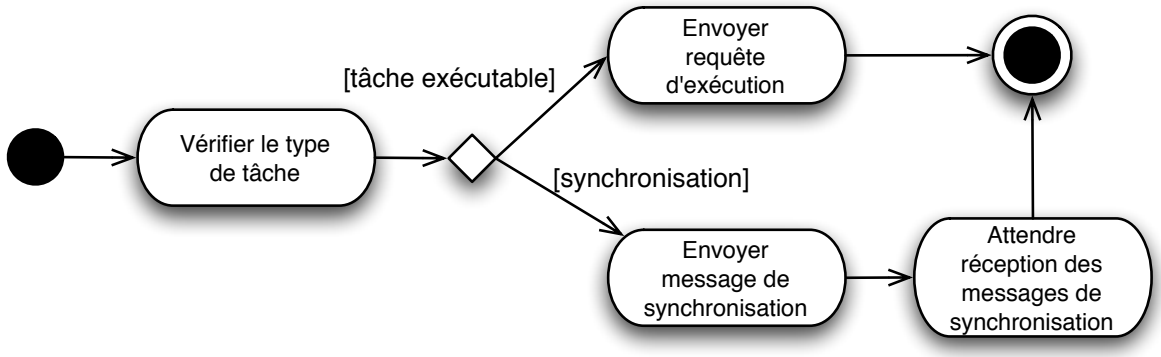


FIG. III.3 – Exécution de tâche : graphe de transitions

Les synchronisations ont une importance capitale dans les missions multi-robots : permettre la réalisation de synchronisations au niveau de l'EMD introduit une modalité de coordination de "bas niveau" : ces synchronisations s'articulent et s'opèrent à un niveau très proche des tâches élémentaires réellement exécutables (couche fonctionnelle).

III.2.4 Le cas particulier des tâches jointes

Bien que nous ne les ayons pas encore définies à ce niveau, une classe de tâches particulières, les tâches jointes (détaillées dans les chapitre IV et V), donnent lieu à un traitement à part : si elles apparaissent de la même manière que les autres tâches dans les données de l'EMD, leurs traitements ne donne pas lieu à des requêtes vers le CPE du robot. Lorsqu'une telle tâche est activée (ces préconditions sont vérifiées), l'EMD le fait savoir à la couche délibérative, via le superviseur de CD. C'est ensuite au niveau de la couche délibérative que les traitements de ces tâches surviennent : l'état de traitement d'une tâche jointe est reçu par l'EMD, et cet état est mis à jour en conséquence. Une tâche jointe peut également être l'objet de conditions de sorties, exactement de la même manière qu'une tâche standard.

Lorsqu'une tâche jointe est en cours de traitement, alors toutes les tâches qui se déroulent en parallèle le font dans le cadre de cette tâche jointe : par conséquent, lorsqu'une tâche jointe est en cours de traitement, tous les status des autres tâches sont remontés au niveau de la couche délibérative, et transmis vers le gestionnaire d'interaction. Ceci est important pour le fonctionnement du GI, comme nous le verrons dans le prochain chapitre.

III.3 Exemple de traitements dans l'EMD

Nous présentons dans ce paragraphe un exemple de création et exécution d'un plan simple d'EMD. Dynamiquement, des tâches sont insérées dans le plan en cours d'exécution et certaines de ces tâches sont annulées, selon les modalités introduites dans le paragraphe précédent.

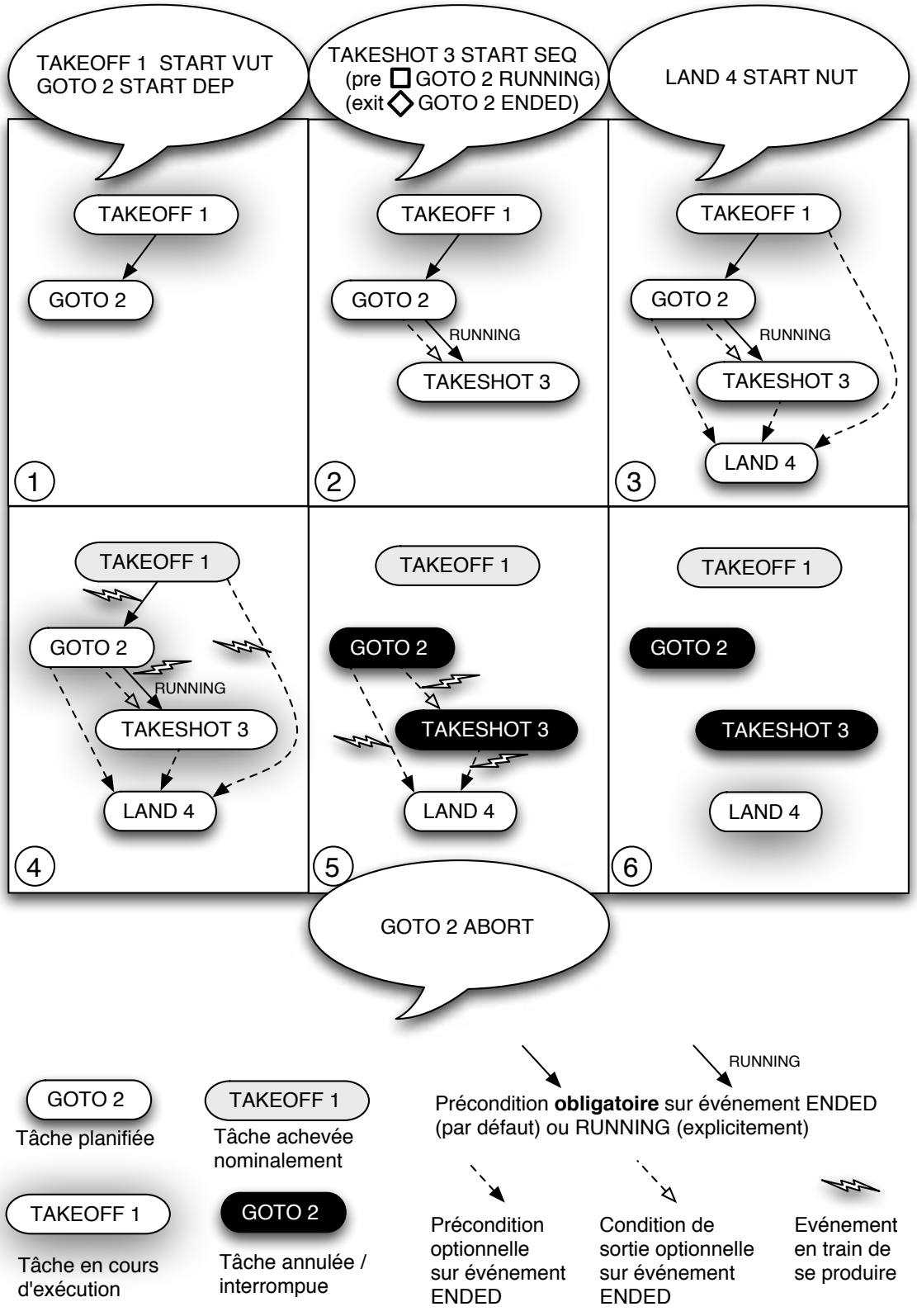


FIG. III.4 – Insertion et annulation de tâches : exemple

Sur la figure III.4, on peut suivre les 6 étapes suivantes :

1. Insertion de la tâche TAKEOFF 1, avec modalité VUT (le plan étant vide jusqu'à présent, la modalité d'insertion n'a pas d'effet particulier : la tâche peut démarrer immédiatement), et insertion de la tâche GOTO 2, avec modalité DEP (crée des dépendances de type "obligatoire" sur les événements `ended` de chacune des tâches déja présente, en l'occurence TAKEOFF 1).
2. Insertion de la tâche TAKESHOT 3, avec la modalité SEQ : les préconditions et conditions de sortie sont données explicitement. En l'occurence, le début de cette tâche est conditionné à l'événement running de GOTO 2 (obligatoire), et la tâche a également une condition de sortie sur l'événement `ended` de GOTO 2 (optionnel).
3. Insertion de la tâche LAND 4, avec la modalité NUT : des dépendances de type préconditions optionnelles sur l'événement `ended` de chacune des autres tâches du plan sont créées.
4. La tâche TAKEOFF 1 vient de se terminer, produisant un événement `ended`. Celui-ci permet de satisfaire les préconditions de la tâche GOTO 2, qui est exécutée. GOTO 2 produit alors l'événement `running`, qui permet de satisfaire les préconditions de TAKESHOT 3. Cette tâche est donc elle aussi exécutée.
5. Un opérateur décide d'interrompre la tâche GOTO 2. GOTO 2 arrive alors dans un état `aborted`, ce qui a pour effet de satisfaire la condition de sortie du TAKESHOT 3 (en effet, la condition de sortie est optionnelle : comme elle n'est plus satisfiable, l'état `ended` de GOTO2 ne pouvant plus être atteint, alors la satisfaction est accordée (voir conditions obligatoires / optionnelles, paragraphe III.1.3). De la même manière, les préconditions portant sur le LAND 4 sont considérées satisfaites, puisque elles sont optionnelles.
6. La tâche LAND 4 peut alors être exécutée, bien que les tâches précédentes aient été interrompues.

L'exemple présenté ici ne contient pas de synchronisation : il illustre uniquement les mécanismes d'insertion et annulation de tâches présentés auparavant, dans le cadre d'un UAV. Dans le chapitre des résultats (VI), nous présentons des applications réelles et en simulation dans lesquelles des synchronisations sont mises en œuvres.

III.4 Conclusion

L'EMD a une position clef dans l'architecture dans la mesure où il interface, en terme de décision, les couches fonctionnelles des différents UAVs avec les "donneurs d'ordre" que sont les NDD et le NDC. L'EMD propose des mécanismes de gestion de plans partiellement ordonnés de tâches, en exploitant différentes modalités d'insertion de tâches ainsi qu'un système de dépendances entre tâches basé sur des préconditions et sur des conditions de sortie.

Le chapitre suivant développe plus en avant l'architecture délibérative proprement dite, c'est à dire la couche délibérative du NDD associé à chaque UAV, et se substituant au NDC dans les configurations les plus avancées de prise de décision distribuée (degrés 4 et 5 d'autonomie décisionnelle).

Chapitre IV

Délibération et coordination dans un système multi-UAV hétérogène

Après avoir présenté les fondements et la vue d'ensemble de notre architecture (chapitre II), et précisé le fonctionnement de l'EMD (chapitre III), nous nous attachons à décrire ici le plus haut niveau de l'architecture, lié à la prise de décision autonome et à la programmation de haut niveau d'activités coopératives.

Ce chapitre est au cœur de notre contribution. Il décrit la couche délibérative (CD) que nous proposons pour les robots d'un système multi-UAV : l'architecture globale, les formalismes de données manipulées, les composantes et leurs interactions, et les mécanismes de traitement sous-jacents. Cette couche délibérative s'inscrit en tant qu'instance de la *prise de décision* dans les NDD, pour les configurations de prise de décision étiquetées en tant que "hauts degrés d'autonomie décisionnelle" (voir le chapitre II).

Nous nous situons plus précisément au degré 4 d'autonomie décisionnelle : il s'agit de doter les UAV du système de capacités autonomes de délibération et de coopération. A ce niveau, nous ne nous considérons pas encore la possibilité de réallouer dynamiquement les tâches entre les UAV du système : il s'agit du degré 5 d'autonomie décisionnelle, qui relève des perspectives.

La figure (fig. IV.1) reprend la présentation des composants de la CD présentée au chapitre II (fig. II.8). Elle est d'un support appréciable pour bien appréhender les formalismes et mécanismes développés dans ce chapitre, et le lecteur pourra s'y référer lorsque nécessaire.

La structure de ce chapitre est la suivante : nous commençons par décrire les formalismes de tâche et de plan manipulés entre les composants de la CD. Nous détaillons ensuite les mécanismes liés au planificateur symbolique (PS) et ses interrelations avec les raffineurs spécialisés (RS). Puis nous introduisons sans le détailler le gestionnaire d'interactions : il s'agit d'un composant essentiel pour la réalisation d'activités coopératives coordonnées dans le système d'UAV, que nous

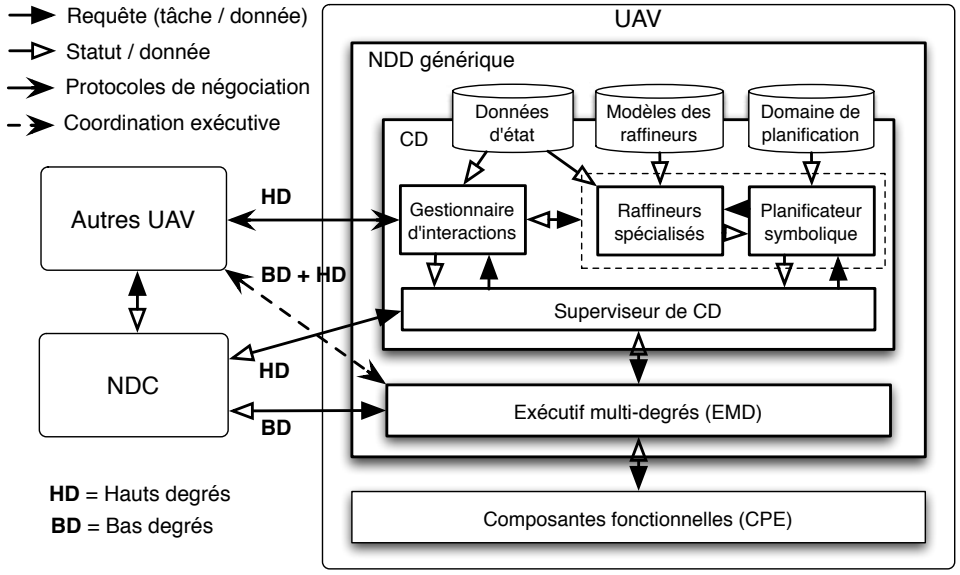
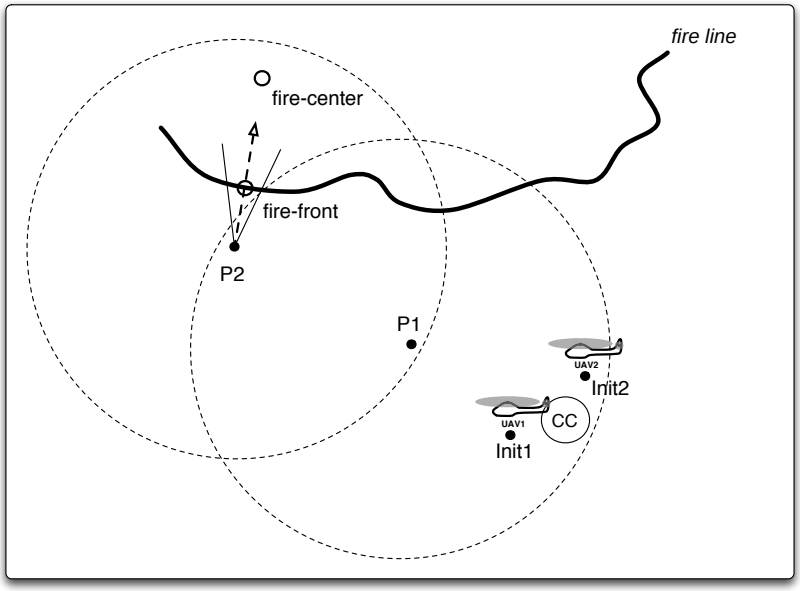


FIG. IV.1 – Architecture décisionnelle : schéma général (rappel)

détaillons dans le chapitre V.
Enfin nous présentons le superviseur de CD (SCD), en précisant la mécanique globale de la couche délibérative à travers les opérations dont ce superviseur est en charge.

Tout au long de ce chapitre (et du chapitre suivant), nous proposons un exemple illustratif de mise en œuvre des formalismes et mécanismes introduits : nous l'appelons *exemple de référence*. Il met en jeu deux UAV : UAV1 et UAV2. Afin de les repérer plus facilement, les développements incrémentaux de l'exemple (en fin de chaque section) sont mis en relief par une ligne verticale sur le coté gauche.



Nous appelons *Mref* la mission globale, dont la spécification et le traitement permettront d'illustrer les formalismes et mécanismes de la couche délibérative. La figure ci-après illustre le contexte de la mission : deux UAV, UAV1 et UAV2, doivent réaliser une supervision d'un feu.

Au moins un UAV doit réaliser des perceptions, et celles-ci doivent être transmises en direct vers le centre de contrôle. Les perceptions doivent être réalisées face au feu, selon la direction (fire-front, fire-center). Comme illustré sur la figure, ce scénario de supervision nécessite un relais de communication. Les points P1 et P2 ne sont pas connus a priori : il s'agit des positions schématiques que les UAV devront atteindre afin de réaliser leur mission. Les grands cercles en pointillé représentent le rayon de communication des UAV depuis les points P1 et P2.

IV.1 Représentations de tâches

Nous définissons dans cette section les différents modèles associés à la représentation de tâches et de plans, nécessaires à la mise en œuvre des capacités délibératives.

IV.1.1 Tâches individuelles et tâches jointes

Avant de poser véritablement les formalismes de tâche employés, il nous faut préciser une distinction fondamentale entre deux catégories de tâches :

– **Les tâches individuelles (TI)** sont des tâches qui peuvent être décomposées / raffinées dans un cadre mono-robot uniquement, sans faire appel aux connaissances et compétences d'autres robots. Nous appelons tâche individuelle élémentaire (TIE) une TI qui ne peut être davantage décomposée, et qui est donc directement exécutable.

– **Les tâches jointes (TJ)** au contraire sont des tâches qui nécessitent, au cours de leurs raffinements, des coordinations non-triviales (qui ne se résument pas à des synchronisations exécutives) subordonnées à des délibérations multi-robots que nous qualifierons de *négociations*. La gestion de ces coordinations non-triviales est la raison d'être du gestionnaire d'interaction (GI).
Pendant son affinement, une TJ donne lieu à un ensemble de sous-tâches pouvant rassembler des TI et des TJ. Nous appelons tâche jointe élémentaire (TJE) une tâche provenant de l'affinement d'une TJ, et qui ne peut être affinée davantage dans le contexte mono-robot : c'est alors le GI qui prend en charge son traitement.

IV.1.2 Tâches élémentaires et plans élémentaires

Pour un UAV donné, un plan élémentaire P est défini de la façon suivante :

Définition IV.1

$\mathsf{P_{el}} = (V, T^i_{el}, T^j_{el}, alloc)$ où :
- V est l'ensemble des UAV du système,
- T^i_{el} est l'ensemble des tâches individuelles élémentaires de cet UAV,
- T^j_{el} est un ensemble de tâches jointes élémentaires, auxquelles cet UAV doit prendre part,
- $alloc : T^j_{el} \rightarrow V^k (k >= 1)$ est une configuration d'allocation pour les taches jointes : à chaque tâche jointe est associé un groupe d'UAV censés réaliser conjointement les parties jointes de cette tâche

et une tâche élémentaire $\mathsf{T_{el}}$ (individuelle ou jointe) est définie ainsi :

Définition IV.2

$\mathsf{T_{el}} = (t_{type},\ t_{id},\ t_{dep},\ I_{start},\ I_{duration},\ timeConst,\ params)$, où :
- t_{type} est le type de tâche,
- t_{id} est un identifiant unique de tâche,
- t_{dep} est un ensemble de dépendances sur les autres tâches, définissant un ordre partiel de cette tâche par rapport à d'autres,
- I_{start} est un intervalle temporel relatif de début de tâche,
- $I_{duration}$ est un intervalle temporel décrivant la durée possible de la tâche,
- $timeConst$ définit des contraintes temporelles absolues sur le début et / ou la fin de la tâche,
- $params$ est un ensemble de paramètres d'exécution pour $\mathsf{T_{el}}$.

Remarque : la définition des tâches élémentaires au sein de la couche délibérative est à rapprocher de la définition de tâche telle que nous la définissons pour l'EMD (voir le chapitre III) : $\mathsf{T} = (t_{type}, t_{id}, \Pi, \Xi,\ params)$
Dans les faits, une tâche élémentaire au niveau de la couche délibérative pourra donner lieu à une tâche au sein de l'EMD, par application d'un opérateur au sein du superviseur de la CD (détaillé dans la section IV.4). Au cours de cette opération, les dépendances t_{dep} et données temporelles I_{start}, $I_{duration}$, $timeConst$ sont traduites en préconditions (ensemble Π) et conditions de sorties (ensemble Ξ).

IV.1.3 Tâches de haut niveau

L'activité d'une couche délibérative est alimentée et influencée par différentes sources d'informations : dans le cas d'une autonomie entière du système (sans intervention humaine directe), ce sont les données perçues par le robot dans l'environnement qui sont potentiellement déclencheurs d'activités délibératives. Cependant, dans la mise en œuvre de nombreux robots, l'humain dispose de moyens privilégiés d'opérations sur les capacités décisionnelles du robot : il s'agit de donner des ordres au robot par un moyen direct, i.e. en amenant l'information (les

ordres) directement aux entités décisionnelles concernées, au niveau de la couche délibérative du robot.

Les deux types d'activation des capacités délibératives ne sont pas antagonistes : un robot peut vraisemblablement prendre l'initiative (et activer par là même des capacités délibératives) suite à la réception et l'analyse de données perçues, tout en étant réceptif à d'éventuelles requêtes de la part d'opérateurs.

Dans le cadre de l'architecture proposée, nous autorisons des moyens directs de communication entre un opérateur et le robot concerné.

Un opérateur du système transmet des requêtes à un haut niveau d'abstraction, que nous appelons *tâche de haut niveau* : il s'agit de tâches n'appartenant pas à l'ensemble des tâches élémentaires. Dans la suite de ce chapitre, c'est ainsi que nous définirons les tâches de haut niveau. Celles-ci sont exprimées de façon unique et monolithique ; leurs raffinements, peuvent conduire à les décomposer en différents ensembles possibles de sous-tâches (elles-mêmes également tâches de haut niveau ou tâches élémentaires).

Compte tenu de l'approche de planification HTN proposée, une tâche de haut niveau doit nécessairement correspondre à une expression d'appel d'une *méthode* dans le planificateur, c'est à dire une modalité de décomposition, quelque part dans la représentation hiérarchique du réseau de tâches. Pour autant, l'ensemble des tâches de haut niveau disponible est extensible, dans la mesure où des méthodes sont complétées en conséquence dans l'HTN, afin d'y répondre.

Formellement, une tâche de haut niveau se définit de la façon suivante :

Définition IV.3

$T = (enonce, parametres, [contraintes\ temporelle\ absolues])$

Les contraintes temporelles sont optionnelles : elles permettent de préciser des contraintes de type *avant* ou *après* sur les instants de début et de fin d'une tâche de haut niveau. Il serait par ailleurs envisageable de considérer des contraintes temporelles relatives entre tâches de haut niveau : nous n'avons cependant pas développé cette extension.

IV.1.4 Missions

Une mission unitaire est, pour un robot donné, un ensemble de tâches de haut niveau, accompagné d'un contexte de planification (état initial). L'expression d'une mission sous formes de tâches de haut niveau est logique et naturelle compte tenu du système de planification à base de HTN (comme nous le proposons dans notre approche). Au contraire, dans des systèmes de planification plus classiques, un problème de planification est plutôt exprimé comme un couple (état initial, état final).

Définition IV.4

$M_{unit} = (\mathsf{T}, E_{initial})$, où :
– T est un ensemble ordonné de tâches de haut niveau à réaliser,
– $E_{initial}$ est un ensemble de faits décrivant l'état initial.

Remarques sur les tâches et missions :
L'ensemble des tâches est ordonné pour une mission unitaire donnée : il s'agit d'activités de haut niveau transmises par un utilisateur du robot, et nous considérons que l'ordre fourni a une certaine importance, c'est à dire qu'il est caractéristique de cette mission unitaire. A supposer qu'un utilisateur souhaite transmettre des tâches de haut niveau à un robot sans les ordonner, il s'agirait alors pour lui de définir plusieurs missions unitaires, et de les transmettre distinctement au robot. Le robot dispose alors de moyens de fusionner les missions unitaires de façon à minimiser le temps total ou le coût, par exemple. Nous discutons de la possibilité de traiter l'insertion dynamique de tâches de haut niveau dans la section IV.4.

Un robot peut avoir un certain nombre de missions unitaires à réaliser : celui-ci doit donc gérer un "pool" de missions unitaires. Nous introduisons une notion de *plan de haut niveau* dans ce but : il s'agit d'un ordre total établi entre toutes les tâches de haut niveau de toutes les missions unitaires à réaliser, pour un robot donné. Nous reviendrons sur cette notion de plan de haut niveau dans la section IV.4, où nous discutons des possibilités de fusion de missions unitaires. Notons cependant qu'il s'agit ici de travaux ouverts. Dans la suite du manuscrit, nous parlerons de plan pour désigner un plan de tâches élémentaires, sauf lorsqu'une ambiguité justifiera de distinguer plan élémentaire et plan de haut niveau.

Compte tenu de la définition d'une mission unitaire, une mission globale sera vue comme un ensemble de missions unitaire. Au degré 4 d'autonomie décisionnelle (celui qui est considéré ici), nous supposons que l'allocation des missions unitaires est réalisée une unique fois, par l'utilisateur du système. Le degré 5 introduit alors la possibilité pour un utilisateur de définir des missions globales et de laisser ensuite les robots décider de l'allocation des missions unitaires sous-jacentes (et éventuellement de remettre en cause la distribution de celles-ci en cours de mission). Le degré 5 fait partie des perspectives, et dans la suite, une mission désignera une mission unitaire, à moins qu'un risque d'ambiguité ne justifie la distinction explicite.

Exemple de référence - partie 1 : définition des tâches

Comme nous l'avons précisé dans cette section, une mission unitaire s'exprime comme un ensemble de tâches de haut niveau. Les missions unitaires *Mref-1* et *Mref-2* concernent respectivement UAV1 et UAV2.
Selon le formalisme défini précédement pour les tâches de haut niveau, nous définissons ces missions comme suit.

Mref-1 regroupe les tâches de haut niveau suivantes :
– T1 : (takeoff, (), (after-9 :00 :00))
– T2 : (monitoring-with-relay, (fire-front, fire-center, (UAV1, UAV2), 900))
– T3 : (land, (), (before-9 :40 :00))

Mref-2 regroupe les tâches de haut niveau suivantes :
– T4 : (takeoff, (), (after-9 :00 :00))
– T5 : (monitoring-with-relay, (fire-front, fire-center, (UAV1, UAV2), 900))
– T6 : (land, (), (before-9 :40 :00))

Nous ajoutons que les états initiaux respectifs de *Mref-1* et *Mref-2* spécifient que les UAV

sont au sol, respectivement aux positions Init1 et Init2.

Les contraintes temporelles exprimées dans les tâches représentent des souhaits d'un utilisateur du système, à savoir que la mission ne commence pas avant 9h00, et ne se termine pas après 9h40. On remarquera par ailleurs que les tâches T2 et T5 sont des tâches jointes : leur développement va conduire à la génération de TJE, ne pouvant être traitée que dans un contexte multi-robots, au sein du GI. Les paramètres de T2 et T5 sont respectivement deux points (référence pour la perception), une liste d'UAV impliqués, et la durée pendant laquelle la tâche coopérative doit avoir lieu.

IV.2 Le couple planificateur symbolique / raffineurs...

...ou "exploiter des modèles géométriques dans un planificateur symbolique".

Nous exploitons différents mécanismes subsidiaires afin de rendre exploitable le planificateur HTN dans un contexte opérationnel. Nous précisons en particulier comment permettre l'interaction du planificateur avec les raffineurs.

Une introduction aux principes de base du planificateur HTN Shop2 est fournie en annexe A : des exemples y sont proposés pour illustrer le système de planification HTN, ainsi que l'introduction du mécanisme des *timelines*, permettant de considérer les aspects temporels et de générer des plans partiellement ordonnés.

Nous nous contenterons ici de rappeler ici qu'un domaine de planification HTN est défini comme un ensemble de méthodes, permettant de décomposer une tâche donnée en sous-tâches, et un ensemble d'opérateurs, qui sont les tâches élémentaires (ou les "feuilles", dans la hiérarchie HTN).

IV.2.1 Propagation d'identifants de tâche de haut niveau

Afin de conserver une connaissance sur l'origine des tâches élémentaires, nous laissons la possibilité de propager en cours de planification les identifiants respectif des tâches de haut niveau d'origine, composant la mission, dans la décomposition en tâches plus élémentaires.

Ainsi, un ensemble de tâches de haut niveau d'une mission M sera considérée comme achevée si les tâches élémentaires issues (suite à décomposition) de ces tâches de haut niveau sont elles-mêmes correctement exécutées.

Nous sommes donc en mesure de faire état de succès ou d'échecs de sous-parties d'une mission : en cas d'échec d'une tâche élémentaire, la tâche de plus haut niveau dont la décomposition a engendré cette tâche élémentaire, est elle-même considérée en état d'échec. Ce principe est exploité dans des opérateurs de manipulation de plan, au niveau du superviseur de la couche délibérative (voir la section IV.4.2).

Des exemples de plans dans lesquel ce mécanisme de propagation est employé sont illustrés dans le chapitre VI, au niveau des résultats en simulation.

IV.2.2 Utilisation des raffineurs

En plus de simples conditions sur des faits, Shop2 introduit également la possibilité, dans les préconditions des méthodes et opérateurs, d'assigner explicitement des valeurs aux variables Shop2. Ce mécanisme permet en particulier, lors d'appel à des opérateurs Shop2, d'instancier une variable en appelant une fonction ou routine extérieure à Shop2. Cette instanciation, en tant que précondition, est un préalable à l'application de cet opérateur dans le plan en cours de construction. Dans notre système, ce sont les raffineurs spécialisés (RS) qui vont être utilisés en tant que routines extérieures.

Avant de proposer un exemple d'interaction entre le PS et les RS, nous décrivons rapidement les caractéristiques et algorithmes des RS que nous avons utilisés (plus de détails sur les raffineurs spécialisés sont disponibles dans [Hattenberger 04]).

Modèles et formalismes

Les principaux modèles utilisés par les raffineurs spécialisés sont les suivants :

– **modèle d'environnement :** regroupe des informations liées au sol et des informations liées à l'espace aérien. Le modèle de sol exploite une grille régulière, pouvant contenir des informations liées au domaine (dans le cadre de COMETS par exemple, des informations sur le caractère inflammable ou encore sur l'état de cartographie sont ainsi disponibles). Le modèle aérien est une discrétisation de l'espace sous forme de voxels (volumic elements) réguliers. Dans ces cellules tridimensionnelles sont exhibées des informations telles que l'occupation ou le coût de franchissement.

– **modèle d'UAV :** celui-ci décrit les capacités de vol (plages de vitesses, rayon de giration, possibilité de vol stationnaire...), et fournit également des informations telles que l'autonomie énergétique.

– **modèle de perception** : le modèle de perception définit les caractéristiques des capteurs, telles que la surface perçue (ouverture), la qualité de perception par rapport à la distance de ce qui est perçu, le coût d'utilisation d'un capteur donné. Ces données permettent d'inférer les positionnements les plus adéquats pour réaliser des actions de perception satisfaisantes (on parlera de planification de perceptions).

– **modèle de communication :** un modèle relativement simple de communication est utilisé. Il se base sur un principe de visibilité, en considérant la puissance d'émission et la sensibilité de réception par rapport à la distance entre entités communicantes.

Mise en œuvre algorithmique

Sur la base des modèles introduits précédemment, les raffineurs proposent un ensemble de fonctions permettant d'une part d'évaluer les faisabilité, coûts et temps de réalisation de

tâches géométriques et/ou de perception, et d'autre part de raffiner et décomposer en tâches plus élémentaires (exécutables par les UAV) les tâches de haut niveau transmises aux raffineurs. Voici quelques uns des algorithmes implémentés dans cette optique :

– **planification de perceptions :** étant donné un lieu à percevoir dans l'environnement, les raffineurs permettent de calculer les meilleures positions de perception compte tenu du modèle de l'UAV et de celui du capteur à employer. Les occultations (obstacles / relief du terrain) ainsi que les contraintes extérieures (zones interdites de survol) sont prises en compte dans le calcul. Parmi les configurations possibles, les meilleures sont évaluées en terme d'utilité de perception.

– **planification de chemin et "TSP" :** La planification de chemin entre un point P et un point P' est réalisée sur la base d'un simple A* dans les voxels de l'environnement. Le TSP (Traveller Salesman Problem, ou problème du voyageur du commerce) consiste à déterminer la tournée la moins coûteuse dans un ensemble de lieux. Une approche stochastique est adoptée, afin de déterminer une solution approchée en un temps raisonable. De façon itérative, différentes configurations sont testées, avec des évolutions de type permutations dans l'ordre des points de passage. Après un temps arbitraire, le calcul est interrompu, et la meilleure solution courante est retournée.

– **cartographie :** (ou "couverture") il s'agit de déterminer un chemin de passage qui, compte tenu des contraintes de déplacement de l'UAV, permette de couvrir intégralement une zone donnée, tout en minimisant un critère comme le temps de survol ou le coût occasionné. La solution approchée adoptée tente de minimiser les virages nécessaires [Mazza 04]. Les virages sont considérés comme critiques pour plusieurs raisons : en premier lieu un UAV est susceptible de ralentir tandis qu'il opère un virage. Ensuite, il est généralement plus difficile (et donc coûteux) d'asservir un UAV le long d'une courbe que le long d'une ligne droite, et il en est de même pour le contrôle de la zone perçue durant cette opération. Nous reparlons d'une mise en œuvre de la cartographie en vue d'une application multi-UAV, dans le chapitre VI.

– **détection :** il s'agit ici davantage d'une tâche de surveillance que de couverture. Cette activité nécessite, pour un UAV, de survoler une zone en réalisant des perceptions pendant un temps donné, et en minimisant la durée entre deux survols d'une même cellule au sol. Les cellules de la zone contiennent des informations relatives à un risque d'occurence d'un événement qui fait l'objet de la détection : dans COMETS, il s'agit du départ d'un incendie. Le risque est variable selon les cellules, et la fréquence de survol doit logiquement être proportionnelle au risque.

Ce problème se prète bien à une solution basée sur les champs de potentiels : chaque cellule voit son potentiel exprimé sous la forme :

$$P = e^{-r \cdot \Delta T}, \tag{IV.1}$$

où $r \in [0,1]$ est le facteur de risque et ΔT est le temps écoulé depuis le survol précédent. A chaque pas de temps, ΔT est incrémenté proportionnellement à un facteur k :

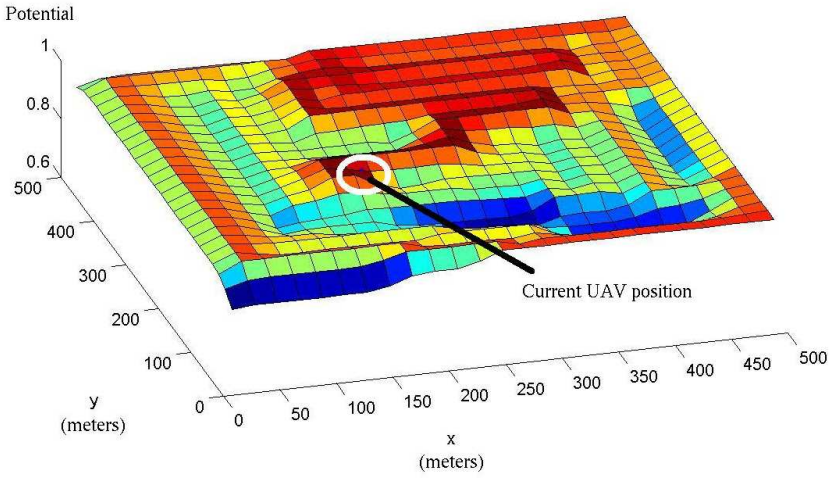


FIG. IV.2 – Détection avec un champ de potentiel, en simulation

$$k = 1 - e^{-\left(\frac{(x-x_u)^2+(y-y_u)^2}{\sigma^2}\right)}, \tag{IV.2}$$

où (x,y) est la position de la cellule considérée, (x_u,y_u) la position de l'UAV et σ un paramètre correspondant à l'ouverture du capteur.

Le déplacement de l'UAV est réalisé selon la direction du plus fort gradient de potentiel. La figure IV.2 représente l'état du potentiel au cours d'une simulation.

– **contraindre les traitements de base :** il est par ailleurs possible de contraindre les traitements présentés précédemment, en terme de maintien de communication durant l'opération (en particulier pour la planification de chemin ou de perceptions), ou en terme de position initiale, qui peut avoir une influence sur toute la suite du traitement (le coût de déplacement, la consommation d'énergie pour se rendre sur le lieu du traitement principal, doivent être pris en considération).

Exemple d'emploi des raffineurs

La figure IV.3 illustre l'utilisation d'un appel à une routine extérieure (en l'occurence les raffineurs) afin d'évaluer le coûts et la durée d'une tâche de déplacement.

Sur la figure IV.3, la méthode general-gotoxyz (en haut) fait appel à un opérateur de calcul, reposant sur un appel aux raffineurs (au milieu), et sur l'opérateur gotoxyz proprement dit (en bas). Le principe de l'opérateur de calcul est d'insérer dans les faits, au cours de la planification, le résultat du calcul réalisé par les raffineurs, puis immédiatement, d'exploiter ce résultat dans l'opérateur proprement dit. Ainsi, au niveau des lignes (1), (2), (3) et (4), on récupère toutes les informations nécessaires produites par les raffineurs (durée, points intermédiaires, coût...) et stockées dans la base de faits, dans la variable "result".

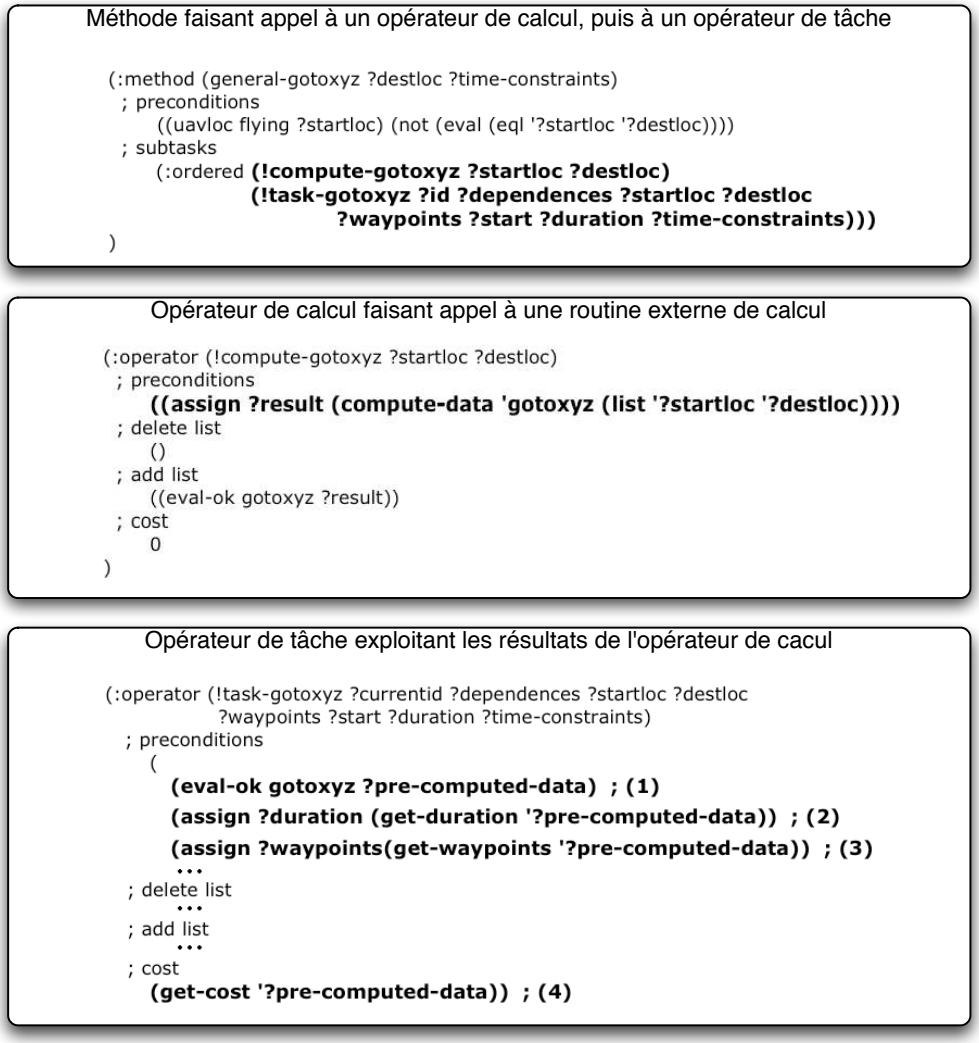


FIG. IV.3 – Exemple simple de mise en œuvre des raffineurs depuis le planificateur

Dans la suite de notre exemple de référence, la mission proposée est traitée auprès du couple planificateur / raffineurs. Un plan de tâches élémentaires est obtenu, pour chacun des UAV.

Exemple de référence - partie 2 : raffinement des tâches

Une fois transmises aux UAV, les missions *Mref-1* et *Mref-2* sont traitées par leurs superviseur de couche délibérative (entité décrite plus loin, dans la section IV.4). Au niveau de chaque UAV, la mission est transmise au couple planificateur / raffineurs pour générer un plan de tâches élémentaires (jointes et / ou individuelles).

Le formalisme est celui introduit pour les tâches élémentaires, dans la section IV.1.2.

Pour *Mref-1* :

1. T1 : la décomposition est assez triviale. Le résultat est une tâche élémentaire de la forme : t1.1 = (take-off, 11, (), 9 :00 :00, 30, ()),
2. T2 : cette tâche est une tâche jointe : elle est donnée pour un ensemble d'UAV (UAV1, UAV2). Elle est partiellement raffinée, donnant le résultat suivant :
   t2.1 = (tje-superviser-et-relayer, 21, ((11 ENDED)), 9 :00 :30, 900, (fire-front, fire-center, control-center, (UAV1, UAV2), 900))
3. T3 : la décomposition est relativement triviale : il s'agit pour l'UAV de procéder à l'atterissage :
   t3.1 = (land, 31, ((21 ENDED)), 9 :15 :30, 45, ())

Pour *Mref-2* :

1. T4 : la décomposition est assez triviale. Le résultat est une tâche élémentaire de la forme : t4.1 = (take-off, 41, (), 9 :00 :00, 30, ()),
2. T5 : cette tâche est une tâche jointe : elle est donnée pour un ensemble d'UAV (UAV1, UAV2). Elle est partiellement raffinée, donnant le résultat suivant :
   t5.1 = (tje-superviser-et-relayer, 51, ((41 ENDED)), 9 :00 :30, 900, (fire-front, fire-center, control-center, (UAV1, UAV2), 900))
3. T6 : la décomposition est relativement triviale : il s'agit pour l'UAV de procéder à l'atterissage :
   t6.1 = (land, 71, ((51 ENDED)), 9 :15 :30, 45, ())

Plusieurs remarques peuvent être faites sur ces raffinements :

– en premier lieu, dans la mesure où il s'agit d'un exemple simple, les tâches de haut niveau n'ont respectiement donné lieu qu'à une seule tâche élémentaire. En cours de planification, les méthodes de haut niveau ont cependant fait appel aux raffineurs, comme présenté auparavant dans cette section : c'est de cette façon que les informations sur les durées (respectivement 30' et 45') ont été obtenues.

– Ensuite, les dépendances entre tâches proviennent de l'utilisation des timelines, telles que présentées en annexe : chacune des tâches dépend de la fin de la tâche précédente, dans la mesure où toutes ces tâches mobilisent la ressource "déplacement" de l'UAV.

– Enfin, peu d'information est disponible sur la tâche jointe en cours de planification, la destination correspondante n'est pas connue. C'est seulement au moment du traitement joint de cette tâche dans le GI que celle-ci sera détaillée (chapitre suivant).

Le plan élémentaire pour UAV1 est donc :

P^1_{el} = ((UAV1, UAV2), (t1.1, t3.1), (t2.1), (t2.1 :(UAV1, UAV2)))

Et de même, le plan élémentaire pour UAV2 est :

P^2_{el} = ((UAV1, UAV2), (t4.1, t6.1), (t5.1), (t5.1 :(UAV1, UAV2)))

Ces plans sont prêt à être transmis aux EMD respectifs des UAV pour exécution. Les tâches jointes élémentaires t2.1 et t5.1 ne peuvent être raffinées d'avantage à ce niveau : elles seront transmises au GI pour être traitées. C'est le superviseur de CD qui manipule le plan pour l'exécuter, ou pour transmettre des requêtes de traitement de tâches jointes au GI.

IV.3 Gérer les interactions

La gestion des interactions entre UAV (coordination pour la réalisation de tâches coopératives) repose sur le gestionnaire d'interactions. Cette entité est un sous-système qui emploie des modèles d'interactions à base de rôles pour raffiner et mettre en œuvre conjointement, dans un contexte multi-robots, des activités coopératives.

Nous ne détaillons par le gestionnaire d'interactions dans ce chapitre, car les modèles, formalismes et mécanismes sous-jacents justifient de le traiter dans un chapitre à part entière (prochain chapitre). Nous le présentons donc ici comme une boîte noire, dont nous donnons les entrées, sorties et effets au sein de la couche délibérative.

Le gestionnaire d'interaction reçoit en entrée des requêtes portant sur des tâches jointes élémentaires. Il s'agit de requêtes demandant le prétraitement, le traitement proprement dit où l'interruption du traitement de tâches jointes :

– **prétraitement d'une TJE :** lorsqu'une requête de prétraitement de tâche jointe est transmise au GI, celui-ci prépare son exécution prochaine. Il s'agit d'instancier la représentation de la TJE au sein du GI (cette représentation prend le nom de *modèle d'interactions*), compte tenu du contexte courant, et de négocier d'une part l'allocation initiale des rôles de ce modèle d'interaction, et d'autre part l'intervalle temporel dans lequel la TJE sera réalisée.

– **traitement d'une TJE :** lorsqu'une requête de traitement d'une tâche jointe (préalablement prétraitée) est transmise au GI, l'activité coopérative sous-jacente s'opère par l'intermédiaire du gestionnaire d'interactions. En sortie de celui-ci, des tâches de haut niveau coordonnées sont produites : il reste alors à les raffiner (avec le couple PS / RS) avant d'insérer les tâches individuelles élémentaires dans le plan de l'EMD pour exécution.

– **annulation ou interruption d'une TJE :** l'annulation concerne une TJE prétraitée, mais pas encore en cours de traitement. L'interruption au contraire concerne l'arrêt du traitement en cours d'une TJE. Lors de la réception de cette requête, le GI procède à l'arrêt immédiat des opérations, et retourne un statut d'échec pour cette tâche jointe.

Le traitement d'une TJE dans le gestionnaire d'interaction produit des statuts de traitement de la TJE qui sont comparables aux statuts d'exécution d'une tâche dans l'EMD :

– `scheduled` : TJE prétraitée dans le GI, prête à être traitée.
– `running` : TJE en cours de traitement dans le GI.
– `aborting` : TJE en cours d'annulation / interruption dans le GI.
– `aborted` : TJE annulée / interrompue.
– `ended` : TJE nominalement traitée.

Pendant la durée du traitement d'une TJE dans le GI, des tâches de haut niveau, i.e. qui nécessitent des raffinements avant exécution, sont produites. Le superviseur de CD opère leurs raffinements dans le couple planificateur symbolique / raffineurs spécialisés, avant de les transmettre à l'EMD en vue de leur exécutions. Les tâches de haut niveau retournées par le GI correspondent à des buts, au sens du planificateur HTN : accompagnées d'états initiaux de planification (déduits en partie de l'état courant de l'UAV), elles définissent des *problèmes* au sens de la planification HTN. La résolution de ces problèmes permet de déterminer des plans de tâches élémentaires correspondant. Un tel plan est alors directement traitable par l'EMD dans le contexte courant.

Exemple de référence - partie 3 : traitement des tâches jointes

Nous décrivons ici en terme d'entrées et sorties les tâches jointes élémentaires de notre exemple, à savoir t2.1 et t5.1, respectivement pour UAV1 et UAV2.

Lorsqu'une requête pour le traitement d'une tâche jointe élémentaire est reçue par le GI, celui-ci, prépare le traitement proprement dit. Du point de vue de la CD, cette étape est assez transparente : il s'agit d'une opération du superviseur de CD (voir section suivante) qui n'a généralement pas d'effet immédiat (sauf problème de chargement de la TJE dans le GI). Nous noterons que cette étape concerne essentiellement l'attribution de rôles dans la partie jointe de l'interaction et la négociation de créneau temporel pour la réalisation de la tâche jointe élémentaire.

Sans les détailler ici, le premier rôle concernera la perception proprement dite, et l'autre rôle concernera le relais de communication. A l'issue de ce prétraitement, une allocation initiale des rôles existe au sein du GI.

Par la suite, lorsqu'au sein de l'EMD la TJE est déclenchée (i.e. ses préconditions sont satisfaites), le superviseur de CD transmet l'information au GI, qui traite la coordination des activités jointes, afin de réaliser des actions cohérentes avec celles des autres UAV. En sortie, les tâches de haut niveau suivantes sont produites :

Pour UAV1 :

- T7 : (general-goto, (P2))
- T8 : (synchro, ((S = (UAV1,UAV2), (R = (UAV1,UAV2)))))
- T9 : (takeshot-while-waiting, 900)
- T10 : (synchro, ((S = (UAV1,UAV2), (R = (UAV1,UAV2)))))
- T11 : (general-goto, (Init1))

Pour UAV2 :

- T12 : (general-goto, (P1))
- T13 : (wait-until-synchro, ((S = (UAV1,UAV2), (R = (UAV1,UAV2)))))
- T14 : (wait-until-synchro, ((S = (UAV1,UAV2), (R = (UAV1,UAV2)))))
- T15 : (general-goto, (Init2))

La résolution individuelle successive de ces tâches donnera lieu à des tâches élémentaires que les UAV exécuteront dans les cadres respectifs des TJE t2.1 et t.5.1. La façon dont ces tâches de haut niveau sont produites est détaillé dans le chapitre suivant.

Les tâches fournies sont par ailleurs susceptibles d'être révisées : en particulier lorsque des contingences surviennent en cours d'exécution (nous illustrerons ceci dans le chapitre suivant).

En exécutant ces tâches élémentaires, les UAV doivent dans un déroulement idéal se rendre respectivement sur le lieu de perception et le lieu de relais, puis réaliser la perception relayée pendant 900 secondes, puis terminer de façon synchronisée la perception relayée, puis revenir à la verticale de leurs points d'atterrissage respectifs. Tout ceci est réalisé dans le cadre la tâche jointe traitée.

Nous allons cependant introduire des contingences, afin de montrer comment le système peut évoluer lorsque nécessaire : c'est dans le cadre du gestionnaire d'interactions que ces contingences seront considérées (chapitre suivant).

L'exploitation du GI dans la couche délibérative est assujettie aux opérations du superviseur de CD : la section suivante introduit les différents opérateurs de manipulation de plan au sein de la CD, que le superviseur de CD met en œuvre.

IV.4 Superviser la délibération

Maintenant que les données et traitements exploitables ont été définis dans les précédentes sections, nous introduisons avec le superviseur de la couche délibérative (SCD) les moyens mis en œuvre pour traiter les requêtes de haut niveau envoyées depuis un NDC, pour effectuer des opérations sur les plans et pour assurer la cohérence des flux de données asynchrones entre l'EMD et les différentes composantes de la CD.

IV.4.1 A propos de la représentation de plan

En premier lieu, notons que le superviseur de CD exploite et repose sur une représentation de plan (définie dans la section IV.1) que l'on peut assimiler à un STNU (Simple Temporal Network with Uncertainties : [Vidal 97]).

Un STN [Schwalb 97] est un graphe orienté où les nœuds correspondent à des événements ou des actions ponctuelles (on parlera de points temporels, ou *timepoints* en anglais), et les arêtes indiquent les transitions autorisées, en précisant, sous la forme d'intervalles [min, max], les durées séparant les événements de part et d'autres des arêtes.

Formellement, un STN peut être décrit comme un 4-tuple $< N, E, l, u >$, où N est un ensemble de nœuds, E est un ensemble d'arêtes, et $l : E \rightarrow \Re \cup \{+\infty\}$ et $u : E \rightarrow \Re \cup \{-\infty\}$ sont des fonctions qui mettent en correspondance les arêtes avec des nombres réels, qui sont les bornes inférieures et supérieures de l'intervalle des durées possibles.

Un STNU introduit une distinction des arêtes, selon deux catégories : *arêtes contingentes*, et *arêtes requises*. La seconde catégorie est celle usuellement considérée dans un STN simple. Une telle arête est considérée comme contrôlable lors de l'application du STN (la date d'occurence d'un point temporel est contrôlée, dans la limite des bornes de l'intervalle de durée). La première catégorie représente des arêtes non-contrôlable : la seule hypothèse est que l'événement se produira dans les bornes de l'intervalle de durée, mais on ne peut pas savoir quand (contingence). La représentation sous forme de STNU est bien adaptée à une représentation de plan devant être exécuté par un robot, dans la mesure où l'on ne contrôle pas toujours les points temporels associés aux tâches : ainsi, si l'on contrôle généralement bien le lancement d'une tâche, la fin de son exécution peut-être beaucoup plus difficile à prévoir. Dans notre représentation, les arêtes menant aux événements de fin de tâche sont nécessairement contingentes.

La définition d'un STNU est proche de celle d'un STN : il s'agit d'un 5-tuple $< N, E, l, u, C >$, où C est le sous-ensemble des arêtes contingentes. Par ailleurs, chaque arête contingente doit sa-

tisfaire : $0 < l(e) < l(u) < \infty$.

Les plans générés par le planificateur symbolique sont donc représentés dans le superviseur de CD en tant que STNU. Ils sont nécessairement consistants, c'est à dire que le mode de production des plans dans le planificateur symbolique garantit que le plan retourné est suceptible d'être exécuté, et donc qu'il n'y a pas d'incohérence temporelle. C'est le système de planification de Shop2 et l'usage des timelines (voir A) qui permet de s'en assurer. En effet, le planificateur planifie dans l'ordre où les tâches seront exécutées. A chaque opérateur appliqué, les timelines sont le cas échéant modifiées pour tenir compte des usages des ressources entre tâches : les dates possibles de début et de fin de tâche sont donc propagées à la volée en cours de planification, sans que des incohérences soient possibles.

La construction du plan dans le planificateur conduit à l'obtention de plans qui, représentés sous forme de STNU, nous semblent répondre à la propriété de *controlabilité dynamique* (nous ne l'avons pas formellement démontré). Cette notion [Vidal 01] caractérise des STNU pour lesquels des stratégies d'exécution viables existent quelque soient, au moment de l'exécution, les dates obtenues (dans les intervalles donnés) lors des transitions sur les arêtes contingentes. Autrement dit, en supposant que les intervalles de durées précisés sur les arêtes du STNU sont valides, le plan ne peut pas échouer.

En réalité, dans le cadre d'une application robotique, il est pratiquement impossible de garantir la validité absolue de ces intervalles de temps de façon réaliste : il y a toujours un risque que la durée de transition entre deux événements du STNU sortent des bornes spécifiées. Une telle situation conduit nécessairement à réviser, d'une manière ou d'une autre, le plan courant devenu inconsistant (plus aucune garantie n'existe sur la possibilité d'achever le plan).

Dans cette optique, la sous-section suivante définit les modes opératoires du SCD, et en particulier l'opération de réparation de plan, pour lequelle nous évoquons des stratégies possibles de traitement (et de recouvrement) de plan en cas de contingence fatale.

IV.4.2 Opérations de plan

Le superviseur de CD fonctionne en appliquant des opérations pour manipuler et traiter convenablement les plans (élémentaire et de haut niveau) dans la couche délibérative, de la réception des requêtes de missions (à un haut niveau) jusqu'à la prise en charge des problèmes survenant à l'exécution, le cas échéant. Nous définissons pour cela un ensemble d'opérateurs de manipulation de plan, sur lesquels repose la gestion des plans dans la couche délibérative.

Ces opérateurs sont partiellement développés : nous présentons donc, dans certain cas, notre vision des opérateurs tels qu'ils devraient être développés.

Voici les opérateurs proposés :

– **SCD-0 : Traitement d'une requête utilisateur :** il s'agit de traiter convenablement une requête, qui peut être essentiellement de deux types :

. Ajout de tâches de haut niveau, à travers la donnée d'une nouvelle mission unitaire. Ces nouvelles tâches sont considérées par rapport au plan courant : si le plan de haut niveau courant est vide (aucune mission unitaire pour cet UAV), l'insertion est assez triviale. Si dans un cas plus général le plan courant comprend un certain nombre de tâches de haut niveau (missions

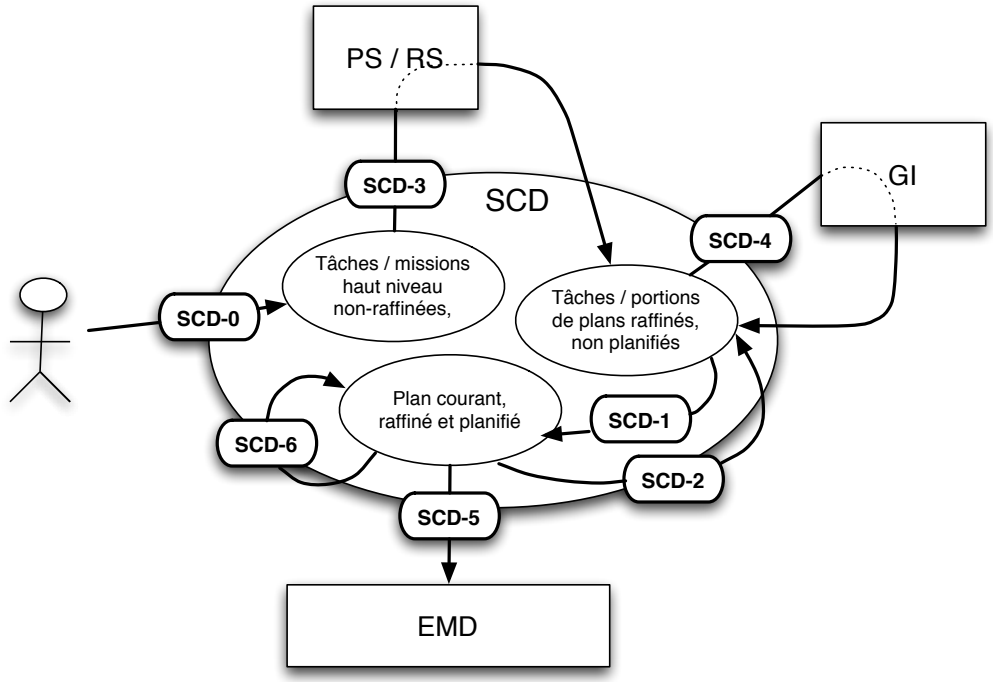


FIG. IV.4 – Entrées et sorties des opérateurs du SCD

unitaires déja prévues), différentes politiques sont envisageables. Un opérateur d'insertion de tâches de haut niveau (SCD-1) est défini à cet effet ci-après.

. Annulation / interruption de tâche. Il s'agit soit d'annuler des tâches de haut niveau planifiées en attente d'exécution, soit d'interrompre des tâches élémentaires en cours d'exécution. Un opérateur d'annulation / interruption de tâches (SCD-2) est défini dans cette optique.

– **SCD-1 : Ajout de tâches de haut niveau :** ajouter une mission unitaire à l'ensemble courant des missions unitaires d'un UAV consiste d'abord à raffiner cette nouvelle mission pour évaluer les coûts, étendues et contraintes temporelles, puis à évaluer les options possibles d'insertion des tâches de la nouvelle mission parmi les tâches des missions déja planifiées.
Si la nouvelle mission ne fait pas état de contraintes temporelles dans ses tâches de haut niveau, une option simple consiste à imposer, dans le plan de haut niveau du robot, une précédence des tâches de haut niveau des missions déja planifiées sur les tâches de haut niveau de cette nouvelle mission.
Mais il est également envisageable de fusionner plus efficacement (coût, temps...) les nouvelles tâches de haut niveau avec celles des missions déja planifiées. Nous ne considérons pas la fusion des tâches élémentaires : en effet, la fusion au niveau des plans élémentaires est loin d'être triviale, compte tenu des informations disponibles sur les tâches et les plans au niveau du SCD : les plans disponibles n'expriment pas de lien de causalité, et ne conservent pas l'historique de la façon dont ils ont été obtenus (mis à part la trace de la tâche de haut niveau d'origine).

En considérant, pour le plan de haut niveau, l'ordonnancement des tâches de haut niveau, plusieurs pistes sont envisageables :
- Vérifier et exploiter la compatibilité temporelle des tâches : il s'agit de savoir *où* les tâches respectives sont susceptibles d'être insérées, abstraction faite de leur nature et effet, dans le plan de haut niveau courant.
- Vérifier et exploiter la compatibilité spatiale : en analysant les déplacements possiblement mis en œuvre par les tâches de haut niveau, il serait possible d'évaluer, parmi les tâches de haut niveau déjà planifiées, lesquelles sont susceptibles de s'exécuter dans le voisinage d'une nouvelle tâche en cours d'insertion. Une façon de procéder peut consister à considérer toutes les tâches de haut niveau du plan de haut niveau courant, plus les nouvelles tâches à insérer, du point de vue des lieux respectifs prévus : en faisant appel à l'algorithme de TSP des raffineurs spécialisés (voir la section IV.2.2), il est alors envisageable d'ordonner les tâches de haut niveau d'une façon qui tendra à minimiser les déplacements.

  En effectuant une replanification sur le plan de haut niveau courant, avec l'ensemble de ces tâches de haut niveau, ordonnées selon le résultat du TSP, on peut obtenir un plan élémentaire cohérent spatialement (évitant des déplacements inutiles).
  Dans cette optique, pour les tâches de haut niveau donnant lieu à de multiples déplacements au dessus d'une zone, soit un point arbitraire est fourni, soit un couple de points (entrée,sortie). On peut également appliquer le TSP avec des zones à la place de certains points : les raffineurs sont alors en mesure de considérer les différentes possibilités, au niveau de chaque zone, pour construire un chemin.

– **SCD-2 : Annulation / interruption de tâches :**
Deux cas de figure peuvent se présenter : si l'opérateur est appliqué suite à une sollicitation d'un utilisateur (par SCD-0), SCD-2 affecte le plan de haut niveau (les missions, tâches de haut niveau et plan de haut niveau sont les seules informations accessibles à un utilisateur).
Sinon, s'il s'agit d'une application par un autre type de sollicitation (événements traités dans le SCD), SCD-2 peut aussi bien concerner le plan de haut niveau que le plan élémentaire. Une annulation ou interruption de tâche peut elle même provoquer en cascade l'annulation ou l'interruption d'autres tâches du plan.

– **SCD-3 : Raffinement / décomposition d'une tâche de haut niveau :** Cet opérateur permet de faire appel au couple PS / RS pour raffiner une tâche d'un niveau d'abstraction donné (correspondant à une méthode donnée dans le planificateur symbolique). L'application de cet opérateur à une tâche non-encore raffinée conduit à obtenir un plan de tâches raffinées, prêt à être coordonné et à être traité pour exécution.

– **SCD-4 : Chargement d'une tâche jointe élémentaire :** cette opération est réalisée pour charger dans le GI une TJE obtenue suite au raffinement d'une tâche de haut niveau (SCD-3). Cette opération est uniquement appliquée pour des TJE de plans qui sont destinés à être exécutées peu après (dans certains cas, SCD-3 n'est appliqué que dans une optique d'évaluation ou de prédiction, sans que le plan retourné par le planificateur soit transmis pour exécution vers l'EMD ou vers le GI).
Le chargement d'une tâche jointe élémentaire nécessite de déterminer un créneau temporel

commun pour tous les UAV concernés par cette TJE.
Pour cela, le chargement d'une TJE doit s'accompagner d'une négociation sur l'intervalle temporel correspondant. Le SCD doit déterminer un ensemble de préférences temporelles pour cette TJE, compte tenu du plan courant. Un plan élémentaire possède une certaine étendue temporelle, et la réalisation d'une tâche jointe, dans le cadre de ce plan, implique des limites temporelles pour sa réalisation. Qui plus est, à supposer que la TJE considérée provienne d'une insertion de tâches dans un plan déjà existant, plusieurs configurations sont susceptibles d'être satisfaisantes. Il est alors possible et nécessaire d'évaluer et classer les configurations satisfaisantes (avec un critère de minimisation de l'étendue temporelle du plan élémentaire, par exemple). Les préférences de créneaux d'interaction sont transmises au GI en même temps que la TJE à charger : des tractations ont alors lieu entre les UAV afin de décider du créneau le plus approprié. Ce créneau est retourné au SCD, qui ajuste les contraintes temporelles du plan élémentaire en fonction des bornes temporelles convenues (il s'agit de propager des décalages dans les dates de début des tâches élémentaires, dans la limite de leurs contraintes temporelles absolues). Lors de son exécution, la TJE se déroulera alors au même moment au niveau de tous les UAV impliqués.

– **SCD-5 : Exécution d'un plan / d'une portion de plan :** cette opération a pour effet de transmettre vers l'EMD le contenu d'un plan élémentaire (d'une portion de plan élémentaire). Cette opération place les tâches correspondantes dans un horizon que l'on peut voir comme proche de l'exécution : les tâches transmises à l'EMD ne sont plus susceptibles d'être remises en cause en terme d'ordre ou moment d'exécution, si le SCD veut appliquer l'opérateur SCD-1 par exemple. Ainsi, la frontière entre le SCD et l'EMD correspond également à l'horizon de révision autorisée de plan. Cela dit, il est toujours possible, sur ordre explicite d'un utilisateur du système, d'interrompre ou annuler des tâches qui sont déjà transmises au niveau de l'EMD.

– **SCD-6 : Révision / réparation / replanification de plan en échec :** pour cette opération, il s'agit de réviser un plan qui se retrouve en situation d'échec, suite à une contingence non gérée ou à une intervention prioritaire d'un utilisateur (annulation de tâches de haut niveau...). Un premier choix peut être de considérer tout échec d'exécution de tâche élémentaire comme un échec complet sur le plan courant. Deux possibilités sont alors envisageables : considérer l'intégralité de la mission comme un échec, et le faire savoir à l'utilisateur du système. Ou bien replanifier "depuis le haut" (tâches de haut niveau/ missions spécifiées par l'utilisateur), pour obtenir un nouveau plan complet cohérent, si tant est que les contraintes temporelles sous-jacentes soient encore applicables.

Une autre politique peut consister à annuler ou interrompre les tâches élémentaires posant problème et celles qui en dépendent directement (avec a priori la même tâche de haut niveau d'origine), mais sans pour autant déprogrammer l'intégralité du plan élémentaire courant. Cela nécessite de pouvoir clairement désigner les tâches en échec.
Il faut noter que la représentation disponible des plans, venant de données générées par un planificateur HTN, n'est pas très propice à l'exploitation d'informations sur la causalité ou l'historique de génération des tâches du plan. Cependant, de la même manière que nous proposons de propager l'identifiant de la tâche de haut niveau d'origine au cours de la planification (grâce au mécanisme de propagation introduit dans la section IV.2.1), il pourrait être tentant

de propager en cours de planification l'historique cumulé de passage sur les méthodes, en conservant pour chaque méthode les effets en terme de faits retirés et faits ajoutés. Au final, le plan obtenu contiendrait alors, pour chaque tâche élémentaire, l'historique qui y a mené. Il serait alors envisageable de reprendre la planification avec l'état courant, tout en minimisant les effets sur le plan courant (il s'agirait donc de reprendre chronologiquement les dernières méthodes appliquées, et de tenter de replanifier compte tenu de l'état courant : un retour arrière chronologique, en quelque sorte). Cette démarche mériterait d'être d'avantage développée et évaluée dans de futurs travaux.

La figure IV.4 schématise les entrées (débuts de flèches) et sorties (fin de flèches) des opérateurs, avec, le cas échéant, l'utilisation des composantes de la CD (plus l'EMD). Par ailleurs, ces opérateurs ne sont pas indépendants : leur usage peut nécessiter l'application successive ou simultanée d'autres opérateurs. Nous illustrons ci-après ces dépendances entre opérateurs du SCD IV.5.

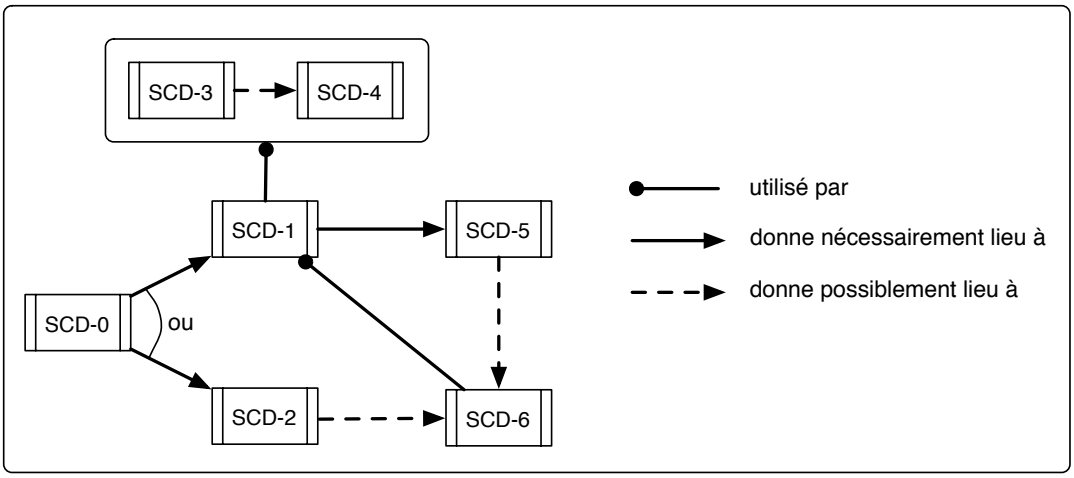


FIG. IV.5 – Dépendances entre opérateurs du SCD

Les opérateurs du SCD ne sont encore que partiellement développés au moment de la rédaction de ce mémoire. Leur implémentation est indispensable pour réellement expérimenter la couche délibérative sur des robots.

Notons finalement que le SCD doit toujours être disponible pendant l'application des opérateurs : il doit toujours être en mesure de répondre à des sollicitations, que ce soit des événements en provenance de la CD (GI, PS/RS) ou de l'EMD, ou encore sur requête d'un utilisateur du système.

Exemple de référence - partie 4 : opérateurs de SCD

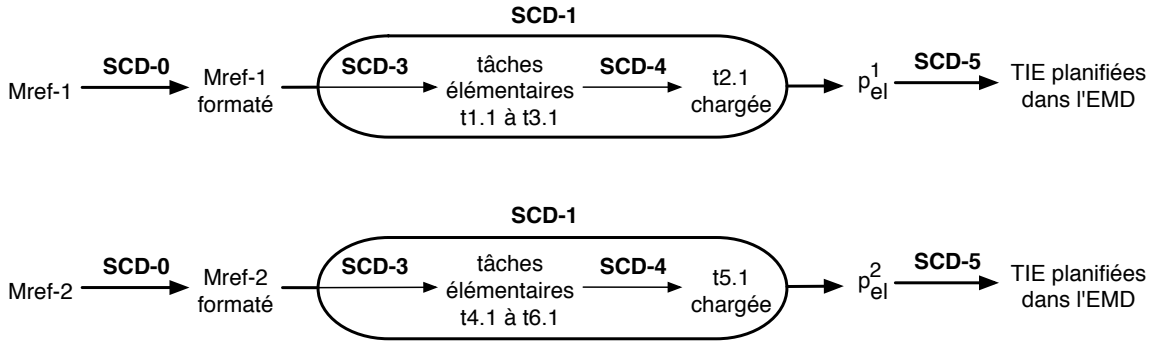
Dans cette partie de l'exemple de référence, nous décrivons comment sont appliqués les principaux opérateurs de SCD dans les couches délibératives respectives des UAV.

La première étape de l'exemple de référence concernait la saisie et la transmission des missions par un utilisateur du système. Il s'agit pour le SCD d'appliquer l'opérateur SCD-0

sur les missions transmises (*M1* et *M2*), pour chacun des UAV. Ensuite (partie 2 de l'exemple de réference), le SCD applique l'opérateur SCD-1 pour l'insertion de tâches, qui va lui-même exploiter SCD-3 pour le raffinement des tâches de haut niveau. Un plan élémentaire est obtenu. Ce plan contient une tâche jointe élémentaire : elle doit être traitée par le GI (partie 3 de l'exemple de référence), et le SCD applique donc immédiatement l'opérateur SCD-4 pour demander au GI de prendre en charge ce traitement. Cette application nécessite de déterminer les préférences sur les intervalles temporels pour les tâches jointes respectives (t2.1 et t5.1). Le superviseur de CD de chaque UAV évalue ces préférences à partir de l'étendue des plans élémentaires respectifs et des contraintes temporelles des tâches élémentaires qu'ils contiennent. Pour les deux UAV, la préférence d'intervalle suivante est calculée : $[9:00:30; max - 930[$. La borne maximum correspond au temps maximum (aucune valeur n'a été spécifiée dans l'énoncé de la mission) moins la durée de la tâche jointe élémentaire, moins la durée de l'atterissage $(900 + 30)$. Le SCD de chaque UAV transmet donc cette préférence aux GI respectifs, et après négociation, c'est l'intervalle temporel effectivement adopté.

Le SCD envoie ensuite le plan vers l'EMD pour exécution, en appliquant l'opérateur SCD-5 sur le plan élémentaire. L'EMD commence le traitement du plan pour exécution.

La figure suivante illustre l'application des opérateurs de SCD pour les deux UAV :



Accessoirement, l'opérateur SCD-6 peut être utilisé pour réviser le plan, le cas échéant. Nous ne considérons cependant pas cette éventualité dans le cadre de cet exemple de référence.

IV.5 Récapitulatif

Nous venons de présenter l'ensemble des composants de la couche décisionnelle : le superviseur de CD exploite les capacités du planificateur symbolique et des raffineurs spécialisés pour décomposer et affiner des tâches de haut niveau afin d'obtenir des plans composés de tâches élémentaires (plans élementaires). Ces tâches élémentaires partiellement ordonnées peuvent alors être transmises à l'EMD en vue de l'exécution. Parmi celles-ci, les tâches jointes élémentaires sont déléguées au gestionnaire d'interaction pour être affinées et coordonnées dans un cadre multi-robots : c'est alors dans le contexte d'exécution proprement dit que ces tâches jointes élémentaires sont traitées.

Chapitre V

Le gestionnaire d'interactions

Dans le chapitre IV, nous avons introduit le gestionnaire d'interaction comme une boîte noire, simplement en termes d'entrées et sorties.

Ce chapitre fournit le détail du gestionnaire d'interactions (GI), qui regroupe différents composants nécessaires au traitement des interactions entre UAV. Nous présentons les formalismes et modèles qu'il requiert, et les mécanismes de traitement associés. Il est important de noter que ce qui est présenté ici n'est pas nécessairement entièrement implémenté : en fin de chapitre, nous récapitulerons et distinguerons ce qui est implémenté de ce qui ne l'est pas encore.

V.1 Vue d'ensemble du gestionnaire d'interactions

V.1.1 Composants

La figure V.1 présente une vue d'ensemble des différentes parties du GI. En termes de données manipulées, les *modèles d'interaction* (MI) décrivent le cadre des interactions souhaitées entre UAV : ils reposent sur des *rôles*, définissant des ensembles d'instructions , des *modèles de contraintes spatiales* (MCS), permettant de contraindre spatialement le cadre des interactions, et des tables de coordination, articulant les rôles et les MCS.

Au œur du système, le *moteur de traitement* est l'entité qui prend en charge le déroulement de *modèles d'interaction* (MI), à travers le traitement des rôles. Ce déroulement donne lieu soit à l'activation de tâches de haut niveau (le GI communique alors de façon interne avec le superviseur de CD pour gérer le raffinement et l'exécution de ces tâches), soit à l'activation de sessions de négociation, via le *gestionnaire de négociations* (GN) : pour ce faire, des *modalités de négociation* (MN) décrivent des protocoles de négociation entre UAV, afin de coordonner différents types d'actions jointes.

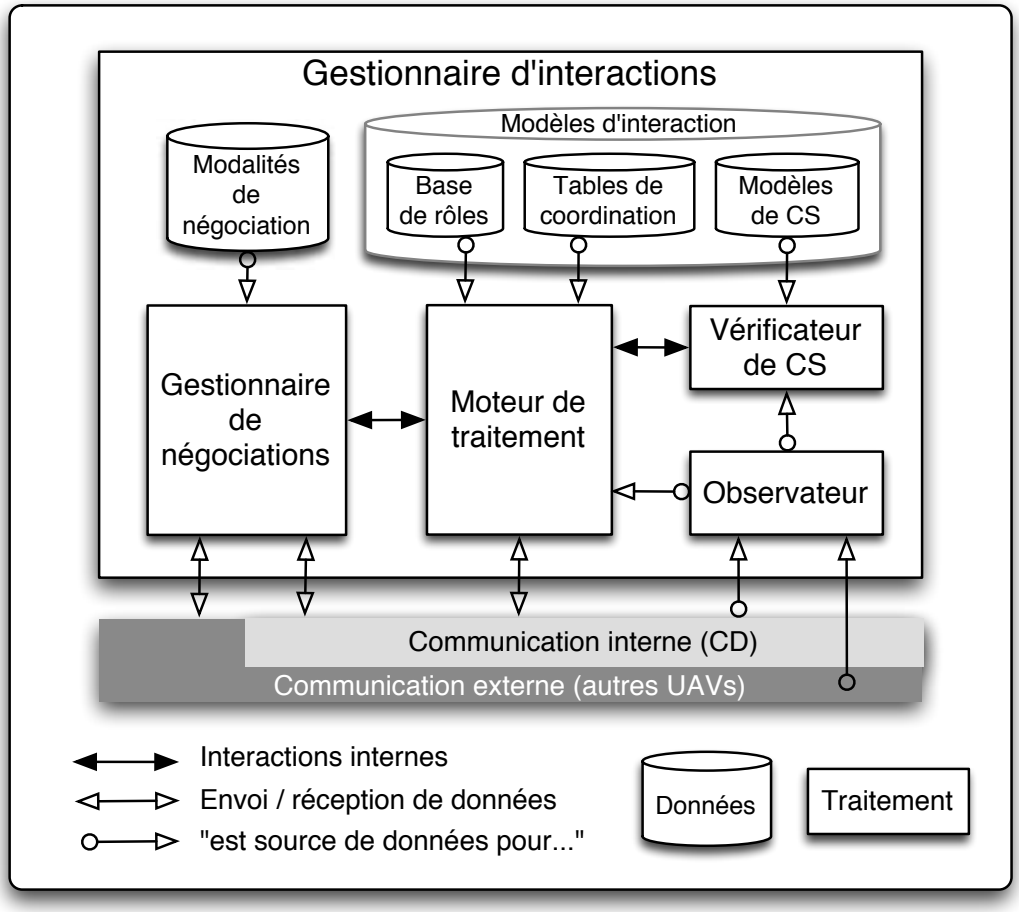


FIG. V.1 – Composantes du gestionnaire d'interactions (GI)

Par ailleurs, les modèles d'interaction peuvent exploiter des MCS : la satisfaction de contraintes spatiales dans un contexte courant est évaluée par le *vérificateur de contraintes spatiales* (VCS).

Enfin, *l'observateur* est une interface intégrée qui permet de souscrire à des sources de données pertinentes (données géométriques, états des UAV...) dans l'optique du traitement des modèles d'interaction et de la vérification des contraintes spatiales.

D'une façon générale, cette architecture de GI décrit distinctement un moteur de traitement de modèles d'interaction, et un gestionnaire de négociations qui offre les services de coordination que peuvent nécessiter les interactions, à travers des modalités de négociation. Il est important de noter que nous parlons de "modalités" de négociation plutôt que de "modèles", à l'inverse des "modèles d'interaction" : en effet, du point de vue d'un utilisateur de ce système, nous proposons une modélisation et un cadre bien formels pour la définition et le traitement des interactions, alors que nous laissons ouverts les choix et moyens d'implémentation des modalités de négociation. En effet, si l'on peut envisager quelques modalités de négociation communes à tous systèmes,

il nous parait souhaitable de laisser une (relative) liberté à l'utilisateur pour la conception des modalités susceptibles de l'intéresser, dans son propre contexte d'application.

V.1.2 Principes de fonctionnement

Le GI entre en action à partir du moment où une tâche coopérative (nous parlons de tâches jointes, et plus exactement de tâches jointes élémentaires (TJE) dans la couche délibérative) est produite dans le plan courant de la couche délibérative. A un plus haut niveau, cela signifie qu'un utilisateur du système a probablement envoyé une mission à cet UAV (et vraisemblablement à d'autres UAV), avec, parmi les tâches demandées, au moins une tâche jointe. Au niveau de chaque UAV recevant la tâche jointe élémentaire à exécuter, les traitements suivant sont réalisés par le gestionnaire d'interaction :

Le GI entre en action à partir du moment où une tâche jointe élémentaire est produite dans le plan courant de la couche délibérative. Notons qu'il existe une bijection entre les tâches jointes élémentaires et les modèles d'interaction.

– **chargement d'un modèle d'interaction :** Le GI met en relation la TJE reçue avec un modèle d'interaction *MI* Le modèle est chargé, appliqué dans le contexte courant, et préparé à être activement traité le moment venu.

– **allocation initiale de rôles :** au moment du chargement du modèle d'interaction, les UAV concernés négocient l'attribution des rôles proposés dans *MI*. Cette opération est réalisée à travers le gestionnaire de négociations (GN), qui dispose d'une modalité de négociation spécifique, dédiée à l'allocation de rôle.

– **activation et traitement du rôle :** le modèle d'interaction reste chargé aussi longtemps que la TJE qui en est à l'origine est en attente d'exécution (parmi les tâches planifiées dans l'EMD). Au moment où la TJE est déclenchée dans l'EMD, le rôle initialement alloué à cet UAV est activé : le moteur de traitement du GI traite le contenu du rôle

– **sessions de négociations :** le traitement des instructions d'un rôle peut donner lieu à des sessions de négociations entre UAV, selon des modalités de négociation prédéfinies : les UAV échangent alors des informations liées au contexte, avec un protocole spécifique à la modalité de négociation adoptée.

– **vérification de contraintes spatiales :** durant le traitement du rôle, des modèles de contraintes spatiales peuvent s'appliquer. Quand c'est le cas, le vérificateur de contraintes spatiales vérifie continuellement la satisfaction des contraintes. En cas de violation d'une contrainte, il génère un événement qui peut être intercepté pour modifier le cours du traitement du rôle, le cas échéant.

– **génération de tâches individuelles :** le traitement du rôle génère des tâches qui sont transmises au superviseur de CD pour raffinement puis exécution dans le contexte courant. Lorsque le déroulement du rôle prend fin, la TJE qui en est à l'origine prend fin également.

Remarque sur l'allocation initiale de rôles : Nous supposons que les missions sont transmises aux UAV simultanément : ainsi pour chaque UAV, lorsque les missions contiennent des tâches jointes, les traitements initiaux respectifs (chargement de modèle d'interactions) se produisent approximativement dans le même intervalle de temps.

Nous supposons de plus que les tâches jointes contenues dans les missions respectives sont cohérentes : les tâches jointes élémentaires résultantes doivent en effet être les même, afin que chaque UAV impliqué charge dans son GI le même modèle d'interaction. Sans cette cohérence, les UAV pourraient essayer de charger des modèle d'interactions différents, avec des ensembles de rôles différents : il ne serait alors pas possible de négocier une allocation de rôle cohérente (cela n'aurait généralement pas de sens). En formulant cette hypothèse, nous nous plaçons explicitement au degré 4 d'autonomie décisionnelle : les UAV se coordonnent de façon autonome, mais ne remettent pas en question l'allocation des tâches de la mission multi-robots, qui doit être fournie de façon cohérente dès le début. Au niveau 5 (notons que nous n'avons pas travaillé à la spécification et au développement de ce niveau) on peut au contraire imaginer qu'une mission globale donnée soit formulée sans se soucier de la cohérence entre les tâches : il s'agirait alors, pour les robots du système, d'organiser par eux-même la décomposition et la distribution des tâches, et éventuellement de la remettre en cause en cours de mission. Une piste à exploiter dans cette optique est d'étendre et systématiser l'utilisation du gestionnaire d'interactions, au plus proche de la planification de tâche : il s'agit alors d'évaluer différents plans, compte tenu des tâches à répartir, et à procéder à des sessions de négociation pour réaliser l'allocation. Il est alors véritablement question de planification distribuée et concertée.

V.2 Modélisations pour l'expression de tâches coopératives

Avant de détailler la mécanique de traitement au sein du GI, nous présentons en premier lieu les principaux formalismes et modèles sur lesquels repose ce sytème de gestion d'interactions.

V.2.1 Introduction aux modèles d'interaction

Dans notre approche, nous avons choisi de définir explicitement des modèles d'interaction pour spécifier la nature, la configuration, les caractéristiques des interactions souhaitées dans un groupe de robots aériens. Pour cela, nous avons adopté une approche **modulaire** et **intégrative**, plutôt que de redéfinir systématiquement tous les éléments des modèles d'interaction à partir de zéro :

– **modulaire**, car les modèles d'interactions sont définis avec une certaine généricité, dans une optique de réutilisabilité : un même modèle peut en effet être utilisé dans des contextes différents et des situations différentes.

– **intégrative**, car les modèles d'interactions sont construit en agrégeant un certain nombre de composants plus génériques, définis indépendamment les uns des autres.

Ainsi, la base de rôles comprend un ensemble de rôles abstraits, hors d'un contexte particulier. Il en est de même pour les modèles de contraintes spatiales, qui définissent des relations spatiales entre des entités abstraites, qui ne trouveront une signification qu'au moment de l'instanciation effective de ces modèles dans un contexte d'exécution donné.

La figure V.2 ci après schématise les différents éléments exploités dans un modèle d'interaction.

Un modèle d'interaction rassemble un certain nombre de variantes[1] qui peuvent être considérées comme autant d'options pour répondre au traitement d'une tâche jointe donnée.

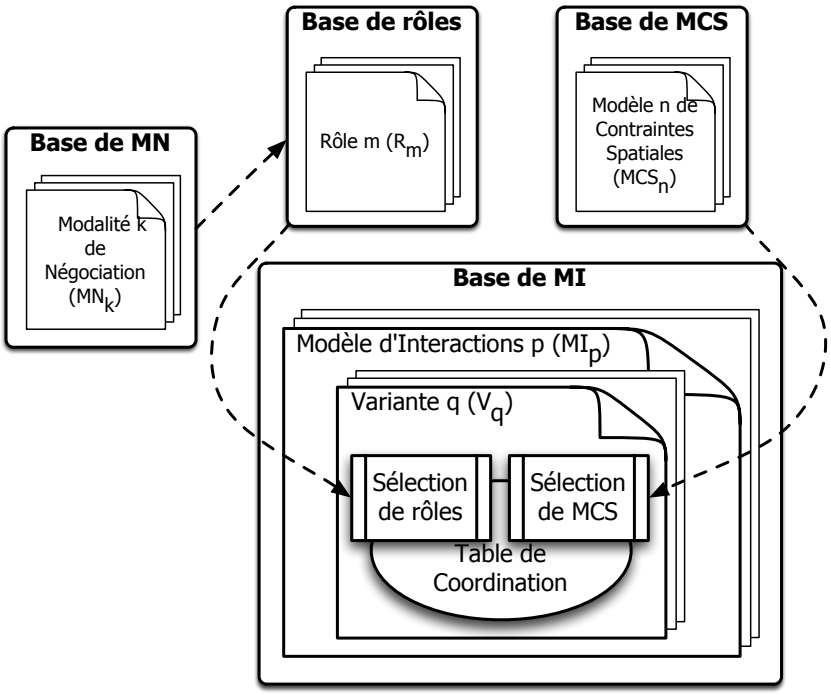


FIG. V.2 – Modèle d'interaction : composantes

Chaque variante de MI fait appel à un certain nombre de rôles, et précise explicitement les articulations entre ces rôles par différents moyens regroupés au sein d'une table de coordination. Lorsqu'un UAV endosse un rôle il applique le contenu du rôle dans le contexte d'exécution courant.

Les rôles définissent les tâches à accomplir dans le cadre des modèles d'interaction. Chaque rôle peut être endossé par un certain nombre d'UAV, en fonction d'une arité fixée (dans la table de coordination) pour ce rôle. Pour qu'une modalité de négociation puisse être appliquée avec succès, il est nécessaire que l'ensemble des UAV impliqués endosse les rôles définis dans la variante choisie, en respectant les arités notifiées pour chaque rôle.

Les contraintes spatiales définissent des relations géométriques qui doivent être respectées pendant la réalisation d'instructions des rôles.

La table de coordination permet de concrétiser la définition d'une variante de modèle d'interaction, en liant d'une part les rôles entre eux, et d'autre part les modèles de contraintes spatiales avec le contenu de ces rôles.

[1]Dans un souci de simplicité, dans la suite du manuscrit, nous désignerons parfois par "modèle d'interaction" une variante elle-même, sans systématiquement préciser "variante de modèle d'interaction".

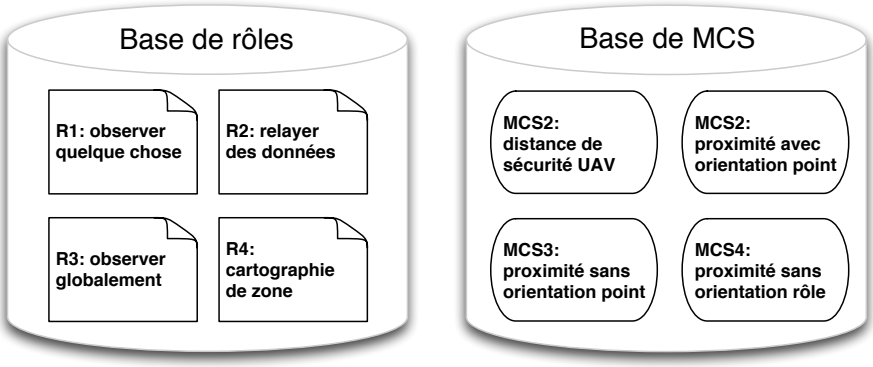


FIG. V.3 – Exemple de base de rôles et base de MCS

La figure V.3 illustre une base de rôles contenant quatre rôles différents et une base de modèles de contraintes spatiales (MCS) contenant quatre modèles. Ces différents modèles sont fournis hors-ligne par un utilisateur du système, en vue d'application futures.

Les rôles contiennent un certains nombre de données dont le détail sera précisé dans la suite, de même que le détail des contraintes spatiales des MCS.

Une variante de modèle d'interaction possède un certain nombre de paramètres : ceux-ci doivent être instanciés au moment de l'application de la variante dans un contexte d'exécution, suite au traitement de la TJE correspondante. Les paramètres de la variante sont alors instanciés avec les paramètres passés à la TJE (il doit exister une correspondance typées entre les paramètres respectifs).

Une table de coordination précise en particulier :

– **l'instanciation des MCS utilisés dans ce modèle d'interaction** : il s'agit d'instancier explicitement les paramètres des contraintes spatiales (valeur arbitraire fournie par un utilisateur), ou bien de déclarer un lien entre un paramètre de contrainte spatiale donné et un paramètre du modèle d'interaction.

– **les correspondances entre MCS et rôles** : sur quels rôles (et dans les faits, sous quelles sous-parties de rôles) s'appliquent les MCS instanciés.

– **l'arité relative à chacun des rôles en présence** : combien d'UAV doivent endosser chaque rôle.

– **les correspondances entre instructions dans les rôles** : quels sont les liens entre les instructions donnant lieu au traitement de modalités de négociation.

La figure V.4 représente la création d'une telle table de coordination, à partir des rôles et MCS donnés précédement.

Un aspect majeur de notre approche est la possibilité d'exploiter des définitions de rôle dans des modèles d'interaction très différents. La figure V.5 ci-après illustre une réutilisation des rôles employés dans (MI1,v1) dans d'autres MI, de même que la réutilisation de certains MCS. Ce sont les tables de coordination respectives qui permettent de donner tout leur sens à ces assemblages

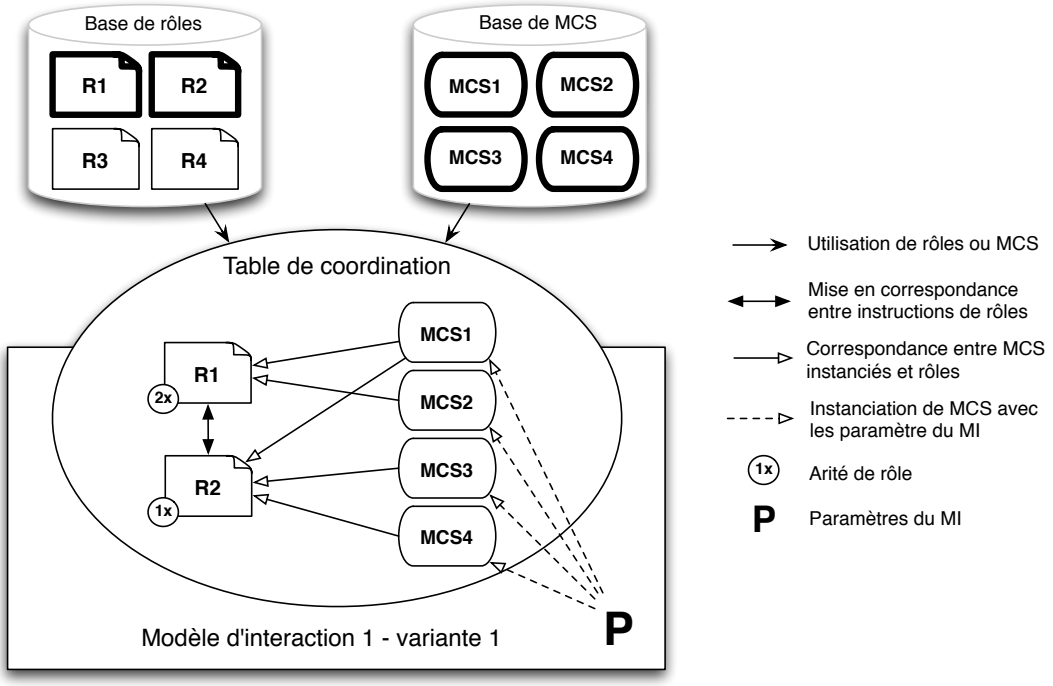


FIG. V.4 – Table de coordination d'une variante v1 d'un modèle d'interaction MI1

de rôles et MCS : et c'est l'application d'une variante de MI dans un contexte d'exécution qui donne finalement tout son sens à un MI donné.

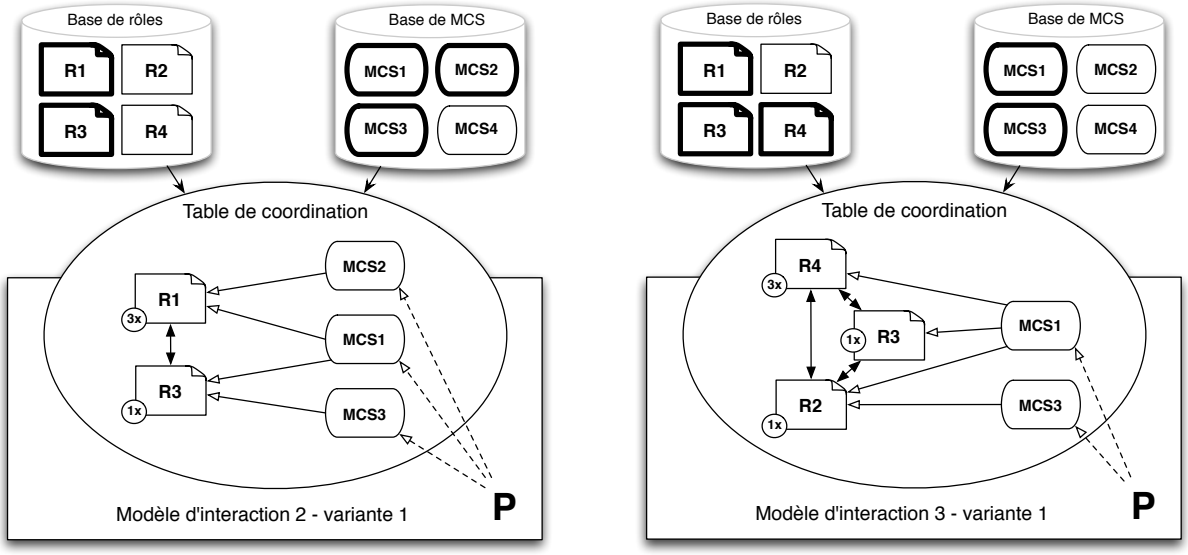


FIG. V.5 – Gauche : MI2 "Poursuite de véhicule" - Droite : MI3 "Cartographie d'une zone"

Sur la figure V.5, la variante MI2.v1 correspond à une configuration de poursuite collaborative d'un véhicule (une voiture par exemple). La table de coordination spécifie des arités de 3 et 1 respectivement pour le rôle de poursuite proprement dit du véhicule et pour le rôle d'observation globale. La variante MI3.v3 décrit quant à elle une configuration de cartographie coopérative avec trois UAV se partageant la tâche de cartographie proprement dite, un UAV survolant plus globalement la zone, et un UAV assurant le relais d'informations (vers un centre de contrôle, par exemple...). Les MCS sont réutilisés de la même manière, et associés aux rôles, après instanciation.

Après cette introduction générale et informelle sur les modèles d'interaction, les paragraphes suivants définissent plus formellement les modèles de rôles, de contraintes spatiales (MCS), les tables de coordination (TC), et finalement les modèles d'interaction (MI) proprement dits.

V.2.2 Modélisation des rôles

Un rôle r est défini de la façon suivante :

Définition V.1 : rôle

$r = (CComp_r, P_r, AE_r)$, où :
- $CComp_r$ est un ensemble de conditions de compatibilité.
- P_r est un ensemble de paramètres à instancier.
- AE_r est un automate d'exécution.

$CComp_r$ définit un ensemble d'exigences en terme de capacités que doit satisfaire l'UAV qui endossera ce rôle. Les conditions de compatibilité peuvent concerner l'UAV lui-même (vitesse, autonomie, rayon de giration, possibilité de vol stationnaire...) ou bien les capacités de perception (type de capteur, caractéristiques de perception...). C'est un préalable à toute évaluation plus fine de l'utilité d'endosser un rôle.

P_r définit des paramètres qui représentent des quantités (coûts...), des durées, des objets dans l'environnement, d'autres UAV. La mise en relation de ces paramètres avec leurs valeurs dans le contexte courant s'opère lorsque le modèle d'interaction qui fait appel à ce rôle est activé. Les cibles des paramètres sont donc des arguments de la tâche jointe à l'origine de l'activation.

AE_r définit les tâches et démarches (actions de coordination...) à entreprendre afin de répondre au "cahier des charges" de ce rôle.

Un AE_r est constitué de trois types de composants (que nous regroupons sous l'appelation *componels* ("components elements"). Ces componels sont les *blocs*, les *tâches*, et les *actions de coordination* :

– les blocs sont des "conteneurs" qui permettent d'appliquer un certain nombre de propriétés à leur contenu (en particulier, ils permettent de définir des préconditions et des conditions de sortie sur le déroulement de leur contenu). Leur contenu est constitué d'un certain nombre de componels.

– les tâches décrivent des actions à réaliser, à des niveau de granularité variable. Elles correspondent à des tâches de haut niveau, individuelles ou jointes.

– les actions de coordination correspondent à des appels explicites à des modalités de négociation, et permettent donc à l'UAV de prendre part aux sessions de négociation pendant que le modèle d'interaction est en cours.

Nous présentons la définition formelle des différents componels dans la section V.4, liée au mécanisme de traitement des rôles : en effet, nous considérons que le détail de ces componels nécessite de présenter conjointement leur cadre de traitement algorithmique, dans un souci de clarté.

Exemple de référence - partie 5 : représentation des rôles

Nous présentons ici les rôles mis en jeu dans le modèle d'interaction utilisés dans notre exemple de référence. Le modèle d'interaction utilisé est celui qui correspond aux tâches jointes élémentaires t2.1 et t5.1 (voir la section IV.2.2).

Nous appellerons MI_{ref} le modèle d'interaction utilisé dans notre exemple de référence. Nous mettons en œuvre 2 rôles dans MI_{ref} : R^1_{ref} et R^2_{ref}. Ils vont respectivement correspondre aux tâches pour la perception du feu et aux tâches pour le relais des communications. Voici pour chacun de ces rôles les conditions de compatibilité et les paramètres attendus :

– R^1_{ref} (pour les perceptions) :
– $\mathsf{CComp_r} = (vol\text{-}stationnaire, camera\text{-}basique)$
– $\mathsf{P_r} = (< Obj > front, < Obj > center, < Point > ground\text{-}point, < Time > duration, < UAVlist > uavs)$
– R^2_{ref} (pour le relais) :
– $\mathsf{CComp_r} = (vol\text{-}stationnaire, capacite\text{-}de\text{-}relais)$
– $\mathsf{P_r} = (< Obj > front, < Obj > center, < Point > ground\text{-}point, < Time > duration, < UAVlist > uavs)$

Les paramètres des rôles sont typés : le type $< Obj >$ représente le nom symbolique d'un objet connu de *l'observateur* : cet objet pourra être observé dans son évolution en cours de mission. Le type $< Obj >$ est compatible partout où le type $< Point >$ est demandé. Le type $< Point >$ représente un nom symbolique connu de tout le système pour désigner un point fixe donné, ou bien directement des coordonnées d'un point arbitraire. Le type $< Time >$ désigne un format de temps correspondant à une heure absolue ou bien à une durée. Le type $< UAVlist >$ est une liste d'identifiants d'UAV (nous ne détaillerons pas tous les autres types possibles de paramètres). Dans ces paramètre, *front* et *center* seront instanciés avec les données connues sur le feu à percevoir, et *ground-point* correspondra à la position du centre de contrôle au sol.

Les conditions de compatibilité, qui reflètent les différentes capacités nécessaires pour endosser un rôle, peuvent être précisées aussi bien symboliquement que numériquement. Nous ne détaillons pas les capacités envisageables (elles dépendent essentiellement du type d'activité, du type d'UAV pouvant être considéré, etc.).

Par ailleurs, nous ne précisons pas ici la troisième partie du rôle, à savoir l'automate d'exécution et son contenu : en effet, celui repose principalement sur la notion de bloc, qui sera défini par la suite, dans le contexte du traitement proprement dit de l'automate d'exécution (section V.4).

V.2.3 Modèles de contraintes spatiales

Les MCS visent à décrire un ensemble de contraintes spatiales entre des entités (UAV ou environnement), selon les besoins des missions à réaliser. Ces MCS s'appliquent aux blocs des rôles, c'est à dire à certaines sections d'instructions à réaliser dans les rôles. Lorsqu'un bloc auquel est attaché un MCS est actif, alors le MCS en question est appliqué. Concrêtement, cela se traduit par une vérification cyclique de l'ensemble des contraintes spatiales définies dans ce MCS et appliquées autour de cet UAV, dans son contexte d'exécution. Les MCS des blocs actifs sont traités par le vérificateur de CS, ou VCS, dont il est question dans les mécanismes de traitement des modèles d'interaction (section V.4).

Formellement, un modèle de contraintes spatiales est défini comme suit :

Définition V.2 : modèle de contraintes spatiales

$mcs = (id, ListeCS)$, où :
– *id* est un l'identifant de ce MCS,
– *ListeCS* est une liste de contraintes spatiales (*CS*).

Les *CS* sont à considérer du point de vue de l'UAV qui endossera un rôle donné dans lequel s'appliquent ces *CS*. Autrement dit, une composante commune à toutes les *CS* est la position de l'UAV. Deux types de contraintes spatiales sont proposées à partir de là : il s'agit des contraintes de distance, et des contraintes d'orientation.

Les contraintes de distance peuvent être exprimées de deux manières. La première consiste à fournir un point P, une valeur numérique V, et une relation de comparaison R devant être satisfaite. Dans ce cadre, la contrainte exprime que la distance de l'UAV à P d'un côté et la valeur V de l'autre côté, dans cet ordre, doivent satisfaire R. Cette première manière est utile pour définir une comparaison de distance par rapport à une valeur numérique connue ou imposée.

exemple : $cs_0 = (P_0, 200, <)$, sera compris comme "la distance de l'UAV à P_0 doit être inférieure à 200".

La seconde manière consiste à fournir d'un côté un point P_0 définissant la distance à l'UAV, et de l'autre côté deux points P_1 et P_2 définissant la distance devant être comparée ainsi que la relation R à vérifier. Cette seconde manière est utile pour définir une comparaison de distance par rapport à d'autres entités de l'environnement, et non plus seulement par rapport à une valeur arbitraire.

exemple : $cs_1 = (P_0, P_1, P_2, <)$, sera compris comme "la distance de l'UAV à P_0 doit être inférieure à la distance de P_1 à P_2".

Les contraintes d'orientation sont évaluées sur la base d'un référentiel qui doit être explicitement spécifié dans la contrainte spatiale. Le référentiel est défini par deux points : le premier point Q_0 permet de spécifier un plan parallèle au "sol". Sur le plan ainsi défini, nous considérons

les coordonnées célestes : azimuth et altitude. Pour cela, le second point Q_1 permet de définir la direction d'azimuth 0 (voir schéma V.6 ci après).

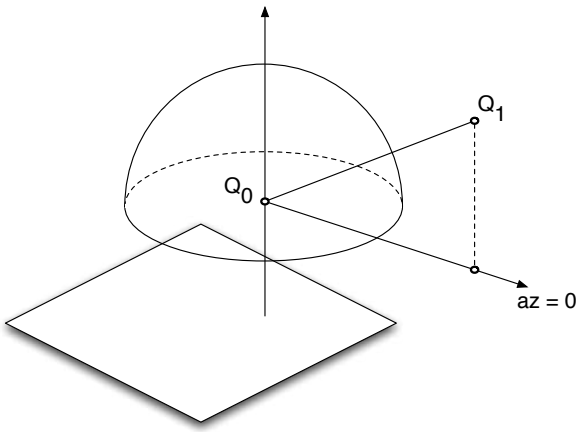


FIG. V.6 – Définition du référentiel céleste à partir de deux points

Dans un repère céleste, les coordonnées des points ne sont considérées que du point de vue de leur orientation, ce qui est parfaitement adapté à la définition de contraintes d'orientation. Les coordonnées de points dans un tel repère sont l'azimuth, angle variant de 0 à 2π radians dans le plan "horizontal" du repère, croissant dans le sens anti-trigonométrique, et l'altitude, angle variant de $-\pi/2$ à $\pi/2$ de "bas en haut".

De même que pour les contraintes de distance, les contraintes d'orientation peuvent s'exprimer de deux manières qui permettent respectivement de faire une comparaison par rapport à une valeur numérique, ou de faire une comparaison par rapport à d'autres entités de l'environnement.

La première consiste à comparer l'angle de l'UAV par rapport à une valeur numérique V, dans un repère céleste défini par deux points Q_0 et Q_1. L'angle peut être soit l'azimuth, soit l'altitude.

exemple : $cs_2 = (Q_0, Q_1, ALTITUDE, \pi/2, =)$, définit une contrainte pour laquelle l'altitude de l'UAV doit être égale à $\pi/2$ dans le repère céleste défini par Q_0 et Q_1.

La seconde consiste à comparer l'angle de l'UAV (azimuth ou altitude) dans un repère céleste défini par deux points Q_0 et Q_1 avec l'angle d'un autre point P dans ce repère.

exemple : $cs_3 = (Q_0, Q_1, AZIMUTH, P, >)$, définit une contrainte pour laquelle l'azimuth de l'UAV doit être supérieur à l'azimuth de P dans le repère céleste défini par Q_0 et Q_1.

Les valeurs numériques données dans les exemples sont illustratives. En réalité, elles ne sont explicitement fournies qu'au moment d'instancier les modèles de contraintes spatiales, dans la table de coordination d'un modèle d'interaction (voir la section V.2.4).

En combinant des contraintes d'orientation et de distance, il est possible de définir explicitement des configurations spatiales qui doivent être respectées pendant la réalisation de tâches jointes.

Nous verrons par ailleurs que ces contraintes spatiales peuvent être exploitées pour déduire les positions valides d'activité, pendant la réalisation d'une tâche jointe : le VCS dispose en effet de moyens de déterminer des positions de l'espace valides, compte tenu des contraintes. Nous en parlons dans la section V.4.3.

Exemple de référence - partie 6 : modèles de contraintes spatiales

Nous définissons dans cette partie des modèles de contraintes spatiales qui sont employés dans le modèle d'interaction de notre exemple de référence. Trois modèles de contraintes seront définis :

Le premier, mcs_1, est assez basique. Il ne contient qu'une seule contrainte spatiale, et vise à conserver une distance de sécurité entre les UAV. Nous écrivons donc la contrainte comme suit : $cs_1 = (R, V, >)$. Nous fixons arbitrairement cette distance de sécurité à 20 mètres.

Le deuxième, mcs_2, concerne la tâche de perception du feu. Nous définissons deux contraintes de distance, et deux distances d'orientation :

– Une première contrainte concerne la distance au feu : la contrainte s'écrit :
$cs_{21} = (P, V, >)$
Nous fixons V à 25 mètres, pour une question de sécurité.

– La deuxième contrainte concerne aussi la distance au feu : pour réaliser des perceptions de bonne qualité, la distance au feu doit en effet être inférieure à 40 mètres.
$cs_{22} = (P, V, <)$

– La troisième contrainte concerne la direction de perception du feu : il s'agit d'une perception de face, et l'angle d'azimuth, doit donc être nul par rapport aux deux points de référence :
$cs_{23} = (P_0, P_1, AZIMUTH, V, =)$
La valeur de V sera fixée à 0 lors de l'instanciation du modèle de contraintes spatiales.

– La quatrième contrainte concerne la hauteur de perception du feu : l'angle d'altitude est arbitrairement fixé à $\pi/6$ pour notre exemple de référence :
$cs_{24} = (P_0, P_1, ALTITUDE, V, =)$

Le troisième, mcs_3, concerne la tâche de relais de communication. Nous définissons deux contraintes de distance :

– Une première contrainte concerne la distance à l'UAV relayé : elle ne doit jamais être supérieure à la distance de communication, arbitrairement fixée à 100 mètres.
$cs_{31} = (R, V, <)$

– La deuxième contrainte concerne la distance au centre de contrôle, considéré comme un point : la distance ne doit jamais être supérieure à la distance de communication, arbitrairement fixée à 100 mètres.

$cs_{32} = (P, V, <)$

Ces modèles de contraintes spatiales sont pour l'instant déclarés de façon abstraite : celui conçu pour le relais de communication peut très bien être utilisé dans beaucoup d'autres cas de figure que notre exemple de référence, par exemple. De même, celui destiné à la perception de feu peut être utilisé dans d'autres contextes de perception de cible. Les valeurs numériques, rôles et points seront instanciés dans la table de coordination, qui fait l'objet de la sous-section suivante.

V.2.4 Formalisme de table de coordination

La table de coordination (TC) dans un MI, est la partie qui lie les différents composants en présence (rôles et MCS), et donne sa cohérence à ce MI. Voici les composants d'une TC :

1. **la donnée de l'arité respective de chaque rôle :** il s'agit de déterminer combien de robots peuvent endosser chacun des rôles. Cela peut être une valeur numérique, ou une relation du type "au moins x" ou "au plus x" ou "entre x et y".
2. **la mise en relation des actions de coordinations entre les AE des différents rôles**
   Cette mise en relation est exprimée à travers une liste d'éléments de mise en correspondance $corresp$ définis comme suit :

   Définition V.3 : mise en correspondance d'un AC

   $corresp = (id_{targetR}, id_{targetAC}, arity[, num])$, où :
   – $id_{targetR}$ est l'identifiant du rôle ciblé,
   – $id_{targetAC}$ est l'identifiant de l'action de coordination ciblée dans ce rôle,
   – $arity$ prend l'une des valeurs suivante :`everyone` ou `num`,
   – et num est une valeur numérique, à fournir lorsque arity est de type `num`.

   La valeur de $arity$ permet de spécifier explicitement combien d'individus de la population des UAV endossant le rôle ciblé devront prendre part à l'action de coordination.
   Il est à noter que les actions de coordination mises en relation doivent nécessairement être de même type : cette mise en correspondance n'a pas de signification dans le cas contraire. Un mécanisme de vérification de cohérence doit donc être appliqué lors de la création d'une table de coordination dans un MI, pour s'assurer qu'elle est formée de façon cohérente.

3. **l'instanciation des MCS :** l'instanciation de modèles de contraintes spatiales est possible dans la limite des informations disponibles lors de la création de la table de coordination : à savoir d'une par les rôles en présence, et d'autre part la signication particulière (la sémantique) du cadre du MI pour laquelle la table de coordination est créée. Ainsi, c'est ici que sont spécifiés les rôles et points des contraintes spatiales. Les rôles peuvent être choisis parmi ceux en présence, et les points sont associés à des points définis en paramètre de la variante de modèle d'interaction (nous reviendrons sur ce point dans la définition formelle d'un modèle d'interaction, section V.2.5).

Par ailleurs la sémantique, c'est à dire la signification que l'on veut attacher au cadre d'interaction, permet de choisir et d'attribuer des valeurs numériques arbitraires aux variables de valeur numériques des contraintes spatiales, par exemple pour la définition d'une distance de sécurité entre UAV.

4. **la mise en relation explicite des MCS instanciés** avec les blocs des AE, dans les rôles, est exprimée de la façon suivante :

   Pour chaque modèle instancié *mcs*,

   Définition V.4 : mise en correspondace d'un MCS avec des blocs

   $corresp_{mcs} = (id, ListOfBlocks)$, où :
   – id est l'identifiant de cette mise en correspondance,
   – $ListOfBlocks$ est une liste de couples (id_{role}, id_{blocs}) rassemblant tous les blocs des rôles en présence pour lesquels ce modèle doit être appliqué, lorsque actifs.

   Lorsqu'un des blocs référencé pour un MCS instancié donné devient actif, alors le MCS instancié est continuellement vérifié pour l'UAV qui endosse ce rôle, dans son contexte d'exécution.

5. **la mise en correspondance des paramètres de rôles** avec les paramètres du modèle d'interaction. A chaque paramètre de rôle doit correpondre un des paramètres du modèle d'interaction. Dans les faits, le modèle d'interaction est construit des sorte que ce soit le cas (voir par la suite la définition du modèle d'interaction : section V.2.5).

Exemple de référence - partie 7 : table de coordination

Nous décrivons ici la table de coordination tc_{ref} qui va permettre de lier rôles et modèles de contraintes spatiales, en vue d'un modèle d'interaction donné. Nous avons besoin pour cela d'évoquer des données qui n'ont pas encore été définies : il s'agit de composantes de l'automate d'exécution, qui seront intégralement détaillées dans la section V.4.

– Arité des rôles : nous spécifions ici qu'un seul UAV doit endosser le rôle R^1_{ref}, et un seul UAV doit endosser le rôle R^2_{ref} (donc exactement 2 UAV au total sont requis pour cette tâche jointe).

– Mise en correspondance des actions de coordination : n'ayant pas encore détaillé le contenu des automates d'exécution des rôles, les actions de coordination n'ont pas encore été données. Nous les évoquons ici seulement par un nom symbolique, et nous préciserons ce à quoi elles correspondent au moment de détailler le contenu des automates d'exécution des rôles, dans la section V.4. Le rôle R^1_{ref} comprend deux actions de coordination : ac_{11} et ac_{12}. Le rôle R^2_{ref} comprend une seule action de coordination : ac_{21}. La mise en correspondance des actions de coordination est la suivante :
. $ac_{11}Match = (1, ((R^2_{ref}, ac_{21}, num, 1)))$
. $ac_{21}Match = (2, ((R^1_{ref}, ac_{11}, num, 1)))$

Notons l'existence d'une symétrie : une des deux mises en correspondance pourrait de façon réaliste être automatiquement déduite de l'autre.

– Instanciation des MCS :
Cette instanciation nécessite la disponibilité des paramètres du modèle d'interaction. Ceux ci sont définis en même temps que le modèle d'interaction est lui-même défini (sous-section suivante). Nous préciserons seulement ici que ces paramètres sont au nombre de cinq, avec les types suivants :
– p_0 : type Obj
– p_1 : type Obj
– p_2 : type $Point$
– p_3 : type $Time$
– p_4 : type $UAVlist$

Voici donc les modèles de contraintes spatiales instanciés :
. mcs1 :
$cs_1 = (R*, 20, >)$

. mcs2 :
$cs_{21} = (p_0, 25, >)$
$cs_{22} = (p_0, 50, <)$
$cs_{23} = (p_0, p_1, AZIMUTH, 0, =)$
$cs_{24} = (p_0, p_1, ALTITUDE, \pi/6, =)$

. mcs3 :
$cs_{31} = (R1, 100, <)$
$cs_{32} = (p_2, 100, <)$

– Mise en relation explicite des MCS instanciés : nous les évoquons ici seulement par un nom symbolique, et nous préciserons ce à quoi ils correspondent au moment de détailler le contenu des automates d'exécution des rôles, dans la section V.4.
. Rôle R^1_{ref} : 8 blocs (b_{10}, b_{11}, ..., b_{17}) seront définis ici. Cependant, seulement 2 blocs sont sujets à des modèles de contraintes spatiales :
$b_{10} \rightarrow mcs1$, et
$b_{12} \rightarrow mcs2$.
. Rôle R^2_{ref} : 7 blocs (b_{20}, b_{21}, ..., b_{26}) seront définis ici. Cependant, seulement 2 blocs sont sujets à des modèles de contraintes spatiales :
$b_{20} \rightarrow mcs1$, et
$b_{22} \rightarrow mcs3$.

– Mise en correspondance des paramètres des rôles avec les paramètres du modèle d'interaction : il s'agit de définir une fonction de mise en correspondance des paramètres des rôles avec ceux du modèle d'interaction. Voici cette mise en correspondance :
– $R^1_{ref} : (front \rightarrow p_0, center \rightarrow p_1, ground\text{-}point \rightarrow p_2, duration \rightarrow p_3, uavs \rightarrow p_4)$
– $R^2_{ref} : (front \rightarrow p_0, center \rightarrow p_1, ground\text{-}point \rightarrow p_2, duration \rightarrow p_3, uavs \rightarrow p_4)$

V.2.5 Modèle d'interaction

Dans les paragraphes précédent, nous venons de définir formellement toutes les composants requis pour la définition d'un modèle d'interaction. Voici à présent la définition formelle du modèle d'interaction lui même.

Un modèle d'interaction s'exprime comme un ensemble V de variantes considérées comme équivalentes, une variante étant définie comme suit :

Définition V.5 : variante de modèle d'interactions

$v_i = (R_i, C_i, TC_i, P_i)$, où :
- R_i est un ensemble de rôles,
- C_i est un ensemble de contraintes spatiales,
- tc_i est la table de coordination de cette variante.
- P_i est un ensemble de paramètres à instancier.

Les paramètres P_i définissent le prototype de cette variante de MI : il doit exister une tâche jointe élémentaire de même prototype qui, transmise au GI, conduira à sélectionner cette variante. Ces paramètres correspondent aux différents paramètres des rôles en présence, donnés dans R_i. Il existe donc une fonction qui à chaque paramètre p d'un rôle r donné, $r \in R_i$, associe un paramètre $p^{'} \in P_i$: cette fonction est définie dans la table de coordination tc_i, qui rappelons-le est le "liant" d'une variante de modèle d'interaction.

Exemple de référence - partie 8 : modèle d'interaction

Nous définissons enfin, dans cette partie, le modèle d'interaction utilisé dans notre exemple de référence.
- Roles utilisés : R^1_{ref} et R^2_{ref}
- Modèles de contraintes spatiales utilisés : mcs_1, mcs_2, et mcs_3
- Table de coordination : tc_{ref}
- Paramètres de ce modèle d'interaction :

. p_0 : type Obj. Il s'agira, lors de l'instanciation, du point à cibler pour les perceptions, sur la ligne de feu. Il s'agira d'un objet disponible via la composante *observateur* du GI : cela signifie que les coordonnées correspondantes pourront évoluer, compte tenu des informations disponibles au niveau de l'observateur.

. p_1 : type Obj. Il s'agira, lors de l'instanciation, du point indiquant "l'intérieur" du feu. Lui aussi est un objet observé, susceptible d'évoluer.

. p_2 : type $Point$. Il s'agira, lors de l'instanciation, de la position du centre de contrôle. C'est un point statique, définit définitivement à l'initialisation.

. p_3 : type $Time$. Il s'agira, lors de l'instanciation, de la durée demandée pour la supervision.

. p_4 : type $UAVlist$. Il s'agira, lors de l'instanciation, de la liste d'UAV impliqués dans l'activité jointe. Cette liste sera utilisée dans les tâches de synchronisation.

Les paramètres définis ici sont mis en correspondance avec les paramètres respectifs des rôles, comme nous l'avons vu dans la table de coordination. Ainsi, l'instanciation de ces paramètres permet directement l'instanciation des paramètres du rôle à endosser en temps voulu, pour un UAV donné.

V.3 Modalités de négociation

Les modalités de négociation définissent des protocoles pour négocier avec les autres UAV du système, protocoles dont la portée, le niveau d'abstraction et de généralisation peuvent être très variables.

V.3.1 Conception et application de modalités de négociation

Les modalités de négociation sont des processus délibératifs permettant de réaliser des coordinations d'activité entre UAV : il peut s'agir de coordination spatiale et temporelle, aussi bien que de raisonnements concertés sur des ressources à partager pour la réalisation d'une activité conjointe. Il peut enfin s'agir de s'accorder sur des rôles à endosser dans le cadre d'une tâche jointe (sous la forme d'un modèle d'interaction dans le GI, nécessitant un certain nombre de roles).

C'est le déclenchement d'une *action de coordination*, lors du traitement de l'automate d'exécution d'un rôle, qui donne lieu à l'emploi d'une modalité de négociation, à travers une *session de négociation*. L'action de coordination comprend une référence à la modalité de négociation à employer, et la table de coordination permet de savoir quels rôles (et donc, lors de l'exécution, quels UAV) doivent prendre part à cette session.

Une modalité de négociation n peut être (en partie) formellement définie, de la façon suivante :

Définition V.6 : modalité de négociation

$n = (id,\ params_{in},\ params_{out}, [\ effects], oper)$, où :
- id est l'identifiant de la modalité, qui sert de référence pour les actions de coordination,
- $params_{in}$ est un ensemble de paramètres d'entrée, exploités dans les traitements associés à la modalité,
- $params_{out}$ est un ensemble de paramètres sortie : ces paramètres sont mis en relation avec les paramètres de l'action de coordination qui a fait appel à cette modalité de négociation. Les données peuvent alors être exploités dans le contexte d'appel.
- $effects$ (optionnel) décrit des sorties ou effet additionnels, qui peuvent être de deux types : nouvel automate, ou déclenchement d'un changement de rôle (voir ci-après).
- $oper$ est le mode opératoire de la modalité de négociation : c'est le cœur de la modalité, pour lequel nous ne donnerons par une définition entièrement formelle (voir ci-après).

Au niveau des $effects$, la modalité de négociation peut retourner un automate d'exécution, comparable à l'automate d'exécution d'un rôle : il est constitué de blocs et autre componels, et est instancié dans le contexte courant. Par ailleurs, un "effet spécial" peut également être déclenché : il s'agit en particulier de la question du changement dynamique de rôle (cas d'une modalité d'allocation de rôle).

Le mode opératoire soulève un point important : il s'agit de la spécification du protocole de communication à proprement parler, pour cette modalité de négociation. Nous avons choisi de laisser au libre arbitre du concepteur la création de modalités de négociation : selon le champ d'application, les besoins opérationnels, différents protocoles de communication et de négociation à proprement parler peuvent être développés. Il peut s'agir par exemple de négociations sur une base de CNP, à base d'économie de marché. Il peut s'agir aussi de protocoles de fusion de plan, pour coordonner des tâches comme des déplacements. Il peut s'agit d'une relation maitre – esclaves entre un robot donneur d'ordres et un ensemble de robots recevant ces ordres, ou bien au contraire une modalité de prise de décision complètement distribuée, sans leadership. Nous proposons en dernière partie de cette section quelques modalités intéressantes à développer dans le contexte d'un domaine et d'opérations comme celui du projet COMETS.

Les points communs à toutes les modalités de négociations sont le formalisme des entrées et sorties, et un support d'échange de messages basique, par lequel tout type d'information peut être échangé. En allant de l'avant, nous pouvons envisager d'enrichir l'expressivité des communications à travers un langage agent tel KQML [Finin 97], mais cela ne change en rien la volonté de laisser ouvert le mode opératoire de la modalité de négociation, qui est au cœur de l'interaction entre UAV. Celui-ci peut mettre en œuvre différents types de calculs à partir des données disponibles en entrée, et exploiter les capacités externes au gestionnaire d'interaction, à savoir le couple planificateur symbolique / raffineurs spécialisés (PS / RS), pour évaluer des tâches ou des plans. Pour résumer les possibilités de conception du mode opératoire d'une modalité de négociation, disons qu'un concepteur a accès aux moyens suivants :

– tous protocoles de communications destinés à résoudre des problèmes ou à coordonner des activités. Ces protocoles s'appuient sur des primitives de transmission et de récupération de données, que ce soient des données de contrôle du protocole (mise en relation, confirmation de réception...) ou des données véritablement informatives.

– appels au couple PS / RS pour évaluer (coût, temps) des tâches elles-mêmes, ou bien les conséquences de l'insertion de tâches dans le plan courant d'exécution (donc compte tenu du plan courant).

– tous calculs sur la base des données disponibles en paramètres d'entrée de la modalité de négociation, et sur la base des données récupérées suite à l'exploitation de moyens externes.

C'est le gestionnaire de négociations (GN) qui procède aux traitements des modalités de négociation, dans le cadre de sessions de négociation. Pour un UAV donné, lors du déclenchement d'une action de coordination, trois cas de figure peuvent se présenter :

– soit la session correspondante est déjà en cours, et l'UAV se joint alors à la session,

– soit il est le premier UAV à prendre part à la session, et dans ce cas il l'initie,

– soit, pour une raison quelconque, l'UAV est en retard pour la session, et celle-ci s'est déjà soldée par un échec chez les autres UAV.

Au cours d'une session de négociation, les nouveaux messages reçus de la part des autres UAV sont cycliquement récupérés et analysés dans le GN : les contenus des messages donnent alors lieu à des traitements propres à la modalité de négociation, qui peuvent en particulier donner lieu à des envois d'informations ou à de nouvelles requêtes (demandes d'informations) à destination des "coopérants".

Une session de négociation s'achève sur un commun accord de fin de session (succès) ou en cas d'interruption prolongée de communication entre certains membres de la coalition (échec).

Nous pouvons schématiquement représenter une modalité de négociation en cours de traitement de la façon illustrée sur la figure V.7.

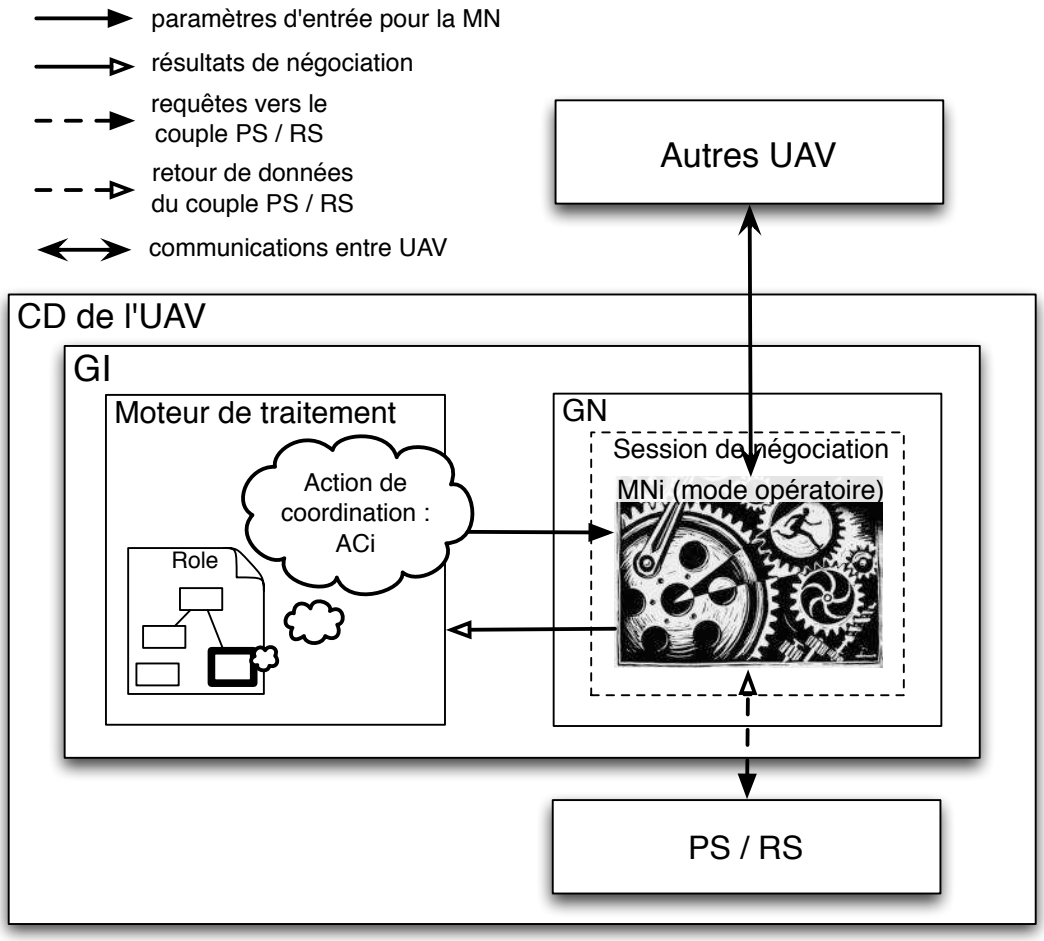


FIG. V.7 – Gestionnaire de négociations : traitement d'une modalité de négociation

V.3.2 Cas de l'allocation de rôle

Cette modalité a un statut particulier au sein des modalités de négociation : elle est toujours disponible et elle ne dépend pas du domaine d'application.

L'allocation de rôle consiste à déterminer, entre des coéquipiers, une configuration pertinente d'endossement des rôles. Il s'agit en conséquence d'être en mesure, pour chaque UAV, d'évaluer les rôles potentiellement endossables, d'émettre des préférences, et finalement de négocier en cas de conflit sur ces préférences.

Au final, cette modalité de négociation de rôle dispose du moyen d'interrompre l'automate d'exécution du rôle en cours, et de passer à l'exécution d'un autre rôle. C'est cette capacité qui

rend particulière cette modalité.

L'évaluation d'un rôle est une nécessité pour émettre des préférences vis à vis des rôles disponibles. Deux moyens sont appliqués successivement pour cela :

– **adéquation des capacités :** un rôle comprend dans sa définition un ensemble d'exigences (les conditions de compatibilité CComp_r, voir sectionV.2) relatives aux capacités de l'UAV : rayon de giration, type de caméra, etc. La vérification de l'adéquation est assez triviale, c'est un filtre qui peut potentiellement éliminer des rôles incompatibles avec les capacités de l'UAV.

– **estimation du coût d'application d'un rôle :** il s'agit d'analyser, pour chaque rôle, les implications (temps, coût, autres ressources...) compte tenu des modèles d'environnement et d'UAV, et vis à vis de l'état courant et du plan courant de l'UAV.
Cette estimation n'est pas triviale, dans la mesure où les modèles d'interaction et les rôles qu'ils mettent en œuvre sont destinés à des situations où des contingences peuvent subvenir : les "schémas de plans" introduits à travers les automates d'exécution des rôles sont en quelque sorte des plans conditionnels, dont le chemin de déroulement réel, et les données réellement prises en comptes (positions d'entités mobiles dans l'environnement...) sont inconnus jusqu'à la réalisation effective.
Plusieurs politiques peuvent être considérées pour l'évaluation d'un rôle : parmi les choix les plus simples, on peut envisager une fonction arbitraire et explicitite associée au rôle, et dépendant vraisemblablement des capacités du robot. A l'opposé, une simulation du comportement de l'UAV endossant ce rôle, et une évaluation des différentes instructions de ce rôle, peut aider à l'évaluation ; elle ne résoud toutefois pas le problème du manque d'information sur l'application effective du rôle, aussi longtemps que l'UAV n'est pas en train de le réaliser.
Nous proposons une méthode d'évaluation hybride (non encore implémentée) qui exploite les lieux géographiques supposés d'un rôle. Il s'agit de considérer un "chemin nominal" dans l'automate d'exécution d'un rôle : ce chemin pourrait correspondre à des indications arbitraires sur les instructions du rôle, fournies par le concepteur de ce rôle. Il est alors possible, pour les instructions rencontrées le long du chemin, de faire appel aux raffineurs spécialisés pour en évaluer le coût et la durée. Les données recueillies pour un rôle donné peuvent alors être traitées pour évaluer l'impact, compte tenu du plan courant, d'endosser ce rôle (compatibilité spatiale et temporelle...). Cette évaluation au niveau du plan aurait alors lieu au niveau du SCD, voir la section IV.4.
L'objectif de cette opération est d'être en mesure de définir un ordre total sur les rôles disponible en terme de préférence, et si possible, de quantifier le coût (ou son dual : l'utilité) d'endossement relatif à chacun des rôles.

Une fois l'ordre de préférence établi, il est alors possible au niveau des UAV d'appliquer un algorithme incrémental de négociations sur la base des préférences exprimées (choix collectif), voir par exemple les travaux de [Meyer 04].

Finalement, le retour de cette modalité particulière conduit soit à l'interruption immédiate de l'exécution de l'automate du rôle courant (nouveau rôle à endosser), soit à poursuivre, de façon transparente, le rôle courant (pas de changement de rôle).

V.3.3 Cas de la négociation d'intervalle temporel

De la même manière que la modalité d'allocation de rôle, cette modalité est toujours disponible et ne dépend pas du domaine d'application. Elle est également appliquée systématiquement, lors du chargement d'un modèle d'interaction dans le GI.

Il s'agit pour les UAV de s'accorder sur un intervalle temporel, compte tenu des préférences respectives. Ces préférences sont fournies par le superviseur de CD au moment de la transmission d'une TJE, sous la forme d'une liste d'intervalles.

Différents moyens de sélection du "meilleur" intervalle peuvent être envisagés, centralisé (toutes les préférences respectives sont transmises à l'un des UAV qui désigne la meilleure solution), ou distribuée (un protocole adéquat doit être spécifié pour s'assurer d'une convergence dans les négociations).

Au final, dans le cas d'un chargement de modèle d'interaction, l'intervalle sélectionné est retourné aux superviseurs de CD respectifs.

V.3.4 D'autres modalités...

Nous évoquons ici d'autres types de modalités de négociation susceptibles d'être utilisées dans des missions réalistes :

– Partage de zone dans une mission de "couverture" : dans le cadre d'une activité de cartographie, par exemple, il est intéressant de distribuer la zone à cartographier entre un certain nombre d'UAV. Notons que nous avons étudié ce problème particulier dans le cadre applicatif d'un travail réalisé pendant un séjour de trois mois à l'université de Linköping, en Suède. Nous en présentons les résultats et conclusions de ces travaux en annexe B.

– Coordination de positions de perception, pour optimiser une tâche de surveillance. Il s'agit ici pour les UAV de se concerter pour appliquer une stratégie permettant d'optimiser la couverture simultanée de perception, compte tenu d'une surveillance à réaliser. Il peut par exemple s'agir de surveiller le déplacement une voiture, en anticipant sur les directions qu'elle est le plus susceptible d'emprunter.

– Coordination d'une chaîne de relais de communication. Il s'agit de déterminer quels UAV jouent le rôle de quels maillons dans une chaîne de communication, sachant que cette chaîne peut être amenée à évoluer dans l'espace, et que des UAV de la chaîne peuvent être amenés à rompre la chaîne (autre activité plus urgente).

– Fusion de plan pour des activités fortement couplées. Il s'agit de synchroniser des plans en termes spatial et temporel, en ajoutant judicieusement des synchronisations dans les plans de tâches respectifs des UAV. Les plans peuvent ainsi être incrémentalement coordonnés (pour des détails sur l'approche de fusion de plan, voir par exemple **??**).

Exemple de référence - partie 9 : modalité de négociation

Dans notre exemple de référence, nous utilisons la modalité de réallocation de rôle : $n_{realloc} = (1,\ (),\ (), apply\text{-}realloc,\ oper)$,

avec un mode opératoire évaluant les rôles disponibles, et permettant de s'accorder avec les autres UAV sur le rôle respectif à adopter. Aucun paramètre explicite d'entrée ou de sortie n'est nécessaire ici : les données manipulées par la modalité sont des "méta-données", à savoir les rôles à pourvoir dans ce modèle d'interaction.

Nous utilisons également la modalité de négociation d'intervalle temporelle : $n_{timeInterval} = (2,\ (preferences),\ (interval), (),\ oper)$,

avec en entrée les préférences transmises depuis le TJE, en sortie l'intervalle choisi, et avec un mode opératoire permettant la sélection de l'intervalle le plus satisfaisant.

Dans l'exemple de référence, nous ne précisons pas le détail des modes opératoires.

V.4 Mécanique de traitement des modèles d'interaction

V.4.1 Etape préliminaire : chargement d'un modèle d'interaction

Cette étape préliminaire survient sur requête de la part du superviseur de CD : lorsqu'un plan contenant des tâches jointes est produit, le superviseur demande au gestionnaire d'interactions de charger un modèle d'interaction par tâche jointe. L'intitulé de la tâche jointe, ainsi que les paramètres associés (et en particulier les identifiants des UAV impliqués dans la réalisation de cette TJE) sont alors transmis au gestionnaire d'interaction.

Lors de la réception d'une requête de chargement, le gestionnaire d'interactions associe à la TJE un modèle d'interaction (l'ensemble des MI et l'ensemble des TJE étant, rappelons le, bijectif). Une variante de modèle d'interaction est choisie : dans l'idéal, le choix devrait dépendre de la population d'UAV impliquée dans la réalisation de la tâche jointe.

Une fois la variante déterminée, il s'agit de sélectionner un rôle parmi les rôles proposés dans cette variante. Une session de négociation avec la modalité d'allocation de rôle est immédiatement déclenchée, pour négocier les rôles avec tous les UAV impliqués. Dans le cadre de cette allocation initiale, aucun timeout de négociation n'est fixé a priori : à ce niveau, la seule situation problématique à court terme est le déclenchement proprement dît de l'exécution de la tâche jointe. Si la session de négociation pour l'allocation initiale de rôle n'aboutit pas avant cet événement, alors la tâche jointe est considérée comme ayant échoué.

L'allocation de rôle nécessite une évaluation des conséquences de l'endossement des différents rôles disponibles (voir le paragraphe V.3.2, sur la modalités de négociation pour l'allocation de rôle).

Cependant, il faut noter que les informations disponibles pour l'évaluation sont peu précises, dans la mesure où cette étape à lieu antérieurement à la réalisation proprement dîte de la TJE. Par la suite, la réallocation de rôle dans le contexte d'exécution est donc susceptible d'améliorer la pertinence de l'allocation des rôles, le cas échéant.

Une fois un rôle endossé, les paramètres du rôle chargé sont mis en correspondance avec les données disponibles : soit des données observées à travers l'observateur (voir la figure V.1), soit

des paramètres transmis avec la requête de chargement.

Les UAV doivent maintenant s'accorder sur le meilleur moment pour commencer cette tâche jointe.

Comme nous l'avons précisé dans le chapitre IV, le chargement d'une TJE (opérateur SCD-4 dans le superviseur de CD) s'accompagne d'une liste de préférences de créneaux temporels : il s'agit donc de négocier, entre les UAV concernés, le créneau le plus satisfaisant. La modalité de négociation d'intervalle temporel est employée pour cela. L'intervalle sélectionné est alors retourné au superviseur de CD.

Le modèle d'interaction, à travers la variante choisie et chargée, est alors prêt à être appliqué, au moment où la TJE correspondante sera exécutée dans le plan de tâches, au niveau de l'EMD.

Exemple de référence - partie 10 : chargement du modèle d'interaction

Dans cette partie, nous procédons au chargement du modèle d'interaction auprès de UAV1 et UAV2. Les tâches jointes élémentaires (respectivement t2.1 et t5.1) dans les plans des UAV (voir le chapitre IV) conduisent les superviseurs de CD à transmettre une requête de chargement vers les GI des couches délibératives respectives. Les TJE sont alors mis en relation avec le modèle d'interaction que nous avons défini pour notre exemple de référence (MI_{ref}), et le modèle est donc instancié avec les paramètres des TJE (voir dans le chapitre IV la partie 2 de l'exemple de référence) :

- $p_0 \rightarrow fire\text{-}front$,
- $p_1 \rightarrow fire\text{-}center$,
- $p_2 \rightarrow control\text{-}center$,
- $p_3 \rightarrow (UAV1, UAV2)$,
- $p_4 \rightarrow 900$,

Les UAV entrent dans une session de négociation pour l'allocation initiale des rôles. Les conditions de compatibilité respectives de rôles sont adéquates pour chacun de UAV : le choix passe donc par une évaluation des rôles à pourvoir. Les lieux d'activité respectifs des rôles sont déterminés : compte tenu des distances courantes des UAV, ceux-ci s'accordent sur l'allocation : $UAV1 \rightarrow R^1_{ref}$, et $UAV2 \rightarrow R^2_{ref}$

La négociation de l'intervalle pour la réalisation de la TJE conduit à fixer le début de la TJE à 9 :00 :30 (même préférences respectives pour les deux UAV, dans cet exemple : $[9:00:30; max - 930[$).

Les GI respectifs sont prêt à traiter les rôles, lorsque les TJE seront prêtes, dans les MLE respectifs.

V.4.2 Activation et traitement d'un modèle d'interaction chargé

L'exécution de la tâche jointe dans l'EMD donne lieu, au niveau du superviseur de CD, à l'envoi d'une requête "START" pour le modèle d'interaction chargé. Celui-ci devient alors actif.

L'activation produit aussitôt un évènement "`running`", transmis au superviseur de la DL, puis à l'EMD. Cet évènement est interprété à ce niveau comme l'événement `running` de la tâche jointe, jusqu'à présent dans un état `scheduled`.

L'activation d'une tâche jointe dans le gestionnaire d'interactions se traduit par le mise en route de l'automate défini dans le rôle pré-sélectionné. Ce traitement est opéré par le *moteur de traitement* du GI (voir le schéma V.1, section V.1).

Modèles des componels

Avant de décrire ce mécanisme de traitement de l'automate d'exécution, nous définissons maintenant le formalisme des componels, qui est étroitement lié aux processus utilisés pour ce traitement.

Les **tâches** décrivent des actions à réaliser, à un niveau qui peut être très variable : elles correspondent à des tâches de haut niveau, individuelle ou jointe. Dans un automate d'exécution d'un rôle, une tâche est définie de la façon suivante :

Définition V.7 : tâche dans automate d'exécution

$t = (t_{id},\ t_{type},\ b_{id},\ params_{types})$, où :
- t_{id} est l'identifiant unique de la tâche,
- t_{type} est le type de la tâche : les types disponibles correspondent aux tâches de haut niveau et tâches élémentaires (individuelles ou jointes) qui peuvent être traitées par le couple planificateur symbolique, dans la couche délibérative.
- b_{id} est l'identifiant du bloc qui contient cette tâche.
- $params_{types}$ est le prototype des paramètres nécessaires pour le traitement de cette tâche : c'est au moment du traitement du rôle dans lequel cette tâche est définie que les paramètres vont être réellement affectés, dans le cadre d'exécution courant.

De façon assez similaire aux tâches, une **action de coordination** est définie de la façon suivante :

Définition V.8 : action de coordination

$ac = (ac_{id},\ ac_{type},\ b_{id},\ params_{in},\ params_{out})$, où :
- ac_{id} est l'identifiant unique de l'action de coordination,
- ac_{type} est le type de l'action de coordination : les types disponibles correspondent aux modalités de négociations existantes.
- b_{id} est l'identifiant du bloc qui contient cette action de coordination.
- $params_{in}$ est le prototype des paramètres nécessaires pour le traitement de cette action de coordination : c'est au moment du traitement que les paramètres de cette action de coordination vont être réellement affectés, dans le cadre d'exécution courant.
- $params_{out}$ Ces paramètres seront modifiés suite à l'application de cette action de coordination. Les paramètres en question sont définis localement, dans le cadre de blocs : ils sont formellement définis dans la description du modèle de bloc, sous l'appelation *paramètres localement calculés* (PLC), dans la section V.4

Voici finalement la définition formelle d'un **bloc** dans ce cadre :

Définition V.9 : bloc

$b = (id, type, endMode, CS, PreC, ExitC, PL, content)$, où :
- *id* est un identifiant unique pour ce bloc,
- *type* est le type du bloc parmi deux possibilités : `loop` ou `plan`,
- *endMode* est l'effet de la fin de ce bloc sur le bloc supérieur : `immediate` ou `wait`,
- *CS* est un ensemble de modèles de contraintes spatiales qui doivent être satisfaits aussi longtemps que le bloc est actif.
- *PreC* est un ensemble de conditions dont la satisfaction déclenche l'activation du bloc.
- *ExitC* est un ensemble de conditions dont la satisfaction provoque l'interruption du bloc, ou le rend inactivable si celui-ci n'est pas actif.
- *PL* est une liste de paramètres locaux au bloc : il n'ont pas d'existence en dehors du bloc.
- *content* est l'ensemble des componels directement contenus dans (et accessibles depuis) *b*.

Le *type* permet de spécifier des blocs fonctionnant en boucle (`loop`) : dans ce cas, seule la satisfaction des conditions de sortie peut mettre fin à l'activité du bloc. Autrement, lorsque le contenu du bloc est achevé, le bloc se ré-active immédiatement.

Le *endMode* précise l'effet de la fin de ce bloc sur un bloc de niveau supérieur. Si le mode est `immediate`, alors le bloc supérieur passe lui même dans un état de terminaison, en interrompant au besoin les autres componels non-encore terminés qu'il contient. Dans le cas `wait`, la terminaison du bloc ne suffit à déclencher la fin du bloc supérieur : il faut soit qu'un autre bloc avec le mode `immediate` se termine, soit que tous les blocs inclus soient terminés.

CS désigne un ensemble de modèles de contraintes spatiales parmi les modèles notifiés dans la table de coordination. Les modèles sont continuellement vérifiés aussi longtemps que le bloc est actif. Si des contraintes spatiales sont violées, l'exécution courante du bloc est interrompue. Il est alors envisageable d'exploiter l'évènement correspondant comme déclencheur (préconditions) d'un autre bloc, afin de réaliser des traitements pertinents, le cas échéant.

PreC correspond aux préconditions à satisfaire pour activer le bloc. Les préconditions sont exprimées de la même manière que les préconditions de tâches dans l'EMD (section III.1.1) : sous forme normale conjonctive (FNC). Cette forme permet en effet d'exprimer des combinaisons plus riches de conditions à satisfaire, qu'une simple conjonction élémentaire de conditions.

ExitC correspond aux conditions de sortie, dont la satisfaction inactive ou empêche l'activation du bloc. Elles sont elles-aussi exprimées en tant que FNC.

PL permet de spécifier des paramètres "locaux", qui peuvent être calculés à travers l'application d'actions de coordination. Il est possible de fournir, pour un paramètre de ce type, une méthode d'initialisation qui fournit au paramètre une valeur calculée au moment où les préconditions du bloc sont vérifiées. Une telle méthode doit être spécifiée par un utilisateur du système. Elle est typée, et peut exploiter les paramètres locaux disponibles dans les blocs supérieurs et les paramètres (non instanciés) définis dans le rôle. Nous parlerons de "paramètre calculé localement".

content regroupe les componels contenus et accessibles depuis ce bloc, que ce soient d'autres blocs, des tâches ou des actions de coordination.

Nous explicitons à présent le formalisme de préconditions et conditions de sortie employé dans notre définition du bloc.

Les préconditions et les conditions de sortie sont exprimées de façon assez semblable. En fait, une condition de sortie doit en plus définir un "effet", dans le cas où elle est définie pour un bloc de type `loop`. Cet effet peut être une sortie définitive du bloc (`exit`) ou bien une interruption suivie d'un nouveau lancement (`continue`).

Voici les formalismes respectifs des préconditions et conditions de sortie :

Définition V.10 : précondition et condition de sortie

$preC = (ev, m)$,
$exitC = (ev, m, effet)$, où :
- ev est un événement (voir ci-après),
- m est la modalité de condition : `optionnel` ou `obligatoire`,
- $effet$ est l'effet engendré sur un bloc de type `loop` : `exit` ou `continue`. Dans le cas d'un bloc `plan`, l'effet est nécessairement `exit`.

Nous définissons le détail des différents types d'événements ci-après. La modalité d'une condition est comparable aux modalités de condition définies dans l'EMD (voir le chapitre III) : elle spécifie la façon d'évaluer la condition si il advient que l'événement associé ne peut plus se réaliser : par exemple l'état `ended` d'une tâche est attendu, et la tâche passe de l'état `running` à l'état `aborted`. Dans le cas où m = `optionnel`, la non-satisfiabilité d'un événement est assimilé à sa satisfaction : autrement dit, dans ce cas, on considèrera la condition comme satisfaite. Dans le cas où m = `obligatoire`, la condition sera considérée comme non-satisfaite et non-satisfiable. De la même manière que pour les conditions exprimées dans l'EMD, cette modalité de condition est pertinente pour des configurations particulières de dépendances entre tâche (voir la tâche LAND sur le schéma III.4).

Finalement, l'application de l'effet `exit`/`continue` dans les blocs `loop` est expliquée plus en détail dans le mécanisme proprement de traitement proprement dit (ci-après).

Les événements considérés dans les conditions sont de trois sortes :

1. Les événements d'évolution de statut liés au traitement d'une instruction (au sens large, c'est à dire une componel) : il peut s'agir d'une tâche, mais aussi d'une action de coordination, ou même d'un bloc au sein du gestionnaire d'interactions.

   La catégorie d'événement d'évolution de statut est formellement définie de la façon suivante :

Définition V.11 : événement d'évolution de statut

$ev_{ees} = (id_{ev}, id_{instr}, state)$, où :

- id_{ev} est l'identifiant attribué à cet événement,
- id_{instr} est l'identifiant de l'instruction ciblée,
- $state$ est l'état de traitement de l'instruction.

2. Les événements liés à des messages : réception et lecture d'un message, dont l'origine peut être interne (message en provenance d'un des composants de la CD, ou même du GI), ou externe (provenance d'un autre UAV, du CC, etc...).
   Les états possibles de traitement d'une instruction sont assez semblables à ceux définis dans le chapitre III relatif à l'EMD. Ces états sont :

   - `scheduled` : planifié, potentiellement déclenchable à tout moment
   - `running` : confirmation de l'activité de l'instruction
   - `aborting` : considérée comme sur le point d'être interrompue, mais pas encore confirmé.
   - `aborted` : confirmation d'interruption.
   - `ended` : confirmation de fin nominale.

   Lors du déroulement nominal d'une instruction, elle passe successivement par les états `scheduled`, `running`, `ended`. En cas de déroulement non nominal, les états `scheduled` et `running` peuvent être suivis des états `aborting` et (ou) `aborted`.

   La catégorie d'événement de message est formellement définie de la façon suivante :

   Définition V.12 : événement de message

   $ev_{em} = (id_{ev}, state, content)$, où :
   - id_{ev} est l'identifiant attribué à cet événement,
   - $state$ est l'état de traitement du message (`pending` / `read`),
   - $content$ est le contenu du message (sous la forme d'une chaîne de caratères)

   Un message nouvellement reçu génère un événement dont l'état est PENDING, soit non-encore lu. Le traitement effectif du message reçu génère quant à lui un événement READ.

3. Les événements temporels : liés à l'écoulement du temps, dans le référentiel du robot.

   Finalement, la catégorie d'événement temporel est formellement définie de la façon suivante :

Définition V.13 : événement temporel

$ev_{et} = (id_{ev}, state, val, absOrRel)$, où :
- id_{ev} est l'identifiant attribué à cet événement,
- $state$ est l'état de l'événement (`occured` / `not-yet-occured`),
- val définit la valeur horaire à considérer,
- $absOrRel$ précise si le temps précisé est absolue (référentiel général commun aux différents UAV) ou relatif (durée écoulée depuis le début de traitement d'un modèle d'interaction).

Remarque : Il pourrait également être intéressant de considérer des temps écoulés depuis l'occurence d'un événement tiers : pour cela, il serait nécessaire de marquer les événements de leur date d'occurence.

Les componels que nous venons de définir sont donc les briques permettant de bâtir les automates d'exécution des rôles en faisant abstraction du cadre précis d'application.

Les paramètres des componels sont mis en relation avec l'ensemble des paramètres $\mathsf{P_r}$ du rôle. Autrement dit, les paramètres du rôles correspondent à l'intersection des paramètres d'entrée des componels de l'automate d'exécution du rôle : de cette façon, lorsque le rôle est endossé par un UAV, et appliqué dans un contexte donné, l'instanciation des paramètres du rôle produit automatiquement l'instanciation des paramètres des componels de son automate d'exécution.

Le principe d'instanciation des paramètres d'un rôle sera précisé par la suite, en présentant la mécanique globale de l'IM (section V.4).

Exemple de référence - partie 11 : détail des automates d'exécution

Nous précisons ici intégralement le contenu des automates d'exécution respectifs des deux rôles.

La notion de bloc est essentielle dans la définition de cet automate. Comme nous l'avons évoqué dans la partie 7 de l'exemple de référence, les rôles R^1_{ref} et R^2_{ref} comprennent chacun 6 blocs.

Afin d'aider à la compréhension des structures des automates d'exécution des rôles, nous proposons des illustrations respectives automates d'exécution de R^1_{ref} et R^2_{ref} en annexe (voir C).

Automate d'exécution du rôle R^1_{ref}

Pour ne pas alourdir l'expression des données, nous employons ici une description semi-formelle seulement (en particulier, nous ne détaillons pas jusqu'au formalisme des événements).

Tout d'abord, nous définissons tous les blocs de cet automate d'exécution :

1. $b_{10} = (10,$ `plan`, `immediate` $, (mcs_1), (), ExitC_{10}, (pl_{10}), (t_{11}, b_{11}, b_{12}, b_{16}, b_{17}))$
2. $b_{11} = (11,$ `plan`, `wait` $, (), (PreC_{11}), (), (), (t_{12}))$
3. $b_{12} = (12,$ `loop`, `immediate` $, (mcs_2), PreC_{12}, (), (), (b_{13}, b_{14}, b_{15}))$

4. $b_{13} = (13, \texttt{plan}, \texttt{wait}, (), PreC_{13}, (), (), (ac_{11}))$
5. $b_{14} = (14, \texttt{plan}, \texttt{wait}, (), (), ExitC_{14}, (), (t_{13}))$
6. $b_{15} = (15, \texttt{plan}, \texttt{immediate}, (), PreC_{15}, (), (pl_{15}), (t_{14}))$
7. $b_{16} = (15, \texttt{plan}, \texttt{wait}, (), PreC_{16}, (), (), (t_{15}))$
8. $b_{17} = (15, \texttt{plan}, \texttt{immediate}, (), PreC_{17}, (), (), (t_{16}))$

Les modèles de contraintes spatiales mcs_1, mcs_2 et mcs_3 ont déjà été définis. Pour les autres éléments, tout doit être défini.

Rappelons que les paramètres du rôle R^1_{ref} sont les suivants :

$\mathsf{P_r} = (< Obj > front, < Obj > center, < Point > ground\text{-}point, < Time > duration, < UAVlist > uavs)$

Dans b_{10} :

– $ExitC_{10}$: il s'agit d'une condition de sortie pour ce bloc. Cette condition est de type message, et spécifie que la réception d'un message avec un contenu indiquant une interruption provoque l'interruption du bloc.

– pl_{10} : il s'agit d'un paramètre local défini dans le cadre du bloc. Il est initialisé par une fonction calculant la meilleure position de perception : *compPercepLoc*$(< Point >, < Point >)$. Cette fonction va faire appel aux raffineurs, pour déterminer la meilleure position pour réaliser la perception. Nous ne définissons pas ici le détail de cette fonction, afin de ne pas alourdir l'exemple. En sortie, un point est retourné.

– t_{11} : tâche de déplacement *goto*, prenant en paramètre la destination. Cette destination est pl_{10}. Nous avons donc :
$t_{11} = (11, \texttt{goto}, b_{10}, (pl_{10}))$

Dans b_{11} :

– $PreC_{11}$: il s'agit d'une condition portant sur l'événement de fin de tâche t_{11} (événement d'évolution de statut). Il s'agit autrement dit de la définition d'une précédence.

– t_{12} : tâche de a but de synchronisation (*wait-until-synchro*). Avec le formalisme défini pour les componels de type tâche, nous avons donc :
$t_{12} = (12, \texttt{wait-until-synchro}, b_{11}, ((uavs), (uavs)))$,
où le dernier champ correspond aux paramètres de la synchronisation (liste d'UAV émetteurs, et liste d'UAV récepteurs du message de synchronisation, voir chapitre III).

Dans b_{12} :

– $PreC_{12}$: il s'agit d'une condition portant sur l'événement de fin de bloc t_{11} (événement d'évolution de statut).

Dans b_{13} :

– $PreC_{13}$: il s'agit d'une condition portant sur un énénement de message : en l'occurence, sur le message généré si le modèle de contrainte spatiale mcs_3 venait à être violé.

– ac_{11} : il s'agit d'une action de coordination faisant appel à la modalité de négociation de changement de rôle, que nous avons défini auparavant dans notre exemple de référence :
$ac_{11} = (11, n_{realloc}, b_{13}, (), ())$

Dans b_{14} :

– $ExitC_{14}$: il s'agit d'une condition de sortie portant sur l'événement de fin du bloc b_{13}.

– t_{13} : tâche de haut niveau `takeshot-while-waiting`, qui permet la prise de vue pendant un vol stationnaire. Elle prend en paramètre la durée requise pour l'activité jointe :
$t_{13} = (13, \texttt{takeshot-while-waiting}, b_{14}, (p_3))$

Dans b_{15} :
– $PreC_{15}$: il s'agit d'une précondition portant sur l'événement de fin du bloc b_{13}.
– pl_{15} : il s'agit d'un paramètre local défini dans le cadre du bloc. Il est initialisé par une fonction calculant une position valide, compte tenu des contraintes spatiales courantes : *compSCGoodLoc*(). La fonction fait appel pour cela aux capacités du vérificateur de contraintes spatiales (VCS), afin de déterminer une position spatiale valide. Un point est retourné.
– t_{14} : tâche de déplacement *goto*, prenant en paramètre la destination. Cette destination est pl_{15}. Nous avons donc :
$t_{14} = (14, \texttt{goto}, b_{15}, (pl_{15}))$
Dans b_{16} :
– $PreC_{16}$: il s'agit d'une précondition portant sur l'événement de fin du bloc b_{12}.
– t_{15} : tâche de synchronisation *synchro* :
$t_{15} = (15, \texttt{synchro}, b_{16}, ((p_4), (p_4)))$
Dans b_{17} :
– $PreC_{17}$: il s'agit d'une précondition portant sur l'événement de fin du bloc b_{16}.
– t_{16} : tâche de déplacement *goto*, pour objectif le point de départ ground-point (tel qu'il a été fourni en paramètre du modèle d'interaction) :
$t_{16} = (16, \texttt{goto}, b_{17}, (ground - point))$

Automate d'exécution du rôle R^2_{ref}

Voici tout d'abord les blocs de cet automate d'exécution :

1. $b_{20} = (20, \texttt{plan}, \texttt{immediate}, (mcs_1), (), ExitC_{20}, (pl_{20}), (t_{21}, b_{22}, b_{23}, b_{26})$
2. $b_{21} = (21, \texttt{plan}, \texttt{wait}, (), (PreC_{21}), (), (), (t_{22}))$
3. $b_{22} = (22, \texttt{loop}, \texttt{immediate}, (mcs_3), PreC_{22}, (), (), (b_{24}, b_{25}))$
4. $b_{23} = (23, \texttt{loop}, \texttt{wait}, (), PreC_{23}, (), (), (ac_{21}))$
5. $b_{24} = (24, \texttt{plan}, \texttt{immediate}, (), PrecC_{24}, (), (pl_{24}), (t_{23}))$
6. $b_{25} = (25, \texttt{plan}, \texttt{immediate}, (), (), (), (), (t_{24}))$
7. $b_{26} = (26, \texttt{plan}, \texttt{immediate}, (), PreC_{26}, (), (), (t_{25}))$

Les modèles de contraintes spatiales mcs_1 et mcs_3 ont déjà été défini. Pour les autres éléments, tout doit être défini.

Rappelons que les paramètres du rôle R^2_{ref} sont les suivants :

$\mathsf{P_r} = (< Obj > front, < Obj > center, < Point > ground\text{-}point, < Time > duration,$
$< UAVlist > uavs)$
Dans b_{20} :
– $ExitC_{20}$: il s'agit d'une condition de sortie pour ce bloc. Cette condition est de type message, et spécifie que la réception d'un message avec un contenu indiquant une interruption provoque l'interruption du bloc.
– pl_{20} : il s'agit d'un paramètre local défini dans le cadre du bloc. Il est initialisé par une fonction calculant la meilleure position de relais de communication : *compRelayLoc*$(< Point >, < Point >)$. Cette fonction va faire appel aux raffineurs, pour déterminer la

meilleure position pour réaliser le relais. Nous ne définissons pas ici le détail de cette fonction, afin de ne pas alourdir l'exemple. En sortie, un point est retourné.

– t_{21} : tâche de déplacement *goto*, prenant en paramètre la destination. Cette destination est pl_{20}. Nous avons donc :
$t_{21} = (21, \texttt{goto}, b_{20}, (pl_{20}))$

Dans b_{21} :

– $PreC_{21}$: il s'agit d'une condition portant sur l'événement de fin de tâche t_{11} (événement d'évolution de statut). Il s'agit autrement dit de la définition d'une précédence.

– t_{22} : tâche de a but de synchronisation (*wait-until-synchro*). Avec le formalisme défini pour les componels de type tâche, nous avons donc :
$t_{22} = (22, \texttt{wait-until-synchro}, b_{21}, ((p_4), (p_4)))$,
où le dernier champ correspond aux paramètres de la synchronisation (liste d'UAV émetteurs, et liste d'UAV récepteurs du message de synchronisation, voir chapitre III).

Dans b_{22} :

– $PreC_{22}$: il s'agit d'une condition portant sur l'événement de fin de bloc t_{21} (événement d'évolution de statut).

Dans b_{23} :

– $PreC_{23}$: il s'agit d'une condition portant sur un énénement de message : en l'occurence, sur un message en provenance de l'autre rôle, et demandant la participation à une session de négociation.

– ac_{21} : il s'agit d'une action de coordination faisant appel à la modalité de négociation de changement de rôle, que nous avons défini auparavant dans notre exemple de référence :
$ac_{21} = (21, n_{realloc}, b_{23}, (), ())$

Dans b_{24} :

– pl_{24} : il s'agit d'un paramètre local défini dans le cadre du bloc. Il est initialisé par une fonction calculant une position valide, compte tenu des contraintes spatiales courantes : *compSCGoodLoc*$()$. La fonction fait appel pour cela aux capacités du vérificateur de contraintes spatiales (VCS), afin de déterminer une position spatiale valide. Un point est retourné.

– t_{23} : tâche de déplacement *goto*, prenant en paramètre la destination. Cette destination est pl_{24}. Nous avons donc :
$t_{23} = (23, \texttt{goto}, b_{24}, (pl_{24}))$

Dans b_{25} :

– t_{24} : tâche de a but de synchronisation (*wait-until-synchro*). Avec le formalisme défini pour les componels de type tâche, nous avons donc :
$t_{24} = (24, \texttt{wait-until-synchro}, b_{25}, ((p_4), (p_4)))$,
où le dernier champ correspond aux paramètres de la synchronisation (liste d'UAV émetteurs, et liste d'UAV récepteurs du message de synchronisation, voir chapitre III).

Dans b_{26} :

– $PreC_{26}$: il s'agit d'une précondition portant sur l'événement de fin du bloc b_{22}.

– t_{25} : tâche de déplacement *goto*, pour objectif le point de départ ground-point (tel qu'il a été fourni en paramètre du modèle d'interaction) :
$t_{25} = (25, \texttt{goto}, b_{26}, (ground - point))$

Mécanique de traitement des automates d'exécution

Suite à cette définition des componels, nous précisons à présent la mécanique de traitement d'un automate d'exécution de rôle dans le moteur de traitement.

Celui-ci est doté d'une routine d'écoute des événements : une base d'événements reçoit continuellement les nouveaux événements. Ceux-ci sont susceptibles de satisfaire des conditions de déclenchement de blocs (préconditions) ou au contraire des conditions d'interruption/annulation de blocs, au sein de l'automate d'exécution.

Dans son fonctionnement, le moteur de traitement considère en permanence deux listes de blocs : une liste dont les préconditions ne sont pas encore satisfaites (blocs inactifs, ou BI), et une liste en cours de traitement, i.e. dont les préconditions ont été satisfaites (blocs actifs, ou BA). Le moteur de traitement procède de façon cyclique à trois étapes de traitements sur ces listes de blocs, en considérant à chaque cycle les nouveaux évènements survenus depuis le cycle précédent :

1. **Vérification des conditions de sortie des blocs de BI** : les blocs de BI dont les conditions de sortie sont satisfaites sont retirés de BI, et archivés. Les blocs concernés ne seront donc jamais activé ni exécutés.

2. **Vérification des conditions de sortie des blocs de BA** : les blocs de BA dont les conditions de sortie sont satisfaites sont traités de la façon suivante. Dans le cas d'un bloc "`loop`", deux cas de figure se présentent :
   - si toutes les conditions ayant conduit à la satisfaction des conditions des sorties sont de type `continue`, alors le bloc n'est pas retiré de la liste BI. Cependant, tous les componels contenus dans le bloc sont interrompus, puis le bloc est ré-activé (voir ci-après pour la définition de l'activation d'un bloc).
   - si une au moins des conditions ayant conduit à la satisfaction des conditions de sortie est de type `exit`, alors le bloc est désactivé (voir ci-après, pour la définition de la désactivation d'un bloc).

   Dans le cas d'un bloc "`plan`", la satisfaction des conditions de sortie conduit directement à la désactivation du bloc.

3. **Vérification des préconditions des blocs de BI** : les blocs de BI dont les préconditions sont vérifiées sont transférés à la liste BA, avant d'être activés.

Cet ordre de traitement a une certaine importance : il permet d'éviter des redondances de vérifications et traitements. En effet, la première étape élimine les blocs de façon anticipée, évitant une possible activation suivie d'une interruption (cas où les préconditions et les conditions de sorties sont simultanément satisfaites). Par ailleurs, le fait de réaliser la seconde étape avant la troisième permet d'éviter de balayer les conditions de sorties de blocs devenus actifs, alors qu'elles ont déjà été vérifiées lors de la première étape (et dans ce cas de figure, elles n'étaient pas satisfaites). En revanche, la première et deuxième étape pourraient être permutées sans conséquence. Les différents traitements relatifs à ces trois étapes sont illustrés sur les figures V.8 et V.9.

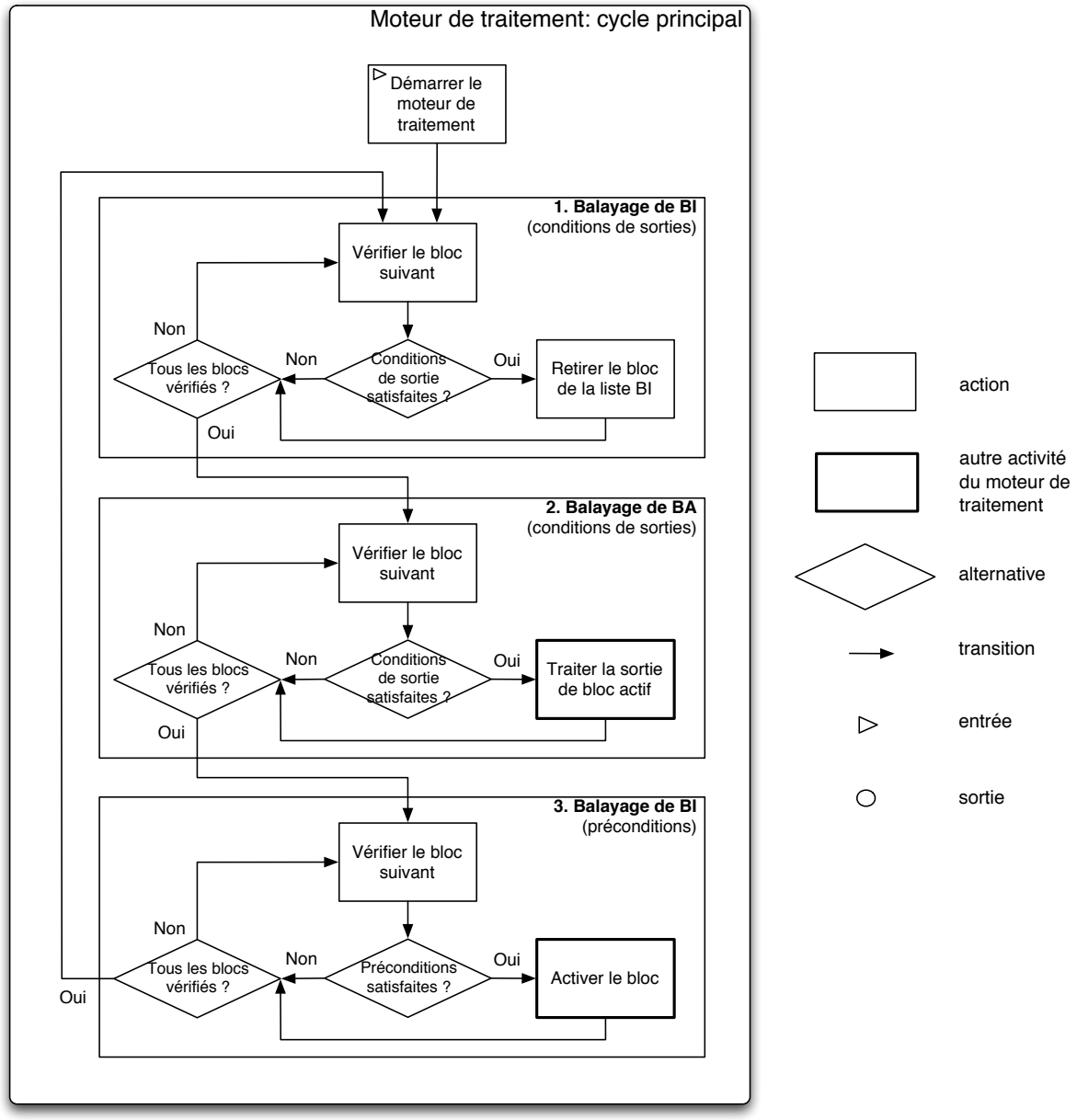


FIG. V.8 – Moteur de traitement : cycle de base

L'activation d'un bloc consiste d'une part à produire un évènement `running` relatif à ce bloc, et d'autre part à appliquer les traitements appropriés à son contenu (différents *componels*) :

– Cas d'un bloc : celui-ci est placé dans la liste BI, en attendant d'être traité dans le cycle suivant du gestionnaire d'interactions. En parallèle, l'évènement `scheduled` de ce bloc est généré.

– Cas d'une tâche : une demande de traitement de tâche est transmise au superviseur de CD. Si la tâche est déjà raffinée, le superviseur de CD transmet directement une requête d'exécution

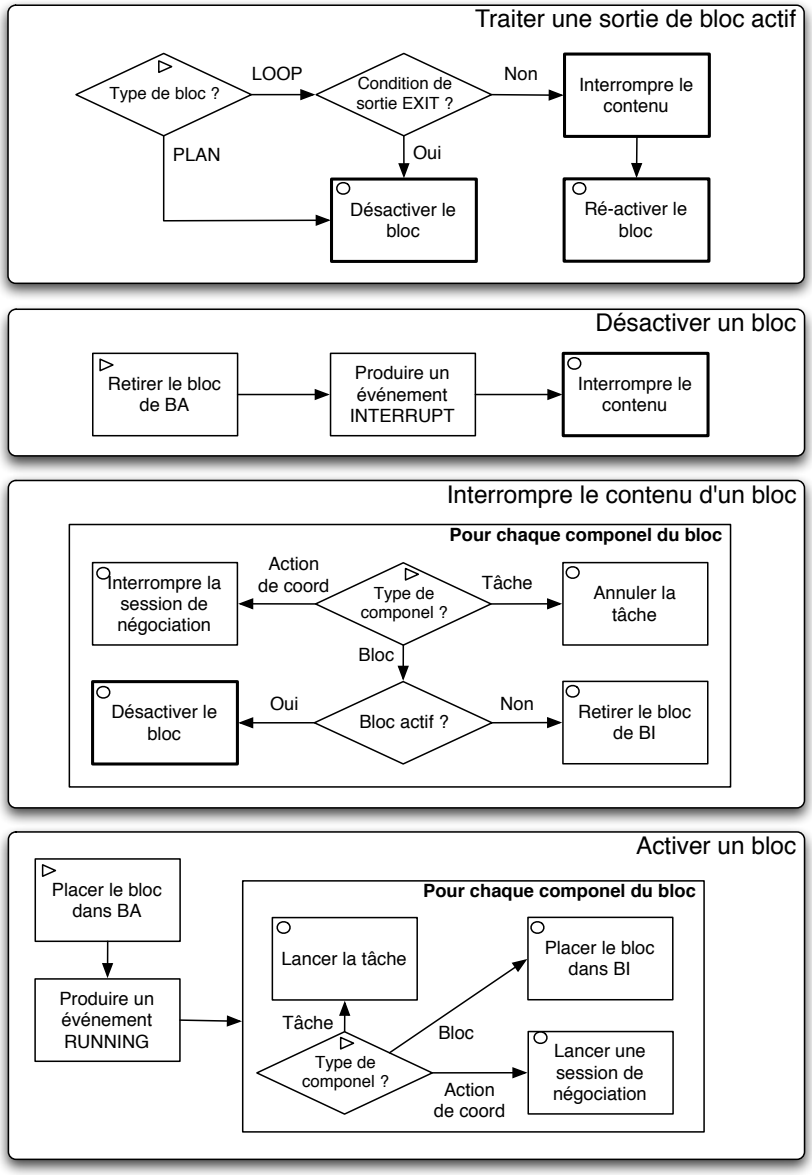


FIG. V.9 – Autres activités du moteur de traitements

de tâche à l'EMD pour exécution. Un statut de tâche `scheduled` est alors retourné et remonte jusqu'au GI. Si la tâche n'est pas encore raffinée, le superviseur de CD demande au couple planificateur symbolique / raffineurs de la traiter. La requête d'exécution de cette tâche est alors transmise de la même manière à l'EMD.

– Cas d'une action de coordination : celle-ci est transmise au gestionnaire de négociations, et une session de négociation est entreprise. Lorsque cette session de négociation s'active, un évènement `scheduled`, puis `running` sont immédiatement produits. L'action de coordination correspond nécessairement à une modalité de négociation, telle que définie dans la section V.3.

La désactivation d'un bloc actif consiste d'abord à générer un évènement INTERRUPT pour ce bloc, avant d'interrompre / annuler tous les componels contenus dans ce bloc actif :

– Cas d'un bloc : si le bloc est dans la liste BI, alors il est simplement retiré de cette liste (devient inactivable). Si le bloc est dans la liste BA (il est actif), alors le bloc fait lui-même l'objet d'une désactivation.

– Cas d'une tâche : une demande d'annulation de tâche est transmise au superviseur de CD, qui envoie la requête à l'EMD. Celui-ci annule alors la tâche si celle-ci est planifiée, mais pas encore déclenchée pour exécution, et procède à l'interruption de la tâche si celle-ci est déjà en cours d'exécution. Il en résulte un événement d'annulation de tâche `aborted`, qui parvient finalement au GI.

– Cas d'une action de coordination : le moteur de traitement transmet au gestionnaire d'interaction une requête d'interruption de la session de négociation correspondante. Lorsque la session de négociation prend fin, l'évènement `aborted` est transmis en retour.

Lorsqu'un modèle d'interaction préalablement chargé est activé, le GI commence par placer le bloc principal de l'automate d'exécution dans la liste BI, tandis que BA est vide. Le moteur de traitement est alors démarré, et le cycle principal est amorcé. Au premier passage, le bloc principal est déplacé de BI à BA (pas de précondition) dans la troisième étape du cycle, au cours de son activation. Les componels contenus dans le bloc sont alors eux même activés, selon leurs nature (voir "Activer un bloc" sur la figure V.8).

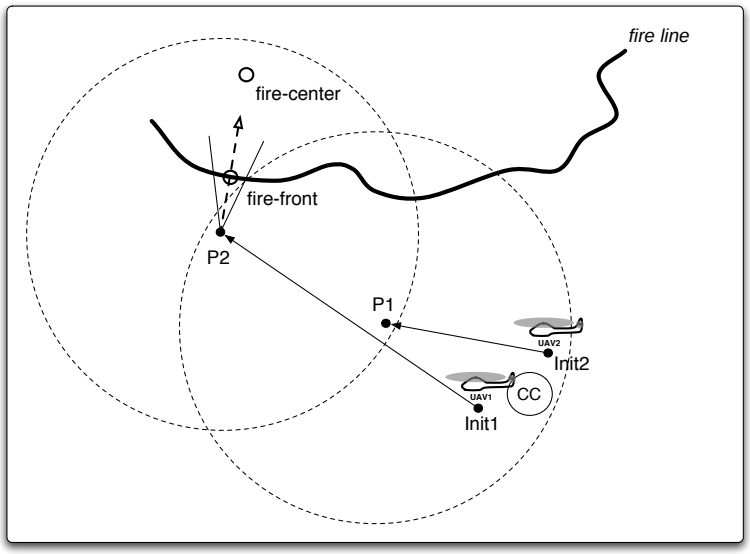
Le traitement se poursuit, tandis que le moteur de traitement écoute les nouveaux événements, susceptibles de satisfaire les préconditions ou les conditions de sorties des blocs de BI et BA.

Il est à noter qu'une tâche jointe, au niveau de l'EMD, est tout à fait susceptible d'être interrompue suite à des contingences : le superviseur de CD peut donc être amené à transmettre une requête d'interruption ABORT à l'interaction manager, que ce soit pour un modèle d'interaction actif (en cours d'exécution) ou chargé mais pas encore actif. Le gestionnaire d'interaction procède alors en conséquence, et reste à l'écoute de nouvelles requêtes (chargement d'une nouveau modèle, activation d'un modèle chargé, etc.).

Exemple de référence - partie 12 : traitement des automates d'exécution

Nous expliquons ici les grandes lignes du déroulement du traitement des rôles, dans un cas nominal. Le chargement des rôles a déjà eu lieu, et UAV1 s'est vu attribué le rôle R^1_{ref}, et UAV2 le rôle R^2_{ref}. Dans l'EMD, les TJE ont le feu vert pour être traitées, et les superviseurs de CD envoient donc une requête de début de traitement aux GI pour le traitement effectif des TJE. Voici les étapes :

1. Entrée dans les blocs principaux respectifs de l'automate d'exécution. pl_{10} et pl_{20} sont initialisés : respectivement avec la position calculée pour réaliser les perceptions, et la position pour le relais de communication.
2. Les tâches de `goto` t_{11} et t_{21} sont activées avec leurs objectifs respectifs. Les moteurs de traitement transmettent ces tâches de haut niveau vers les superviseurs de CD, qui appliquent les opérations nécessaires pour les raffiner, avant de transmettre le résultat vers l'EMD pour exécution immédiate, dans le cadre de la tâche jointe (en parallèle de celle-ci, dans les faits). Nous avons appelé T7 et T12 ces tâches de haut niveau, dans le chapitre IV (partie 3 de l'exemple de référence).



3. Une fois ces tâches de déplacement achevées par les UAV (UAV2 finit vraisemblablement son déplacement vers P1 avant qu'UAV2 n'atteigne P2), une synchronisation doit avoir lieu avant de commencer l'activité proprement dite de perception relayée. Il s'agit des tâches `wait-until-synchro`, respectivement t_{12} et t_{22}. Elles donnent lieu aux tâches de haut niveau respectives T8 et T13 dans la partie 3 de l'exemple de référence.
4. Une fois la synchonisation réalisée, les UAV entrent dans la partie centrale de l'activité jointe : UAV1 commence ses perceptions, tandis que UAV2 effectue passivement le relais de communication (respectivement t_{13} (`takeshot-while waiting`) et t_{24} (`wait-until-synchro`)).
5. Si tout se passe nominalement, une fois le temps écoulé, l'exécution des tâches élémentaires issues de t_{13} prend fin : cela met fin au blocs b_{14} puis b_{12}, et permet donc le déclenchement de la synchronisation pour l'UAV1 (tâche t_{15}). UAV2 étant de son côté en attente de synchronisation (suite à t_{24}), la synchronisation a lieu, et les tâches finales `goto` t_{16} et t_{25} peuvent s'accomplir : retours respectifs aux points de départ.

Ce déroulement est celui qui est nominal : c'est pourquoi certaines parties des automates respectifs n'ont pas été sollicitées. En particulier, aucune session de négociation n'a eu lieu ici. Dans la dernière partie de notre exemple de référence, nous introduirons des contingences qui vont amener le plan à se dérouler différement, et en particulier à faire appel dynamiquement à une redistribution des rôles.

V.4.3 Le vérificateur de contraintes spatiales

Validité de contraintes spatiales

Tandis qu'un rôle est en cours de traitement, les MCS attachés aux blocs actifs sont continuellement vérifiés dans le contexte courant d'exécution. Un MCS est dit *actif* lorsqu'au moins un bloc lié à ce MCS, dans le rôle en cours de l'UAV, est actif.

Le VCS considère en permanence l'ensemble des MCS actifs, et vérifie de façon cyclique que chaque MCS est toujours satisfait. S'il advient qu'une contrainte d'un MCS soit violée, alors le

VCS génère un événement de type "message", portant l'information et précisant la contrainte violée. Ce message peut éventuellement être "intercepté" dans les instructions de ce rôle : il peut en effet donner lieu à la satisfaction de préconditions ou conditions de sortie de blocs. Ainsi, une contrainte spatiale violée peut donner lieu à un traitement permettant de revenir dans une configuration spatiale adéquate, ou alors peut déclencher une session de négociation pour redistribuer les rôles compte tenu des positions courantes des UAV.

Déterminer des positions valides

L'autre fonctionnalité majeure du VCS est son aptitude à calculer, à partir d'un ensemble de contraintes spatiales, des positions valides dans l'espace.

A cet effet, plusieurs solutions sont envisageables : une première solution consiste à considérer le problème comme un problème de satisfaction de contraintes numériques. En utilisant un moteur de résolution de systèmes de contraintes, il est possible de déterminer des solutions qui satisfont ces contraintes. Cependant, la programmation efficace d'un tel moteur est loin d'être triviale, et les solutions existantes sont lourdes, et nécessiteraient un important travail d'intégration pour être exploitées ici.

Nous avons donc opté pour une autre solution, plus adaptée à notre contexte : nous ne définissons, pour le moment, que des conjonctions de contraintes dans nos modèles de contraintes spatiales. Toutes les contraintes exprimées doivent donc nécessairement être satisfaites : l'espace valide est donc un espace connexe, correspondant à la superposition de toutes les contraintes. Nous considérons donc une solution valide pour la première contrainte, dans un premier temps : cela est assez trivial pour une contrainte donnée. Ensuite, pour chaque nouvelle contrainte, nous déplaçons localement la solution afin de trouver une nouvelle solution qui satisfasse l'ensemble des contraintes déjà considérées. Une solution possible est d'étendre stochastiquement des solutions dans les voisinages de la solution courante, et en ne considérant parmi les points tirés aléatoirement que ceux qui ne violent pas les contraintes précédement satisfaites. Parmi ces points, on évalue le "degré" de violation de la nouvelle contrainte considérée. Cette évaluation est assez simple à réaliser :

Dans le cas d'une contrainte de distance, il suffit de comparer, pour la distance du point au point de référence par rapport à la distance de référence. Le quotient des distances est un bon indice du degré de violation : si la relation est $dist < distRef$, alors plus le quotient est élevé, plus la contrainte est violée. Si la relation est $dist > distRef$, alors plus le quotient est proche de 0, plus la contrainte est violée. Dans le premier cas, nous considérons donc la valeur de violation comme le quotient des distances. Dans le second cas, nous considérons la violation comme (1 - le quotient des distances). Nous ordonnons tous les points du voisinage selon une valeur croissante de violation de contraintes : nous sommes alors en mesure de considérer le point "le moins" en violation avec la contrainte. Nous réitérons alors le processus à partir de ce point, jusqu'à trouver une solution valide avec cette nouvelle contrainte (à supposer que cette solution existe : si l'on tombe sur un minimal local depuis lequel on ne parvient plus à converger, c'est qu'il n'y a pas de solution).

Dans le cas d'une contrainte d'orientation, le "degré" de violation est calculé comme l'écart entre l'orientation du point et l'orientation de référence (peut importe la relation de comparaison). Nous classons les points selon ce critère, et nous sélectionnons donc celui qui est "le moins" en violation avec la contrainte d'orientation. Nous réitérons alors le processus, comme pour les

contraintes de distance, jusqu'à l'éventuelle solution.

Cette approche fonctionne tant que l'on ne considère pas de disjonction sur des contraintes : dans ce cas, une approche à base de satisfaction de contraintes semble inévitable.

Exemple de référence - partie 13 : vérificateur de contraintes spatiales

Nous allons introduire dans cette dernière partie de l'exemple de référence des contingences, qui vont conduire à appliquer différement le modèle d'interaction. Nous reprenons le déroulement nominal du scénario jusqu'au point 4 (voir la partie 12 de l'exemple de référence). Les UAV sont en train de réaliser la perception relayée : UAV1 effectue les perceptions, et UAV2 relaye les données vers le centre de contrôle.

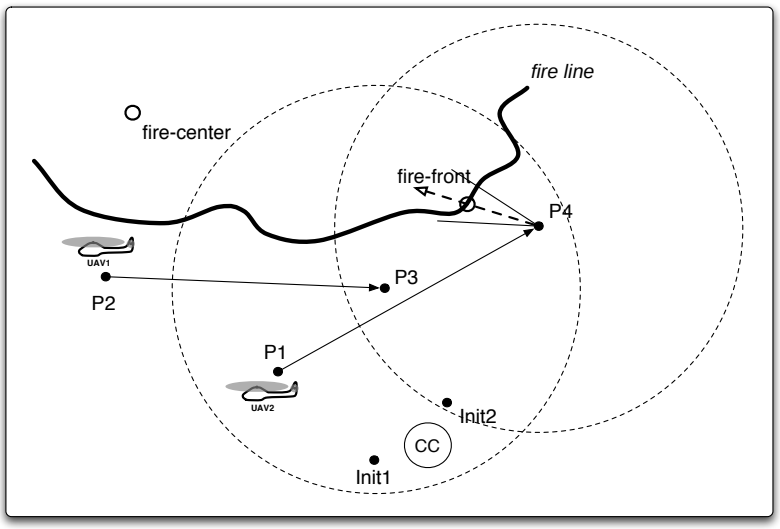
L'objet fire-front est observé par *l'observateur* : les données disponibles sont ainsi tenues à jour. Si le point localisé comme le "devant" du feu se déplace très légèrement, l'UAV n'a pas nécessairement besoin de réagir : les contraintes spatiales sont toujours vérifiées. Si le déplacement est plus important, alors il peut arriver que des contraintes spatiales soient violées : il va s'agir d'une violation de mcs_2. Cette violation, au niveau du VCS, génère un message qui est réceptionné en interne, comme un événement de message. Cet événement va satisfaire la précondition du bloc b_13, ce qui va conduire l'action de coordination ac_{11} à se déclencher. UAV1 va donc initier une session de négociation pour la réallocation de rôles. Il transmet à UAV2 un message de début de session de négociation. Ce message apparait comme un événement dans le moteur de traitement de UAV2. Cet événement est celui qui satisfait le bloc b_{23} : l'action de coordination ac_{21} se déclenche alors, et UAV2 entre dans la session de négociation pour la réallocation de tâche initiée par UAV1. La négociation commence : chacun ré-évalue, compte tenu de sa position courante, le rôle qui lui convient le mieux, et négocie avec l'autre pour déterminer la meilleure solution globale.

Deux cas de figure sont alors envisageables :

– Pas de changement de rôle : pour UAV2, rien ne se passe, le bloc b_{23} boucle, prêt à être de nouveau activé si un message d'UAV1 pour la réallocation de rôle est reçu. Pour UAV1, l'action de coordination ac_{11} se termine, ainsi que le bloc b_{13} qui la contient. La fin de b_{13} est un événement qui provoque la fin du bloc b_{14} (condition de sortie), et qui déclenche donc l'interruption de la tâche t_{13} : au niveau de l'EMD, les tâche élémentaires résultants du raffinement de t_{13} sont interrompues. Le bloc b_15 devient alors actif, ce qui conduit à l'initialisation du paramètre local pl_{15} avec une position valide pour les contraintes spatiales, compte tenu de l'évolution de la position de *fire-front*. Le VCS retourne un point valide pour pl_{15}. Ce paramètre local est alors utilisé comme objectif dans la tâche `goto` t_{14} : UAV1 se déplace donc jusqu'à la position valide. Une fois le déplacement achevé (et donc la tâche t_{14} arrivée dans un état `ended`), le bloc b_{15} se termine, et le bloc b_{12} aussi en suivant. Cependant, celui-ci étant de type LOOP, il est immédiatement de nouveau activé : le bloc b_{14} est en suivant relancé (pas de préconditions), et en suivant la tâche t_{13} qu'il contient, c'est à dire le `takeshot-while-waiting`. L'activité jointe est donc revenue dans une situation "nominale", en attendant d'éventuelles nouvelles contingences.

– Changement de rôle : le changement occasionné par le changement de fire-front est très important, et les UAV évaluent qu'un changment de rôle est bénéfique.

Cela peut par exemple correspondre à la configuration schématisée ci-dessous :



l'"objet" observé fire-front est déplacé subitement, loin de sa position initiale. Il peut par exemple s'agir d'un déplacement ayant pour origine un changement de la direction de propagation principale du feu (suite par exemple à un changement important de la direction du vent...).
Les UAV doivent donc subir un changement de rôle courant, dynamiquement : UAV1 adopte R^2_{ref} et UAV2 adopte R^1_{ref}. Comme on peut le voir sur la figure précédente, une telle allocation peut s'expliquer par les distances réciproques à parcourir qui sont inférieures avec ce changement de rôle. UAV2 va donc réaliser les perceptions, et UAV1 le relais. En premier lieu, le changement de rôle conduit chacun des UAV à interrompre / annuler toutes les activités en cours dans le cadre du traitement du rôle précédent. Puis les nouveaux rôles endossés sont traités. Les déplacements d'UAV1 en P3 et UAV2 en P4 sont réalisés suite au traitement du bloc principal de l'automate d'exécution de leur nouveaux rôles respectifs. Ainsi, les paramètres locaux pl_{10} et pl_{20} sont initialisés respectivement pour le lieu où réaliser les nouvelles perceptions, et le lieu où réaliser le nouveau relais. Puis les tâches t_{11} et t_{21} conduisent respectivement UAV2 et UAV1 sur les nouveaux lieux. L'activité jointe reprend son cours nominal, mis à part que les UAV ont inversé leur rôles.

Cet exemple montre à la fois comment les modèles de contraintes spatiales peuvent intervenir dans la réalisation de l'activité jointe, et comment une modalité de négociation peut être utilisée dans le cadre d'une réallocation de rôles.

V.5 Bilan

Nous avons présenté dans ce chapitre le gestionnaire d'interactions, qui est l'entité en charge du traitement des activités jointes au sein de notre couche délibérative. Dans le cadre de notre paradigme de gestion des interactions entre UAV, nous avons introduit la notion de de modèle d'interaction, au sein duquel des rôles, des modèles de contraintes spatiales et leurs relations sont explicitements définis via une table de coordination. Définir des rôles permet de spécifier explicitement des assymétries dans les activités coopératives à réaliser. La définition explicite que

nous proposons pour les rôles répond selon nous aux besoins de systèmes de robots critiques, comme un système multi-UAV : le cadre des activités jointes est clairement défini : au niveau de la nature des activités entreprises, au niveau temporel, et également au niveau spatial (avec l'application de contraintes spatiales explicites). L'ouverture laissée au niveau de la spécification des négociations est une nécessité compte tenu de la variété des contextes possibles et des méthodes de négociation dans un système multi-robots.

Notre paradigme de définition d'interactions permet à un utilisateur de cadrer explicitement le champ d'interaction qu'il désire, tout en laissant le choix des protocoles de négociation qui paraissent le plus propice.

Comme nous l'avons précisé en introduction de ce chapitre, certains éléments ou traitements que nous proposons au sein du GI n'ont pas été implémentés.

En premier lieu, peu de modalités de négociation ont été développées actuellement. La modalité de réallocation de tâches que nous exploitons est une modalité de test, dans laquelle les rôles sont choisis aléatoirement. Un échange de confirmations permet de vérifier que les rôles sont différents. La modalité de négociation d'intervalle temporel n'est pas encore développée.

Au niveau du vérificateur de contraintes spatiales, l'algorithme proposé pour trouver des positions valides dans l'espace compte tenu des contraintes spatiales n'est pas implémenté.

Les événements temporels ne sont pas encore traités dans les préconditions et conditions de sortie des blocs.

Tout le reste du gestionnaire d'interaction est implémenté, et partiellement testé.

Chapitre VI

Résultats

Ce chapitre regroupe les résultats obtenus d'une part dans un cadre d'expérimentation réel au cours du projet COMETS, et d'autre part en simulation.

VI.1 Résultats en application réelles dans COMETS

Le projet COMETS a défini un cadre de développement et d'expérimentation du plus grand intérêt. Les applications visées par ce projet ont conduit à réaliser des expérimentations réunissant plusieurs UAVs, dans des conditions difficiles à reproduire en dehors d'un projet de ce type (mise à feu de petits foyers sous contrôle de pompiers, espace aérien propice à des expérimentations multi-UAV...).

Trois sessions d'expérimentations ont ainsi été menées entre 2003 et 2005, permettant d'éprouver les progrès dans les développements, au niveau du contrôle des UAVs, des moyens de perception et des traitements associés, au niveau des communications entre les différentes entités du système, et au niveau de l'intégration des entités du système au sein de l'architecture.

Au cours de ces expérimentations, nous avons testé les trois premiers niveaux d'autonomie décisionnelle, c'est à dire des configurations où la prise de décision est essentiellement réalisée au sol, par un NDC (centre de contrôle).

VI.1.1 Mise en œuvre de l'architecture dans le système COMETS

Nous décrivons ici l'architecture et ses composantes, telles que développées et expérimentées dans le projet, pour chacun des trois segments : sol, communication, aérien.

Segment sol :

Le segment sol consiste en un centre de contrôle composé d'un système de planification de mission (SPM), permettant la création assistée (semi-automatique) de missions et leur exécution, et d'un système de contrôle des activités (GUI utilisateur), fournissant les données télémétriques, ainsi que les données de perception retournées à l'opérateur. Le segment sol comprend également le système de perception (SP), en charge de traitements synthétiques des données perçues (en particulier pour la détection et la modélisation de feu).

Segment de communication :

Le segment de communication est un "tableau noir" distribué multi-plateformes du nom de BBCS (Black Board Communication System [Remuß 04]), développé à l'université de Berlin pour les besoins de l'hélicoptère Marvin. Dans ce système de communication, une topologie du réseau est mise en place en définissant des nœuds de communication ainsi que des canaux (channels) entre certains de ces nœuds. Chaque nœud comprend un certains nombre d'emplacements de données ("slots"). Les emplacements sont mis en relation entre les nœuds, en spécifiant, pour chaque emplacement, la fraction de bande passante qui lui est réservée. Le système de nœuds est alors en mesure de propager, là où elles sont déclarées, les données définies. Ce système s'est révélé extrèmement satisfaisant pour les communications entre entités du système (aussi bien pour des données de contrôle que pour des données "utiles" comme la télémétrie ou les images perçues), et a été exploité aussi bien pour des communications sol-sol que pour des communications sol-air, sur une base TCP ou UDP. De plus, un réseau BBCS est robuste aux coupures locales dans le réseau : la transmission des données peut s'opérer par différents chemins, à la condition que chaque nœud du chemin déclare les données en question. Lors d'une rupture de connection entre deux nœuds, le système reconfigure automatiquement la topologie du réseau.

Segment aérien

Le segment aérien regroupe les UAV à proprement parler, ainsi que les composants virtuellement embarquables, mais laissés au sol pour des questions de ressources embarquées (disponibilité de puissance de calcul, énergie, poids). Dans le projet, trois plateformes ont été utilisées (figure VI.1) :

– L'hélicoptère Marvin : hélicoptère développé par l'université de Berlin, il dispose de capacités autonomes avancées de déplacement (passage par points de passages, décollage et atterissage notamment). Doté d'un GPS centimétrique, il s'est trouvé être très bien adapté à la réalisation de tâches de balayage (détection...), ou de placement précis pour des activités de perception (monitoring).

– L'hélicoptère Heliv : hélicoptère développé par la PME Helivision et l'université de Séville. Heliv est destiné à être téléopéré, le pilote ayant connaissance des requêtes de déplacement destinées à Heliv. Heliv, muni de caméras fonctionnant dans le visible et l'infrarouge, s'est trouvé être bien adapté à des tâches de confirmation, dans lesquelles il était intéressant de se rendre rapidement sur place. Pour des tâches répétitives ou demandant une précision dans la

FIG. VI.1 – Haut à gauche, Karma - Haut à droite, Marvin - En bas, Heliv

réalisation, cet UAV téléopéré s'est trouvé être moins performant que Marvin.

– Le dirigeable Karma : dirigeable développé par le LAAS-CNRS, utilisé de façon téléopérée pendant les expérimentations du projet, mais destiné à naviguer de façon autonome (points de passage). Cet UAV s'est trouvé être très intéressant pour des tâches peu contraintes de longue haleine, possiblement très éloignées (la taille de l'appareil le rend visible et opérable de beaucoup plus loin qu'hélicoptère), comme la cartographie depuis une altitude élevée (jusqu'à 200 mètres au dessus du sol, en cours d'expérimentations). Karma est par contre beaucoup moins maniable et précis que les hélicoptères, et est donc peu adapté à des tâches de monitoring, par exemple.

Les Nœuds Décisionnels Distribués (NDD) font partie du segment aérien. Lors des expérimentations, ils ont fonctionné au sol, faute de moyens de traitement suffisant à bord des UAV. En expérimentations réelles, seuls les bas degrés d'autonomie décisionnelle (1 à 3) ont été effectivement testés, avec une prise de décision réalisée de façon centralisée (NDC, au niveau du centre de contrôle, avec le support d'un opérateur).

Implémentation des composants :

Au niveau du segment sol, le centre de contrôle (CC) et le système de perception (SP) ont été réalisés respectivement par l'équipe de l'industriel GMV et l'équipe de l'université de Séville (AICIA). Les développements ont été fait en langage C. Pour ce qui est du segment sol, les développements de BBCS ont été réalisés par l'équipe de l'université de Berlin (TUB) en langage C. Pour le segment aérien, les composants propriétaires embarqués (CPE) ont été réalisé par les propriétaires respectifs des 3 UAV, tandis que les NDD ont été réalisé par le LAAS.

La partie EMD des NDD comprend une interface (codée en C) pour la connection au réseau BBCS, et une partie réalisant effectivement les mécanismes de supervision, développée avec OpenPRS.

VI.1.2 Missions et scénarios COMETS, champs d'application

Nous définissons ici les scénarios applicatifs choisis pour les expérimentations du projet.

1. **Détection et confirmation d'alarme :** au cours de cette activité, il s'agit pour un UAV (ou un groupe d'UAV) de réaliser une détection de feu au dessus d'une zone donnée. La détection consiste à survoler une zone délimitée, en favorisant les parties de la zone considérées comme potentiellement les plus dangereuses (ou étiquetées comme telles dans le modèle de la zone dont disposent les UAVs). La détection peut par exemple être réalisée avec des cellules IR bon marché, mais approximatives.

   Une fois une alarme détectée, au moins 1 autre UAV doit venir confirmer l'alarme, vraisemblablement avec d'autres types de capteurs (visible, ou caméra IR plus précise). Soit il s'agit d'une fausse alarme (moteur de véhicule par exemple...), soit elle est avérée (en considérant des éléments comme le panache de fumée par exemple).

2. **Monitoring d'alarme confirmée :** une fois l'alarme confirmée, plusieurs (au moins 2) UAV doivent se coordonner pour réaliser le monitoring. Dans l'idéal, les UAV impliqués doivent être placés de façon à percevoir le feu sous des angles différents (de face, de coté....). Ils doivent se synchroniser au début de la perception coopérative. Ensuite, pendant une durée donnée, ils réalisent des prises de vue de concert, tandis que les données perçues sont traitées au sol et intégrées dans un modèle de feu, au sein du SP.

3. **Cartographie :** il s'agit de créer ou de mettre à jour une carte tridimensionnelle de l'environnement. Cette activité doit dans l'idéal être réalisée en temps réel, afin de tenir informé l'utilisateur du système des données les plus récentes. Dans le projet COMETS, après des essais avec un bans de caméra stéréo embarqué de trois mètres, des techniques à base de SLAM (Simultaneous Localization And Mapping) monoculaire ont conduit à réaliser des tests avec une unique caméra.

Mission proposée

La mission proposée applique les trois activités introduites précédement. En premier lieu, il s'agit de réaliser une détection de feu pendant une durée D sur une zone Z. Un UAV C1 doté de cellules infrarouges doit se rendre aux environs de Z, et réaliser une détection. Si une alarme est levée (à la position P), il se met en attente en vol de sécurité. Sinon, lorsque D est écoulé, il revient à la base et atterrit.

Si une alarme est levée, un second UAV C2 muni d'une caméra fonctionnant dans le visible doit décoller, se rendre sur aux abords de P et réaliser une tâche de perception. Si l'alarme est infirmée, C1 poursuit sa tâche de détection tandis que C2 revient à la base. Si elle est confirmée, C1 et C2 doivent réaliser une surveillance coordonnée du feu (monitoring). Ils se placent aux positions adéquates, et commencent leur opération de surveillance. Dans le même temps, un troisième UAV B1 doit décoller et réaliser une cartographie de Z, en commençant aux environs de P. Finalement, lorsque l'opérateur met fin à la surveillance, C1, C2 et B1 doivent revenir à la base. La mission prend fin.

Scénario proposé

La mission précédente est instanciée dans le scénario suivant : en premier lieu, Marvin endosse le rôle de C1, Heliv celui de C2, et Karma celui de B1.

Marvin doit se rendre en Z pour une tâche de détection. Tandis qu'il réalise la détection, une alarme est repérée à la position P. Il se place à l'écart, en sécurité, en vol stationnaire. Heliv décolle pour aller confirmer l'alarme. Il se rend à proximité de P et réalise des perceptions. L'alarme est confirmée. Heliv et Marvin se placent sous deux angles différents (de face et de coté), et se synchronisent. Ils réalisent alors des perceptions, et les données perçues sont transmises en temps réel au système de perception au sol pour traitement. Dans le même temps, Karma décolle pour cartographier les environs du lieu de l'alarme. Après un certain temps, l'opérateur met fin au monitoring coordonné, ainsi qu'à la cartographie. Marvin, Heliv et Karma reviennent à proximité du CC, et atterrissent. La mission est terminée.

Activité du NDD dans ce scénario

Le NDD a fonctionné au niveau 3 dans ce scénario : le CC envoyait des séquences de tâches à réaliser, avec certaines dépendances (préconditions et conditions de sorties) à respecter sur ces tâches.

Les plans transmis aux EMD respectifs de chacun des UAV se présentent sous la forme d'une liste de tâches, accompagnées de leurs paramètres et des dépendances à considérer entre tâches. Les paramètres sont enregistrés au niveau de l'interface de l'EMD (ils ne sont pas transmis à l'exécutif proprement dit, qui ne raisonne que sur les dépendances entre tâches, pas sur leurs paramètres).

A chaque UAV était associé un EMD, qui traitait les tâches selon leur dépendance, déclenchant ou interrompant l'exécution des tâches le cas échéant. Les EMD, au niveau 3, étaient également

en charge de la réalisation de synchronisations directes entre EMD, au cours de la réalisation des tâches.

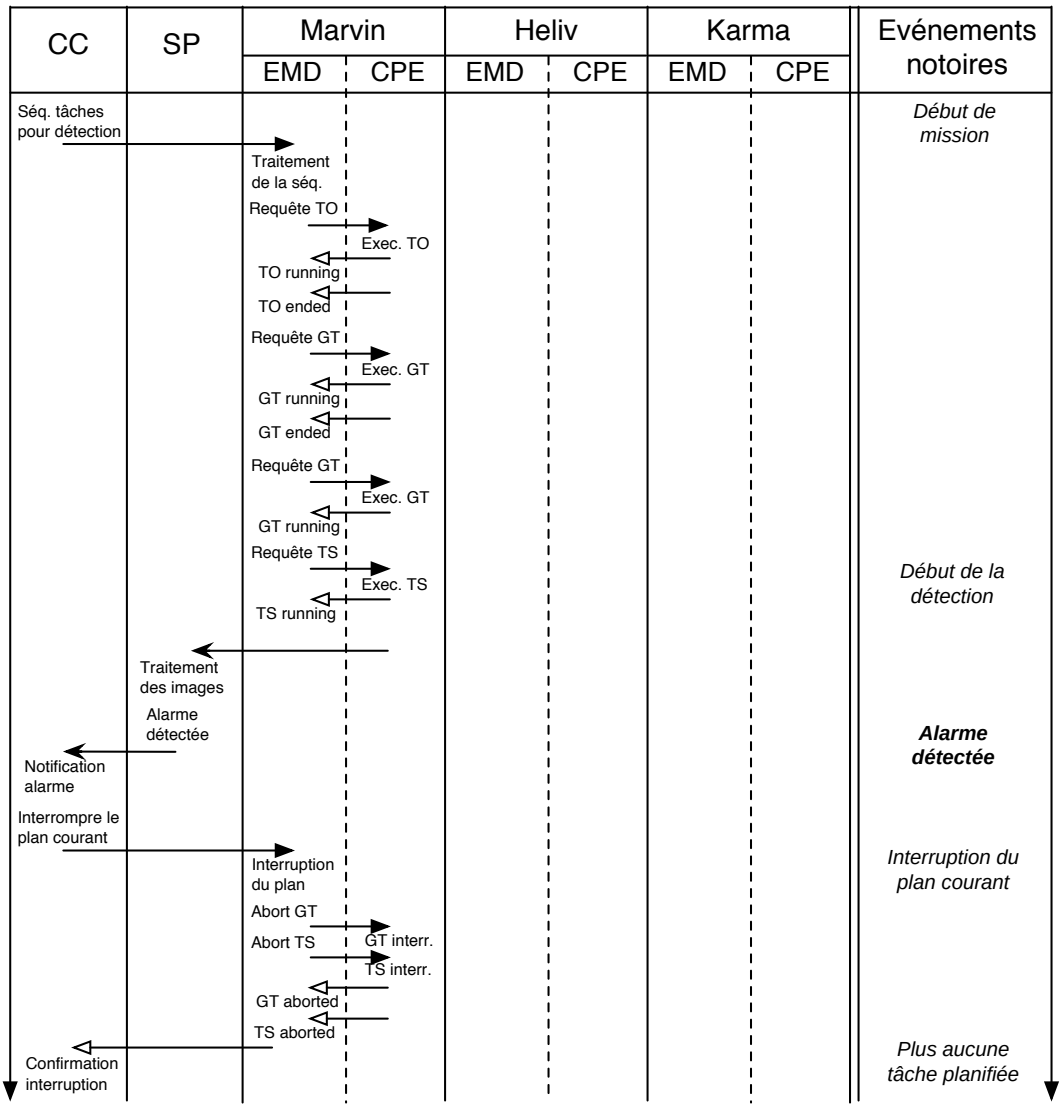


FIG. VI.2 – Première partie : détection d'une alarme

VI.1.3 Résultats obtenus en expérimentations réelles

Le scénario proposé a été, à quelques nuances près, réalisé tel quel avec succès en Mai 2005 au Portugal, lors des expérimentations finales du projet COMETS. Les diagrammes ci-après explicitent le détail des activités des différentes entités du systèmes au cours de la mission.

Le diagramme de la figure VI.2 illustre la première phase : la détection d'une alarme. Au

cours de celle-ci, seulement Marvin est actif. Au moment ou l'alarme est levée, le CC annule toutes les tâches en cours. C'est une étape transitoire de courte durée, dans l'attente de la suite des tâches. Bien que ce n'était pas le cas au cours de ces expérimentations, il est tout à fait envisageable de définir au niveau de l'EMD des routines de "fond", qui se déclenchent lorsqu'aucune autre tâche n'est active. Il pourrait par exemple s'agir d'une tâche "wait", qui place l'UAV dans une situation de vol sécurisé (vol stationnaire pour un hélicoptère, ou vol circulaire en lieu sur pour un dirigeable ou un drone...).

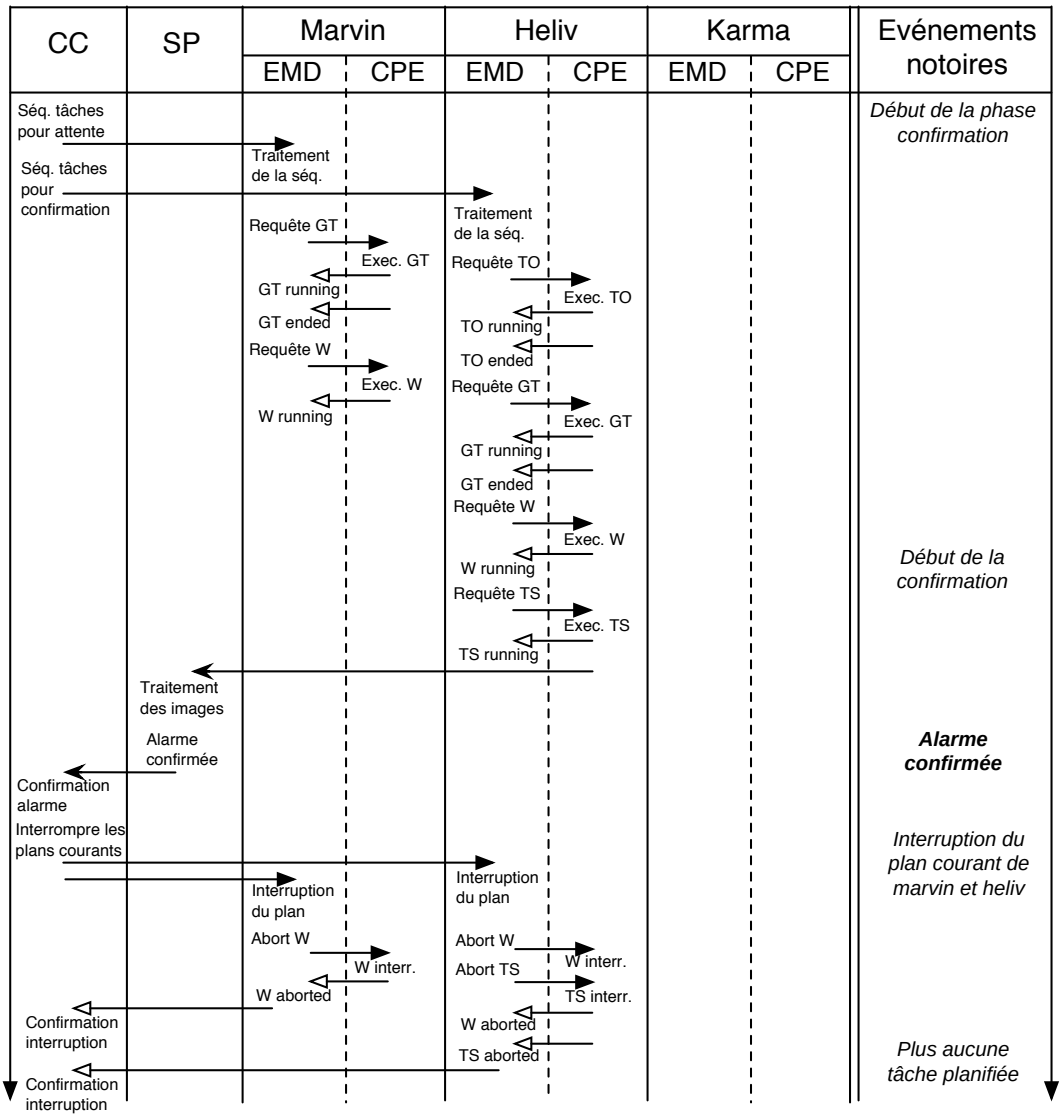


FIG. VI.3 – Deuxième partie : confirmation d'une alarme

Le diagramme de la figure VI.3 illustre la deuxième phase : la confirmation de l'alarme. Dans cette phase, Marvin est mis un petit peu à l'écart, le temps pour Heliv de réaliser les perceptions

pour la confirmation. Lorsque la confirmation est effective, le CC, comme pour la première phase, ordonne l'interruption des plans courants, avant, à la troisième phase, de transmettre la suite des plans en conséquence.

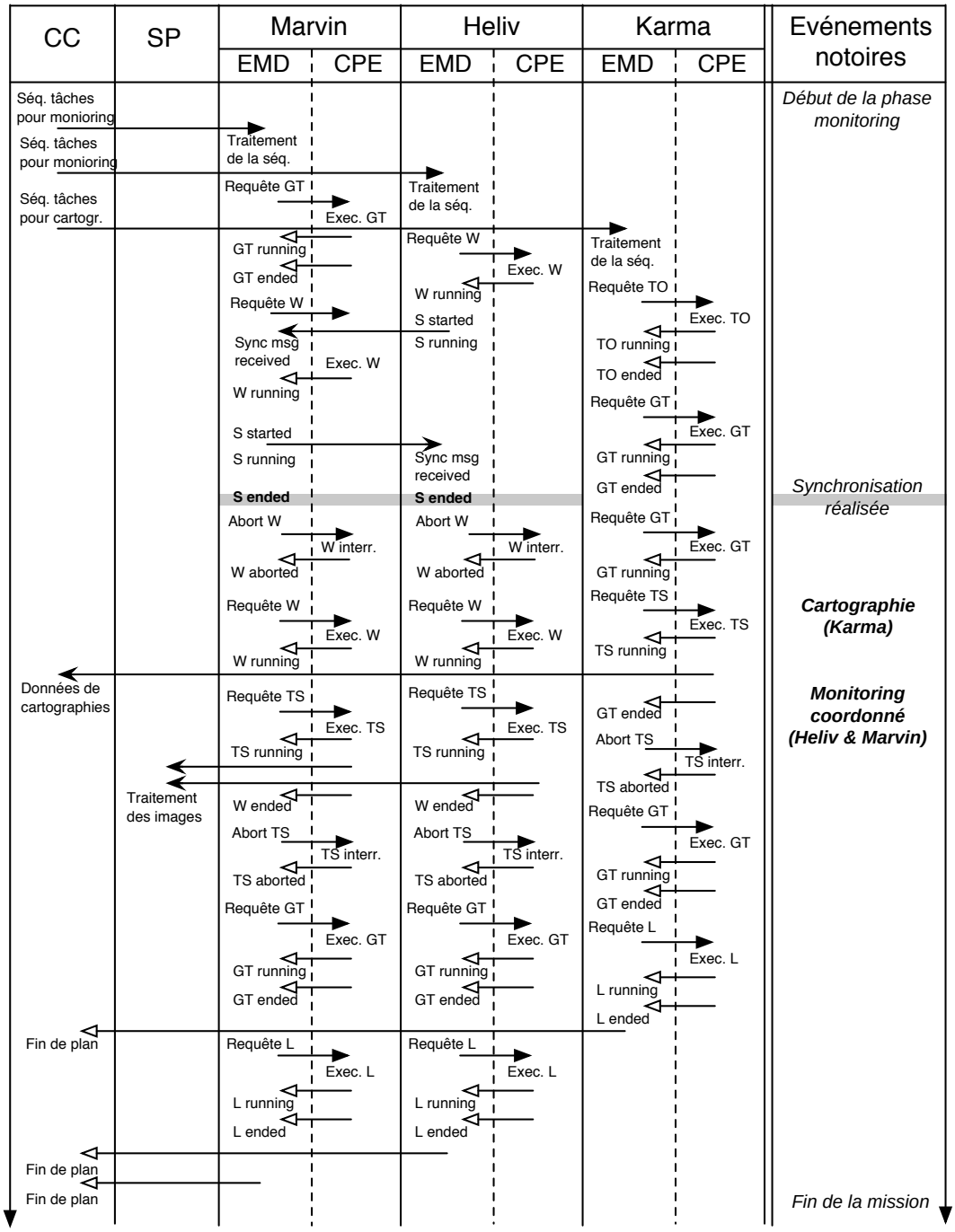


FIG. VI.4 – Troisième partie : surveillance coordonnée et cartographie

Au cours de la troisième phase (figure VI.4), Heliv et Marvin doivent réaliser une surveillance coordonnée du feu (alarme avérée) pendant un certain temps, tandis que Karma doit cartographier les environs du lieu de l'alarme. Une synchronisation est demandée avant de commencer la surveillance coordonnée : Marvin et Heliv exécutent une tâche d'attente (Wait), et lorsque la synchronisation est effective, celle ci est annulée. Une nouvelle tâche Wait est alors planifiée avec un temps donné (2 minutes), avec en parallèle une tâche de perception (Take Shot). La fin du Wait (lorsque le temps est écoulé) déclenche alors la fin du Take Shot, par le biais des dépendances de tâches (condition de sortie) dans les EMD.

Lorsque les activités principales sont terminées (temps écoulé), les UAV reviennent à la base avant de se poser.

Au cours des expérimentations réelles, les couches délibératives distribuées (NDD) n'ont pas été mises à contribution. La section suivante présente les résultats en simulation obtenus avec l'architecture développée pour COMETS.

FIG. VI.5 – Monitoring coopératif d'un foyer par Marvin et Heliv

La photo de la figure VI.5 a été prise à Lousa, Portugal, au mois de Mai 2005, pendant la 3ème session d'expérimentation du projet COMETS. Les UAV sont en train de réaliser une tâche de perception coordonnée du foyer (3ème phase du scénario). Le dirigeable Karma (hors champ) est en train de réaliser une cartographie des lieux.

VI.1.4 Conclusions sur les expérimentations réelles

Les expérimentations réalisées ont démontré l'utilité de disposer d'exécutifs génériques, gérant de façon cohérente, proche du temps réel, le déroulement des plans partiellement ordonnés. Au degré 3 d'autonomie décisionnelle, les synchronisations directes entre EMD se sont opérées de façon transparente pour les autres entités du systèmes, que ce soit le CC ou les CPE. Cette mise en œuvre effective a représenté un très important effort d'intégration, les EMD se trouvant, en terme de communication de données, aux interfaces entre le CC et les CPE propres à chacun des UAV.

VI.2 Résultats obtenus en simulation

Dans cette partie, nous décrivons l'implémentation de la couche délibérative et les tests réalisés en simulation, sur une base de "problèmes de type COMETS".

VI.2.1 Développement liés à la couche délibérative

La couche délibérative regroupe 4 composants principaux : un planificateur symbolique fortement lié à un ensemble de raffineurs géométriques, un gestionnaire d'interaction en charge de la résolution de tâches jointes, et un superviseur de la couche délibérative dont l'objectif est d'assurer la cohérence de la délibération.

En termes de développements, notre démarche à consiste, dans un premier temps, à se munir du moyen de générer des plans au niveau d'un UAV, abstraction faite des interactions entre UAV. Deux motivations à l'origine de ce choix :

– la nécessité d'être rapidement en mesure de générer des plans mono-robot sans reposer sur les capacités du Centre de Contrôle (contraintes de développement liées au projet COMETS).

– un choix de paradigme orienté vers une coordination de plan *a posteriori*, et non en cours de construction du plan. Dans cette optique, il était plausible de réaliser des plans sans considérer explicitement le contexte multi-robots, puis coordonner les plans dans un second temps.

Le couple planificateur / raffineur que nous proposons répond à cette aspiration, sans pour autant fermer les portes à la considération future de tâches raffinables uniquement dans un contexte multi-robots, i.e. reposant sur des processus de coordination multi-robots. La sous-section suivante présente les tests réalisés avec le couple planificateur symbolique / raffineurs spécialisés uniquement.

VI.2.2 Implémentation : choix techniques

– Le planificateur symbolique, comme nous l'avons déja introduit précédement, est construit autour du planificateur Shop2. Celui-ci est programmé en langage Lisp, et nous avons choisi

l'implémentation Lisp "CMUCL" [CMUCL 05], après différents tests d'environnements Lisp : CMUCL est gratuit (la plupart du code est dans le domaine public), compatible sparc-solaris, linux, et est maintenu par une équipe très active. Par ailleurs, cette implémentation fournit des mécanismes intéressants pour interagir (transmission de messages par "pipes") avec d'autres processus. En l'occurence, nous exécutons les raffineurs spécialisés (voir le point suivant) en tant que processus fils depuis Lisp, et communiquons par transmission de messages de processus à processus ("streams" vers flux de données d'entrées et sorties respectives).

– Les raffineurs spécialisés sont implémentés en java, dans leur réalisation actuelle : il s'agit du travail de stage de DEA de Gautier Hattenberger [Hattenberger 04]. Dans un soucis d'homogénéité des sous-systèmes de la couche délibérative, une ré-implémentation des raffineurs en C ou C++ serait sans doute profitable.

– Le gestionnaire d'interaction est développé en C++ : la représentation des données relatives au modèle d'interaction se prête très bien à une implémentation objet. Par ailleurs, le bénéfice de l'utilisation de librairies standards (ou en passe de le devenir) comme STL (manipulation de listes, vecteurs...) ou BOOST (utilisé en l'occurence pour du multi-threading) est très appréciable.

– Le superviseur de la couche délibérative est réalisé avec OpenPRS : le paradigme que propose OpenPRS est très pertinent pour cette entité de supervision, qui doit gérer et réagir à de multiples entrées d'informations. Par ailleurs, l'implémentation OpenPRS du superviseur de CD permet de rapprocher celui-ci de l'EMD, lorsque c'est possible. On obtient alors un superviseur étendu à deux couches : l'une proche de l'exécution, et l'autre proche de la délibération.

– La communication entre les entités de la couche délibérative est assurée par un serveur de communication qui porte le nom de *message passer* : celui-ci fonctionne sur un principe assez basique de transmission de messages par sockets. Tout processus enregistré auprès du *message passer* peut recevoir et envoyer des messages de et vers tout autre processus également enregistré auprès du *message passer* ("peer to peer" ou "broadcast"). Ce système de communication, développé au même moment qu'OpenPRS (et par la même personne : Felix Ingrand), est encore utilisé au LAAS, principalement pour des travaux avec des développements exploitant OpenPRS.

– Un module de simulation d'UAV (très élémentaire) a été développé sur la base d'un module GenoM [Fleury 97] : il s'agit d'un outil de conception et de développement de modules fonctionnels. GenoM permet d'encapsuler les fonctions opérationnelles dans des modules indépendants réutilisables. Les modules GenoM peuvent alors être interfacés avec un exécutif (développé par exemple avec OpenPRS) qui pourra leur adresser des requêtes (lancement, interruption...) et recevoir (et traiter en conséquence) des retours d'exécution. GenoM est un outil utilisé pour tous les robots du LAAS.

– Un module de visualisation d'UAV simulé, a été également développé sur une base de GenoM. Cette interface exploite un autre outil développé au LAAS : GDHE [GDHE 05]. Celui-ci est un outil de visualisation 3D orienté application robotique, et utilisant la librairie OpenGL. Le module de visualisation simule l'environnement : il permet de connecter les différents UAV (sous forme de modules UAV simulés, comme décrits au point précédent), et interface leurs affichages respectifs vers GDHE. Les images de simulation présentées dans cette section sont générées par ce modules, et rendues avec GDHE.

VI.2.3 Planification et raffinement de tâches dans la couche délibérative

Nous présentons ici des résultats obtenus en simulation sur l'utilisation du planificateur symbolique et des raffineurs spécialisés.

Configuration de tests

Nous nous plaçons dans un contexte ou la couche délibérative, associée à l'EMD d'un UAV, interragit avec un niveau fonctionnel simulé. Au cours de nos tests, la configuration employée est illustrée sur la figure VI.6.

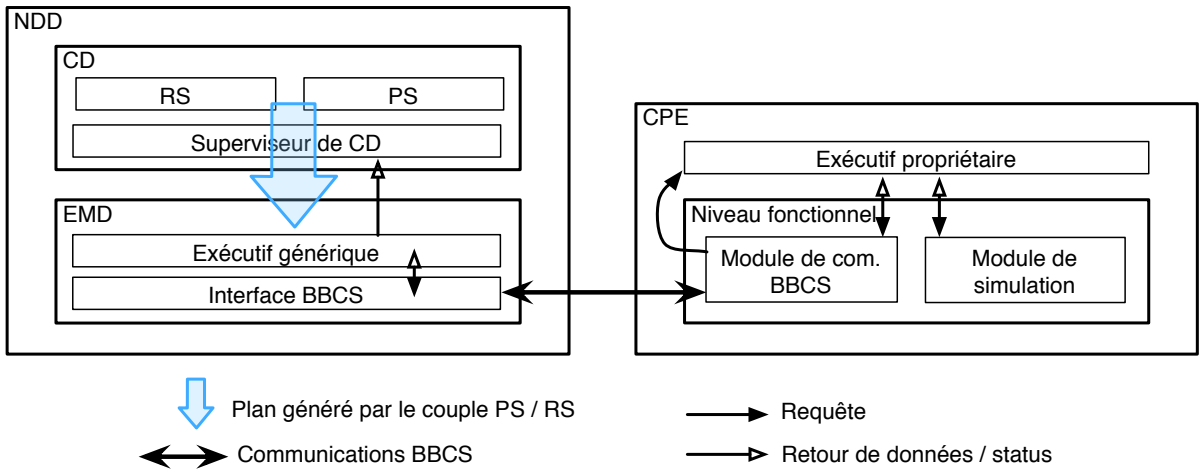


FIG. VI.6 – Configuration de test de plans produits par le couple planificateur / raffineurs

Dans cette configuration, les plans produits dans la CD sont transmis à l'EMD pour exécution. Coté CPE, nous simulons le fonctionnement de la partie embarquée d'un UAV, dans une configuration proche de celle de Karma : un exécutif propriétaire pilote les modules fonctionnels, à savoir un module de communication vers BBCS et un module de simulation proprement dit. Les statuts d'exécution retournés, bien que remontant vers la couche délibérative (au niveau du superviseur de CD), ne sont pas traités : nous sommes, dans ces tests, en boucle ouverte. Pour une mise en œuvre réelle (et donc pour fermer la boucle), des processus de reprise d'erreur et (possiblement) de réparation de plan, devraient être définis au sein de la CD.

Dans cette configuration de simulation, nous transmettons une mission (un ensemble de "buts", ou plus exactement un ensemble d'instructions initiales, puisque nous sommes dans un paradigme de planification HTN) à la CD par l'intermédiaire du superviseur de CD.

Exemples d'élaboration de plan

La figure VI.7 schématise le domaine de planification COMETS dans lequel nos tests ont été réalisés. La hiérarchie des méthodes en particulier y figure clairement, à travers les flèches représentant les décompositions. Le domaine est ouvert : de nouvelles méthodes de plus haut niveau peuvent tout à fait être ajoutées à ce domaine. Les flêches illustrent les dépendances entre

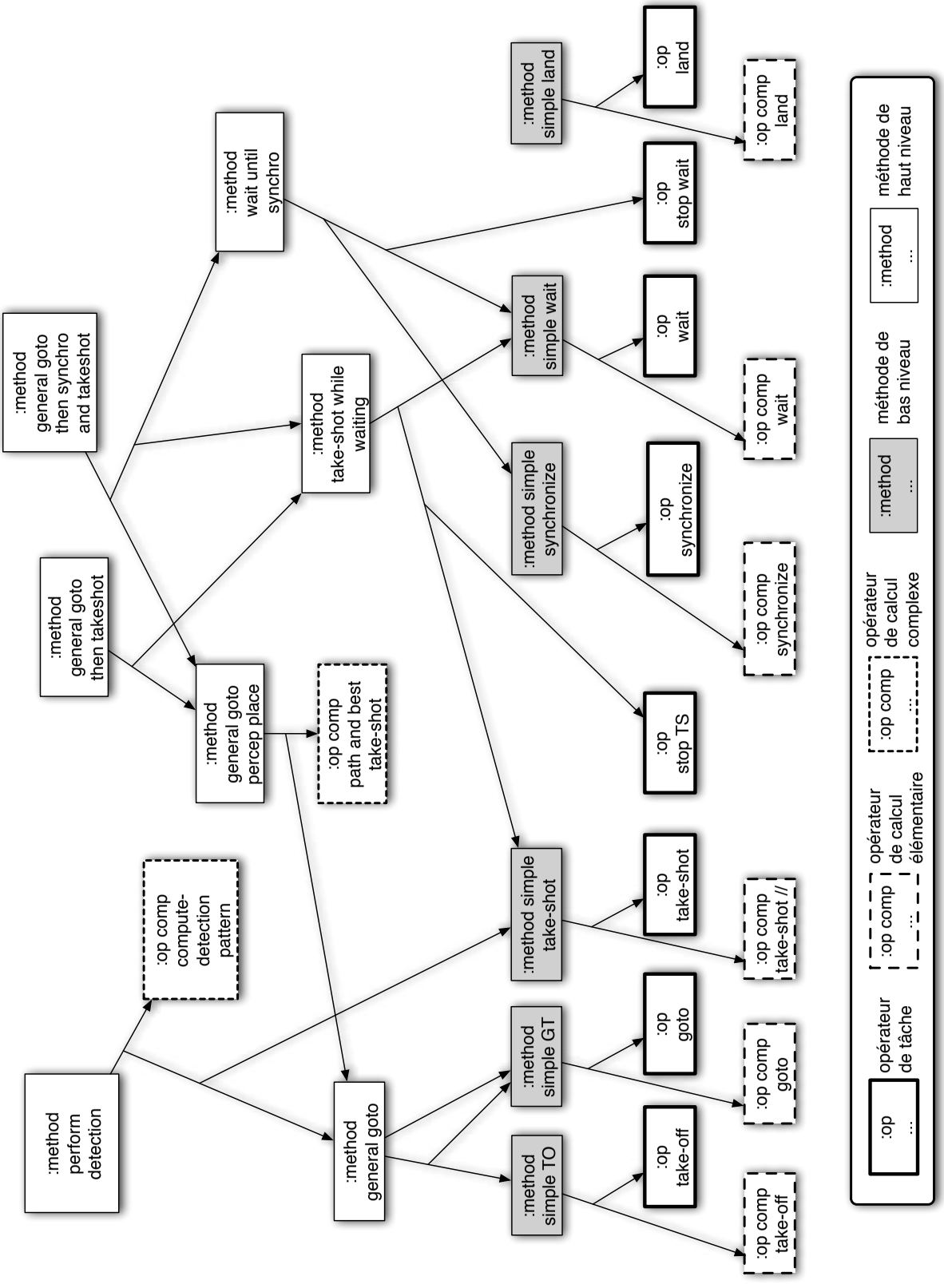


FIG. VI.7 – Domaine de planification COMETS : hiérarchie des méthodes

méthodes. Une flêche F1 dont l'origine appartient à une autre flêche F2 signifie que F2 précède F1, dans la décomposition de la méthode. Deux flèches issues d'une même méthode signifient par défaut une union de décompositions, sans ordre fixé. Si le symbole XOR est spécifié, alors il s'agit d'une disjonction sur les décompositions concernées.

```
(defproblem exemple_these comets-domain2
  ;;; Etat initial
  (
   (current-plan main)
   (takeoffloc zero)
   (landloc zero)
   (uavloc grounded zero)
   (my-write-time uavloc ((num 0) (num 0)) end none)
   (my-read-time uavloc ((num 0) (num 0)) end none)
   (perception-device free 'singlecam)
   (my-write-time perception-device 'singlecam ((num 0) (num 0)) end none)
   (my-read-time perception-device 'singlecam ((num 0) (num 0)) end none)
   (task-id 100)
   )

  ;;; Buts : appel ordonné à des tâches de haut niveau
  (
   (:task general-takeoff nil 700)
   (:task general-combined-goto-and-takeshot P1 20 nil 701)
   (:task perform-detection Z2 P2 60 nil 702)
   (:task general-land nil 703)
  )
)
```

FIG. VI.8 – Définition d'un problème Shop pour Karma

Pour illustrer l'utilisation des raffineurs en cours de planification, nous proposons un exemple simple d'une mission de haut niveau adressée à un UAV unique. Il s'agit, pour le dirigeable Karma, de décoller de la base P0, de se rendre aux environs du point P1 pour percevoir P1 pendant 20 secondes, puis de se rendre au dessus de la zone Z2, pour réaliser une détection d'alarme pendant 5 minutes. Enfin, Karma doit revenir à la base P0 avant d'atterir.

Pour l'UAV, le problème sera décrit par l'état initial du monde (et en particulier de l'UAV), et l'ensemble des tâches de haut niveau que l'UAV doit accomplir. L'énoncé de ce problème est donné sur la figure VI.8. Les identifiants (arbitraires) donnés à la fin de la formulation de chaque tâche du but, sont propagés dans les décompositions jusqu'aux tâche élémentaires, telles qu'elles sont exprimées dans le SCD. Les tâches élémentaires obtenues (telles quelles), prêtes à être traitées, sont présentées sur la figure VI.9. Comme on peut le constater dans ces tâches élémentaires, on peut identifier la tâche de haut niveau d'origine par l'identifiant donnée à la fin

de chaque tâche.

Les tâches obtenues expriment des dépendances les unes par rapport aux autres. Dans le traitement de ces tâches élémentaires, pour traitement et envoi à l'EMD (il s'agit de l'application de l'opérateur SCD-5 dans le SCD, voir la section IV.4.2), ces dépendances vont donner lieu à des préconditions entre tâches, exception faite des tâches élémentaires "stop-takeshot", dont les dépendances vont être interprétées comme des conditions de sortie pour les tâches "takeshot" de même identifiant. Ainsi, dans les tâches élémentaires retournées, représentées sur la figure VI.9, la tâche "takeshot" 103 est suivie d'un "stop-takeshot" portant ce même identifiant, et définissant une dépendance sur l'événement ENDED de la tâche 102. En conséquence, dans l'EMD, la tâche "takeshot" 103 aura comme condition de sortie la fin de la tâche 102, et la prise de vues prendra donc fin immédiatement après la fin de la tâche "wait" 102. La figure VI.10 ci-après illustre l'exécution du plan produit pour Karma, pendant la détection d'alarme. Le chemin parcouru par l'UAV est celui calculé dans les raffineurs, avec l'algorithme pour la détection qui est décrit dans la section IV.2.2, décrivant l'utilisation et les algorithmes des RS.

```
(SHOP-PLAN

(. TASKREQ TASK-TAKEOFF 100 (DEPENDENCES (. .))
  (PARAMS
        ZERO
        40.000000
        (START-TIME (NUM 0) (NUM 0))
        (DURATION (NUM 4) (NUM 6))
        (TIME-CONSTRAINTS NIL NIL NIL NIL)
        700)
.)


(. TASKREQ TASK-GOTOXYZ 101 (DEPENDENCES (. (. ENDED 100 .) .))
  (PARAMS
        ZERO
        (LOC 10.000000 0.000000 70.000000)
        (WPLIST (. (TERM 0.000000 0.000000 80.000000 0 0 0 0 -1)
                   (TERM 10.000000 0.000000 70.000000 14 1 -1 1 -1)
        .))
        (START-TIME (NUM 4) (NUM 6))
        (DURATION (NUM 1) (NUM 81.000000))
        (TIME-CONSTRAINTS NIL NIL NIL NIL)
        701)
.)


(. TASKREQ TASK-WAIT 102 (DEPENDENCES (. (. ENDED 101 .) .))
  (PARAMS
      20
      (START-TIME (NUM 5) (NUM 87.000000))
      (DURATION (NUM 20) (NUM 20))
      (TIME-CONSTRAINTS NIL NIL NIL NIL)
      701)
.)


(. TASKREQ TASK-TAKESHOT 103 (DEPENDENCES (. (. RUNNING 102 .) .))
  (PARAMS
        'SINGLECAM
        (LOC 10.000000 0.000000 70.000000)
        UNLIMITED
        (START-TIME (NUM 25) (NUM 107.000000))
        (DURATION (NUM 0) (NUM 0))
        (TIME-CONSTRAINTS NIL NIL NIL NIL)
        701)
.)


(. TASKREQ TASK-STOP-TAKESHOT 103 (DEPENDENCES (. (. ENDED 102 .) .))
  (PARAMS
  'SINGLECAM
  (START-TIME (NUM 25) (NUM 107.000000))
  (DURATION (NUM 0) (NUM 0))
  (TIME-CONSTRAINTS NIL NIL NIL NIL)
  701)
.)


(. TASKREQ TASK-GOTOXYZ 105 (DEPENDENCES (. (. ENDED 104 .) .))
  (PARAMS
        P2
        (LOC 70.000000 -80.000000 70.000000)
        (WPLIST (.
                (TERM 70.000000 -60.000000 50.000000 0 0 0 0 -1)
                (TERM 80.000000 -70.000000 70.000000 14 1 -1 1 -1)
                (TERM 70.000000 -80.000000 70.000000 28 2 -1 3 -1)
                (TERM 50.000000 -80.000000 70.000000 48 4 -1 6 -1)
                (TERM 30.000000 -80.000000 70.000000 68 5 -1 8 -1)
                (TERM 20.000000 -70.000000 70.000000 82 6 -1 10 -1)
                (TERM 30.000000 -60.000000 70.000000 96 8 -1 12 -1)
                (TERM 40.000000 -50.000000 70.000000 110 9 -1 13 -1)
                (TERM 40.000000 -30.000000 70.000000 130 10 -1 16 -1)
                (TERM 50.000000 -20.000000 70.000000 144 12 -1 18 -1)
                (TERM 70.000000 -20.000000 70.000000 164 13 -1 20 -1)
                (TERM 80.000000 -30.000000 70.000000 178 14 -1 22 -1)
                (TERM 80.000000 -50.000000 70.000000 198 16 -1 24 -1)
                (TERM 80.000000 -70.000000 70.000000 218 18 -1 27 -1)
                (TERM 70.000000 -80.000000 70.000000 233 19 -1 29 -1)
                (TERM 50.000000 -80.000000 70.000000 253 21 -1 31 -1)
                (TERM 30.000000 -80.000000 70.000000 273 22 -1 34 -1)
                (TERM 20.000000 -70.000000 70.000000 287 23 -1 35 -1)
                (TERM 20.000000 -50.000000 70.000000 307 25 -1 38 -1)
                (TERM 20.000000 -30.000000 70.000000 327 27 -1 40 -1)
                (TERM 30.000000 -20.000000 70.000000 341 28 -1 42 -1)
                (TERM 50.000000 -20.000000 70.000000 361 30 -1 45 -1)
                (TERM 60.000000 -30.000000 70.000000 375 31 -1 46 -1)
                (TERM 60.000000 -50.000000 70.000000 395 32 -1 49 -1)
                (TERM 50.000000 -60.000000 70.000000 409 34 -1 51 -1)
                (TERM 30.000000 -60.000000 70.000000 429 35 -1 53 -1)
                (TERM 20.000000 -70.000000 70.000000 443 36 -1 55 -1)
                (TERM 30.000000 -80.000000 70.000000 457 38 -1 57 -1)
                (TERM 50.000000 -80.000000 70.000000 477 39 -1 59 -1)
                (TERM 70.000000 -80.000000 70.000000 497 41 -1 62 -1)
      .))
        (START-TIME (NUM 33) (NUM 195.000000))
        (DURATION (NUM 300) (NUM 300))
        (TIME-CONSTRAINTS NIL NIL NIL NIL)
        702)
.)


(. TASKREQ TASK-TAKESHOT 106 (DEPENDENCES (. (. RUNNING 105 .) (. ENDED 103 .) .))
  (PARAMS
        'SINGLECAM
        (LOC 70.000000 -80.000000 70.000000)
        UNLIMITED
        (START-TIME (NUM 333) (NUM 495.000000))
        (DURATION (NUM 0) (NUM 0))
        (TIME-CONSTRAINTS NIL NIL NIL NIL)
        702)
.)


(. TASKREQ TASK-STOP-TAKESHOT 106 (DEPENDENCES (. (. ENDED 105 .) .))
  (PARAMS
        'SINGLECAM
        (START-TIME (NUM 333) (NUM 495.000000))
        (DURATION (NUM 0) (NUM 0))
        (TIME-CONSTRAINTS NIL NIL NIL NIL)
        702)
.)


(. TASKREQ TASK-GOTOXYZ 107 (DEPENDENCES (. (. ENDED 106 .) (. ENDED 105 .) .))
  (PARAMS
        (LOC 70.000000 -80.000000 70.000000)
        ZERO
        (WPLIST (.
                (TERM 80.000000 -70.000000 90.000000 0 0 0 0 -1)
                (TERM 70.000000 -60.000000 90.000000 14 1 -1 1 -1)
                (TERM 60.000000 -50.000000 90.000000 28 2 -1 3 -1)
                (TERM 50.000000 -40.000000 90.000000 42 3 -1 5 -1)
                (TERM 40.000000 -30.000000 90.000000 56 4 -1 7 -1)
                (TERM 30.000000 -20.000000 90.000000 70 5 -1 8 -1)
                (TERM 20.000000 -10.000000 90.000000 84 7 -1 10 -1)
                (TERM 10.000000 0.000000 90.000000 98 8 -1 12 -1)
        .))
        (START-TIME (NUM 333) (NUM 495.000000))
        (DURATION (NUM 8) (NUM 98.000000))
        (TIME-CONSTRAINTS NIL NIL NIL NIL)
        703)
.)


(. TASKREQ TASK-LAND 108 (DEPENDENCES (. (. ENDED 107 .) .))
  (PARAMS
        ZERO
        25.000000
        (START-TIME (NUM 341) (NUM 593.000000))
        (DURATION (NUM 4) (NUM 6))
        (TIME-CONSTRAINTS NIL NIL NIL NIL)
        703)
.))
```

FIG. VI.9 – Tâches élémentaires obtenues pour Karma

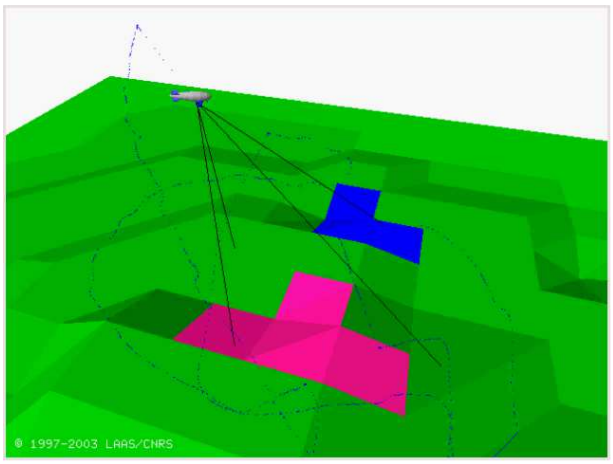


FIG. VI.10 – Test en simulation : détection d'alarme par Karma

Elaboration de plan avec tâches de synchronisations

Un deuxième exemple de planification avec le couple PS/RS est présenté ici, dans le cadre du scénario COMETS : cet exemple met en jeu une synchronisation entre deux UAV. L'hélicoptère Marvin est déjà en vol, et attend pour une confirmation d'alarme de l'hélicoptère Heliv au point "lousa-fire". Heliv doit décoller, faire des prises de vues dans un premier temps en solo pour confirmer l'alarme, puis, après synchronisation avec Marvin, réaliser un "monitoring" coordonné de l'alarme avec Marvin durant 2 minutes. Enfin, les hélicoptères doivent revenir se poser à la base.

La figure VI.11 décrit les problèmes de planification des missions respectives de Heliv et Marvin. On peut en particulier voir dans la tâche de haut niveau "combinée" pour réaliser un "monitoring" coordonné, un certain nombre de paramètres liés à la synchronisation : 551 correspond à un identifiant arbitraire commun aux deux UAV pour mettre en relation les tâches de synchronisation. Les deux couples (20 21) représentent ensuite les identifiants des UAV émetteurs et récepteurs des messages de synchronisation : les deux ensembles étant identiques, il s'agit d'un rendez-vous (ou d'une synchronisation totale) entre les deux UAV (Marvin ayant l'identifiant 20, et Helivision l'identifiant 21, dans le ce cas).

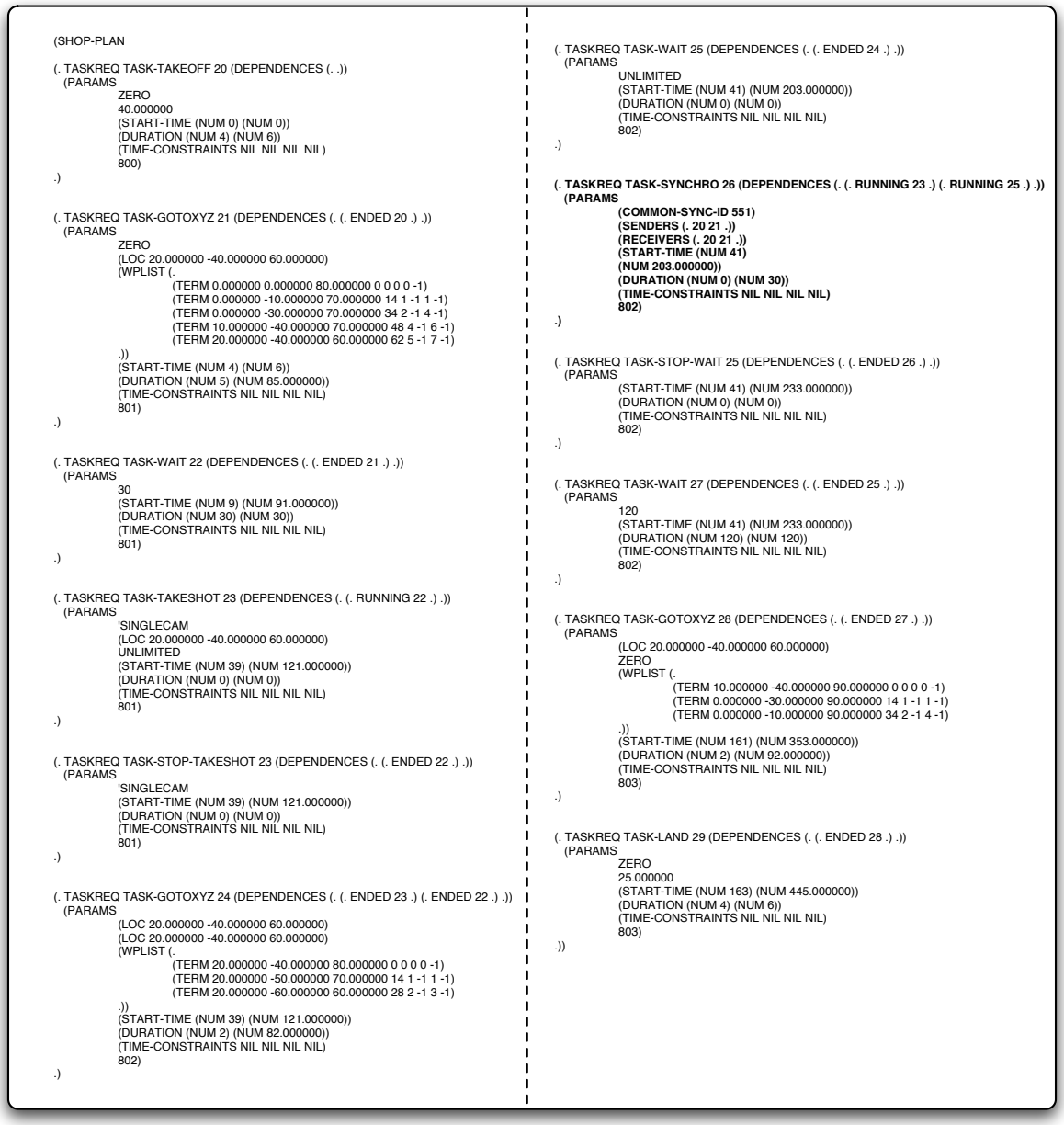
Les ensembles de tâches élémentaires produites par le planificateurs sont donnés respectivement sur la figure VI.12 pour Heliv, et sur la figure VI.13 pour Marvin. Les tâches de synchronisation proprement dites sont illustrées en gras.

Ces plans ont été joués en simulation, dans la configuration de test présentée en début de section (figure VI.6). L'image montrée ci-après sur la figure VI.14 correspond a une prise de vue coordonnée d'Heliv et Marvin pendant l'exécution de ces plans, peu après la synchronisation.

```
(defproblem heliv1 comets-domain2

  ;;; Etat initial
  (
   (current-plan main)
   (takeoffloc zero)
   (landloc zero)
   (uavloc grounded zero)
   (my-write-time uavloc ((num 0) (num 0)) end none)
   (my-read-time uavloc ((num 0) (num 0)) end none)
   (perception-device free 'singlecam)
   (my-write-time perception-device 'singlecam ((num 0) (num 0)) end none)
   (my-read-time perception-device 'singlecam ((num 0) (num 0)) end none)
   (task-id 20)
  )

  ;;;Buts
  (
   (:task general-takeoff nil 800)
   (:task general-combined-goto-and-takeshot lousa-fire 30 nil 801)
   (:task general-combined-goto-and-synchro-and-takeshot 551 (20 21) (20 21) lousa-fire 120 nil 802)
   (:task general-land nil 803)
  )
)
```

```
(defproblem marvin2 comets-domain2

  ;;; Etat initial
  (
   (current-plan main)
   (takeoffloc zero)
   (landloc zero)
   (uavloc flying (LOC 30.0 -30.0 40.0))
   (my-write-time uavloc ((num 0) (num 0)) end none)
   (my-read-time uavloc ((num 0) (num 0)) end none)
   (perception-device free 'singlecam)
   (my-write-time perception-device 'singlecam ((num 0) (num 0)) end none)
   (my-read-time perception-device 'singlecam ((num 0) (num 0)) end none)
   (task-id 40)
   )

  ;;; Buts
  (
   (:task general-combined-goto-and-synchro-and-takeshot 551 (20 21) (20 21) lousa-fire 120 nil 900)
   (:task general-land nil 901)
  )
)
```

FIG. VI.11 – Définition d'un problème Shop pour Heliv (haut) et Marvin (bas)



```
(SHOP-PLAN

(. TASKREQ TASK-TAKEOFF 20 (DEPENDENCES (. .))
  (PARAMS
        ZERO
        40.000000
        (START-TIME (NUM 0) (NUM 0))
        (DURATION (NUM 4) (NUM 6))
        (TIME-CONSTRAINTS NIL NIL NIL NIL)
        800)
.)


(. TASKREQ TASK-GOTOXYZ 21 (DEPENDENCES (. (. ENDED 20 .) .))
  (PARAMS
        ZERO
        (LOC 20.000000 -40.000000 60.000000)
        (WPLIST (.
                (TERM 0.000000 0.000000 80.000000 0 0 0 0 -1)
                (TERM 0.000000 -10.000000 70.000000 14 1 -1 1 -1)
                (TERM 0.000000 -30.000000 70.000000 34 2 -1 4 -1)
                (TERM 10.000000 -40.000000 70.000000 48 4 -1 6 -1)
                (TERM 20.000000 -40.000000 60.000000 62 5 -1 7 -1)
        .))
        (START-TIME (NUM 4) (NUM 6))
        (DURATION (NUM 5) (NUM 85.000000))
        (TIME-CONSTRAINTS NIL NIL NIL NIL)
        801)
.)


(. TASKREQ TASK-WAIT 22 (DEPENDENCES (. (. ENDED 21 .) .))
  (PARAMS
        30
        (START-TIME (NUM 9) (NUM 91.000000))
        (DURATION (NUM 30) (NUM 30))
        (TIME-CONSTRAINTS NIL NIL NIL NIL)
        801)
.)


(. TASKREQ TASK-TAKESHOT 23 (DEPENDENCES (. (. RUNNING 22 .) .))
  (PARAMS
        'SINGLECAM
        (LOC 20.000000 -40.000000 60.000000)
        UNLIMITED
        (START-TIME (NUM 39) (NUM 121.000000))
        (DURATION (NUM 0) (NUM 0))
        (TIME-CONSTRAINTS NIL NIL NIL NIL)
        801)
.)


(. TASKREQ TASK-STOP-TAKESHOT 23 (DEPENDENCES (. (. ENDED 22 .) .))
  (PARAMS
        'SINGLECAM
        (START-TIME (NUM 39) (NUM 121.000000))
        (DURATION (NUM 0) (NUM 0))
        (TIME-CONSTRAINTS NIL NIL NIL NIL)
        801)
.)


(. TASKREQ TASK-GOTOXYZ 24 (DEPENDENCES (. (. ENDED 23 .) (. ENDED 22 .) .))
  (PARAMS
        (LOC 20.000000 -40.000000 60.000000)
        (LOC 20.000000 -40.000000 60.000000)
        (WPLIST (.
                (TERM 20.000000 -40.000000 80.000000 0 0 0 0 -1)
                (TERM 20.000000 -50.000000 70.000000 14 1 -1 1 -1)
                (TERM 20.000000 -60.000000 60.000000 28 2 -1 3 -1)
        .))
        (START-TIME (NUM 39) (NUM 121.000000))
        (DURATION (NUM 2) (NUM 82.000000))
        (TIME-CONSTRAINTS NIL NIL NIL NIL)
        802)
.)


(. TASKREQ TASK-WAIT 25 (DEPENDENCES (. (. ENDED 24 .) .))
  (PARAMS
        UNLIMITED
        (START-TIME (NUM 41) (NUM 203.000000))
        (DURATION (NUM 0) (NUM 0))
        (TIME-CONSTRAINTS NIL NIL NIL NIL)
        802)
.)


(. TASKREQ TASK-SYNCHRO 26 (DEPENDENCES (. (. RUNNING 23 .) (. RUNNING 25 .) .))
  (PARAMS
        (COMMON-SYNC-ID 551)
        (SENDERS (. 20 21 .))
        (RECEIVERS (. 20 21 .))
        (START-TIME (NUM 41)
        (NUM 203.000000))
        (DURATION (NUM 0) (NUM 30))
        (TIME-CONSTRAINTS NIL NIL NIL NIL)
        802)
.)


(. TASKREQ TASK-STOP-WAIT 25 (DEPENDENCES (. (. ENDED 26 .) .))
  (PARAMS
        (START-TIME (NUM 41) (NUM 233.000000))
        (DURATION (NUM 0) (NUM 0))
        (TIME-CONSTRAINTS NIL NIL NIL NIL)
        802)
.)


(. TASKREQ TASK-WAIT 27 (DEPENDENCES (. (. ENDED 25 .) .))
  (PARAMS
        120
        (START-TIME (NUM 41) (NUM 233.000000))
        (DURATION (NUM 120) (NUM 120))
        (TIME-CONSTRAINTS NIL NIL NIL NIL)
        802)
.)


(. TASKREQ TASK-GOTOXYZ 28 (DEPENDENCES (. (. ENDED 27 .) .))
  (PARAMS
        (LOC 20.000000 -40.000000 60.000000)
        ZERO
        (WPLIST (.
                (TERM 10.000000 -40.000000 90.000000 0 0 0 0 -1)
                (TERM 0.000000 -30.000000 90.000000 14 1 -1 1 -1)
                (TERM 0.000000 -10.000000 90.000000 34 2 -1 4 -1)
        .))
        (START-TIME (NUM 161) (NUM 353.000000))
        (DURATION (NUM 2) (NUM 92.000000))
        (TIME-CONSTRAINTS NIL NIL NIL NIL)
        803)
.)


(. TASKREQ TASK-LAND 29 (DEPENDENCES (. (. ENDED 28 .) .))
  (PARAMS
        ZERO
        25.000000
        (START-TIME (NUM 163) (NUM 445.000000))
        (DURATION (NUM 4) (NUM 6))
        (TIME-CONSTRAINTS NIL NIL NIL NIL)
        803)
.))
```

FIG. VI.12 – Tâches élémentaires produites pour Heliv

```
(SHOP-PLAN

(. TASKREQ TASK-GOTOXYZ 40 (DEPENDENCES (. .))
  (PARAMS
        (LOC 30.000000 -30.000000 40.000000)
        (LOC 20.000000 -40.000000 60.000000)
        (WPLIST (.
              (TERM 30.000000 -20.000000 70.000000 0 0 0 0 -1)
              (TERM 20.000000 -30.000000 70.000000 14 1 -1 1 -1)
              (TERM 20.000000 -40.000000 60.000000 28 2 -1 3 -1)
        .))
        (START-TIME (NUM 0) (NUM 0))
        (DURATION (NUM 2) (NUM 72.000000))
        (TIME-CONSTRAINTS NIL NIL NIL NIL)
        900)
.)


(. TASKREQ TASK-WAIT 41 (DEPENDENCES (. (. ENDED 40 .) .))
  (PARAMS
        UNLIMITED
        (START-TIME (NUM 2) (NUM 72.000000))
        (DURATION (NUM 0) (NUM 0))
        (TIME-CONSTRAINTS NIL NIL NIL NIL) 900)
.)


(. TASKREQ TASK-SYNCHRO 42 (DEPENDENCES (. (. RUNNING 41 .) .))
  (PARAMS
        (COMMON-SYNC-ID 551)
        (SENDERS (. 20 21 .))
        (RECEIVERS (. 20 21 .))
        (START-TIME (NUM 2) (NUM 72.000000))
        (DURATION (NUM 0) (NUM 30))
        (TIME-CONSTRAINTS NIL NIL NIL NIL) 900)
.)


(. TASKREQ TASK-STOP-WAIT 41 (DEPENDENCES (. (. ENDED 42 .) .))
  (PARAMS
        (START-TIME (NUM 2) (NUM 102.000000))
        (DURATION (NUM 0) (NUM 0))
        (TIME-CONSTRAINTS NIL NIL NIL NIL) 900)
.)


(. TASKREQ TASK-WAIT 43 (DEPENDENCES (. (. ENDED 41 .) .))
  (PARAMS
        120
        (START-TIME (NUM 2) (NUM 102.000000))
        (DURATION (NUM 120) (NUM 120))
        (TIME-CONSTRAINTS NIL NIL NIL NIL)
        900)
.)


(. TASKREQ TASK-TAKESHOT 44 (DEPENDENCES (. (. RUNNING 43 .) (. ENDED 42 .) .)
  (PARAMS
        'SINGLECAM
        (LOC 20.000000 -40.000000 60.000000)
        UNLIMITED
        (START-TIME (NUM 122) (NUM 222.000000))
        (DURATION (NUM 0) (NUM 0))
        (TIME-CONSTRAINTS NIL NIL NIL NIL)
        900)
.)


(. TASKREQ TASK-STOP-TAKESHOT 44 (DEPENDENCES (. (. ENDED 43 .) .))
  (PARAMS
        'SINGLECAM
        (START-TIME (NUM 122) (NUM 222.000000))
        (DURATION (NUM 0) (NUM 0))
        (TIME-CONSTRAINTS NIL NIL NIL NIL)
        900)
.)


(. TASKREQ TASK-GOTOXYZ 45 (DEPENDENCES (. (. ENDED 44 .) (. ENDED 43 .) .))
  (PARAMS
        (LOC 20.000000 -40.000000 60.000000)
        ZERO
        (WPLIST (.
              (TERM 10.000000 -40.000000 90.000000 0 0 0 0 -1)
              (TERM 0.000000 -30.000000 90.000000 14 1 -1 1 -1)
              (TERM 0.000000 -10.000000 90.000000 34 2 -1 4 -1)
        .))
        (START-TIME (NUM 122) (NUM 222.000000))
        (DURATION (NUM 2) (NUM 92.000000))
        (TIME-CONSTRAINTS NIL NIL NIL NIL) 901)
.)


(. TASKREQ TASK-LAND 46 (DEPENDENCES (. (. ENDED 45 .) .))
  (PARAMS
        ZERO
        25.000000
        (START-TIME (NUM 124) (NUM 314.000000))
        (DURATION (NUM 4) (NUM 6))
        (TIME-CONSTRAINTS NIL NIL NIL NIL) 901)
.))
```

FIG. VI.13 – Tâches élémentaires produites pour Marvin

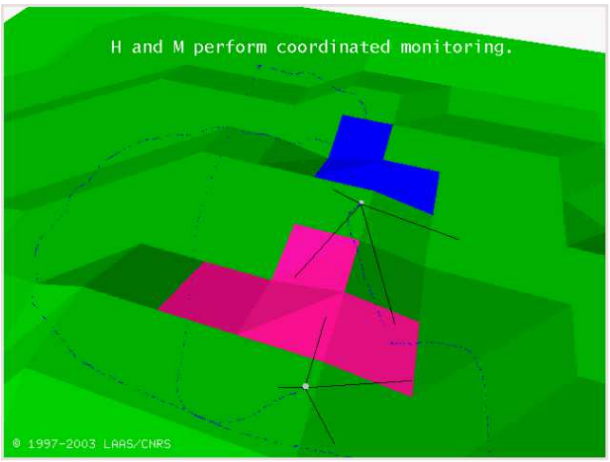


FIG. VI.14 – Test en simulation : perceptions coopératives entre Heliv et Marvin, après synchronisation

VI.2.4 Gestionnaire d'interactions : mise en œuvre partielle

Nous décrivons dans cette sous-section les tests réalisés avec le gestionnaire d'interactions : d'abord déconnecté du reste du système (tests internes ou "unitaires"). Puis avec le reste des éléments de la couche délibérative.

Tests du gestionnaire d'interaction seul

Nous nous sommes attachés à tester dans un premier temps les principales fonctionnalités et mécanismes du GI, sans interfacer celui-ci avec le reste de la CD. Ces tests ont donc concerné la modélisation de tâche jointe avec les modèles d'interaction, l'activation et le traitement d'un modèle d'interaction, la modélisation et la vérification de contraintes spatiales actives, et la mise en œuvre et le traitement de modalités de négociations élémentaires.

Nous décrivons brièvement ici les tests réalisés pour valider le fonctionnement des différents traitements dans le GI :

– **Description d'un modèle d'interaction :** nous avons en premier lieu vérifié la cohérence des structures et relations entres structures pour définir un modèle d'interaction. Nous avons écrit plusieurs rôles factices, plusieurs modèles de contraintes spatiales, et mis en relations ces données au sein d'une table de coordination, pour obtenir une variante de modèle d'interaction applicable.

– **Mécanismes de traitement de MI :** le moteur de traitement du GI a été éprouvé en effectuant des opérations de *chargement* de MI (voir la mécanique du GI, section V.4), puis en démarrant véritablement le traitement du rôle préselectionné dans la phase de chargement. Le moteur de traitement fonctionne de la façon prévue.

– **Modèles de contraintes spatiales :** nous avons vérifié le fonctionnement du VCS en définissant des contraintes spatiales d'orientation et de distance sur un rôle actif (en cours de traitement), et dépendant de données (positions de points dans l'espace) récupérées à partir de l'observateur : les données observées correspondent à des coordonnées de points mises à jour à travers le medium de communication BBCS. Il s'agit par exemple de positions d'UAV ou position d'alarmes d'incendie simulées, et dynamiquement mises à jour en cours. Nous avons pu nous assurer de la génération d'un événement de "viol de contraintes spatiales", lorsqu'il advient qu'un modèle de contraintes spatiales imposés ne soient plus satisfait dans le contexte courant d'exécution (suite à l'évolution des objets observés dans l'environnement).

– **Modalités de négociation :** le gestionnaire de négociations fournit des capacités d'envoi de messages entre UAV, avec un support pour des informations de types différents : chaîne de caractère, valeurs numériques, temporelles, mais aussi des informations géométriques comme des points. Une modalité de négociation doit donc définir la nature des données à échanger, et la sémantique, dans le cadre de cette modalité, que les UAV impliqués doivent attacher aux informations. Dans le test mis en œuvre, nous nous sommes contentés de définir un choix de rôle sur une base aléatoire : en cas de conflit sur les rôles sélectionnés, l'un des deux choisit aléatoirement parmi les autres rôles disponibles. Il s'agit bien entendu d'une modalité très élémentaire destinée à tester l'application et le traitement d'une action de coordination dans le traitement des rôles. Des modalités beaucoup plus riches et pertinentes devraient être conçues et développées dans le cadre d'applications réelles.

Remarque sur la conception de modèles d'interaction : Le formalisme que nous avons mis au point est riche, mais se prette assez peu à une description "ligne à ligne" par un concepteur : il nécessite en effet la définition d'un nombre important de symboles représentant les componels, les sources d'informations, les événements, et les liens entre les différentes données.

En revanche, il est tout à fait vraisemblable d'imaginer une interface graphique de conception de modèles d'interactions : des associations entre entités à base de "drag'n drop" et des pointages à la souris dans une représentation de l'environnement, paraissent tout à fait plausibles pour définir à un haut niveau et assez intuitivement les données requises pour la définition de modèles d'interaction.

Premiers tests intégrés

Dans un second temps, nous avons réalisé des tests d'intégration du GI au sein d'une couche délibérative : il s'agit là encore de tests restreints, le nœud décisionnel (couche délibérative + EMD) déconnecté d'un système fonctionnel (bas niveau).

– Requête de chargement à destination du GI. Dans ce test, une mission est transmise à la couche déliblérative. La mission est traitée par le superviseur de CD (opérateurs 0,1), et transmise au couple planificateur / raffineurs. Le plan résultant contient une tâche jointe élémentaire : le superviseur de CD transmet une requête de chargement de TJE au GI (opérateur SCD-4). Le GI charge correctement le modèle d'interaction correspondant à la TJE, en mettant en correspondance les paramètres passés.

– Requête de traitement à destination du GI. Le superviseur de CD transmet une requête pour le traitement de la modalité d'interaction chargée suite à la requête initiale de chargement. Le moteur de traitement du GI applique le rôle sélectionné au moment du chargement, et produit les tâches de haut niveau correspondantes : celles-ci sont réceptionnées par le superviseur de CD, qui les transmets pour raffinement au couple planificateur / raffineurs (opérateur SCD-3), avant de les envoyer à l'EMD pour exécution (opérateur SCD-5).

Des tests plus systématiques du GI (et de la couche délibérative), avec une mise en œuvre exhaustive des opérateurs (en particulier l'opérateur SCD-6 pour la replanification, ou l'utilisation de l'opérateur SCD-1 dans des cas où des missions sont déja programmées au niveau des UAV) seraient nécessaires afin de valider le paradigme et les notions introduits, ainsi que l'implémentation que nous en proposons : il s'agit ici de travaux prospectifs.

Dans les différents tests impliquant le gestionnaire d'interactions, c'est un scénario à 2 UAV proche de celui présenté dans les chapitres IV et V qui a été appliqué.

Chapitre VII

Conclusions et perspectives

VII.1 Retour sur les contributions

Nous avons proposé dans ce mémoire une approche pour appréhender le contrôle et la prise de décision dans les systèmes multi-UAV.

Le problème que nous avons considéré est très ambitieux, puisqu'il s'agit de doter un système multi-UAV à la fois d'une architecture de contrôle et de décision flexible, et de proposer un cadre d'interaction puissant et maîtrisable. L'approche choisie est pragmatique, et très orientée utilisateur ; elle peut se résumer ainsi : augmenter l'autonomie décisionnelle des UAV, mais sans pour autant perdre le contrôle du cadre des interactions entre UAV. Ainsi, notre approche permet à un utilisateur de spécifier explicitement ce qu'il attend de ces interactions.

VII.1.1 Récapitulatif synthétique

Voici résumées en quelques points les contributions principales de ce travail :

– Une proposition de classification des degrés d'autonomie décisionnelle d'un robot dans un système multi-robots (conception),
– Une architecture exploitant cette classification, et considérant explicitement une délégation incrémentale de capacités décisionnelles aux robots du système (conception et développement partiel),
– Un exécutif générique autorisant différentes configurations de prise de décision (centralisée ou décentralisée), et facilitant l'inclusion de nouveaux robots dans le système (conception, implémentation, expérimentation),

– Une couche délibérative proposant des fonctions de planification et de coordination de tâches pour les UAV :

1. Planification de tâches sur la base d'un planificateur symbolique HTN associé à des raffineurs géométriques (conception, développement du domaine de planification et du lien avec les raffineurs géométriques),
2. Coordination de tâches à base de modèles d'interaction explicites, et mettant en œuvre des modalités de négociation ouvertes (conception, développemenent partiel),

– Un formalisme de spécification (ou de "programmation") des modalités d'interaction (conception, développement partiel).

VII.1.2 Récapitulatif détaillé

Nous avons d'abord introduit une classification en degrés d'autonomie décisionnelle, permettant de décrire différentes configurations de délégation de capacités décisionnelles entre un nœud décisionnel centralisé (par exemple un centre de contrôle) et des nœuds décisionels distribués (en l'occurence les UAV du système). Point commun à tous les degrés d'autonomie, nous avons introduit un exécutif multi-degrés qui supervise de façon transparente des séquences de tâches à exécuter, qu'elles soient élaborées par un système central pour les bas degrés d'autonomie, ou individuellement au niveau des UAV pour les hauts degrés d'autonomie. Dans les bas degrés d'autonomie décisionnelle (1 à 3), la prise de décision proprement dite est réalisée dans le nœud décisionnel centralisé. Dans les hauts degrés (4 et 5), la prise de décision est réalisée au niveau des UAV. La délégation d'autonomie décisionnelle est donc incrémentale, selon les souhaits de l'utilisateur, les besoins du système et le contexte opérationnel.

Pour les hauts degrés d'autonomie décisionnelle, nous proposons une couche délibérative qui répond, selon nous, aux besoins et contraintes des systèmes multi-UAV. Cette couche comprend un *planificateur symbolique* qui dispose du support de primitives de raffinement géométriques dédiées au contexte multi-UAV, que nous appelons *raffineurs spécialisés*. Nous proposons l'utilisation d'un planificateur HTN dans notre approche, car ce paradigme se prête bien à des domaines opérationnels, où il s'agit de raffiner des missions : les connaissances sur le cadre applicatif peuvent en effet être exploitées dans le processus de planification, ce qui permet des performances de planification appréciables. Les raffineurs permettent d'exploiter les données et modèles géométriques dans la planification.

A côté de ce couple planificateur / raffineurs, nous introduisons un gestionnaire d'interactions : celui-ci tient une place importante dans la couche délibérative, puisque c'est sur cette entité que repose le traitement des tâches coopératives. Le gestionnaire d'interactions exploite des modèles d'interactions : il s'agit d'un formalisme pour concevoir des cadres d'interaction. Un modèle d'interaction comprend des rôles, définissant les activités à accomplir, et des modèles de contraintes spatiales, qui permettent de contraindre explicitement l'espace autorisé pour les activités, en fonction du cadre applicatif. Le traitement des rôles peut donner lieu à des sessions de négociations dédiées (la conception des modalités de négociation est ouverte aux besoins et exigences d'un utilisateur), afin de coordonner des activités dans un cadre précis. Le gestionnaire d'interactions permet de traiter des tâches jointes dans un cadre d'exécution courant, et se veut donc compatible avec les contraintes temporelles d'exécution.

En dernier lieu, la couche délibérative repose sur un superviseur en charge de la gestion des activités de celle-ci. Il manipule les plans de tâches de façon à les affiner ou les coordonner en s'appuyant sur le couple planificateur symbolique / raffineurs spécialisés et sur le gestionnaire d'interactions. Il transmet le cas échéant les plans à l'exécutif multi-degrés pour exécution, et peut disposer de moyens de révision de plan, en cas de situation (ou de prévision) d'échec.

VII.1.3 Etat de développement

Dans le contexte du projet COMETS, les bas degrés d'autonomie décisionnelle (1 à 3) ont été testés en expérimentations réelles : il s'agissait d'une prise de décision centralisée, avec transmission de séquences de tâches aux EMD respectifs des UAV. A travers cette architecture, 3 UAV très hétérogènes ont été supervisés de façon transparente pour le donneur d'ordre (le centre de contrôle), et de façon transparente pour les composants propriétaires.

Pour les hauts degrés d'autonomie décisionnelle, du point de vue de la couche délibérative, le couple planificateur / raffineurs a été testé à plusieurs occasions en simulation avec les partenaires du projet (lors de réunions d'intégration logicielle). Enfin, les développements relatifs au gestionnaire d'interactions et le superviseur de CD ont été testés en interne uniquement, leur développement n'étant pas encore finalisé.

VII.2 Limites et perspectives

VII.2.1 Les limites de notre approche

Nous présentons ici des limites que nous reconnaissons à notre approche, du point de vue de l'architecture et du point de vue des cadres d'interaction proposés.

– Validation de la consistance globale : de nombreux sous-systèmes sont définis dans notre architecture, et garantir les propriétés du système intégré représente un travail important : une validation (incrémentale) de l'ensemble demande une formalisation et une couverture de test plus poussée.
– Notre approche requiert une définition explicite du cadre d'interaction. Le système peut parfois manquer de souplesse face à l'adversité : on ne peut compter sur l'émergence de comportements utiles en cas de contingence non prévue. C'est à la fois la force et la faiblesse de cette approche.
– La conception de modèles d'interaction peut s'avérer laborieuse compte tenu de la quantité d'informations à fournir et à corréler. Ce point est l'objet de perspectives (voir la sous-section suivante).

VII.2.2 Perspectives

Différentes extensions sont envisageables pour dépasser certaines des limites précisées ci-avant, et pour enrichir le système proposé.

– Le degré 5 d'autonomie décisionnelle correspond à une gestion autonome cohérente de l'allocation des tâches parmi les UAV du système. Une mission (ou un ensemble de missions)

est présenté comme telle aux UAV, et ceux-ci doivent être en mesure de se répartir les tâches (individuelles ou jointes). Par ailleurs, l'allocation peut être remise en question de façon opportuniste en cours de réalisation de la mission. La mise en place de ce degré d'autonomie nécessite d'étendre notre architecture de façon à ce que les UAV puissent évaluer et négocier l'allocation de missions, éventuellement à l'aide de modalités de négociations du gestionnaire d'interactions.

– Comme nous l'avons mentionné plus haut, la conception de modèles d'interaction pourrait grandement bénéficier d'une interface graphique de programmation intuitive : cela pourrait concerner la plupart des éléments nécessaires aux modèles d'interaction (rôles, modèles de contraintes spatiales...) ainsi que leur assemblage dans ce modèle.

– Nous avons défendu dans cette approche une définition préalable de modèles d'interaction, destinée à fixer explicitement le cadre des activités jointes. Il pourrait néamoins être intéressant de considérer et évaluer la possibilité, pour les UAV, d'apprendre des modèles d'interaction, dans des contextes applicatifs donnés : à partir d'opérations menées à des degrés d'autonomie inférieurs (1 à 3), des modèles d'interactions pourraient être inférés en vue d'opérations autonomes. Il s'agit d'un autre moyen de répondre à la complexité de conception des modèles d'interaction. Ce point est largement spéculatif : il soulève de nouvelles problématiques complexes.

– Dans notre approche, nous avons évoqué différents degrés d'autonomie, mais nous avons très peu évoqué les transitions entre degrés d'autonomie. Dans quelles situations est il pertinent que le système gagne ou perde en autonomie ? Selon quels critères ? Comment doit s'opérer le transfert de "responsabilité" dans un sens ou dans l'autre, vis à vis du système ? Nous n'avons pas abordé ces problèmes, mais ils sont essentiels au déploiement d'un système réel. Pour ceci, le rôle des opérateurs doit bien entendu être considéré : il y a là encore matière à de nombreux développements.

Annexe A

Introduction au planificateur Shop2

Principes de base

En premier lieu, le domaine de Shop2 est défini à partir de méthodes et opérateurs, exprimés de la façon suivante :

Une **méthode** Shop2 est définie comme suit :
– Une ”tête” : modalité d'appel de la méthode (nom, paramètres...).
– Un ensemble de couples : (Ci,Qi), où Ci est un ensemble de préconditions et Qi est l'ensemble des sous-tâches obtenues en appliquant cette méthode.

Un **opérateur** Shop2 est défini comme suit :
– Une ”tête” : modalité d'appel de l'opérateur (nom, paramètres...).
– Un ensemble de préconditions.
– Un ensemble de faits retirés de la base de faits Shop2.
– Un ensemble de faits ajoutés à la base de faits Shop2.
– Un coût numérique d'application de cet opérateur.

Les figures A.1 A.2 ci-après illustrent un exemple simple de définition d'un domaine et d'un problème Shop2 dans un cadre inspiré du domaine COMETS :

Ce domaine décrit les opérateurs pour décoller (takeoff) et pour se rendre d'un endroit à un autre (goto). Le coût fixe (50 unités) pour le goto sert uniquement les besoins pour cet exemple simple : les fonctionnalités de Shop2 permettent en effet de décrire ce coût comme une variable dont la valeur peut dépendre du contexte (ici, il s'agirait d'évaluer la distance entre deux lieux).

Le problème présenté consiste à amener l'UAV Karma de Toulouse à Lousa, sachant que Karma est initialement sur le sol.

```
;;;Domain's operators
;;;------------------

;;;operator's head
(:operator (!takeoff ?uav)
        ;;;preconditions
          ((grounded ?uav))
        ;;;delete list
          ((grounded ?uav))
        ;;;add list
          ((flying ?uav))
        ;;;cost
          15
)

;;;operator's head
(:operator (!goto ?uav ?starting-loc ?ending-loc)
        ;;;preconditions
          ((flying ?uav)
           (location ?starting-loc ?uav))
        ;;;delete list
          ((location ?starting-loc ?uav))
        ;;;add list
          ((location ?ending-loc ?uav))
        ;;;cost
          50
        )
```

```
;;;Domain's methods
;;;----------------

;;;methods's head
(:method (general-goto ?uav ?ending-loc)

        ;;;1st modality: if already flying
        ;;;preconditions
          ((flying ?uav)
           (location ?starting-loc ?uav))
        ;;;tail / decomposition scheme
          ((!goto ?uav ?starting-loc ?ending-loc))

        ;;;2nd modality: not yet flying
        ;;;preconditions
          ((grounded ?uav)
           (location ?starting-loc ?uav))
        ;;;tail / decomposition scheme
          ((!takeoff ?uav)
           (!goto ?uav ?starting-loc ?ending-loc))
        )

)
```

FIG. A.1 – Exemple simple de domaine Shop2

Le plan obtenu, sans surprise, comprend un “takeoff” suivi d’un “goto”. Dans la méthode “general-goto”, c’est la seconde modalité qui a pu être appliquée, compte tenu de l’état initial (on-ground Karma). Si l’état de Karma avait été “flying” dans le problème, alors le planificateur aurait appliqué la première modalité du “general-goto”, conduisant uniquement à un “goto”, sans “takeoff” préalable.

Le mécanisme de base de Shop2 peut être enrichi à travers différents mécanismes subsidiaires : les sous-sections suivantes introduisent la gestion de domaine temporel, et l’utilisation en cours de recherche de routines externes au planificateur.

Domaine temporel avec Shop2

La gestion du temps nécessite l’utilisation d’une technique que les développeurs de Shop2 appellent *timeline* : il s’agit d’attacher à chaque ressource du problème une marque qui permette de conserver la dernière date d’utilisation de cette ressource. Ainsi, un opérateur Shop2 qui utilise une ressource (simple prise de connaissance de l’état de cette ressource / lecture, ou modification effective de l’état de cette ressource / écriture) marque celle-ci, et l’opérateur suivant qui aura besoin (en lecture ou écriture) de cette ressource prendra en considération la date attachée à

```
;;; Problem definition
;;;-------------------------

(defproblem simple1 comets-simple

    ;;;Initial set of facts (state hypothesis)
    ((location Toulouse karma)
    (grounded karma))

    ;;;Goal
    ((general-goto karma Lousa))
)
```

```
* (find-plans 'simple1)

------------------------------------------------------------------------
Problem SIMPLE1 with :WHICH = :FIRST, :VERBOSE = 3
Totals: Plans Mincost Maxcost Expansions Inferences  CPU time  Real time
             1      65      65          4          6     0.000      0.010
Plans:
(((!TAKEOFF KARMA) 15 (!GOTO KARMA TOULOUSE LOUSA) 50))
```

FIG. A.2 – Exemple simple de problème et résultat obtenu

la dernière marque. De plus, un opérateur qui n'utilise pas cette ressource (ni en lecture, ni en écriture), sera considéré comme une tâche indépendante temporellement de celles qui modifient la ressource : en conséquence, il se peut très bien que la tâche associée à cet opérateur puisse s'exécuter en parallèle de celles utilisant la ressource.

Les figures A.3 et A.4 ci-après illustrent la mise en oeuvre d'une tâche de déplacement (goto) utilisant la ressource position-UAV, et d'une tâche de perception (takeshot) utilisant la ressource visual-camera. Elles peuvent être exécutées simultanément dans la mesure où il n'y pas de conflit de ressource. La source de la durée estimée des tâches n'est pas précisée dans cet exemple : elle est obtenue par l'utilisation de fonctions de calcul auxiliaires, dont il est question dans le paragraphe suivant.

Il est à noter que dans cet exemple, la tache de perception "takeshot" planifiée a une date de début égale à celle du "takeoff", puisque les tâches n'interfèrent pas sur leurs ressources (déplacement de l'UAV pour "takeoff" et "goto", et utilisation de la camera omnidirectionnelle pour "takeshot"). Si le but de l'opération avait été de déclencher le début des perceptions au moment où l'UAV est en l'air (takeoff terminé), il aurait fallu prendre en considération la timeline affectant la position de l'UAV, dans la tâche de perception.

```
;;;Domain's operators
;;;------------------

(:operator (!takeoff ?uav ?start ?duration)
      ;;;preconditions
         ((grounded ?uav) (assign ?duration 15)
          (my-write-time uavloc ?t1)
          (my-read-time uavloc ?t2)
          (assign ?start (max '?t1 '?t2))
          (assign ?end (+ '?start '?duration))
          (assign ?cost 20))
        ;;;delete list
         ((grounded ?uav)
          (my-write-time uavloc ?t1)
          (my-read-time uavloc ?t2))
        ;;;add list
         ((flying ?uav)
          (my-write-time uavloc ?end)
          (my-read-time uavloc ?end))
        ;;;cost
         ?cost)

(:operator (!goto ?uav ?start-loc ?end-loc ?start ?duration)
        ;;;preconditions
         ((flying ?uav)  (location ?start-loc ?uav)
          (my-write-time uavloc ?t1)
          (my-read-time uavloc ?t2)
          (assign ?start (max '?t1 '?t2))
          (assign ?end (+ '?start '?duration))
          (assign ?cost (* 10 '?duration)))
        ;;;delete list
         ((location ?start-loc ?uav)
          (my-write-time uavloc ?t1)
          (my-read-time uavloc ?t2))
        ;;;add list
         ((location ?end-loc ?uav)
          (my-write-time uavloc ?end)
          (my-read-time uavloc ?end))
        ;;;cost
         ?cost)

(:operator (!takeshot ?uav ?percep-device ?start ?duration)
        ;;;preconditions
         ((perception-device free ?percep-device)
          (my-write-time perception-device ?percepdev ?t1)
          (my-read-time perception-device ?percepdev ?t2)
          (assign ?start (max '?t1 '?t2))
          (assign ?end (+ '?start '?duration))
          (assign ?cost (* 2 '?duration)))
        ;;;delete list
         ((perception-device free ?percep-device)
          (my-write-time perception-device ?percepdev ?t1)
          (my-read-time perception-device ?percepdev ?t2))
        ;;;add list
         ((perception-device busy ?percep-device)
          (my-write-time perception-device ?percepdev ?end)
          (my-read-time perception-device ?percepdev ?end))
        ;;;cost
         ?cost)
```

```
;;;Domain's methods
;;;----------------

(:method (general-goto ?uav ?end-loc ?duration)

      ;;;1st modality: if already flying
       ;;;preconditions
        ((flying ?uav)
         (location ?start-loc ?uav))
       ;;;tail / decomposition scheme
        ((!goto ?uav ?start-loc ?end-loc
                ?start ?duration))

      ;;;2nd modality: not yet flying
       ;;;preconditions
        ((grounded ?uav)
         (location ?start-loc ?uav))
       ;;;tail / decomposition scheme
        ((!takeoff ?uav ?start2 ?duration2)
         (!goto ?uav ?start-loc ?end-loc
                ?start ?duration))
        )


(:method (goto-and-takeshot ?uav ?percep-device
                            ?end-loc ?duration)
       ;;;preconditions
        ((location ?start-loc ?uav)
         (perception-device free ?percep-device))

       ;;;tail / decomposition scheme
        ((general-goto ?uav ?end-loc ?duration)
         (!takeshot ?uav ?percep-device
                    ?start ?duration))
        )
```

FIG. A.3 – Exemple de domaine Shop2 avec des “timelines”

```
;;; Problem definition
;;;-------------------------

(defproblem timelines comets-timelines

  ;;;Initial set of facts (state hypothesis)
  ((location toulouse karma)
   (grounded karma)
   (my-write-time uavloc 0)
   (my-read-time uavloc 0)
   (my-write-time perception-device omni-cam 0)
   (my-read-time perception-device omni-cam 0)
   (perception-device free omni-cam))

  ;;;Goals
  ((goto-and-takeshot karma omni-cam lousa 130))
)
```

```
* (find-plans 'timelines)

------------------------------------------------------------------------
Problem TIMELINES with :WHICH = :FIRST, :VERBOSE = 3
Totals: Plans Mincost Maxcost Expansions Inferences  CPU time  Real time
              1     1580     1580              6           25       0.010       0.000
Plans:
(((!TAKEOFF KARMA 0 15) 20
  (!GOTO KARMA TOULOUSE LOUSA 15 130) 1300
  (!TAKESHOT KARMA OMNI-CAM 0 130) 260))
```

FIG. A.4 – Exemple de problème et solution associée dans le domaine avec “timelines”

Annexe B

Travaux expérimentaux pour la couverture de zone

Nous présentons dans cette section quelques résultats liés à notre problématique, et issus de travaux réalisés pendant une visite de 3 mois au laboratoire AIICS de l'université de Linköping, en Suède, durant l'été 2004.

WITAS - tour d'horizon

L'université de Linköping est à l'origine d'un projet recherche en robotique aérienne du nom de WITAS, qui s'est conclu par des expérimentations réelles sur un UAV (Hélicoptère Yamaha R-MAX) début 2004. L'objectif du projet était multiple, considérant d'une part le contrôle, le traitement d'images, et l'interface homme-machine (à travers un système de traitement de la parole, en particulier), mais aussi des aspects architecturaux (logiciels), jusqu'à la décision.

L'architecture logicielle développée est une architecture en couche, où les composants s'échangent des données avec un principe de *services*. Le support de communications employé est à base de Corba. A travers ce système, les traitements sont répartis indifférement entre le sol et l'appareil.

Application testée - mise en oeuvre

Les moyens conséquents mis en oeuvre dans le projet WITAS ont permis à AIICS d'acquérir 2 hélicoptères Yamaha R-MAX : ce laboratoire s'intéresse et s'investit dans les systèmes multi-UAV depuis maintenant une année.

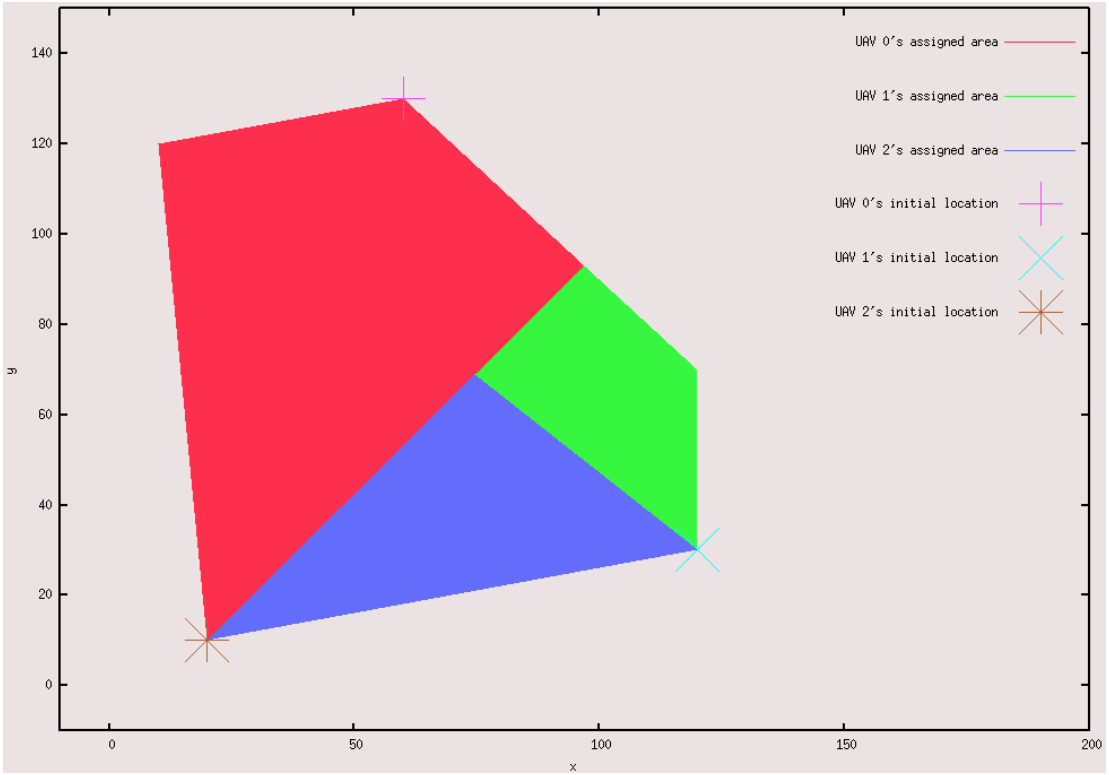


FIG. B.1 – Partage de zone, résultat en simulation

Durant ce séjour de 3 mois, nous avons participé à la conception et à l'implémentation dans l'architecture d'un module élémentaire de communication haut niveau entre UAV, avec une application directe au partage de zone (pour distribuer des tâches de "couverture", pour de la cartographie de zone par exemple).

L'algorithme de partage de zone mis en oeuvre est inspiré des travaux de [Mazza 04], dans lesquels les auteurs suggèrent d'employer, dans le cadre de systèmes d'UAV, un algorithme déterministe de partage de zone pour zone convexe, qui permet de séparer une zone en autant de sous-zones que souhaité, compte tenu des proportions respectives des sous-zones que l'on veut obtenir. Cet algorithme est centralisé, mais nécessite des informations en provenance des chacun des UAV en présence, pour pouvoir être appliqué (capacités de couvertures...). Des échanges d'informations cohérents doivent avoir lieu lorsqu'une entité doit appliquer le processus de partage de zone, en vue d'une mission de couverture.

Avec l'implémentation que nous avons mise en oeuvre (C++ avec différentes plateformes cibles : sparc-solaris, linux, macOS X), des séries de tests ont permis de vérifier la robustesse de l'approche. Deux modes de spécification des données sont possibles : soit explicitement fourni par un utilisateur (données relatives à chacun des UAV), soit échangées entre plusieurs instance du système implémenté (des primitives de communication relativement simples ont été développées à cette occasion : ce sont celles que nous proposons à présent dans le gestionnaire d'interaction, comme support de communication aux modalités de négociations).

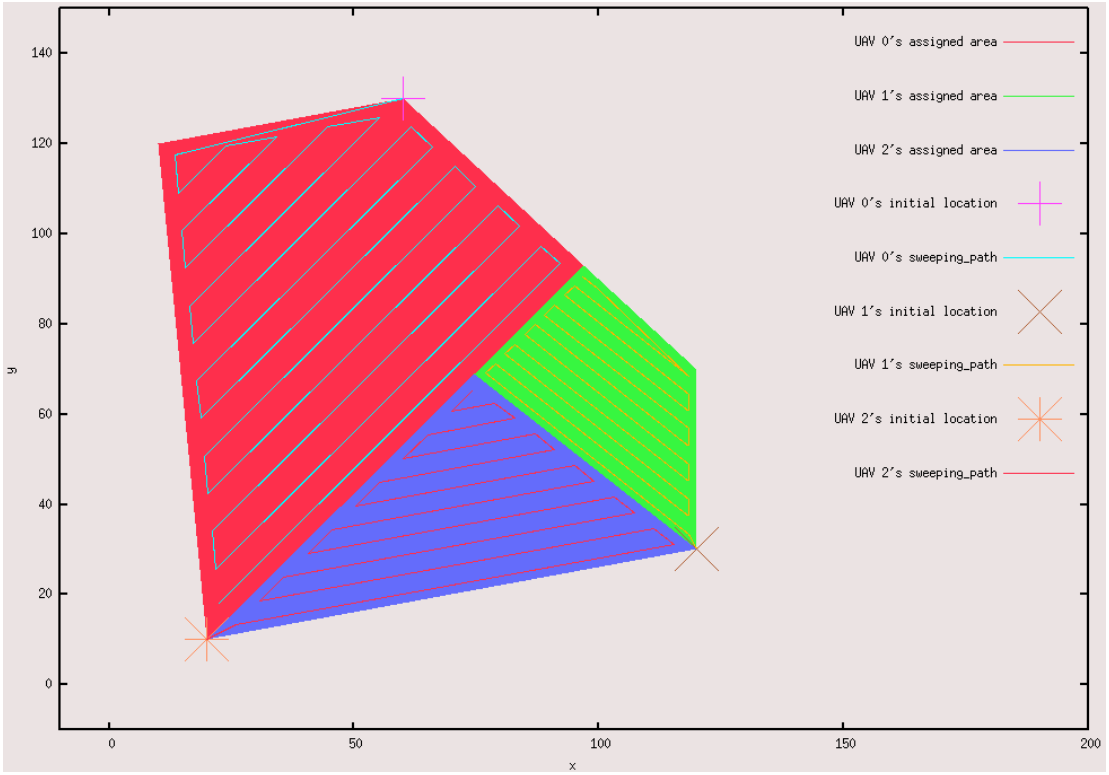


FIG. B.2 – Application de motifs de chemin de survol

En simulation, des fichiers gnuplot sont générés. Les figures B.1 et B.2 illustre des résultats obtenus, pour trois UAV ayant des capacités différentes (vitesse, ouverture de caméra, hauteur de survol...), devant cartographier une zone convexe donnée. La première figure montre le découpage obtenu, avec l'algorithme de partage employé.

La deuxième figure (fig. B.2) illustre l'application d'un motif de parcours, pour chaque UAV, de la zone qui lui est attibué. Ce motif prend en considération la largeur de "balayage" de chacun des UAV, pendant leurs navigations, ainsi que les positions initiales respectives.

Ce module a été intégré dans l'architecture Corba du laboratoire, et utilisé avec succès lors d'essais réels avec deux hélicoptères yamaha : les trajectoires respectives ont été générées et transmises aux deux UAV, qui ont réalisé le suivi de trajectoire correspondant, terminant leur parcourt quasiment en même temps.

Enseignements tirés

Les développements réalisés lors de ce séjour s'inscrivent dans la ligne de pensée du GI. Chronologiquement, ces développements sont venus conforter les idées préliminaires dont nous disposions sur ce que devrait être un gestionnaire d'interactions dans une architecture multi-UAV : un système permettant différentes modalités de négociations, celles-ci devant s'inscrire dans un cadre de gestion d'interactions parfaitement maitrîsé. Ce en quoi, selon nous, le para-

digme des modèles d'interaction explicites répond de façon adéquate.

Annexe C

Exemple de référence : automates d'exécution

Cette annexe regroupe les illustrations des automates d'exécution, pour l'exemple de référence des chapitre IV et V.

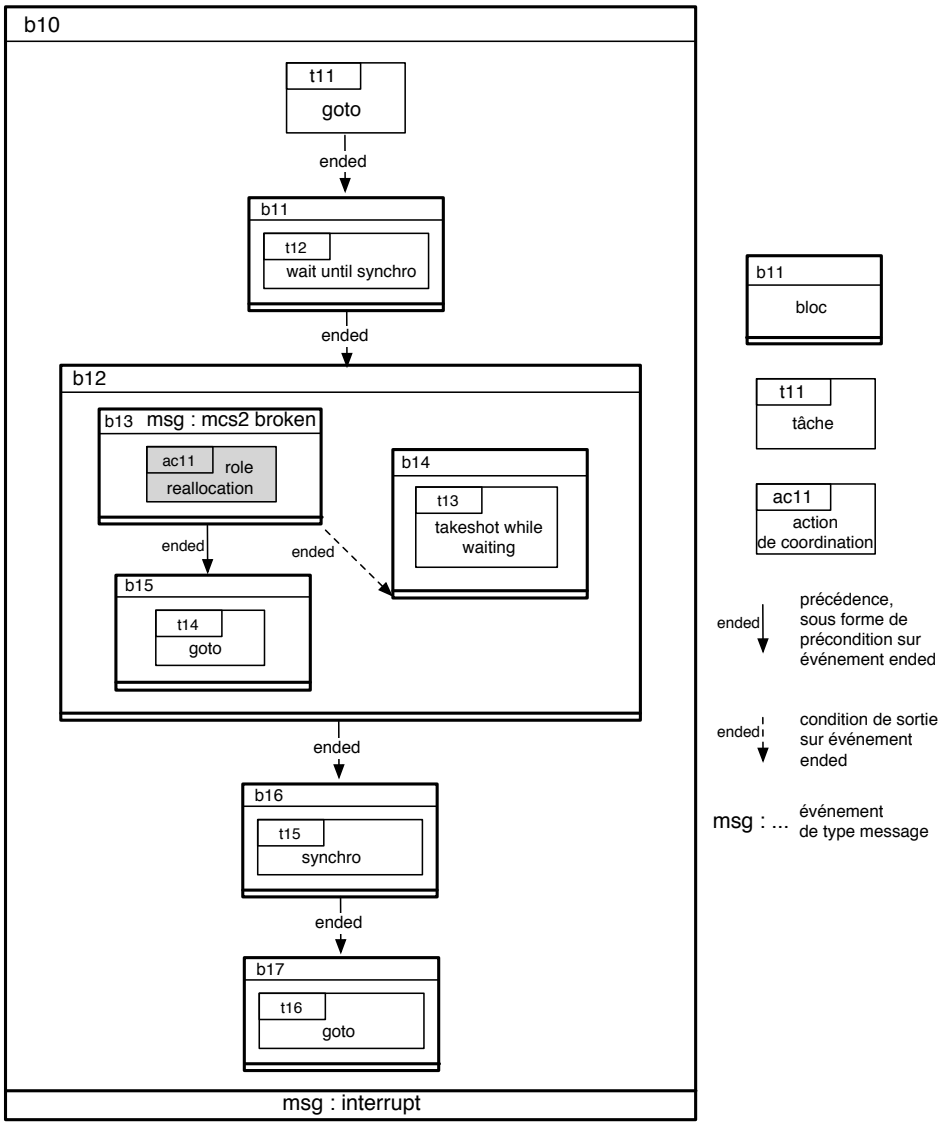


FIG. C.1 – Automate d'exécution du rôle R1

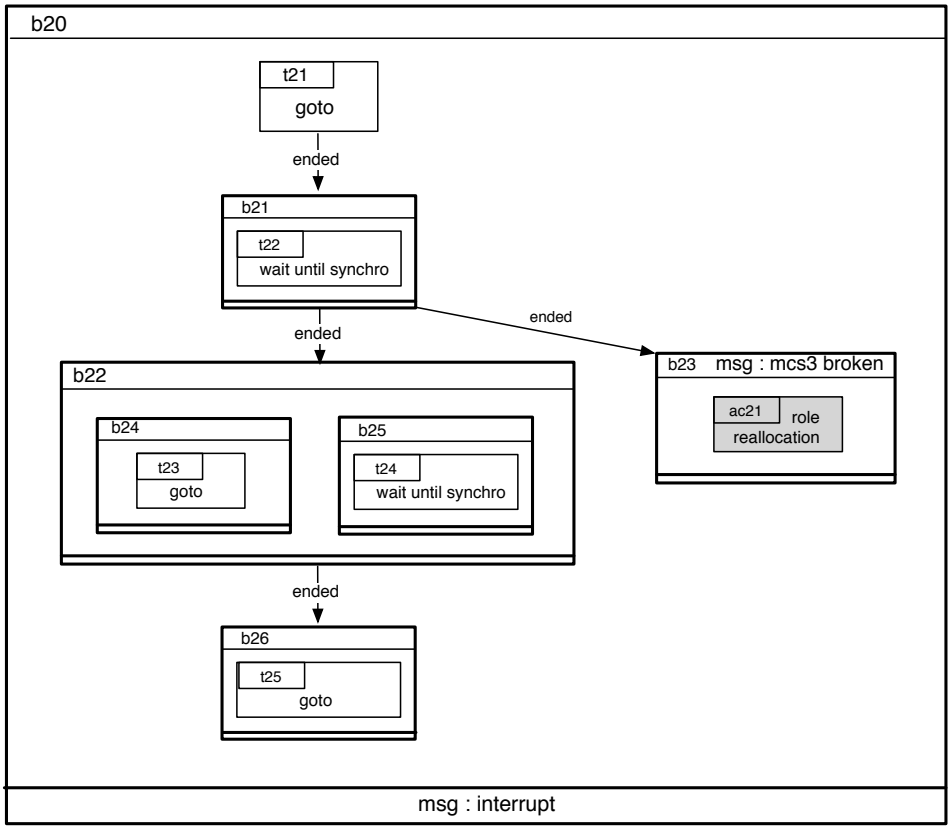


FIG. C.2 – Automate d'exécution du rôle R2

Glossaire

AC : Action de Coordination. Instruction de l'automate d'exécution (AE) d'un rôle qui fait appel à une modalité de négociation (MN).

AE : Automate d'Execution. Ensemble des instructions définies au sein d'un rôle.

CC : Centre de Contrôle. Entité au sol en charge de la prise de décision dans une configuration centralisée du système.

CD : Couche Décisionnelle. Dans un noeud décisionnel distribué (NDD), ensemble des composants permettant la prise de décision.

CS : Contrainte Spatiale. Contrainte de distance ou d'orientation définie explicitement.

CNP : "Contract Net Protocol". Protocole de négociation à base d'engagement (contrats) dans un système d'agent, évoqué à l'origine par Smith [Smith 80].

CPE : Composants Propriétaires Embarqués. Partie propriétaire des UAV qui est nécessairement embarquée dans l'UAV.

EMD : Exécutif Multi Degrés. Exécutif potentiellement embarquable assurant le déroulement des plans générés, qu'ils soit produits par un noeud décisionnel centralisé (NDC) ou distribué (NDD), donc quelque soit le degré d'autonomie décisionnelle des robots du système.

GI : Gestionnaire d'Interactions. Elément de la couche délibérative (CD) dédié au raffinement des tâches jointes (TJ) et aux démarches de coordination, avec les GI des autres robots.

GN : Gestionnaire de Négociations. Elément du gestionnaire d'interactions (GI) en charge du traitement des modalités de négociartion (MI).

HTN : "Hierarchical Task Network", ou réseau hiérarchique de tâches. Représentation hiérarchique de tâches selon leurs décompositions en sous-tâches.

MCS : Modèle de Contraintes Spatiales. Ensemble de contraintes spatiales qui peut être lié aux blocs, dans les rôles.

MI : Modèle d'Interaction. Modèle explicite d'interaction, exprimée sous la forme d'un ensemble de variantes dont chacune consiste en la déclaration d'un certain nombre de rôles et de modèles de contraintes spatiales (MCS), et d'une table de coordination (TC).

MN : Modalité de Négociation. Moyen, protocole de négociation utilisable dans le gestionnaire d'interaction (GI) pour raffiner les tâche jointes (TJ) avec les autres UAV.

NDC : Noeud Décisionnel Centralisé. Toute entité en mesure de réaliser de façon centralisée la prise de décision pour un ensemble de robots.

NDD : Noeud Décisionnel Distribué. Entité en mesure de prendre des décisions en coopération avec d'autres noeuds décisionnels distribiués NDD.

PS : Planificateur Symbolique. Composant de génération de plan de tâches, dans la couche délibérative (CD).

RS : Raffineurs Spécialisés. Composant de raffinement géométriques de tâches, dans la couche délibérative (CD).

SCD : Superviseur de Couche Délibérative. Supervise l'activité de la couche délibérative (CD).

SMC : Système de Monitoring et de Contrôle. Dans COMETS, composante du centre de contrôle (CC) chargée de présenter les données de télémétrie et de perception auprès des utilisateurs du système.

SP : Système de Perception. Dans COMETS, entité du segment sol en charge du traitement des données (détection / confirmation d'alarmes, modélisation du feu…) de perceptions reçues de tous les UAV.

SPM : Système de Planification de Mission. Dans COMETS, composante du centre de contrôle CC chargée de la planification semi-automatique de missions.

STN : "Simple Temporal Network", ou réseau de contraintes temporelles. Représentation particulière des relations temporelles entre tâches.

STNU : "Simple Temporal Network with Uncertainty", ou réseau de contraintes temporelles avec incertitude. Représentation particulière des relations temporelles entre tâches.

TC : Table de Coordination. Support de mise en correspondance explicite entre les rôles entre eux d'une part, et entre les rôles et les modèles de contraintes spatiales (MCS) d'autre part.

TI : Tâche individuelle. Tâche pouvant être raffinées dans un cadre mono-robot.

TIE : Tâche individuelle élémentaire. Tâche individuelle ne pouvant être raffinée d'avantage, et potentiellement exécutable.

TJ : Tâche Jointe. Tâche nécessitant une activité conjointe d'un certain nombre d'UAV, et ne pouvant pas être raffinée/décomposée dans le seul contexte mono-robot d'un robot donné.

TJE : Tâche Jointe Elémentaire. Tâche jointe devant être traitée par le gestionnaire d'interactions dans un contexte multi-robots, afin d'être raffinée.

TSP : "Traveller Salesman Problem", ou problème du voyageur de commerce. Problème classique d'optimisation, dans lequel il s'agit de relier un certain nombres de noeuds dans un graphe en minimisant les coûts (étiquetés sur les arêtes du graphe).

UAV : "Unmanned Aerial Vehicle", ou véhicule aérien non-habité.

Références bibliographiques

[Alami 93] R. Alami, R. Chatila & B. Espiau. *Designing an intelligent control architecture for autonomous robots*. In 6th International Conference on Advanced Robotics (ICAR'93), pages 435–440, 1993.

[Alami 95] R. Alami, F. Robert, F. Ingrand & S. Suzuki. *Multi-robot cooperation through incremental plan-merging*. In Proceedings of the International Conference on Robotics and Automation (ICRA'95), pages 2573–2579, 1995.

[Alami 98a] R. Alami, R. Chatila, S. Fleury, M. Ghallab & F. Ingrand. *An Architecture for Autonomy*. International Journal of Robotics Research, Special Issue on "Integrated Architectures for Robot Control and Programming", vol. 17, no. 4, pages 315–337, 1998.

[Alami 98b] R. Alami, S. Fleury, M. Herrb, F. Ingrand & F. Robert. *Multi Robot Cooperation in the Martha Project*. IEEE Robotics and Automation Magazine - Special Issue on "Robotics & Automation in the European Union", vol. 5, no. 1, 1998.

[Albus 89] J. S. Albus, H. G. McCain & R. Lumia. *NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)*. NIST TN (Supersedes NBS Technical Note 1235, July 1987), 1989.

[Alighanbari 05] M. Alighanbari & J. P. How. *Cooperative Task Assignement of Unmanned Aerial Vehicles in Adversarial Environments*. In To appear at the 2005 IEEE ACC, 2005.

[Arkin 92] R. C. Arkin. *Cooperation without Communication : Multiagent Schema-Based Robot Navigation*. Journal of Robotic Systems, vol. 9, no. 3, pages 351–364, 1992.

[AUVSI 05] AUVSI. *Association for Unmanned Vehicle Systems International (AUVSI) 's International Aerial Robotics Competition*, 2005, http ://avdil.gtri.gatech.edu/auvs/iarclaunchpoint.html.

[B. Damas 04] P. Lima B. Damas. *Stochastic Discrete Event Model of a Multi-Robot Team Playing an Adversarial Game*. In Proceedings of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles - IAV2004, Lisboa, Portugal, 2004.

[Bacchus 01] F. Bacchus & M. Ady. *Planning with Ressources and Concurrency. A Forward Chaining Approach*. In In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'01), 2001.

[Balch 95] T. Balch & R. C. Arkin. *Communication in reactive multiagent robotic systems*. Autonomous Robots, vol. 1, no. 1, pages 27–52, 1995.

[Bonasso 97] R. P. Bonasso, J. Firby, E. Gat, D. Kortenkamp, D. P. Miller & M. G. Slack. *Experiences with an Architecture for Intelligent, Reactive Agents*. Journal of Experimental and Theoretical Artificial Intelligence, vol. 9, no. 2/3, pages 237–256, 1997.

[Botelho 99] S. C. Botelho & R. Alami. *M+ : A Scheme for Multi-robot Cooperation Through Negotiated Task Allocation and Achievement*. In Proceedings of the International Conference on Robotics and Automation (ICRA'99), 1999.

[Brenner 03] M. Brenner. *A Multiagent Planning Language*. In Workshop on PDDL, ICAPS'03, Trento, Italy, 2003.

[Brooks 86] R. A. Brooks. *A Robust Layered Control System for a Mobile Robot.* IEEE Journal of Robotics and Automation, vol. 2, no. 1, pages 14–23, 1986.

[Burgard 00] W. Burgard, M. Moors, D. Fox, R. Simmons & S. Thrun. *Collaborative Multi-Robot Exploration*. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'00), 2000.

[Caloud 90] P. Caloud, W. Choi, J-C. Latombe, C. Le Pape & M. Yim. *Indoor Automation with Many Mobile Robots*. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 67– 72, 1990.

[Chaimowicz 01] L. Chaimowicz, T. Sugar, V. V. Kumar & M.F.M. Campos. *An Architecture for Tightly Coupled Multi- Robot Cooperation*. In IEEE International Conference on Robotics and Automation, 2001.

[Chatila 92] R. Chatila, R. Alami, B. Degallaix & H. Laruelle. *Integrated planning and execution control of autonomous robot actions*. In IEEE Internationl Conference on Robotics and Automation, pages 2689–2696, 1992.

[Chien 00] S. Chien, R. Knight, A. Stechert, R. Sherwood & G. Rabideau. *Using Iterative Repair to Improve the Responsiveness of Planning and Scheduling*. In Proceedings of the International Conference on Planning and Scheduling (AIPS'00), 2000.

[CMUCL 05] CMUCL. *CMUCL web page*, 2005, http ://www.cons.org/cmucl/.

[COMETS 05] COMETS. *Site internet du projet COMETS*, 2005, http ://www.comets-uavs.org.

[de Weerdt 03] M. M. de Weerdt. *Plan Merging in Multi-Agent System*. PhD thesis, Delft University of Technology, Delft, The Netherlands, 2003.

[Dias 04] B. Dias. *TraderBots : A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments*. PhD thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania, 2004.

[Doherty 99] P. Doherty & J. Kvarnström. *TALplanner : An Empirical Investigation of a Temporal Logic-based Forward Chaining Planner*. In Proceedings of the 6th International Workshop on the Temporal Representation and Reasoning (TIME'99), 1999.

[Doherty 00] P. Doherty, G. Granlund, K. Kuchcinski, E. Sandewall, K. Nordberg, E. Skarman & J. Wiklund. *The WITAS Unmanned Aerial Vehicle Project*. In Proc. of the 14th European Conference on Artificial Intelligence (ECAI'00), pages 747–755, Berlin, Germany, 2000.

[Dudek 02] G. Dudek, M. Jenkins & E. Milios. Robot teams - from diversity to polymorphism, chapitre A Taxonomy of Multirobot Systems, pages 3–35. Tucker Balch and Lynne E. Parker, 2002.

[Ephrati 93] E. Ephrati & J. S. Rosenschein. *Multi-agent planning as the process of merging distributed sub-plans.* In Proceedings of the Twelfth International Workshop on Distributed Artificial Intelligence (DAI'93), pages 115–129, 1993.

[Ferber 95] J. Ferber. Les systèmes multi-agents, vers une intelligence collective. InterEditions, 1995.

[Finin 97] T. Finin, Y. Labrou & J. Mayfields. Software agents, chapitre KQML as an agent communication language. Jeff Bradshaw, MIT press, 1997.

[Fleury 97] S. Fleury, M. Herrb & R. Chatila. *GenoM : a Tool for the Specification and the Implementation of Operating Modules in a Distributed Robot Architecture*. In In proceedings of the IEEE Int. Conference on Intelligent Robots and Systems (IROS'97), 1997.

[Frank 00] J. Frank, A. Jonsson & P. Morris. *On the representation of mutual exclusion constraints for planning*. In In proceedings of the Symposium on Abstraction, Reformulation and Approximation, 2000.

[Frank 01] J. Frank, A. Jonsson, R. Morris & D. Smith. *Planning and Scheduling for Fleets of Earth Observing Satellites*. In Proceedings of the 6th International Symposium on Artificial Intelligence, Robotics and Autonomation in Space (i-SAIRAS'01), 2001.

[Furukawa 04] T. Furukawa, F. Bourgault, H. F. Durrant-Whyte & G. Dissanayake. *Dynamic Allocation and Control of Coordinated UAVs to Engage Multiple Targets in a Time-Optimal Manner*. In Proceedings of the IEEE Int. Conference on Robotics and Automation (ICRA'04), pages 2352–2358, 2004.

[Gancet 04] J. Gancet & S. Lacroix. *Embedding heterogeneous levels of decisional autonomy in multi-robot systems*. In Proc. of DARS'04, 2004.

[Gancet 05] J. Gancet, G. Hattenberger, R. Alami & S. Lacroix. *Task planning and control for a multi-UAV system : architecture and algorithms*. In To appear in the Proc. of the IEEE Int. Conference on Intelligent Robots and Systems (IROS'05), 2005.

[Gat 91] E. Gat. *Integrating planning and reacting in a heterogeneous asynchronous architecture for mobile robots*. SIGART Bulletin, vol. 2, pages 17–74, 1991.

[GDHE 05] GDHE. *LAAS Open Software for Autonomous Systems : Gdhe*, 2005, http ://softs.laas.fr/openrobots/tools/gdhe.php.

[Gerkey 02] B. P. Gerkey & M. J. Mataric. *Sold ! : Auction methods for multi-robot coordination*. IEEE Transactions on Robotics and Automation,,special issue on Advances in Multi-Robot Systems, vol. 18, no. 5, pages 758–786, 2002.

[Ghallab 04] M. Ghallab, Nau D. & Traverso P. Automated planning, theory and practice. Morgan Kaufmann Publishers, 2004.

[Hattenberger 04] G. Hattenberger. Planification et affinement de tâches pour un ensemble de drones autonomes. Master's thesis, ENSICA - EDSYS, stage de DEA réalisé au LAAS-CNRS, 2004.

[How 04] J. P. How, E. King & Y. Kuwata. *Flight Demonstration of Cooperative Control for UAV Teams*. In AIAA 3rd Unmanned Unlimited Technical Conference, Workshop and Exhibit, Chicago, IL, 2004.

[Ingrand 92] F. Ingrand, M. P. Georgeff & A. S. Rao. *An Architecture for Real-Time Reasoning and System Control*. IEEE Expert, vol. 7, no. 6, pages 33–44, December 1992.

[Ingrand 05a] F. Ingrand. *LAAS Open Software for Autonomous Systems*, 2005, http ://softs.laas.fr/openrobots/tools/openprs.php.

[Ingrand 05b] F. Ingrand. *page de Felix Ingrand, concepteur de PRS*, 2005, http ://www.laas.fr/ felix.

[Kalra 03] N. Kalra & A. Stentz. *A Market Approach to Tightly-Coupled Multi-Robot Coordination : First Results*. In Proceedings of the ARL Collaborative Technologies Alliance Symposium, May 2003.

[Kalra 05] N. Kalra, D. Ferguson & A. Stentz. *Hoplites : A market-based framework for complex tight coordination in multirobot teams.* In To appear in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), April 2005.

[Kortenkamp 98] D. Kortenkamp, R. Bonasso & R. Murphy. Artificial intelligence and mobile robots : Case studies of successful robot systems. AAAI Press, 1998.

[Kvarnström 01] J. Kvarnström & P. Doherty. *TALplanner : A Temporal Logic Based Forward Chaining Planner*. Annals of Mathematics and Artificial Intelligence (AMAI), vol. 30, pages 119–169, 2001.

[Laborie 95] P. Laborie. *IxTeT : une approche intégrée pour la gestion de ressources et la synthèse de plans*. PhD thesis, Ecole Nationale Supérieure des Télécommunications, 1995.

[Laengle 98] T. Laengle, T. C. Lueth, U. Rembold & H. Woern. *A distributed control architecture for autonomous mobile robots implementation of the Karlsruhe Multi-Agent Robot Architecture (KAMARA)*. Advanced Robotics, vol. 12, no. 4, pages 411–431, 1998.

[Lemai 04] S. Lemai & F. Ingrand. *Interleaving Temporal Planning and Execution in Robotics Domains*. In In proceedings of the National Conference on Artificial Intelligence (AAAI'04), 2004.

[Lemaire 04] T. Lemaire, R. Alami & S. Lacroix. *A Distributed Tasks Allocation Scheme in Multi-UAV Context*. In Proc. of the Int. Conference on Robotic and Automation (ICRA'04), 2004.

[Levesque 90] H. J. Levesque, P. R. Cohen & J. Numes. *On acting together*. In AAAI press, editeur, In proceedings of the National Conference on Artificial Intelligence (AAAI'90), 1990.

[Lundh 04] R. Lundh, L. Karlsson & A. Saffiotti. *Dynamic Configuration of a Team of Robots*. In Proc. of the European Conference on Artificial Intelligence (ECAI'04) Workshop on Agents in Dynamic and Real-Time Environments, 2004.

[Mataric 95] Maja J Mataric. *Issues and Approaches in the Design of Collective Autonomous Agents*. Robotics and Autonomous Systems, vol. 16, no. 2-4, pages 321–331, 1995.

[Mazza 04] I. Mazza & A. Ollero. *Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms*. In Proc. of DARS'04, 2004.

[Meyer 04] T. Meyer, F. Norman, R. Kwok & D. Zhang. *Logical foundations of negotiation : outcome, concession and adaptation*. In Proceedings of the 19th National Conference on Artificial Intelligence (AAAI'04), pages 293–298, 2004.

[Mintzberg 79] H. Mintzberg. The structuring of organizations. Prentice-Hall, Englewoods Cliffs, N.J., 1979.

[Muscettola 02] N. Muscettola, G. A. Dorais, C. Fry, R. Levinson & C. Plaunt. *IDEA : Planning at the Core of Autonomous Reactive Agents*. In Proceedings of the 3rd Internationsal NASA Workshop on Planning and Scheduling for Space, October 2002.

[Musial 01] M. Musial, U. W. Brandenburg & G. Hommel. *Inexpensive System Design : The Flying Robot MARVIN*. In 16th Int. Unmanned Air Vehicle System Conference (UAVs),, volume 23, pages 1–12, 2001.

[Nair 03] R. Nair, M. Tambe & S. Marsella. *Role Allocation and Reallocation in Multiagent Teams : Toward a Practical Analysis*. In Proceedings of the second International Joint Conference on Agents and Multiagent Systems (AAMAS), 2003.

[Nau 03] D. Nau, T.C. Au, O. Ilghami, U. Kuter, W. Murdock, D. Wu & F. Yaman. *SHOP2 : An HTN Planning System*. Artificial Intelligence Research, vol. 20, pages 379–404, 2003.

[Ollero 04a] A. Ollero, G. Hommel, J. Gancet, L.G. Gutierrez, D.X. Viegas, P.E. Forssen & M.A. Gonzalez. *COMETS : A multiple heterogeneous UAV system*. In Proc. of SSRR'04, Bonn, Germany, 2004.

[Ollero 04b] A. Ollero *et al*. *Architecture and perception issues in the COMETS multi-UAV project*. IEEE Robotics and Automation Magazine, special issue on R & A in Europe : Projects funded by the Commission of the E.U., 2004.

[Parker 93] L. E. Parker. *Designing control laws for cooperative agent teams*. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'93), 1993.

[Pellier 05] D. Pellier & H. Fiorino. *Multi-Agent Assumption-Based Planning*. In To appear in the Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'05), 2005.

[Remuß 04] V. Remuß, M. Musial & U. W. Brandenburg. *BBBCS - Robust Communication System for Distributed Systems*. In Proc. of the IEEE International Workshop on Safety, Security and Rescue Robotics (SSRR'04), 2004.

[Robocup 05] Robocup. *RoboCup official web page*, 2005, http ://www.robocup.org.

[Schumacher 00] C. J. Schumacher & S. N. Singh. *Nonlinear Control of Multiple UAVS in Close-Coupled Formation Flight*. In AIAA Guidance, Navigation, and Control Conference, 2000.

[Schwalb 97] E. Schwalb & R. Dechter. *Processing disjunctions in temporal constraint networks*. Artificial Intelligence, vol. 93, pages 29–61, 1997.

[Shim 03] D. Shim, J. Kim & S. Sastry. *Decentralized Nonlinear Model Predictive Control of Multiple Flying Robots in Dynamic Environments*. In 2003 IEEE/RSJ Int. Conf. on Decision and Control, 2003.

[Smith 80] R. G. Smith. *The contract net protocol : High-level communication and control in a distributed problem solver*. IEEE Transactions on Computers, vol. C-29, no. 12, 1980.

[Stone 99] P. Stone & M. Veloso. *Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork*. Artificial Intelligence, vol. 110, no. 2, pages 241–273, 1999.

[Sukkarieh 02] S. Sukkarieh, E. Nettleton, J. Kim, M. Ridley, A. Goktogan & H. Durrant-Whyte. *The ANSER Project : Multi-UAV Data Fusion*. International Journal on Robotics Research, 2002.

[Sukkhatme 02] G. S. Sukkhatme, J. F. Montgomery & R. T. Vaughan. Robot teams - from diversity to polymorphism, chapitre Experiments with Aerial-Ground Robots, pages 345–367. Tucker Balch and Lynne E. Parker, 2002.

[Tambe 97] M. Tambe. *Towards Flexible Teamwork*. Journal of Artificial Intelligence Research, vol. 7, pages 83–124, 1997.

[Veloso 98] M. Veloso & P. Stone. *Individual and collaborative behaviors in a team of homogeneous robotic soccer agents*. In Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS'98), pages 309–316, 1998.

[Vidal 97] T. Vidal & H. Fargier. *Contingent durations in temporal CSPs : From consistency to controllabilities*. In Proc. of IEEE TIME-97 Int. Workshop, 1997.

[Vidal 01] T. Vidal & J. Bidot. *Dynamic Sequencing of Tasks in Simple Temporal Networks with Uncertainty*. proc. of the 7th Int. Conference on Principles and Practice of Constraint Programming, 2001.

[Vidal 02] R. Vidal, S. Sastry, J. Kim, O. Shakernia & D. Shim. *The Berkeley Aerial Robot Project (BEAR)*. In 2002 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems IROS 2002, Workshop on Aerial Robotics, pages 1–10, Lausanne, Switzerland, 2002.

[Volpe 01] R. Volpe, I. Nesnas, T. Estlin, D. Mutz, R. Petras & H. Das. *The claraty architecture for robotic autonomy*. In Proc. of the 2001 IEEE Aerospace Conference, Big Sky, Montana, 2001.