Algorithmes et architecture pour le contrôle de l'exploration d'une zone par une flotte de sous-marins autonomes

Antoine Milot

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)*

Présentée et soutenue le *01/12/2022* **par :**

Antoine Milot

Algorithmes et architecture pour le contrôle de l'exploration d'une zone par une flotte de sous-marins autonomes.

JURY

François CHARPILLET	Directeur de Recherche, INRIA Nancy	Président du Jury
Estelle CHAUVEAU	Ingénieure de recherche, Naval Group	Co-directrice de thèse
Simon LACROIX	Directeur de Recherche, LAAS-CNRS	Co-directeur de thèse
Charles LESIRE	Directeur de Recherche, ONERA	Co-directeur de thèse
Damien PELLIER	Maitre de Conférences, Université Grenoble Alpes	Rapporteur
Olivier SIMONIN	Professeur, INSA Lyon	Rapporteur
Benoit CLÉMENT	Professeur, ENSTA Bretagne	Examinateur

École doctorale et spécialité :
EDSYS : Robotique 4200046
Unité de Recherche :
LAAS-CNRS
ONERA
CIFRE Naval Group
Directeur(s) de Thèse :
Simon LACROIX, *Charles LESIRE* et *Estelle CHAUVEAU*
Rapporteurs :
Damien PELLIER et *Olivier SIMONIN*

RÉSUMÉ

Depuis les années 80, les progrès dans l'ensemble des fonctions nécessaires à l'autonomie des robots sous-marins ont permis le développement de solutions opérationnelles, principalement pour des missions de recueil de données.

Le déploiement en flotte de ces robots pourrait permettre de tirer le plein potentiel de ces progrès, mais les recherches dans ce secteur n'ont pas encore atteint un niveau de maturité suffisant. Une des raisons est que le contexte sous-marin impose des contraintes extrêmement fortes sur les communications, ce qui implique de grandes difficultés à l'établissement de stratégies de coopération.

Un réel besoin pour des systèmes multirobots complexes, capables d'évoluer sous l'eau en coopération existe cependant, c'est le cas du contexte opérationnel entourant la chasse aux mines, qui consiste à détecter et localiser des mines. Cette mission, qui est initialement une mission de couverture, est par nature dynamique, car il faut identifier les objets suspects détectés. Les robots doivent donc réagir aux nouveaux objectifs et les répartir au sein de la flotte. Il peut cependant exister plus d'une manière de réaliser un objectif, et des contraintes spécifiques peuvent être imposées par l'opérateur. Ces éléments induisent des dépendances complexes entre les tâches à exécuter.

Les répartitions de ces objectifs au sein du système définissent des problèmes d'allocation de tâches multirobots (« Multi-Robot Task Allocation » ou MRTA) devant être résolus avant et pendant la mission. Au regard des spécificités de la mission, telles que les contraintes de communication et les dépendances complexes pouvant lier les tâches, la résolution d'un problème de MRTA est difficile.

La solution d'un problème de MRTA définit un plan qui doit être exécuté par les robots. L'exécution de ce plan n'est cependant pas déterministe, chaque robot doit donc superviser l'exécution de son plan afin d'en détecter les divergences. Au même titre que l'intégration de nouveaux objectifs, ces divergences sont des aléas qu'il faut réparer afin de ne pas compromettre la mission.

La mise en place d'algorithmes de décision, de supervision, et de réparation, aptes à répondre à ces défis est au cœur de nos travaux. Nous proposons de faire reposer ces algorithmes sur des méthodes entremêlant allocation par enchères et planification hiérarchique.

Grâce aux enchères nous décentralisons la décision pour faire face aux contraintes de communication, cependant, il est souvent difficile avec ces méthodes d'obtenir des solutions en temps et qualité raisonnables lorsque les tâches comprennent des dépendances complexes. Nous relevons ce défi en exploitant la planification hiérarchique au sein de notre schéma d'enchère. Enfin, nous tirons profit des structures riches utilisées par l'algorithme de décision pour permettre aux robots de superviser et réparer leurs plans.

L'architecture résultante, appelée HaucTioN, a été implémentée et évaluée dans un environnement de simulation reproduisant les principales caractéristiques de la mission, notamment les pertes de communication et la présence d'aléas.

ABSTRACT

Since the 1980s, progresses in all the functions necessary for the autonomy of underwater robots have allowed the development of operational solutions, mainly for data collection missions.

The deployment of these robots in fleets could allow to take advantage of the full potential of these advances. Research in this domain has however not yet reached a sufficient level of maturity. One of the reasons is that the underwater context imposes extremely strong constraints on communications, which imply major difficulties in establishing cooperation strategies.

Nevertheless, cooperating multi-robot systems capable of evolving underwater are needed for complex mission. This is the case of mine hunting missions, which consist in detecting and locating mines. These missions are dynamic by nature because it is necessary to identify the detected suspicious objects. The robots must therefore react to new targets and distribute them within the fleet. However, there may be more than one way to achieve an objective, and specific constraints may be imposed by the operator. These elements induce complex dependencies on the tasks to achieve.

The distribution of these objectives within the system defines "Multi-Robot Task Allocation Problems (MRTA)" to be solved before and during the mission. Given the specificities of the mission, such as communication constraints and complex dependencies that may link tasks, solving an MRTA problem is difficult.

The solution of an MRTA problem defines a plan to be executed by the robots. The execution of this plan is, however, not deterministic. So each robot must supervise the plan execution in order to detect deviations, which must be repaired to not compromise the mission.

The implementation of decision, supervision, and repair algorithms that are able to meet these challenges is at the heart of our work. We propose to base these algorithms on methods mixing auction-based allocation and hierarchical planning.

Auctions allow to decentralize the decision in order to cope with communication constraints. However, with these methods it is often difficult to obtain solutions in reasonable time and quality when the tasks involve complex dependencies. We address this challenge by exploiting hierarchical planning within our auction scheme. Finally, we leverage the rich structures used by the decision algorithm to allow robots to supervise and repair their plans.

The resulting architecture, called HaucTioN, was implemented and evaluated in a simulation environment reproducing the main characteristics of the mission, including communication losses and the presence of hazards.

REMERCIEMENTS

En premier lieu, je tiens à remercier mes merveilleux directeurs de thèse sans qui ce manuscrit n'existerait pas. Si complémentaires, j'irai jusqu'à dire qu'à trois ils formaient la paire.

À Estelle, qui s'est toujours démenée pour faire de cette thèse un projet tangible. C'est grâce à tes qualités humaines et tes nombreux efforts que j'ai pu rapidement m'imprégner de la problématique. Et c'est tes encouragements, dans les hauts comme dans les bas, qui m'ont soutenu durant ce long périple. Plus que tout, j'aimerais te remercier pour ton soutien indéfectible ses trois dernières années.

À Charles, sans qui je n'aurai jamais osé entreprendre cette thèse. Tu m'as toujours conseillé avec précision et rigueur, mais surtout, tu as su me dire les choses quand il le fallait. C'est ta supervision éclairée qui a été la constante de ces quatre dernières années. Je voulais sincèrement te remercier d'avoir cru au stagiaire, et d'avoir su le guider jusqu'au docteur.

À Simon, qui pourrait à lui seul définir ce mélange rare entre bonté et modestie. Malgré une vie bien occupée et de nombreux devoirs, tu as toujours déployé d'immenses efforts pour superviser mon travail. Mais plus que tout, c'est la sollicitude dont tu as toujours fait preuve qui m'a le plus touché. Je ne pourrais jamais assez te remercier d'avoir fait de cette thèse, un espace-temps où je me suis senti si bien.

Je tiens également à remercier tout particulièrement Arthur, pour les nombreuses discussions, conseils, et explications, mais surtout pour tenir un des SAV les plus efficaces au monde.

Je dédis cette thèse à mes deux supers stagiaires, Roland et Jean-Hugues. Sans vous, ces travaux auraient été bien moins ambitieux, mais surtout c'est pouvoir échanger et travailler avec vous qui a défini, pour moi, les périodes les plus agréables de cette thèse.

J'ai également une attention toute particulière pour ma grand-mère, chez qui j'ai trouvé refuge dans les périodes les plus intenses. Et pour la SNCF, qui m'a offert de si nombreuses heures de rédactions.

À mes compagnons de galère de RIS. Aux anciens d'abord, Amandine, Rafa, Guilhem, Kathleen, Guillaume, Martin, Yannick, Idriss et les Florians. Et à tous ceux qui doivent encore ramer, Jérémy, Philippe, Anthony, Ilinka, Fimon, William, Valentin, Smail, Émile, Adrien et Bastien. Merci du fond du coeur d'avoir si bien accompagné ces trois dernières années de ma vie. Merci pour toutes nos pauses café, pour tous nos débats, pour notre vie quotidienne si simple et agréable. Et surtout, merci de m'avoir si souvent laissé gagner au Skull King. Je sais, vous étiez si nombreux que j'ai esquivé les remerciements individuels, mais sincèrement, j'ai adoré discuter, jouer, se plaindre, et rire avec chacun de vous. C'est le coeur gros que je quitte un RIS si chaleureux.

À toutes les formidables personnes du CEMIS. À mes colocs d'open-space ces trois dernières années, Maeve, Paul, Eva, Guillaume, Margaux, Louise, Nathan et Daniel, merci d'avoir fait du bureau un endroit si accueillant, où on prend un réel plaisir à venir partager un moment ensemble. À tous mes camarades de sorties, et en particulier à André, Lucile, Elina, Nicolas et Gautier, j'ai débarqué dans une ville inconnue sans attaches ni contacts, et j'en suis reparti avec de précieux amis. Á vous tous, merci, mais tellement merci, d'apporter cette ambiance si particulière et chaleureuse au CEMIS. C'est grâce à chacun de vous que j'ai toujours adoré mes déplacements à Toulon.

Enfin, à Sabrina. Ma fabuleuse compagne. Je crois que rien de ce que pourrais écrire ne suffirait à exprimer suffisamment la chance que j'ai de partager ta vie. J'espère de tout coeur que ces six ans à tes côtés, à se soutenir dans le pire et à s'aimer dans le meilleur, ne sont que l'introduction de toute une vie remplie d'autant de bonheur.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

LISTINGS

bande passante (bandwidth) La bande passante d'un réseau est le volume de donnée maximum pouvant être transféré dans un laps de temps défini. Elle caractérise le potentiel maximum du réseau avant saturation. Elle est usuellement exprimée en hertz (Hz)

---

adds Ensemble d'effets positifs (d'une tâche primitive).
$\mathbb{C}$ Ensemble de constantes (d'un langage de premier ordre).
$\mathbb{V}$ Ensemble de variables (d'un langage de premier ordre).

$\mathcal{B}$ Ensemble de mises.
ψ Valeur d'une mise.
Ψ Ensemble de valeurs de mises.

$\mathtt{T}_c$ Tâche complexe.
$\mathcal{T}_c$ Ensemble de tâches complexes.
$\mathbb{T}_c$ Ensemble fini de symboles de tâches complexes.
$\boldsymbol{C}$ Ensemble de constantes (d'une tâche).
constraints Ensemble de contraintes (d'une tâche primitive ou une méthode).

dels Ensemble d'effets négatifs (d'une tâche primitive).
$\mathcal{D}$ Domaine de planification.

end Instant de fin (d'une tâche).

$\mathtt{tn}_\mathtt{i}$ Réseau de tâches initial d'un problème de planification.

$\mathtt{s}_\mathtt{i}$ État initial d'un problème de planification.
δ Un *item for sell*.

L Label unique.
λ Ensemble de labels multirobots en vente.
$\mathbb{L}$ Ensemble infini de labels.
$\mathcal{L}$ Langage du premier ordre.

M Méthode de décomposition.
$\mathcal{M}$ Ensemble de méthodes de décomposition.
$\mathcal{G}$ Multi-Robot Graph.
$\mathcal{H}$ Multi-Robot HTN.
$\mathtt{M}_t$ Fonction de correspondance des labels multirobots vers des tâches opérationnelles.

Π Un plan.
$\mathbb{P}$ Ensemble fini de symboles de prédicats.
pres Ensemble de conditions (d'une tâche primitive).
$\mathcal{T}_p$ Ensemble de tâches primitives.
$\mathbb{T}_p$ Ensemble fini de symboles de tâches primitives.
$\mathcal{P}$ Problème de planification.

start Instant de début (d'une tâche).
$\mathcal{S}$ Symbole d'une tâche.

$\mathbb{S}$ Ensemble de symboles de tâches.

$\mathtt{T}$ Une tâche (primitive ou complexe).
$\mathtt{tn}$ Réseau de tâches.
$\mathbb{T}$ Ensemble de tâches (primitives ou complexes).

$\boldsymbol{V}$ Ensemble de variables paramétrant une tâche.

ACRONYMES

AUV Autonomous Underwater Vehicle

CSP Constraint Satisfaction Problem

GdM Guerre des Mines

HDDL Hierarchical Domain Definition Language
HTN Hierarchical Task Network

LTL Linear Temporal Logic

MCM Mine Counter-Measures
MDP Markov Decision Process
MILCO MIne Like COntact
MILEC MIne Like ECho
MILP Mixed Integer Linear Programming
MRS Multi-Robot System
MRTA Multi-Robot Task Allocation
mTSP Multiple Traveling Salesman Problem

PDDL Planning Domain Description Language
PSI Parallel Single-Item auctions

REA Rapid Environmental Assessment
ROV Remotely Operated Vehicle

SA Situation Awareness
SAS Synthetic Aperture Sonar
SI Single-Item
SSI Sequential Single-Item
STN Simple Temporal Network

TDG Task Decomposition Graph
TSP Traveling Salesman Problem

USV Unmanned Surface Vehicle

WDP Winner Determination Problem

1

INTRODUCTION

De nos jours, la plupart des forces armées prônent les doctrines *zéro-perte*, aussi, la simple suspicion de mines suffit à ralentir fortement la progression d'un adversaire en raison de principes de précaution élémentaires. Par la menace invisible qu'elles font peser, les mines représentent une arme psychologique redoutable qu'il faut neutraliser.

Dans l'imaginaire populaire, les opérations de déminage sont réalisées sur terre dans le but de faire passer des troupes, des blindés, ou de sécuriser une zone. Cependant, de nombreuses opérations de déminage ont également lieu en mer, toujours dans le but de faire traverser un champ de mines à des vecteurs militaires ou de neutraliser la menace qu'il représente.

Ces opérations stratégiques d'importance vitale sont coûteuses à mettre en oeuvre en termes de moyens humains, de ressources logistiques, et de temps. De plus, elles comportent des risques importants pour les opérateurs directement impliqués avec les explosifs. Par exemple, pour déminer des mines sous-marines, des vaisseaux parcourent directement le champ de mines à l'aide de capteurs spécialisés. Une fois un objet suspect détecté ; un plongeur démineur est généralement dépêché sur place pour confirmer la menace et neutraliser la mine le cas échéant. Ce type d'opération, appelé *chasse aux mines*, permet un déminage efficace des zones sous-marines, mais implique un risque létal pour les plongeurs démineurs et les opérateurs évoluant au sein du champ de mines. Enfin, l'entrainement et le déploiement de plongeurs spécialisés sont des processus long et coûteux.

Pour ces raisons, de nombreuses marines investissent dans la robotique afin de remplacer leurs systèmes actuels par des systèmes robotisés. Mais les fortes contraintes pesant sur les communications sous-marines ne permettent pas de téléopérer des robots en restant à hors du champ de mines, par conséquent, les robots doivent posséder une grande autonomie. Comme représenté dans la Fig. 1.1, de tels systèmes peuvent évoluer sans interactions avec un opérateur, ainsi, le risque humain de la mission peut être supprimé.

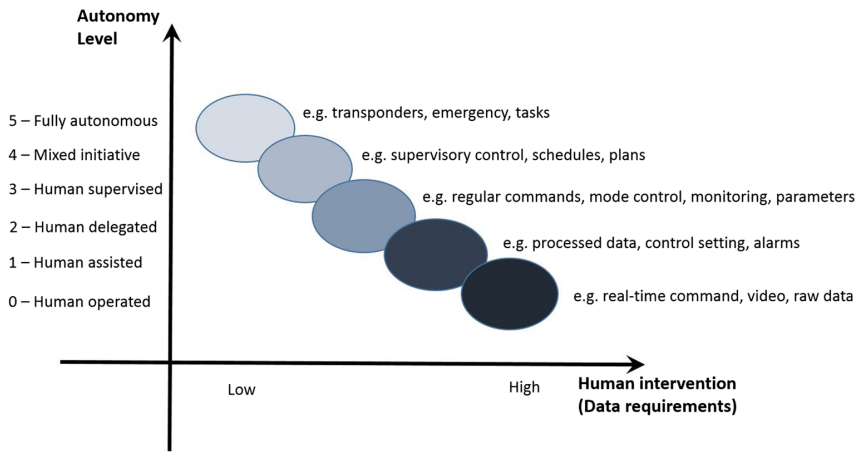


FIGURE 1.1 – Niveau d'autonomie par degré d'interventions humaines [Zol+19].

En sus de la suppression du risque humain, ces systèmes autonomes offrent de nombreux avantages, et ce d'autant plus quand ils sont composés de plusieurs robots. Par exemple, la multiplication des

unités améliore les performances du système, mais également sa résilience face aux aléas du monde réel. Cependant, le déploiement de tels systèmes implique de surmonter de nombreux défis.

Un des principaux défis à relever réside dans la mise en place d'une coopération efficace entre les robots. Par exemple, durant une mission de chasse aux mines la détection d'un objet suspect correspond à l'apparition d'un nouvel objectif devant être pris en charge par le système. Les robots doivent donc déterminer de manière dynamique *qui* doit accomplir *quoi*. Ces questions définissent un problème d'allocation de tâches multirobot ("Multi-Robot Task Allocation (MRTA)"), dont la résolution permet de répartir des objectifs à différentes unités.

Pour résoudre un tel problème, le système exploite des algorithmes de **décision**. Usuellement, un premier problème d'allocation de tâches est résolu au début de la mission avec les objectifs initiaux, mais la nature dynamique de la mission implique de pouvoir résoudre ces problèmes durant la mission. Pour pouvoir prendre efficacement des décisions en ligne, les robots doivent idéalement communiquer afin de partager leurs informations. Cependant, les contraintes de communication constituent un obstacle important à la coopération du système, aussi, elles doivent être prises en compte dans la conception des algorithmes de décision.

De plus, les objectifs à allouer peuvent comporter des contraintes ou être accomplis de différentes manières. Par exemple, la zone à déminer peut être découpée de plusieurs façons en plusieurs sous-zones, et le déminage d'une sous-zone peut être prioritaire par rapport à une autre. Ces éléments créent des *dépendances complexes* entre les objectifs à accomplir, et ces dépendances rendent d'autant plus difficile la prise de décision des robots.

La résolution d'un problème d'allocation de tâches constitue un plan devant être exécuté par le système. Ce plan, qui définit *comment* la mission peut être accomplie, est composé d'un ensemble d'actions à réaliser par les robots. Cependant, l'environnement sous-marin est incertain et les actions des robots ne sont pas déterministes. En conséquence, une divergence peut apparaitre entre le plan décidé et la réalité. Si le système ne corrige pas cette divergence, la mission peut être compromise.

Le système doit donc intégrer des algorithmes de **supervision**, pour détecter les divergences, et des algorithmes de **réparation**, pour réparer les conséquences de ces divergences. Ainsi, les processus du système peuvent être considérés comme liés, la décision mène à la supervision, la supervision mène à la réparation, et la réparation corrige des décisions. Ces processus vitaux et leurs interactions forment l'architecture du système. L'atteinte d'une autonomie suffisante pour les robots passe donc par la mise en place d'une architecture capable de répondre aux défis de la coopération multirobot.

C'est l'objet des travaux présentés dans ce manuscrit, qui proposent le développement d'algorithmes de décision, de supervision, et de réparation pour un système multirobot chasseur de mines.

Dans le Chap. 2, nous définissons le contexte opérationnel de la chasse aux mines et les enjeux du déploiement d'un système multirobot pour accomplir une telle mission. Puis, nous présentons dans le Chap. 3 un état de l'art des travaux connexes à ces enjeux. Sur la base de cet état de l'art, nous présentons dans le Chap. 4 notre contribution, une architecture nommée HaucTioN. Cette architecture entremêle allocation par enchères et planification hiérarchique afin de permettre au système de prendre des décisions de manière décentralisée tout en prenant en compte les dépendances complexes entre les objectifs.

Par la suite, nous introduisons dans le Chap. 5 la pierre angulaire de notre approche, un formalisme de problème de planification hiérarchique intégrant un cadre temporel. Nous détaillons ensuite dans le Chap. 6 notre principale contribution, le protocole de décision de HaucTioN. Puis, nous présentons dans le Chap. 7 comment les structures hiérarchiques complexes issues du protocole de décision sont réutilisées durant la mission dans le cadre de la supervision et de la réparation.

Dans le Chap. 8 nous présentons une évaluation en simulation de notre approche. Enfin, nous concluons sur notre contribution, ses avantages, ses limites, et ses perspectives d'amélioration dans le Chap. 9.

Résumé des publications

Publications

La majeure partie de nos travaux a été dédiée à surmonter les défis soulevés par la mise en place d'un protocole de décision mêlant enchères et planification hiérarchique. Les publications correspondant à ces travaux sont :

— [Mil+21a] Antoine MILOT et al. "Market-based Multi-robot coordination with HTN planning". In : *International Conference on Intelligent Robots and Systems (IROS))*. 2021. DOI : 10.1109/IROS51168.2021.9636286 ;

— [Mil+21b] Antoine MILOT et al. "Solving Hierarchical Auctions with HTN Planning". In : *4th ICAPS workshop on Hierarchical Planning (HPlan)*. 2021. URL : https://hal.archives-ouvertes.fr/hal-03289854 ;

Puis, nous avons montré que ce même protocole de décision pouvait être exploité en ligne afin de réparer le plan de mission. C'est le travail présenté dans :

— [Mil+22b] Antoine MILOT et al. "Autonomous and responsive multi-robot system for minehunting". In : *OCEANS - IEEE*. 2022. DOI : 10.1109/OCEANSChennai45887.2022.9775467.

Enfin, une vision globale de notre approche est résumée dans :

— [Mil+22a] Antoine MILOT et al. "Allocation par enchères et planification hiérarchique pour un système multirobot, application au cas de la chasse aux mines". In : *JFSMA 2022 : 30èmes Journées Francophones sur les Systèmes Multi-Agents*. 2022. URL : https://hal.archives-ouvertes.fr/hal-03694348.

2

CONTEXTE OPÉRATIONNEL ET DÉFINITION DU PROBLÈME

Aperçu

Dans ce chapitre, nous exposons les objectifs de la chasse aux mines et les méthodes usuelles mises en oeuvre pour les atteindre. Puis, nous soulignons les raisons du remplacement de ces méthodes au profit de systèmes robotisés autonomes, dont l'utilisation soulève de nouveaux challenges. Enfin, nous définissons le cadre de notre problème.

Depuis 1945, les mines navales ont endommagé plus de navires de la marine américaine que toutes les autres armes réunies [Eva16]. Leur plus grand intérêt cependant est en tant qu'armes psychologiques. À l'heure où la plupart des pays mettent en pratique une doctrine zéro perte, le simple soupçon de mines suffit à interdire l'accès à une zone et à ralentir fortement la progression de navires. Aussi, des mesures efficaces de lutte contre les mines sont nécessaires en temps de paix comme en temps de guerre.

La chasse aux mines est l'une des contre-mesures possibles pour nettoyer un champ de mines. Elle consiste à détecter et identifier les mines à l'aide de capteurs, puis à les neutraliser. Pour la réaliser, la plupart des pays s'appuient sur des navires chasseurs de mines évoluant directement au sein de la zone minée, exposant les équipages à des risques élevés.

Portées par la course aux armements, les mines évoluent rapidement et leurs contre-mesures se doivent de suivre le rythme. Par conséquent, la chasse aux mines se réinvente en permanence. La principale révolution en cours est le recul de l'homme au profit d'unités robotisées autonomes.

Dans ce chapitre, nous définissons le concept de chasse aux mines, puis nous soulignons le besoin de modernisation des systèmes actuels. Par la suite, nous définissons le contexte que nous considérons dans le cadre de la thèse. Enfin, nous posons le problème à résoudre ainsi que les hypothèses considérées.

2.1 GÉNÉRALITÉS

Les mines navales représentent une menace importante devant être neutralisée. La chasse aux mines est une des contre-mesures possibles contre l'usage de ces mines navales. Dans cette section, nous résumons les principaux enjeux de la chasse aux mines.

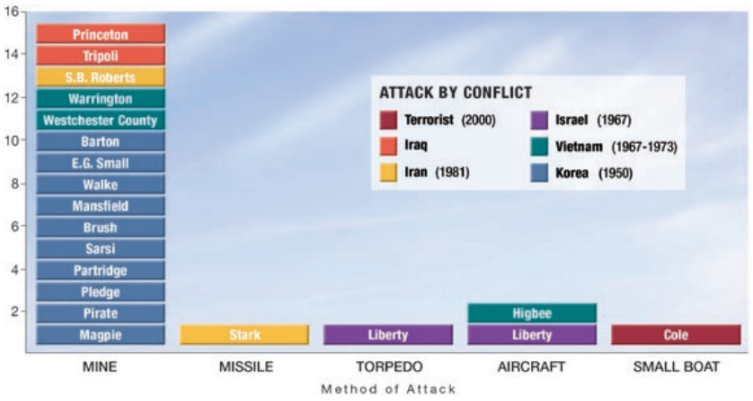
2.1.1 *La menace mine*

L'efficacité des mines navales lors de conflits armés n'est plus à prouver. Les mines sont généralement utilisées à grande échelle pour maximiser leur emprise sur une zone. Il reste courant, même lors de conflits modernes, de subir l'explosion d'une mine. Par exemple, la Fig. 2.1a liste les dommages infligés à la marine américaine par type d'armement au cours des derniers conflits.

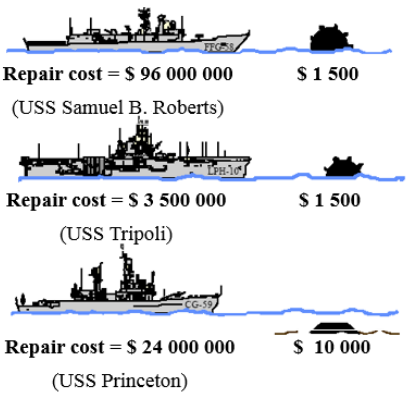
Bien que l'explosion d'une mine ne signe pas systématiquement la perte du vaisseau ciblé, les dommages causés suffisent généralement pour le mettre hors-jeu. Mais, la plupart des marines appliquant une doctrine

zero-mort, leur plus grand impact est avant tout psychologique. En effet, par mesures préventives, la simple suspicion de mines suffit à ralentir fortement un adversaire, voire à lui interdire totalement l'accès à une zone.

Comparativement à d'autres armes, les mines sont très bon marché. À l'opposé, les dégâts qu'elles infligent peuvent atteindre des sommes conséquentes, d'autant plus lorsque les navires concernés sont des atouts militaires. La Fig. 2.1b montre quelques exemples du coût des dommages subis par un navire relativement au coût de la mine qui les a causés. Par ce rapport de force économique, les mines sont parfaites pour les conflits asymétriques.



(a) Résumé des dommages subis par des navires américains par type d'arme depuis 1950 [Exe09].

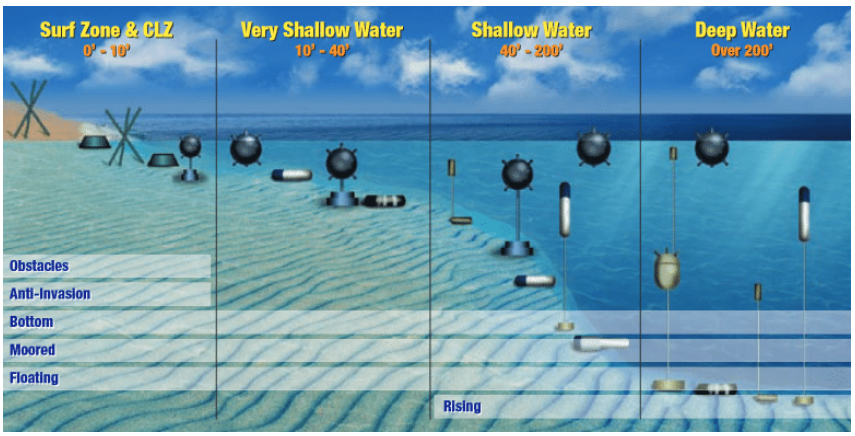


(b) Exemple des dommages causés par des mines à des navires américains [Cor03].

FIGURE 2.1 – Résumé des dommages infligés par des mines sur des navires américains.

Les mines étant qualifiées d'armes du lâche, peu d'États communiquent à leur sujet, cependant, elles ont connu leur propre course aux armements. Ainsi, les mines à orin, boules à piques flottantes au bout d'une chaine, se sont vues remplacées au fur et à mesure des dernières décades par des systèmes complexes. Les mines actuelles peuvent, par exemple, être activées par *influence* (acoustique, électromagnétique, pression), compter le nombre de vaisseaux avant d'exploser, permettant ainsi de frapper le coeur d'un convoi, ou encore, se déplacer lorsque détectées.

Les mines sont classifiées par zone de mouillage (de fond, à orin, à la surface), par système d'activation, et par méthode de mouillage (aérien, sous-marin...) [Mar18].



source: Wikimedia Commons

FIGURE 2.2 – Illustration des différents types de mines par zone d'utilisation.

La Fig. 2.2 résume les différents types de mines existantes. Suivant la profondeur des eaux concernées, différents types de mines peuvent être utilisés. Les mines les plus courantes restent cependant les mines de fonds (“bottom mine”) et les mines à orin (“moored mine”) et peuvent embarquer différents systèmes de mise à feu.

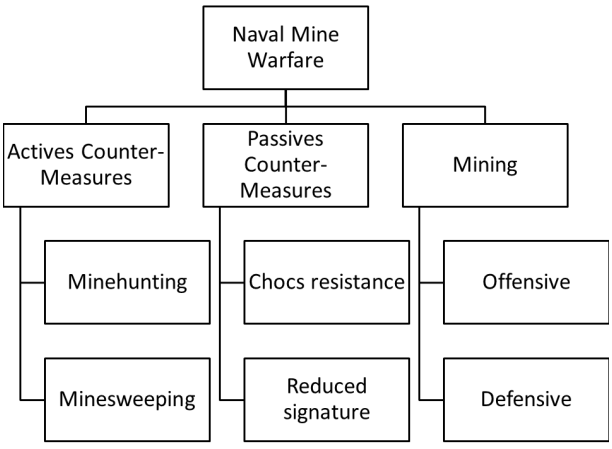
2.1.2 *Guerre des mines et chasse aux mines*

La chasse aux mines est une sous-composante de la Guerre des Mines (GdM) (“Naval Mine Warfare”). La GdM regroupe tous les domaines en lien avec l’utilisation de mines sous-marines et leurs contre-mesures (“Mine Counter-Measures (MCM)”). Les différentes composantes de la GdM sont représentées sur la Fig. 2.3.

Le minage, qui peut être effectué dans un but aussi bien offensif que défensif, nécessite pour la marine adverse d’utiliser des contre-mesures. Par exemple, pour qui voudrait traverser une zone minée, une possible contre-mesure serait de renforcer le blindage de la coque. Ce type de contre-mesure est dit passif. À l’opposé de ces contre-mesures passives on trouve les contre-mesures actives, et parmi elles, la chasse aux mines.

Une première contre-mesure active consiste à draguer les mines à l’aveugle (“minesweeping”). Ce draguage peut, par exemple, être fait de manière mécanique en tirant un filet entre deux vaisseaux. Lorsque le filet rencontre une mine de contact elle explose, il est donc souvent nécessaire de réparer, voire de remplacer le filet. Ce processus est long et fastidieux. De plus, avec cette méthode, il est difficile d’apporter des garanties sur le nettoyage des zones.

À l’inverse, la chasse aux mines fait figure de solution chirurgicale. En effet, elle nécessite de détecter et identifier une mine tout en déterminant précisément sa localisation. Pour toutes ces raisons, quand il s’agit de déminer une zone (ou de garantir qu’elle n’a pas été minée), les marines choisissent généralement la chasse aux mines.



source: NATO Declassified MTP

FIGURE 2.3 – Arborescence des composantes de la guerre des mines.

Objectifs de la chasse aux mines

Une mission de chasse aux mines est composée de trois objectifs [1] : Détection ; Identification ; et Neutralisation.

Lorsque qu'un objet est **détecté**, c.-à-d. un écho significatif est reçu, il est classifié comme *objet suspect* ou *fausse alarme*. L'objet suspect est ensuite **identifié** comme *mine* ou fausse alarme. Si la menace est confirmée, il peut être nécessaire de procéder à la **neutralisation** de la mine. Les différentes étapes de ce processus sont résumées dans la Fig. 2.4. En accord avec la définition des objectifs que nous utilisons, nous considérons que les étapes de détection, qui traitent des MIne Like ECho (MILEC), et de classification, qui traitent des MIne Like COntact (MILCO), appartiennent à l'objectif général de *Détection*.

Il est à clarifier que ces objectifs, s'ils peuvent être vu comme les différents états d'une mine, ne correspondent pas à des phases distinctes et globales de la mission de chasse aux mines. La couverture de la zone étant progressive, les objectifs de détection, d'identification et de neutralisation peuvent être accomplis en parallèle.

La mission consiste tout d'abord en un problème d'exploration, car on ne peut pas prévoir la présence et le nombre de mines dans la zone. Cependant, au cours de la mission, il est nécessaire de réagir à divers événements, et en particulier à la découverte d'un objet suspect. Ainsi, les objectifs d'identification et neutralisation apparaissent de manière imprévisible durant la mission, il est donc nécessaire de s'adapter afin de les prendre en compte. Cette imprévisibilité entraîne un fort besoin de *réactivité*.

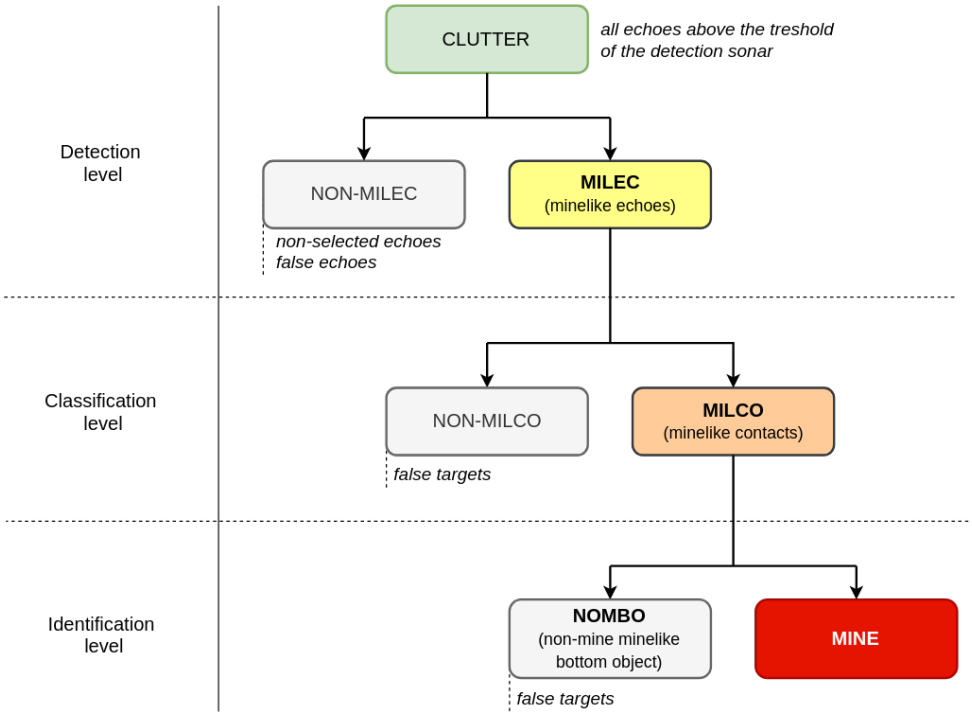


FIGURE 2.4 – Logigramme de la détection à l'identification d'une mine.

1. Ces objectifs peuvent parfois être étendus dans la littérature à : Détection, classification, identification, localisation puis neutralisation ou évitement

2.2 SCÉNARIOS TYPIQUES, FINALITÉS ET CONTRAINTES ASSOCIÉES

Les causes d'une opération de chasse aux mines peuvent être diverses. En conséquence, le but de la mission et les contraintes à considérer le sont tout autant. Il est cependant possible de regrouper les opérations de chasse aux mines dans trois grandes familles : *Port allié*, *Point d'étranglement* et *Débarquement*. Ces trois types de missions sont définis ci-après.

2.2.1 *Port allié*

Definition 2.1 (Port allié ("Home Port"))

L'opération s'inscrit dans une dynamique de surveillance. Afin de s'assurer la souveraineté d'un port stratégique allié, il est nécessaire de vérifier qu'aucune force adverse n'y a mouillé[2] *des mines.*

Principales caractéristiques :

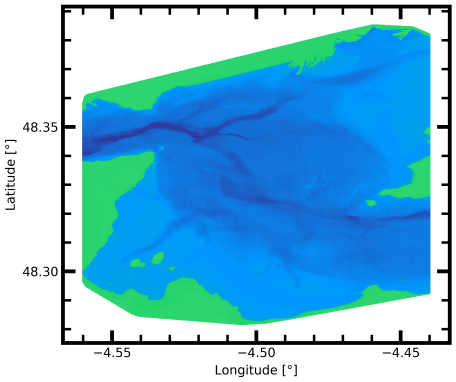
- *Zone alliée ;*
- *Informations (précédents rapports, bathymétrie, nature des fonds...) disponibles ;*
- *Possible utilisation d'atouts (balises de communication ou localisation, aéronefs...)*
- *Pas de limite de temps ;*
- *Niveau de confiance dans le* blanchiment de la zone[3] *proche de 100%.*

Ce type d'opération régulière en zone alliée bénéficie de connaissances approfondies sur l'environnement, voire du déploiement d'atouts spécifiques. La recherche est généralement basée sur la différence entre les cartographies précédentes et celle en cours.

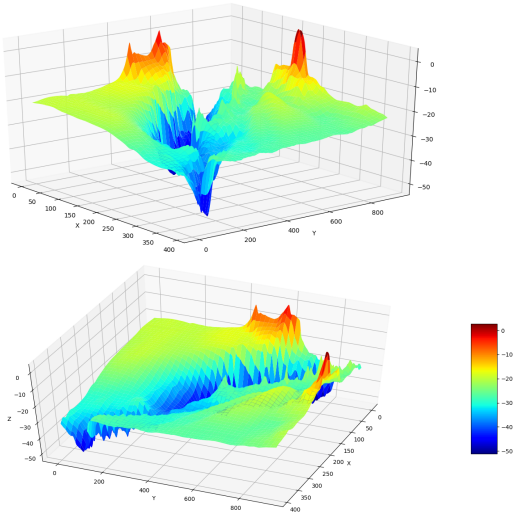
Les missions de type "Home Port" constituent une importante part des tâches permanentes des marines. Même en temps de paix elles demeurent nécessaires. D'une part cela permet aux armées d'entretenir la souveraineté de leurs eaux en anticipant tout type de conflits inattendus. D'autre part, les mines de précédents affrontements peuvent encore être présentes.

Ces missions ont lieu dans les ports stratégiques (c.-à-d. présentant un intérêt économique ou militaire). La rade de Brest en France, visible sur la Fig. 2.5, en est un cas d'école. Par exemple à Brest, une mine allemande de la Seconde Guerre Mondiale a été trouvée en 2020 [20]. Outre l'important trafic maritime commercial et civil, la rade comporte de nombreux sites militaires en faisant une cible prioritaire.

2. Mouiller une mine : Action d'installer une mine navale, généralement effectuée par un vaisseau mouilleur de mines ou par un aéronef
3. Blanchiment d'une zone : la zone est couverte afin d'assurer l'absence de mine avec un certain niveau de confiance
4. https ://wwz.ifremer.fr
5. La Cordelière est un navire ayant sombré le 10 août 1512 au large de Brest. Une importante campagne de recherche a récemment vue le jour pour retrouver l'épave : https ://www.lacordeliere.bzh



(a) Bathymétrie de la rade de Brest (Données IFREMER [4]).



(b) Bathymétrie du goulet de Brest (Données Projet Cordelière [5]).

FIGURE 2.5 – Illustrations de l'environnement de la rade de Brest, port stratégique de grande importance pour la France.

2.2.2 *Point d'étranglement*

Definition 2.2 (Point d'étranglement ("Chokepoint"))

Le but de l'opération est de pouvoir traverser un champ de mines rapidement, par exemple pour faire passer un convoi. Il n'est donc pas nécessaire de déminer toute la zone, mais uniquement de dégager un passage en un minimum de temps.

Principales caractéristiques :

- *Zone de recherche restreinte, la couverture se fait en priorité dans une zone donnée pour dégager un accès ;*
- *Peu d'informations disponibles ;*
- *Contrainte de temps ;*
- *Niveau de confiance dans le* blanchiment de la zone *autour de 80-90%.*

Ce type de mission se déroule le plus souvent en temps de guerre et est d'importance critique pour la suite des opérations. En effet, à l'aide d'un champ de mines placé au bon endroit, une force adverse peut totalement geler les flux militaires, mais aussi économiques, provoquant rapidement une situation de crise.

Le détroit d'Ormuz, représenté sur la Fig. 2.6, est un excellent exemple de zone de grande importance stratégique. Un cinquième du pétrole mondial y transite chaque année et, avec ses 55km de large, il est aisé pour un état de bloquer ce flux à l'aide d'un champ de mines. De par son importance, cette zone a déjà connu de nombreuses opérations de déminage, par exemple lors de la Guerre du Golfe [Cus91]. Les tensions autour de cette zone sont toujours d'actualité et obligent les marines à prendre au sérieux une menace mine dans ce secteur [Woo18].

source: BBC

FIGURE 2.6 – Schéma du contexte maritime et géopolitique du détroit d'Ormuz [19a].

2.2.3 *Débarquement*

Definition 2.3 (Débarquement ("Landing"))

Le but est d'établir un accès sécurisé jusqu'à un rivage adverse afin de procéder à un débarquement. Le tout doit se faire dans la discrétion. Tant que l'adversaire n'a pas connaissance des opérations, il n'y a pas de forte contrainte de temps.

Principales caractéristiques :

- *Zone adverse ;*
- *Peu d'informations disponibles ;*
- *Forte contrainte de discrétion ;*
- *Pas de contrainte de temps ;*
- *Niveau de confiance dans le* blanchiment de la zone *autour de 80-90%.*

Ces missions se déroulent également en temps de guerre. Elles sont vitales pour assurer le débarquement des troupes, mais également le ravitaillement par voie maritime. Par exemple, le débarquement de Normandie, lancé dans la nuit du 5 au 6 juin 1944, a débuté par le déploiement de plus de 200 dragueurs de mines ("minesweeper") pour accomplir une opération de déminage de grande envergure [21d].

Les trois scénarios présentés portent sur des missions ayant une finalité et des impératifs à respecter très différents. Cette finalité et ces impératifs impactent fortement la manière dont une mission de chasse aux mines peut être accomplie. Tous les systèmes ne peuvent pas nécessairement supporter une telle disparité. Par conséquent, le type de mission à considérer est fondamental dans le choix du système à déployer.

2.3 MÉTHODES CONVENTIONNELLES

Historiquement, une mission de chasse aux mines s'organise autour de vaisseaux mères, appelés chasseurs de mines, pouvant s'appuyer sur différents vecteurs en support. Chacun de ces vecteurs joue un rôle dans les différents objectifs de détection, identification, neutralisation.

2.3.1 *Détection*

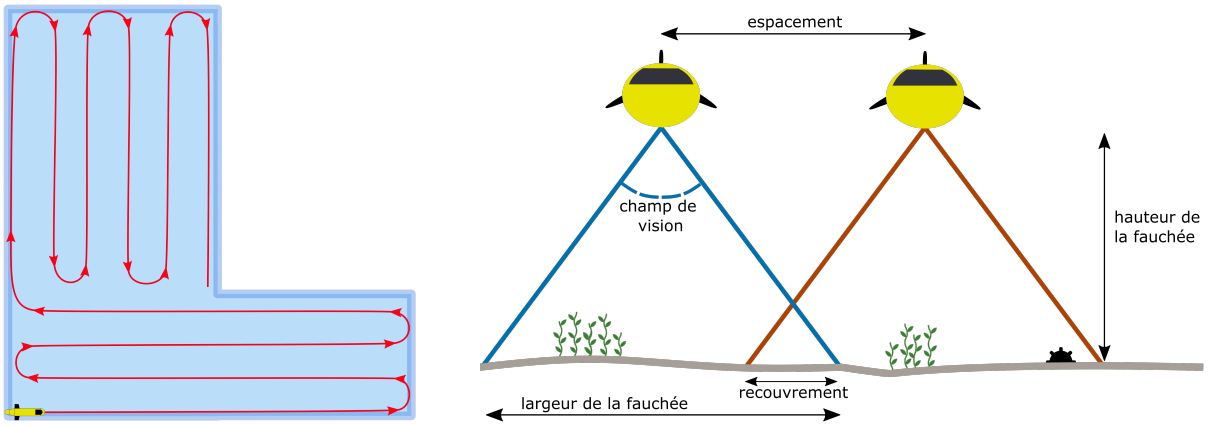
La détection représente la plus grande charge de travail de la mission. Elle est accomplie par les navires chasseurs de mines. Ces vaisseaux s'appuient principalement sur deux capteurs pour cet objectif :

— Sonar de coque ("hull mounted sonar") : Permet de détecter les obstacles dans un faible cône à l'avant du vaisseau. Principalement utilisé pour sécuriser la progression du navire.

— Sonar remorqué ("towed sonar") : Le sonar est relié au vaisseau par un câble, il peut être placé à différentes profondeurs. Le sonar cartographie les fonds marins avec une grande résolution.

Pour remplir l'objectif de détection, la zone est généralement ratissée bande par bande par les vaisseaux chasseurs de mines. Cette couverture suit un schéma appelé *boustrophedon*, représenté sur la Fig. 2.7. La qualité de la fauchée dépend de la largeur des rails et de leur recouvrement ("overlap"), mais également de paramètres environnementaux comme le type de fond (sable, roches...) et le courant sous-marin. Certaines informations sur ces paramètres sont soit déjà connues, soit obtenues par une évaluation environnementale rapide (Rapid Environmental Assessment (REA)). Néanmoins, ces informations sont incomplètes et incertaines.

Si la majeure partie de la détection est accomplie par les vaisseaux chasseurs de mines, d'autres unités peuvent être utilisées en soutien. Par exemple, un hélicoptère peut participer à la mission en détectant les mines flottant à la surface ou en plongeant ponctuellement un sonar dans l'eau ("dipping sonar").



(a) Vue de dessus. La trajectoire suit un motif de type boustrophédon.

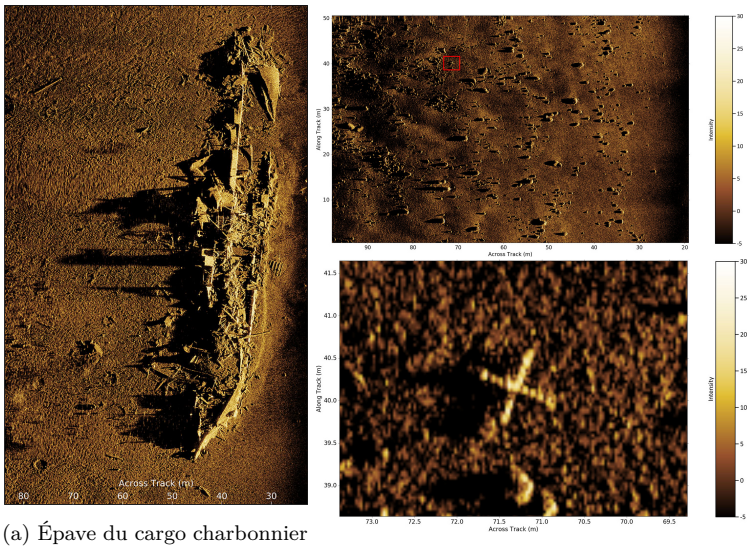
(b) Vue de coupe. Les principaux paramètres de la fauchée sont mis en avant.

FIGURE 2.7 – Illustration d'une fauchée.

La cartographie systématique de la zone par les sonars remorqués permet de détecter les mines, comme les mines de fond, ou leurs traces, telles que les ancres des mines à orin. Ces sonars, habituellement des sonars à balayages latéraux, sont de plus en plus souvent remplacés par des Synthetic Aperture Sonar (SAS) offrants de meilleures performances. Ce type de capteur est essentiel pour une mission de chasse aux

mines. Il permet de détecter et classifier efficacement les objets des fonds marins. Les SAS sont notamment utilisés pour la recherche d'épaves (Fig. 2.8a).

La Fig. 2.8 montre des exemples de cartographies réalisées avec un SAS. Sur la Fig. 2.8b, un objet ressemblant à une mine de fond à été détecté. Sa classification en objet suspect est rendue possible par l'étude de son écho ainsi que son ombre projetée. À l'aide de ces deux paramètres, l'opérateur peut reconstruire la géométrie de l'objet et le classifier comme possible menace. La Fig. 2.9 illustre ce processus.

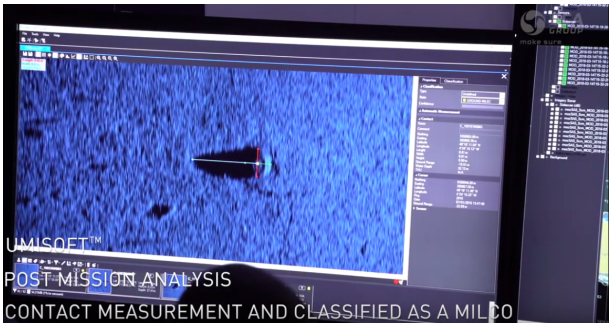


(a) Épave du cargo charbonnier Le Ferrando.

(b) Fond marin comportant une mine de fond.

source: KrakenRobotics

FIGURE 2.8 – Exemples de cartographies obtenues avec un sonar à ouverture synthétique.



source: ECA

FIGURE 2.9 – Exemple de classification d'une mine à partir d'une cartographie.

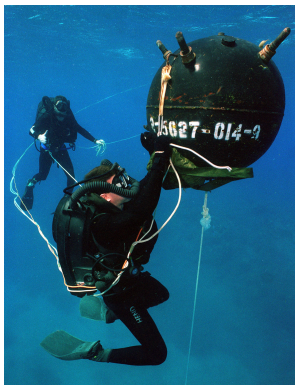
Grâce aux données relevées par le SAS, les opérateurs peuvent classifier les objets détectés comme à identifier ou non, c.-à-d. comme MILCO ou fausse alarme (Voir Fig. 2.4).

2.3.2 *Identification et Neutralisation*

Une fois un objet détecté et classifié comme MILCO, il doit être identifié afin de confirmer ou non la présence d'une mine. Usuellement, cette identification se fait en dépêchant sur place un *plongeur démineur*. Le rôle du plongeur démineur est de confirmer visuellement la présence d'une mine et, si possible, de la classifier parmi un ensemble de mines connues.

Lorsque la mission l'exige, le plongeur peut procéder dans le même temps à la neutralisation de la mine, soit par désamorçage, soit à l'aide d'une charge explosive. C'est cette dernière méthode, comportant le moins de risque pour le plongeur, qui est la plus souvent utilisée. La Fig. 2.10a montre un plongeur démineur neutralisant une mine d'entrainement.

Il est également possible d'envoyer un robot télé-opéré, tel qu'un Remotely Operated Vehicle (ROV), pour réaliser ces étapes d'identification et de neutralisation. Dans ce cas, le robot peut embarquer une caméra ou des capteurs spécifiques, tels que chimiques ou électromagnétiques, pour procéder à l'identification ainsi qu'une charge explosive pour la neutralisation. La Fig. 2.10a illustre le ROV SAAB, actuellement commercialisé, à l'oeuvre.



source: Wikimedia

(a) Plongeur démineur attachant une charge explosive à une mine à orin d'entrainement.

source: SAAB

(b) Illustration du ROV "SAAB" neutralisant une mine.

FIGURE 2.10 – Neutralisation d'une mine avec un plongeur démineur et un ROV.

2.4 DES CONTRE-MESURES EN CONSTANTE ÉVOLUTION

Portée par la course aux armements, les mines ont évolué pour devenir toujours plus discrètes, autonomes, et dangereuses. Afin de conserver leur souvaireneté, les marines doivent suivre le rythme imposé par les avancées technologiques des mines. Par conséquent, de nombreuses marines investissent dans des systèmes chasseurs de mines aptes à contrer ces armes d'un genre nouveau.

2.4.1 *Des doctrines et des besoins nouveaux*

Les tensions de notre siècle ont régulièrement vu la menace mine refaire surface [22]. Loin d'être une arme obsolète, la mine a évolué avec le temps, obligeant les marines à adapter leurs contremesures.

L'utilité fondamentale des mines n'a pas changé : interdire un accès, ou à minima en rendre sa traversée laborieuse, à moindre coût. Cependant, engagées dans la course aux armements, les mines et leurs contremesures ont connu d'importantes évolutions. Le primitif draguage de mines, par exemple, s'est vu totalement remplacé par la solution plus performante de la chasse aux mines.

En ce qui concerne les mines, les principales évolutions ont porté sur la discrétion, l'intelligence embarquée, et les sécurités anti-désamorçage. Par la modernisation de leurs capteurs, elles peuvent, par exemple, compter les signatures électromagnétiques d'un convoi pour se déclencher au moment le plus

opportun. Et, par l'ajout de sécurités contre la neutralisation, les mines modernes rendent la mission d'un plongeur démineur hautement dangereuse.

Les doctrines des marines ont également évolué, le zéro perte ("zero casualty") passant d'objectif optionnel à impératif. Ainsi, les contremesures utilisées se doivent d'être plus sécurisantes pour les opérateurs humains. La majeure partie des recherches menées dans ce but consiste à développer et déployer de nouveaux vecteurs, organiques ou mécaniques, pour toujours plus éloigner l'humain de l'explosif. Entre 1960 et 2010, à l'instar des chiens démineurs, la marine américaine a, par exemple, exploré l'utilisation de dauphins pour trouver et neutraliser des mines [14a]. Cependant, grâce aux progrès du domaine, les vecteurs d'avenir semblent appartenir à la robotique. Durant la guerre du Golfe déjà, des ROV, unités robotisées téléguidées, ont été utilisés en soutien des plongeurs démineurs [Cus91]. La Fig. 2.11 illustre un ROV chasseur de mine récent et commercialisé par la société ECA.

source: ECA

FIGURE 2.11 – Le robot téléguidé H1000 de ECA, peut être utilisé pour identifier et neutraliser des mines.

Néanmoins, ces robots ne remplacent pas totalement l'homme. Il demeure nécessaire aux opérateurs de parcourir le champ de mines pour le cartographier puis de téléguider les ROV grâce à une liaison câblée, les communications sous-marines ne permettant par un guidage sans-fil à distance suffisante. Aussi, malgré l'intégration de ces robots, les plongeurs, et plus généralement tous les opérateurs évoluant au sein du champ de mines, encourent toujours un risque létal important [10 ; 21a]. En conséquence, un fort besoin d'autonomie des unités robotisées se fait ressentir afin de pouvoir, à terme, supprimer le risque humain [14b].

En sus de ces raisons sécuritaires, poussant à une refonte des systèmes engagés dans la chasse aux mines, coexiste une raison économique. En effet, comme souligné dans [Sha20], l'accomplissement d'une mission de chasse aux mines est une tâche chronophage qui nécessite des ressources humaines et des équipements importants. Par exemple, les vaisseaux chasseurs de mines sont parmi les navires militaires les plus chers à produire en raison de leur coque en fibre de verre permettant de réduire leur signature électromagnétique [18]. Le remplacement de ces bâtiments par des vecteurs plus petits et n'embarquant pas d'opérateurs réduirait drastiquement les coûts associés à la construction du système chasseur de mines.

2.4.2 *L'avènement des systèmes multirobots*

Afin de réduire les coûts, gagner en efficacité, et supprimer les risques humains, de nombreuses marines explorent activement l'utilisation de systèmes multirobots (Multi-Robot System (MRS)) pour les missions de chasse aux mines sous-marines. En effet, les unités robotiques sont plus adaptées aux missions de chasse

aux mines que les humains. D'une part, le contexte sous-marin implique des hautes pressions et l'absence de lumière, d'autre part, des explosifs peuvent être présents.

En 2016, l'Établissement Norvégien de Recherche sur la Défense a posé dans [MN16] les bases d'un concept de système multirobot autonome chasseur de mines devant remplacer à terme leurs systèmes actuels. En France, les recherches dans ce domaine sont soutenues par les investissements importants des marines Belge et Néerlandaise. En effet, les entreprises françaises Naval Group et ECA, spécialisées respectivement dans la construction navale de défense et la robotique militaire, ont remporté un marché important visant à remplacer les chasseurs de mines belges et néerlandais par des unités robotiques autonomes [19b]. En Angleterre, la Royal Navy a emboîté le pas en annonçant la prochaine mise au rebut de ses chasseurs de mines au profit de drones [All20].

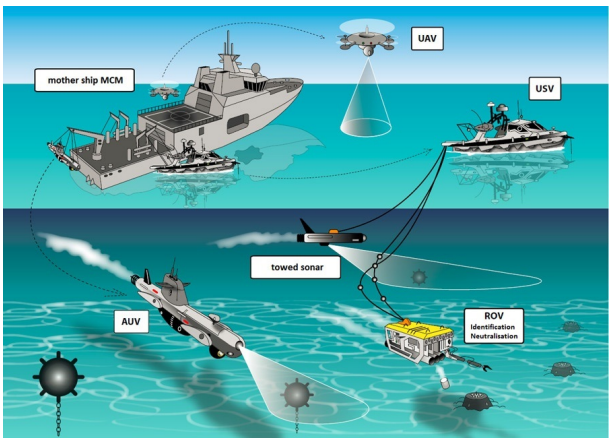
Parmi les vecteurs composant de tels systèmes, on en retrouve trois principaux :

— Autonomous Underwater Vehicle (AUV) : Drone sous-marin dédié à la détection, la classification et l'identification des mines ainsi que, dans certains cas, leur neutralisation à l'aide d'une charge explosive. Suivant leur charge utile, ils peuvent embarquer un SAS pour la détection ou des capteurs chimiques, électromagnétiques, ou optiques pour l'identification. Ils sont au coeur de la force de travail des nouveaux systèmes.

— Unmanned Surface Vehicle (USV) : Drone de surface pouvant remorquer des sonars ou agir en soutien des drones sous-marins en les déployant, les rechargeant ou en établissant des communications avec eux.

— Vaisseau-mère : Embarque des opérateurs humains et sert de base de déploiement des unités robotisées. Il n'évolue pas dans le champ de mines.

Un concept de système multirobot dédié à la chasse aux mines et intégrant ces vecteurs est représenté sur la Fig. 2.12. Ce concept est proposé par Eguermin, une école navale militaire Belgo-Néerlandaise et spécialisée dans la guerre des mines.

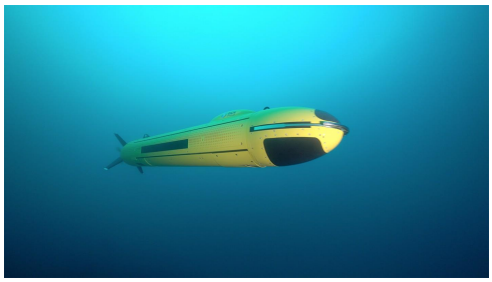


source: Eguermin

FIGURE 2.12 – Illustration d'un système multirobot accomplissant une mission de chasse aux mines.

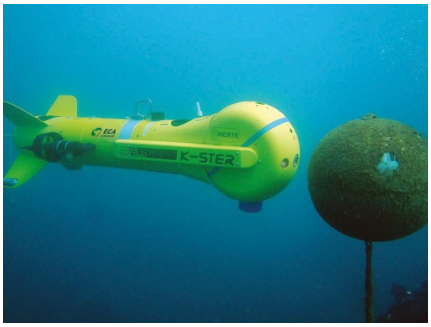
Ces nouveaux systèmes, qui étaient jusqu'alors limités à l'état de concept, tendent à se concrétiser. Par exemple, la Royal Navy a déployé ses premiers démonstrateurs autonomes [21c]. La Fig. 2.13 montre quelques exemples de vecteurs en cours de développement.

Cette tendance au remplacement complet des vaisseaux chasseurs de mines par des systèmes robotiques autonomes s'observe aussi à l'échelle mondiale avec des projets européens de grande ampleur [21b]. L'avenir



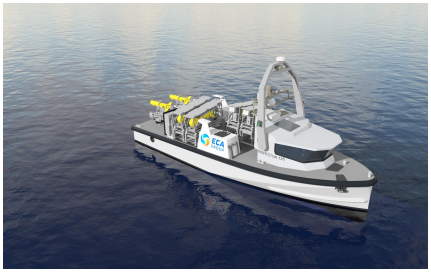
source: ECA

(a) Drone A-18 (ECA), dédié à la détection et classification.



source: ECA

(b) Drone KSTER (ECA), dédié à la neutralisation.



source: ECA

(c) Drone INSPECTOR 125 (ECA), dédié au déploiement d'unités robotisées.

source: Naval Group

(d) Concept du M940 (Naval Group), vaisseau-mère porteur de drones.

FIGURE 2.13 – Exemples de systèmes pouvant être utilisés pour identifier et neutraliser des mines. Ces vecteurs sont issus du programme de renouvellement des marines Belges et Néerlandaise [19b].

de la chasse aux mines semble tracé et ne peut s'envisager sans l'utilisation de systèmes multirobots autonomes.

Fort de ces investissements massifs, le secteur de la robotique marine connaît depuis une dizaine d'années un important essor. Ainsi, comme montré dans [Fer+17b], de nombreux travaux sur les MRS ont été menés pour des applications telles que l'archéologie maritime, l'étude des océans, la lutte anti-sous-marin ou la chasse aux mines. Ces travaux visent en grande partie à relever les nombreux enjeux entourant le déploiement d'un système multirobot dans un environnement sous-marin.

2.5 SYSTÈME MULTIROBOT AUTONOME CHASSEUR DE MINES

Dans cette section, nous nous intéressons aux enjeux relatifs à l'utilisation d'un système multirobot devant accomplir une mission de chasse aux mines. Nous soulignons les principaux défis à relever et les hypothèses considérées. Enfin nous présentons un scénario typique pour lequel un tel système pourrait être utilisé. Nos travaux portant sur la *coopération* entre robots, nous n'aborderons pas en détail les problématiques de perception, navigation ou localisation.

2.5.1 *Objectifs du système*

Indépendamment de sa nature, le coeur d'une mission peut, dans la plupart des cas, se résumer à trois éléments principaux :

— Un but composé d'*objectifs* à accomplir ;

- Des *contraintes de mission* à respecter lors de l'accomplissement des objectifs, ce sont des impératifs portant sur la manière d'accomplir le but ;
- Des *critères d'évaluation* permettant de porter un jugement sur la réalisation des objectifs et la satisfaction des contraintes de mission.

Dans le cas d'une mission de chasse aux mines de type "Home Port", cette synthèse correspond à :

- *Objectifs* : Nettoyer une zone donnée
  - Couvrir la zone de mission ;
  - Identifier les objets suspects découverts lors de la couverture ;
  - Facultatif : Neutraliser les mines identifiées.
- *Contraintes de mission* :
  - Atteindre un certain niveau de confiance dans le nettoyage de la zone de mission ;
  - Rapporter régulièrement ses activités à l'opérateur.
- *Critères d'évaluation* :
  - Niveau de confiance final dans le nettoyage de la zone ;
  - Le temps nécessaire pour accomplir la mission ;
  - Des variables spécifiques telles que le nombre de mines trouvées ou les schémas de mouillage identifiés.

Pour accomplir les objectifs de la mission, comme des objectifs de couverture de zone ou d'identification d'objets suspects, les robots doivent être capables d'évoluer en autonomie. Par exemple, ils doivent être capables de déterminer quel capteur utiliser et comment naviguer dans la zone de mission, ou encore de se répartir les objectifs durant la mission pour gagner en efficacité.

En sus des objectifs inhérents à la chasse aux mines, le système doit répondre à certaines contraintes. Par exemple, l'accomplissement des objectifs avec un certain degré de réussite. D'autres contraintes sont amenées par la nature autonome du système. En effet, la mission à réaliser étant longue et délicate, il peut être nécessaire pour l'opérateur militaire d'en connaître son évolution afin de conserver une connaissance et une confiance suffisantes dans le système.

Par exemple, l'opérateur peut imposer un retour d'information en ligne, c.-à-d. durant la mission, par l'envoi de rapports réguliers. Ces informations lui permettent de conserver un certain niveau de connaissance de la situation ("Situation Awareness (SA)") [Rol+17]. Même si ces informations n'ont pas vocation à servir de base à un contrôle fort par l'opérateur elles sont d'autant plus vitales que le système est autonome et la mission sensible : le système doit être en mesure de satisfaire cette contrainte de flux d'information durant l'exécution.

2.5.2 *Décision*

Pour que les robots puissent accomplir les objectifs de la mission il est nécessaire que le système prenne des *décisions*. Deux notions fondamentales entourent la prise de décision pour un système multirobot : la *planification* et l'*allocation*, c.-à-d. qui doit faire quelle tâche et comment.

La mission de chasse aux mines consiste initialement en un problème de couverture : une zone donnée doit être cartographiée. Si nous considérons un système composé d'un seul robot, il est nécessaire de prévoir *comment* le robot doit se mouvoir et percevoir. Par exemple, quelle trajectoire adopter et quels capteurs utiliser pour cartographier la zone. La prévision de ces éléments consiste à résoudre un problème

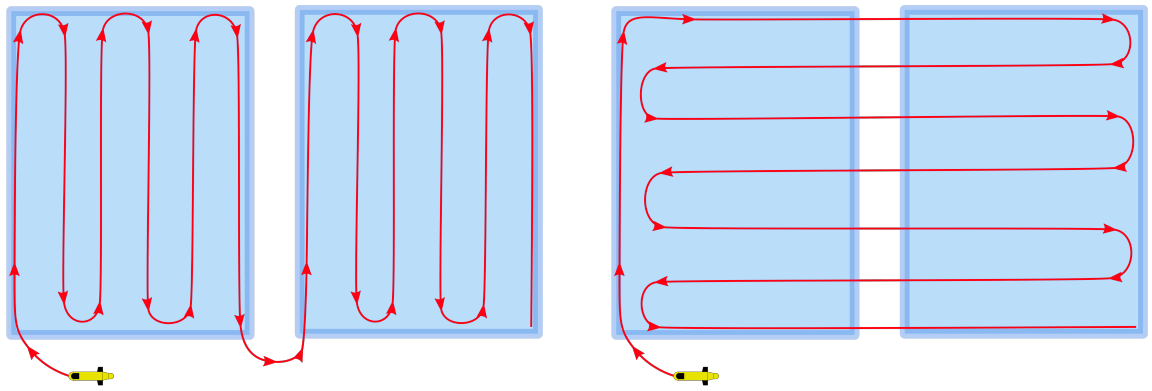
de planification. Cette planification est contrainte par les capacités du robot, mais également par des paramètres extérieurs, tels que le courant sous-marin ou le type de fond. L'ensemble de ces éléments impacte la solution du problème de planification.

Si nous considérons un système composé d'au moins deux robots, une manière de gagner en efficacité est de répartir la force de travail, c.-à-d. répartir les robots sur la zone de mission. Ainsi, chaque robot est en charge d'une partie de l'objectif global. Pour procéder à cette répartition du travail, il est cependant nécessaire de déterminer *qui* doit faire *quoi*. Ces éléments définissent le problème d'allocation. La solution de ce problème tient compte des objectifs et contraintes de la mission, mais également des capacités propres à chaque robot. Par exemple un robot plus rapide qu'un autre devrait se voir allouer une zone plus grande.

Ainsi, planification et allocation sont deux éléments clés du déploiement d'un système multirobot. Ensemble, ils définissent les décisions du système. Si ces deux problèmes peuvent être séparés pour des cas simples, dans le cadre général la résolution de l'un influence l'autre et vice-versa.

Par exemple, la manière dont un robot peut couvrir une zone influencera les portions de zone qui lui seront attribuées, et inversement les portions de zone qu'ils lui seront attribuées influenceront sa manière de les couvrir. Ainsi, si un robot doit couvrir plusieurs zones voisines les unes des autres, il peut être plus intéressant pour lui de planifier une trajectoire parcourant toutes les zones à la fois. Dans un tel cas, représenté sur la Fig. 2.14, le robot n'accomplit pas un objectif, c.-à-d. la couverture d'une zone, puis un autre, mais tous à la fois. Deux zones voisines l'une de l'autre sont couvertes par un Autonomous Underwater Vehicle (AUV). Dans le premier cas, Fig. 2.14a, le robot couvre une zone, puis la suivante. Dans le second cas, Fig. 2.14b, le robot couvre les deux zones en même temps. Ce faisant, le nombre de demi-tours nécessaires au robot a été minimisé.

Des solutions de bonne qualité permettent d'économiser des ressources comme le temps, l'énergie, ou le nombre de robots nécessaires. Ces ressources étant précieuses, l'accomplissement de la mission n'est pas une finalité. Le système doit toujours chercher à améliorer ses performances. Il est donc souvent nécessaire d'intriquer la planification avec l'allocation afin d'atteindre des solutions plus performantes.



(a) Deux zones sont couvertes par deux boustrophédons différents.

(b) Deux zones sont couvertes par le même boustrophédon.

FIGURE 2.14 – Illustration de l'impact de la dépendance des objectifs sur une trajectoire boustrophédon. Avec la seconde méthode, le nombre de demi-tours est minimisé.

Au sein du système multirobot, les processus de décision doivent répondre aux questions :

— **Quoi** : Qu'est-ce qui doit être accompli. Ce sont généralement les tâches obtenues après raffinement de la description générale de la mission ;

— **Qui** : Quel robot doit accomplir telle tâche ;

— **Quand** : À quel moment le robot désigné devra accomplir la tâche ;

— **Comment** : Comment le robot doit-il accomplir la tâche.

Les réponses à ces questions sont guidées par les modèles utilisés pour représenter la mission ainsi que les capacités des robots.

2.5.3 *Exécution, supervision et réparation*

Les décisions prises sont réalisées durant la phase d'*exécution*. Cependant, accomplir ces décisions dans le milieu sous-marin est un réel challenge. La nature non-maitrisée d'un tel environnement, que cela soit par les courants rencontrés, le type de fonds, ou encore la propagation des ondes acoustiques, impose au système de s'adapter en permanence. Les robots peuvent également rencontrer des avaries impactant la bonne tenue de leurs objectifs. Ainsi, l'utilisation d'un système multirobot, qui plus est dans un environnement incertain, soulève des défis importants de résilience.

À ce constat viennent s'ajouter des aléas propres à la mission de chasse aux mines. Par l'impossibilité de prédire le nombre de mines, leur emplacement, voire leur existence, les objectifs d'identification ne peuvent être définis en amont de la mission. Ainsi, même si le système a initialement comme mission de couvrir une zone donnée, il doit pouvoir dévier du plan initial. Dès lors qu'un objet suspect sera détecté, cet aléa doit être géré par de nouvelles décisions.

Par leur existence, les aléas amènent une notion de *dynamique* au sein de la mission. Afin de ne pas compromettre la mission, le système doit être en mesure de détecter et comprendre ces aléas, c'est le rôle de la *supervision*. Une fois ces aléas intégrés, le système doit pouvoir se reconfigurer pour *réparer* les décisions prises.

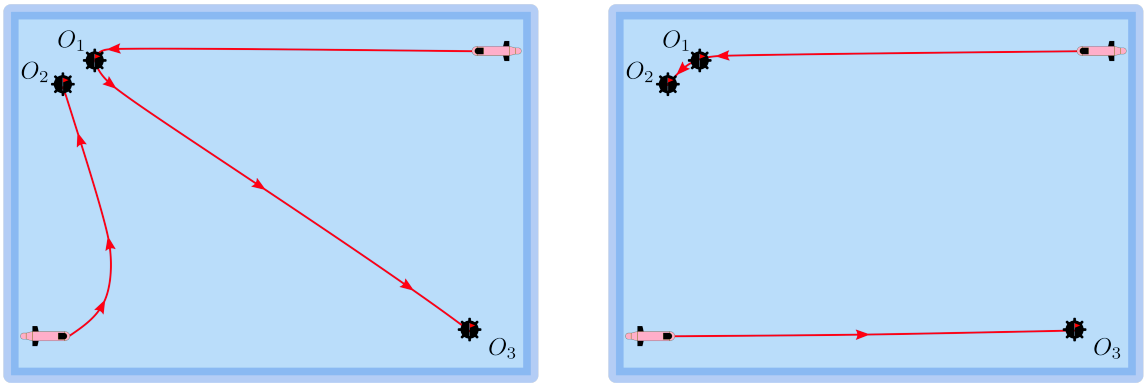
La bonne intégration de ces aléas par le système est définie par la boucle *Percevoir-Planifier-Agir*. Les aléas sont perçus, le système doit reconfigurer son plan de mission afin de les prendre en compte, puis le système exécute son plan.

2.5.4 *Contraintes sur et entre tâches*

Une part importante de la difficulté d'une prise de décision réside dans les liens entre objectifs et les contraintes associées. Ces contraintes et liens peuvent être multiples et plus ou moins stricts. Ils font apparaitre une notion de *dépendance* entre les tâches à accomplir.

Par exemple, une dépendance peut unir des tâches dans la manière de les réaliser. Comme dit précédemment et illustré dans la Fig. 2.14, il peut être plus intéressant pour un agent d'obtenir un ensemble de tâches plutôt qu'une seule. L'accomplissement d'un objectif particulier par un agent sera influencé par le besoin de réaliser ou non un autre objectif. Même dans le cas le plus minimal, sans considérer la manière dont sont réalisés les objectifs, une dépendance existe par la nature même des objectifs qui sont géographiquement répartis. Nous illustrons ce cas sur la Fig. 2.15. Deux agents doivent identifier trois objets O_1, O_2 et O_3. L'agent en charge de O_1 aura plus de facilité qu'un autre à s'occuper de O_2 car il a déjà à effectuer le trajet jusqu'à O_1. Ici la dépendance n'est pas une contrainte sur la faisabilité de la tâche, et donc l'existence d'un plan, mais sur l'optimisation de la solution.

Les objectifs à réaliser peuvent être encore plus fortement liés entre eux par des contraintes. Dans le cas le plus simple, les contraintes sont intrinsèques à l'objectif, p. ex. il faut accomplir cet objectif de couverture avant l'instant t (contrainte d'échéance). Ici, la tâche n'est pas directement dépendante d'une autre. Cependant, les contraintes peuvent également concerner plusieurs objectifs, p. ex. la couverture



(a) Le premier agent identifie O_1 et O_3. Le second identifie O_2.

(b) Le premier agent identifie O_1 et O_2. Le second identifie O_3.

FIGURE 2.15 – Cette mission d'identification illustre la dépendance des objectifs, liés à l'emplacement géographique des objets. Avec la seconde méthode, la longueur de la trajectoire totale des robots est minimisée.

de la zone z_x doit être réalisée avant celle de la zone z_y (contrainte de séquentialité, par exemple pour encapsuler une notion de priorité stricte). Dans ce second cas, la tâche est directement dépendante d'une autre. Cette notion de dépendance doit être explicitement intégrée au problème de prise de décision.

Le Tab. 2.1 liste les contraintes que l'on retrouve sur les objectifs de la chasse aux mines. À l'instar de [Gin17] nous utilisons la terminologie suivante : les contraintes de synchronicité intègrent un temps spécifique, p. ex. *durée* $= 10s$, et les contraintes de précédence comportent uniquement des relations d'ordonnancement, p. ex. tâche A *puis* tâche B. À cette terminologie nous ajoutons les contraintes de périodicité faisant intervenir un temps spécifique et cyclique, p. ex. faire un rapport toutes les 20 minutes. Enfin, nous considérons le cas des contraintes causales qui sont des contraintes entre tâches non-explicitement définies. Elles apparaissent quand la possibilité de réaliser une tâche dépend de la bonne réalisation d'une autre tâche. Par exemple, un robot sous-marin peut agir comme une mule de données en récupérant les données d'un coéquipier avant de remonter à la surface. Dans ce cas, la présence du coéquipier avec les données concernées à un instant et un endroit déterminé établit une contrainte causale avec l'objectif du robot devant transporter les données.

Type	**Temporelles**				**Causales**
Nom	Périodicité	Synchronicité		Précédence	
Portée	Monotâche	Monotâche	Multitâche	Multitâche	Multitâche*
Exemples	Faire un rapport Rafraîchir position GPS	Deadline Durée Fenêtre temporelle	Etablir des relais de communication	Couvrir une zone avant une autre Identifier un objet avant un autre	Transport de données Recharge

TABLE 2.1 – Résumé des contraintes pouvant s'appliquer aux tâches de la chasse aux mines. L'astérisque indique des contraintes non-explicites entre tâches.

Les contraintes de synchronicité font intervenir un temps spécifique et sont diverses. On peut, par exemple, définir une fenêtre temporelle dans laquelle accomplir l'objectif : il faut finir de couvrir cette zone dans les 90 prochaines minutes. Pour ce qui est des contraintes de synchronicité entre plusieurs tâches, nous utilisons les intervalles d'Allen pour les définir [All83]. Ces intervalles sont représentés sur le Tab. 2.2. Dans notre contexte nous utilisons, par exemple, des tâches jointes, c.-à-d. devant s'exécuter en même temps. Ainsi, si un robot doit identifier un objet suspect tout en transmettant les données brutes à l'opérateur, il peut être nécessaire d'établir des relais de communication pour pallier le problème de

portée des communications sous-marines à débit élevé. Dans ce cas, le drone identifiant l'objet et ceux servant de relais doivent accomplir leurs objectifs en même temps.

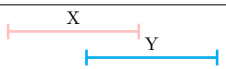
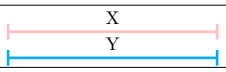
Relation	Illustration	Interpretation
$X < Y$ $Y > X$		X precedes Y Y is preceded by X
X m Y Y mi X		X meets Y Y is met by X (i stands for **i**nverse)
X o Y Y oi X		X overlaps Y Y is overlapped by X
X s Y Y si X		X starts Y Y is started by X
X d Y Y di X		X during Y Y contains X
X f Y Y fi X		X finishes Y Y is finished by X
$X = Y$		X is equal to Y

TABLE 2.2 – Intervalles temporels de Allen [All83].

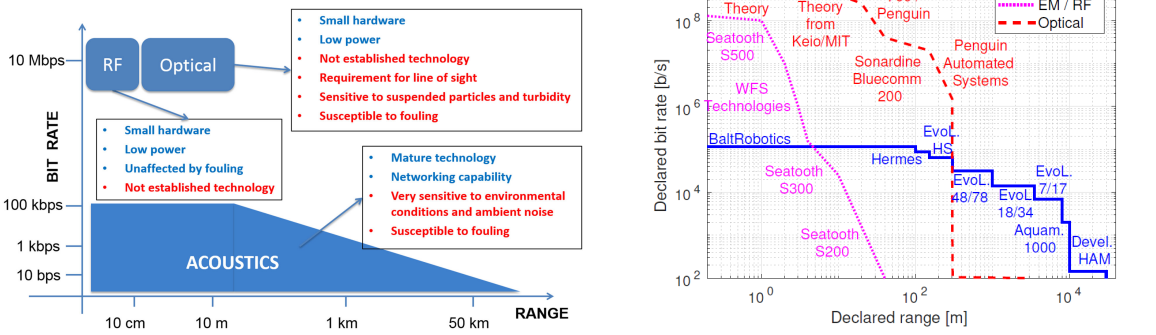
2.5.5 *Coopérer en environnement sous-marin*

Le déploiement de plusieurs unités et la nécessité de faire face à des aléas implique de faire circuler des informations et donc de *communiquer* durant la mission. Cette capacité de communication est fondamentale pour le système multirobot.

Dans un environnement hautement dynamique, le manque de communication impactera d'autant plus la résilience du système. Par exemple, un objet ne pourra jamais être identifié si aucun AUV apte à l'identifier ne reçoit l'alerte issue de sa détection. Sans cette capacité, il est impossible de développer la coopération entre agents hétérogènes, et par conséquent, de prendre des décisions ou de les réparer à l'échelle multirobot. En plus d'être nécessaires pour permettre au système un niveau d'autonomie suffisant, les communications sont indispensables à l'opérateur pour suivre l'évolution de la mission.

En environnement sous-marin, plusieurs supports, tels que l'optique, les ondes électromagnétiques, ou les ondes acoustiques, sont possibles pour transmettre des informations. Chacune de ces technologies est impactée différemment par l'environnement. La Fig. 2.16 montre quelques exemples actuels de moyens de communications sous-marines. En raison des fortes atténuations provoquées par l'eau sur les ondes électromagnétiques et la lumière, il est extrêmement difficile de dépasser des portées de l'ordre de la dizaine de mètres. En revanche, les ondes acoustiques peuvent voyager bien plus loin et même couvrir toute la surface du globe pour ce qui est des ultra basses fréquences. Les communications sous-marines sont donc réalisées quasi-exclusivement par le biais d'ondes acoustiques.

Malgré leur portée importante les communications acoustiques en milieu sous-marins souffrent de nombreux problèmes. La vitesse du son dans l'eau, c.-à-d. environ $1500 m.s^{-1}$, dépend de paramètres environnementaux tels que la salinité et la température. La variation de cette vitesse entraîne des problèmes, comme la perte locale de connectivité ou l'atténuation par trajets multiples ("multi-path fading"). Ces problèmes causent des pertes et retards dans les messages [APM05]. Enfin, bien qu'offrant de grandes portées, le son ne permet pas de communiquer à un débit élevé. La bande passante (bandwidth) est donc



(a) Vue schématique des technologies de communications sous-marines et défis associés [Fer+17b].

(b) Performances des technologies de communications sous-marines [Cam+17].

FIGURE 2.16 – Caractéristiques des moyens de communications sous-marines en termes de portée et de débit.

très limitée, ce qui constitue un obstacle supplémentaire aux communications nécessaires à un système multirobot.

Les communications sous-marines constituent un véritable défi pour la coopération des robots et doivent être prises en compte dans la conception du système [ABB17].

2.6 CADRE DE L'APPROCHE

Dans les précédentes parties de ce chapitre, nous avons souligné les défis entourant le déploiement d'un système multirobot autonome devant accomplir une mission de chasse aux mines. Dans cette section, nous définissons le cadre du système que nous utiliserons pour établir notre approche.

2.6.1 *Agents considérés et rôles*

Chaque objectif de chasse aux mines requiert des capacités différentes. La capacité d'un robot étant définie par sa charge utile, nous devons recourir à une équipe d'agents hétérogènes pour accomplir la mission. Par conséquent, nous définissons les unités suivantes :

— *Explorers* : AUV avec sonars à haute fréquence, en charge de l'objectif de *détection* ;

— *Identifiers* : AUV avec des capteurs chimiques, optiques ou électromagnétiques pour l'*identification* des objets suspects ;

— *Neutralizers* : AUV avec des charges explosives pour la *neutralisation* ;

— *Helpers* : Unmanned Surface Vehicle (USV) qui déploient et rechargent les AUV tout en servant de lien de communication entre les drones sous-marins et les opérateurs.

Dans notre système, nous considérons un système multirobot autonome composé au maximum d'une dizaine d'unités. Étant donnés des critères comme la taille des zones à couvrir, la vitesse moyenne des drones existants en couverture, et le prix des drones, ce nombre d'unités nous semble adéquat pour poser les bases de notre approche. En raison de la forte composante couverture de la mission, les *explorers* sont généralement majoritaires.

De plus, nous considérons également l'existence d'un opérateur humain. Les unités devant évoluer de manière autonome, cet opérateur n'a pas pour but de les téléguider. Cependant, suivant les spécifications de la mission, les unités pourront être dans l'obligation de faire des rapports réguliers à l'opérateur. Une

des prérogatives de l'opérateur est de pouvoir changer les paramètres de la mission lorsqu'il est en contact avec au moins un des robots. Par exemple, durant la mission, l'opérateur peut augmenter la taille de la zone à couvrir. Cette augmentation se traduit par l'ajout de nouveaux objectifs. Il est important de souligner qu'aucune décision n'est associée à cet ajout, c.-à-d. qu'aucun ordre n'est donné à des agents spécifiques. De la même manière que pour un aléa, c'est au système de se réorganiser pour faire face à ces nouvelles informations.

Les liens entre l'opérateur et les agents se résument donc à :

— Des agents vers l'opérateur : Surveillance du système ("monitoring"), par exemple via l'émission de rapports.

— De l'opérateur vers les agents : Modification des paramètres de la mission, par exemple via l'ajout de nouveaux objectifs.

2.6.2 *Hypothèses sur le système et l'environnement*

Nous formalisons notre approche autour de plusieurs hypothèses. Ces hypothèses permettent de s'abstraire de la forte complexité de l'environnement sans dénaturer le problème à résoudre. Elles définissent un cadre théorique permettant d'adresser les défis fondamentaux de la mission considérée.

Communications acoustiques

Modéliser la propagation du son dans l'eau est extrêmement complexe [Ett03]. Cette complexité vient du nombre de paramètres à considérer (température, salinité, pression...) mais également de leur grande variabilité. Des simplifications sont donc nécessaires pour explorer efficacement les nombreux défis entourant l'utilisation d'un système multirobot pour une mission de chasse aux mines. Pour poser ces hypothèses et s'abstraire de la modélisation du son, il faut identifier comment les communications sous-marines sont susceptibles d'impacter le système.

À l'échelle d'un système multirobot, les communications sous-marines via des ondes acoustiques s'illustrent par ces principales caractéristiques [Cam+17 ; Ett03 ; ABB17] :

— Des portées importantes et des débits faibles ;

— Une bande passante limitée ;

— Des altérations de messages, voire des pertes complètes. Ces pertes sont principalement dues à l'atténuation, aux aléas acoustiques (passage d'un navire), et aux phénomènes de chenaux acoustiques provoquant des zones d'ombres. Des pertes peuvent donc arriver, qu'importe la distance entre émetteur et récepteur.

Dans notre cadre théorique, nous considérons donc les hypothèses suivantes :

— Bien que les unités soient hétérogènes dans leurs capacités de perception et de navigation, nous considérons qu'elles embarquent toutes les mêmes moyens de communication ;

— Débit et portée du modem : Environ $3kbps$ pour une portée de $10km$, conformément à ce qui est indiqué dans [Cam+17 ; Yor18] ;

— Bande passante : Quelques kHz ;

— Perte de message : Dans le même esprit que [OKS19], nous considérons un modèle de Bernoulli pour simuler la perte de communication.

Il est important de noter que la portée théorique du modem est suffisamment grande pour couvrir la zone de mission en tout point. Le modèle de Bernoulli utilisé nous permet également de simuler des ruptures de communications pour des robots proches, comme pourraient le causer des chenaux acoustiques provoquant des zones d'ombres.

Afin de gérer les pertes de messages, nous utilisons un protocole d'accusé de réception. Ce protocole permet à l'émetteur d'une information de valider la bonne réception du message, si nécessaire. Nos robots étant équipés des mêmes modems et devant faire face aux mêmes conditions, nous considérons raisonnablement que les communications sont bilatérales. Ainsi, si à un instant t un agent peut communiquer avec un autre, nous considérons que l'inverse est vrai. Par conséquent, la bonne réception de l'accusé de réception ne relève que du protocole. L'hypothèse simplificatrice ici est donc que si un message est reçu, alors son accusé de réception l'est aussi.

Connaissance partielle de l'environnement

Connaître son environnement est une nécessité pour prendre des décisions qui ne seront pas systématiquement remises en question par la réalité du terrain. Cependant, dans un théâtre d'opération aussi vaste et complexe que celui d'une mission de chasse aux mines, il est impossible de connaître parfaitement l'environnement. Par exemple, les robots ne peuvent connaitre à chaque instant la force et la direction du courant sous-marin en tout point.

Néanmoins, s'il est impossible d'être parfaitement informé sur l'environnement, certaines connaissances, bien qu'incomplètes, peuvent être exploitées par le système. Dans le cadre de notre mission, nous faisons l'hypothèse que des **informations partielles** (courant sous-marin, nature des fonds...) sont récupérées via les évaluations environnementales introduites dans la Sec. 2.3.1 et déjà utilisées par les méthodes conventionnelles de chasse aux mines. Par conséquent, le système doit évoluer dans un *environnement incertain*.

2.6.3 *Ambition*

Pour développer notre approche, nous utilisons une mission de type *Port allié*, présentée à la Sec. 2.2.1. Ce type de mission est primordial pour la souveraineté des marines. Bien que ces missions n'imposent pas de limite de temps ou de discrétion sur les communications, comme pour les missions *Point d'étranglement* et *Débarquement*, elles sont riches en aléas et contraintes. Par conséquent, ce type de mission peut être utilisé comme cas d'usage de référence dans le développement d'un système chasseurs de mines autonome.

Notre objectif est de proposer une architecture de décision, supervision et réparation capable de :

- **Allouer** et **planifier** des objectifs ;
- **Exécuter** le plan et **superviser** son exécution ;
- **Réparer** le plan lorsque nécessaire, notamment :
  - à l'échelle de l'agent, c.-à-d. *localement* ;
  - à l'échelle de l'équipe, c.-à-d. *globalement*, en :
    - Intégrant des **nouveaux objectifs en ligne** ;
    - Réparant des **décisions prises**, c.-à-d. sur des plans déjà engagés.

Pour des problèmes **Partiellement-Ordonnés**, pouvant intégrer des contraintes :

- Causales ;

— Temporelles de :
  — Précédence ;
  — Synchronicité ;
  — Périodicité.

Dans un environnement contraint par les communications :

— Perte de message ;
— Bande passante limitée.

Une vue d'ensemble des processus nécessaires à mettre en place est représentée dans la Fig. 2.17. Par le système décrit dans la Sec. 2.6.1 et les hypothèses posées dans la Sec. 2.6.2, nous avons défini un cadre théorique permettant d'explorer les défis soulevés par ces processus entourant la coopération multirobot en environnement sous-marin tout en restant en adéquation avec la finalité opérationnelle du système, c.-à-d. l'accomplissement d'une mission de chasse aux mines.

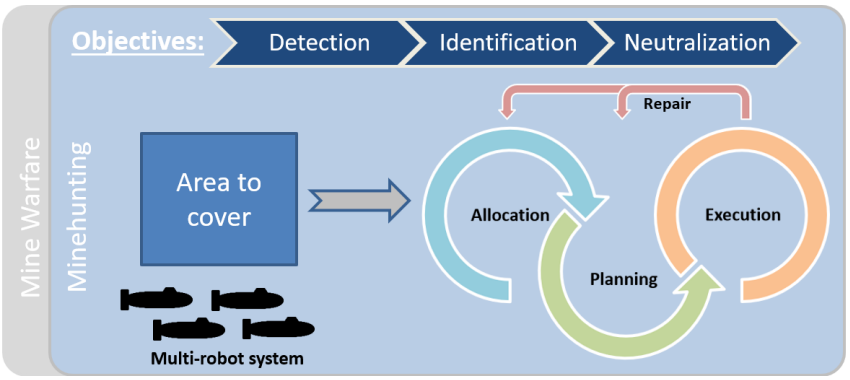


FIGURE 2.17 – Vue d'ensemble des processus nécessaires au système multirobot considéré.

3

ÉTAT DE L'ART : ALLOCATION AU SEIN DE SYTÈMES MULTIROBOTS

Aperçu

Dans ce chapitre, nous présentons une synthèse des travaux dédiés aux fonctions d'allocation, de supervision, et de réparation de tâches pour des systèmes multirobots. Nous introduisons en premier lieu les processus régissant l'autonomie de tels systèmes. Puis, nous définissons le problème d'Allocation de Tâches MultiRobot (MRTA) avant de présenter les principales architectures permettant de le résoudre. Nous détaillons par la suite l'utilisation d'architectures Market-Based, et plus précisémment d'enchères. Enfin, nous dégageons les principales caractéristiques de notre approche par une synthèse des différents éléments abordés.

Les systèmes multirobots sont avant tout un sous-type des *systèmes multiagents*. Dans [Fer95], la définition suivante est proposée pour caractériser un agent : "entité réelle ou virtuelle, évoluant dans un environnement, capable de percevoir et d'agir dessus, qui peut communiquer avec d'autres agents, qui exhibe un comportement autonome, lequel peut être vu comme la conséquence de ses connaissances, de ses interactions avec d'autres agents et les buts qu'il poursuit".

Dit autrement, un robot est une entité réelle composée d'un ensemble de processus lui permettant d'accéder à l'autonomie [Kun+18]. En vue de l'accomplissement d'un objectif, ces processus doivent permettre au robot d'être conscient de son environnement et de pouvoir agir dessus. Ces processus sont usuellement regroupés au sein d'une boucle *Percevoir-Planifier-Agir* [Gat+97]. Le robot perçoit son environnement, au regard des informations perçues il planifie ensuite ses actions pour accomplir un but, enfin il exécute son plan. Ces étapes forment une boucle, car le robot doit superviser l'exécution de son plan, en conséquence il doit percevoir, replanifier et exécuter en continu. Ces processus sont agencés au sein d'une architecture.

La boucle décrite ici permet de caractériser un robot évoluant seul, c.-à-d. un système monorobot. Cependant, dans le cas d'un système multirobot, la présence de coéquipiers impacte les processus de cette boucle et y ajoute également un nouveau processus fondamental, l'*allocation*. Ce processus apparait par la notion de *coopération* propre aux systèmes multirobots.

En repartant de la définition d'un agent présentée dans [Fer95], un système multiagent peut être défini comme un agrégat d'agents, pouvant eux-mêmes être des robots, des processus informatiques ou encore des êtres vivants. Si rien n'oblige ces agents à partager un but commun, c'est généralement le cas quand l'on parle de systèmes multirobots. En effet, la plupart des systèmes explorés sont conçus par l'homme, et ce dans des buts précis. Ces systèmes se définissent par un nombre important de paramètres. Parmi ces paramètres on retrouve en premier lieu le nombre d'agents et leur hétérogénéité, chaque agent pouvant, par exemple, avoir des compétences cognitives ou perceptives différentes. Le grand nombre de paramètres à considérer et leurs interactions rendent ces systèmes très complexes à étudier [Car13].

Ces agents aux capacités diverses et partageant un but commun doivent pouvoir coopérer pour accomplir efficacement ce but. C'est dans la mise en place de cette coopération que réside le plus grand défi du

déploiement de systèmes multiagents. Cette coopération passe par deux éléments absents des architectures monorobots, le besoin de *communiquer* pour pouvoir *allouer* des objectifs. Ces éléments permettent de prendre des décisions à l'échelle multirobot. La prise de décision multirobot étant fondamentale au sein d'un système multirobot, dans ce chapitre nous en établissons un état de l'art.

3.1 LE PROBLÈME D'ALLOCATION DE TÂCHES MULTIROBOT

Afin de mener à bien une mission les robots doivent coopérer. En fonction de ses capacités et de la situation actuelle, chaque robot du MRS peut réaliser différemment un objectif donné : un point essentiel au déploiement d'un MRS est le problème de Multi-Robot Task Allocation (MRTA), qui optimise la répartition au sein de la flotte des tâches à réaliser, en fonction de la situation actuelle.

Definition 3.1 (Multi-Robot Task Allocation (MRTA))

Étant donné $R = (r_1, \ldots, r_n)$ un ensemble de n robots et un ensemble $Q = (t_1, \ldots, t_s)$ de s tâches, résoudre le problème MRTA consiste à trouver une allocation $A : Q \to R$, c'est-à-dire allouer chaque tâche $t \in Q$ à un robot $r \in R$.

Dans cette section, nous introduisons les modèles sur lesquels il est possible de s'appuyer pour résoudre le problème de MRTA. Nous soulignons ensuite comment la présence d'alternatives pour accomplir une tâche peut impacter le processus de décision. Puis, nous montrons qu'en pratique le système ne doit pas résoudre un problème de MRTA mais plusieurs durant la mission. Enfin, nous définissons les principales propriétés que doit posséder le mécanisme de résolution.

3.1.1 *Caractérisation de tâche et fonction objectif*

Afin de pouvoir prendre des décisions, les robots ont besoin de raisonner sur des modèles mathématiques formels. Ces modèles définissent s'ils sont en capacité d'effectuer la tâche, mais peuvent également caractériser d'autres informations. Par exemple, il est possible de déterminer à quel prix (coût) et pour quel degré de réussite (récompense) un robot peut réaliser une tâche. Tous ces éléments nourrissent la valeur de la contribution du robot à l'équipe.

L'éligibilité d'un robot pour une tâche dépend donc non seulement d'un filtrage classique sur les capacités requises, mais également d'une notion de rentabilité. L'établissement d'un modèle mathématique encapsulant ces informations permet de définir une fonction objectif. Cette fonction objectif pourra être minimisée ou maximisée selon des critères définis par l'opérateur afin d'optimiser l'accomplissement de la mission.

Généralement, la définition d'un tel modèle se fait autour des notions de coût et de récompense. Ces notions sont résumées dans [Mos10] :

— *Coût* : Combien coûtera l'exécution de la tâche par le robot, p. ex. en temps ou en énergie ;

— *Aptitude* : À quel point le robot est pertinent pour réaliser la tâche. Par exemple, s'il s'agit d'une tâche où une position doit être rejointe de manière urgente, le robot le plus rapide sera le plus indiqué ;

— *Récompense* : Quel gain sera obtenu par l'accomplissement de la tâche par le robot ;

— *Priorité* : La tâche est-elle à accomplir le plus tôt possible ou peut-elle être repoussée au profit d'autres tâches.

L'optimisation multicritère étant un problème complexe, le plus souvent une combinaison de ces éléments est utilisée pour représenter la valeur d'accomplissement de la tâche. Cette combinaison est appelée *utilité* ("utility") [GM04a].

Un exemple d'une telle combinaison peut être *utilité = récompense - coût*. Cependant l'établissement d'une valeur d'utilité sous forme de scalaire nécessite de pondérer au mieux chacune des composantes.

Il est à noter que les notions d'aptitude et de priorité proposées dans [Mos10] sont le plus souvent incluses dans le coût et la récompense par des fonctions d'utilités individuelles. La mise en place d'une fonction d'utilité individuelle nécessite l'établissement d'une correspondance avec la fonction objectif globale [Dia+06].

L'utilité peut ainsi directement être utilisée au sein de la fonction objectif guidant la résolution du problème de MRTA. Dans [Mos10], différentes optimisations de fonctions objectifs sont décrites, par exemple :

— Minimiser le coût ou maximiser l'utilité du pire agent ;

— Minimiser la somme des coûts individuels / Maximiser la somme des utilités individuelles ;

— Minimiser le coût moyen par tâche.

Il est important de faire attention aux effets secondaires liés à la définition de l'objectif. Par exemple, dans le minmax du pire agent le reste des coûts sera caché, c.-à-d. que les autres agents pourront avoir des solutions éloignées de leur minmax local. Ces coûts cachés peuvent résulter en des solutions locales peu pertinentes. Si une telle liberté de comportement est acceptée par l'opérateur alors cette fonction objectif est viable, sinon une autre, plus globale, devra être utilisée.

Ainsi la définition de la fonction objectif est principalement une affaire de compromis entre les critères d'intérêts. Une solution lorsqu'il n'est pas possible de définir une combinaison de critères est d'appliquer l'optimalité de Pareto sur plusieurs fonctions d'utilités. Cette optimalité garantit qu'aucun critère ne peut être amélioré sans en dégrader un autre. Cependant, les solutions de Pareto ne garantissent pas de minimum, ainsi certains critères utilisés peuvent présenter des résultats sous-optimaux [MM10].

3.1.2 *Allocation et décomposition*

La présence de modèles sur lesquels raisonner permet de guider la résolution du problème de MRTA. Sur la base de ces modèles, il est possible de faire un choix sur le robot devant accomplir une tâche, cependant ces modèles permettent également de faire un choix sur la tâche devant être accomplie. En effet, pour un même objectif il peut exister plusieurs alternatives pour le réaliser. Ainsi, le processus de décision doit faire un choix parmi ces alternatives en fonction des informations disponibles au moment du choix.

À l'échelle de l'agent, ces alternatives permettent de trouver une solution à l'accomplissement de la mission avec une plus grande flexibilité et efficacité. C'est également le cas à l'échelle d'un système multirobot, mais l'allocation de tâches rend l'utilisation d'alternatives plus complexe. Par exemple, si une tâche est allouée à un robot cela signifie qu'un choix a été fait sur la manière d'accomplir la mission, ce choix doit être diffusé auprès des coéquipiers afin de conserver la cohérence du système sur les tâches à accomplir. Si cette information n'est pas diffusée, deux robots peuvent faire deux tâches de deux alternatives différentes, rendant ainsi la mission sous-optimale [GM01].

Il est donc nécessaire de considérer avec prudence comment les alternatives peuvent impacter l'allocation au sein d'un système multirobot.

Pour représenter les alternatives, il est possible d'utiliser des décompositions. Une décomposition spécifie qu'une tâche donnée peut être accomplie par la réalisation de plusieurs sous-tâches. La tâche est

un objectif à accomplir et les sous-tâches correspondent à une manière d'accomplir cet objectif. À chaque alternative peut alors correspondre une décomposition. Ces décompositions sont usuellement encodées au sein d'un réseau de tâches hiérarchiques, formant ainsi un arbre décomposant le problème initial en plusieurs sous-problèmes [EHN94].

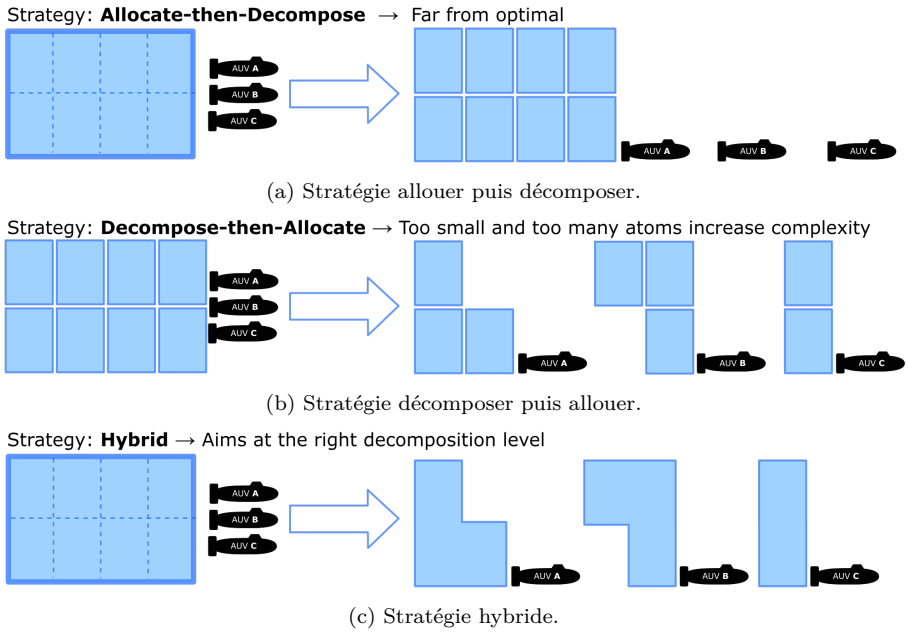
À partir de ce modèle, il est possible d'organiser la décomposition et l'allocation pour prendre en compte ces alternatives. Il existe trois stratégies pour organiser la décomposition et l'allocation [Mos10] :

— *Allouer puis décomposer* ("Allocate then decompose") : Allouer l'objectif de mission puis le raffiner en éléments plus simples à appréhender. À l'issue de la décomposition réalisée par les agents des réallocations supplémentaires peuvent être nécessaires [BA99] ;

— *Décomposer puis allouer* ("Decompose then allocate") : Les tâches complexes sont d'abord décomposées avant d'être allouées aux agents. Les objectifs primitifs ainsi obtenus facilitent la planification et l'allocation. Cependant les éléments importants issus de l'allocation ne peuvent nourrir la décomposition [Cal+90] ;

— *Hybride* ("Hybrid") : La décomposition et l'allocation ne sont pas totalement séparées et peuvent être entrelacées. Les informations pertinentes issues d'une décomposition ou allocation peuvent être prises en compte, menant à des solutions généralement plus abouties au prix de problèmes plus complexes [ZS03].

Ces stratégies sont illustrées sur la Fig. 3.1.



(a) Stratégie allouer puis décomposer.

(b) Stratégie décomposer puis allouer.

(c) Stratégie hybride.

FIGURE 3.1 – Illustration des trois principales stratégies d'allocation et décomposition.

Les deux premières stratégies sont extrêmes et, comme souvent dans un tel cas, un compromis est préférable afin d'arriver à une solution de meilleure qualité. Ce compromis prend la forme d'une approche hybride entremêlant décomposition et allocation. En entremêlant ces deux processus, les robots pourront s'allouer et planifier des tâches de manière plus libre et informée.

3.1.3 *Allocation de tâche et réparation en ligne*

En pratique, la résolution de problème de MRTA n'est pas unique mais multiple. En effet, les approches résolvant les problèmes d'allocation de tâches multirobot peuvent être caractérisées suivant différents critères tels que le type de problème, le mécanisme de décision, l'organisation du système et les techniques de réparation [KHE15]. Ce dernier point implique que dans certains contextes, le problème de MRTA peut devoir être résolu plusieurs fois en ligne, c.-à-d. pendant l'exécution de la mission. Il n'est donc pas suffisant de le résoudre une unique fois lors de la préparation de la mission, aussi appelée planification initiale.

Notamment, [GM03a] compare le problème de MRTA à un problème d'affectation optimale ("Optimal Assignement Problem") devant être résolu de manière périodique. Par conséquent, la résolution du problème d'allocation de tâches doit pouvoir être appelée de manière dynamique en fonction de nouvelles informations ou événements. Cette dynamique implique d'être apte à *réparer* le plan global du système.

Une réparation survient lorsqu'un plan déjà engagé doit être modifié pour intégrer de nouvelles informations. Par exemple, l'intégration de nouveaux objectifs à accomplir, la prise en compte d'un retard sur un objectif, ou la panne d'un capteur. Sans réparation, le plan du robot, voire la mission, peuvent être compromis. Il est donc important pour le robot de pouvoir superviser la bonne exécution de son plan et de pouvoir faire appel à des mécanismes de réparations si nécessaire. Ces mécanismes peuvent permettre de trouver un plan alternatif s'il existe [Bec+18 ; GLB13].

Les réparations peuvent être locales au robot ou globales. Par exemple, si le robot doit changer l'ordre dans lequel il accomplit des tâches qui ne dépendent que de lui, c'est une réparation locale qui n'impacte que son plan. Dans le cas global, le robot doit réparer des tâches dont ses coéquipiers dépendent. La réparation devra idéalement prendre en compte les coéquipiers concernés pour proposer un nouveau plan global incluant les dits coéquipiers [Les+16].

Cette réparation globale est particulièrement complexe à mettre en place. Les systèmes multirobots permettent une meilleure résilience face aux aléas en offrant plus de possibilités de réparation, cet atout s'accompagne cependant d'un inconvénient. Selon le contexte, la propagation des réparations à tous les agents pour atteindre un nouveau plan global peut être difficile à atteindre [CLP15], par exemple, si les coéquipiers concernés par un aléa ne peuvent pas communiquer.

Enfin, la résolution de problèmes de MRTA en ligne devient autrement plus complexe lorsque les tâches sont contraintes entre elles (Sec. 2.5.4). Dans un tel cas, à chaque phase de décision, il sera nécessaire de connaître l'historique des dernières décisions afin de ne pas casser des contraintes qui ont été établies. Sans cela, l'apparition de blocages à l'exécution est possible, p. ex. parce qu'un robot n'a pas tenu compte qu'il devait atteindre la fin de la tâche d'un coéquipier avant d'exécuter la sienne.

3.1.4 *Propriétés requises d'un mécanisme de décision multiagent*

La résolution d'un problème de MRTA passe par l'utilisation d'un processus de décision. Ce processus de décision peut comporter des propriétés qui le rendront plus ou moins adapté à la situation. Par exemple, dans le cas d'une réparation en ligne introduit à la section précédente, la rapidité du processus pour converger vers une nouvelle décision peut être vitale.

Dans [Mos10], un ensemble de propriétés génériques pour caractériser un mécanisme de décision multiagent est proposé :

— *Validité mathématique* : Capacité à apporter des garanties, par exemple sur le temps de calcul ou les performances ;

— *Distribuabilité* : Capacité à répartir la complexité du travail dans tous les agents du système, permettant ainsi un meilleur usage des ressources ;

— *Décentralisation* : Capacité à répartir le processus de décision de manière à ne pas comporter d'éléments centraux dont l'échec peut compromettre la totalité du système ;

— *Passage à l'échelle* : Capacité à conserver de bonnes performances en augmentant la dimension du problème (nombre d'agents).

— *Tolérance aux fautes* : Capacité à corriger les décisions prises lorsqu'elles sont remises en question ;

— *Flexibilité* : Capacité à supporter de nouveaux contextes, par exemple intégrer de nouveaux agents ou objectifs ;

— *Réactivité* : Capacité à réagir rapidement à une nouvelle information.

Ces propriétés, bien que fondamentales, sont difficiles à atteindre et rarement compatibles. Par exemple, la décentralisation du système s'accompagne, dans de nombreuses approches, par la perte de garanties sur le processus de décision [KHE15]. Ainsi, la conception du système passe par des compromis entre ces propriétés. Ces choix sont à faire en fonction de l'ensemble des missions devant être accomplies par le système multirobot.

Nous pouvons illustrer ces propriétés en soulignant les différences entre systèmes multirobots et essaims, dont les limites sont souvent floues. La robotique en essaim a comme qualité principale le passage à l'échelle. Cette propriété est atteinte le plus souvent par l'usage d'agents homogènes relativement peu efficaces face à l'objectif à accomplir s'ils ne coopèrent pas entre eux [Sen+16]. Par le grand nombre d'agents et leur redondance, les essaims sont également très robustes aux pertes d'agents. À l'inverse, les systèmes multirobots sont généralement composés d'un nombre réduit d'agents, le plus souvent hétérogènes [DJM02]. Chacun des agents composant le système peut accomplir une partie des tâches seul, ils ne sont pas impuissants lorsqu'isolés. La perte d'un agent, s'il ne signifie pas l'échec de la mission, n'est pas anecdotique et oblige le système à se reconfigurer.

Propriétés principales pour la chasse aux mines

Au regard de la mission que notre système doit accomplir, définie à la Sec. 2.6.3, il est fondamental de mettre en place un mécanisme de décision permettant aux robots déployés de réparer leurs plans afin de prendre en compte des aléas comme la découverte d'objets suspects.

Les propriétés du mécanisme de décision qui nous semblent les plus importantes à intégrer au système sont résumées ici (Sec. 3.1.4) :

— *Décentralisation de la décision* : L'environnement sous-marin impose des contraintes fortes sur les communications, aussi il n'apparait pas comme raisonnable de se reposer sur une solution centralisée ;

— *Distribuabilité des calculs* : La mission à accomplir étant complexe, il est important de pouvoir répartir au sein des robots les coûts calculatoires de résolution du problème de MRTA ;

— *Tolérance aux fautes* : Les robots devant évoluer au sein d'un environnement incertain, ils doivent être capables de s'adapter à des aléas comme des retards, de nouveaux objectifs ou des pannes.

3.2 CLASSIFICATION DES PROBLÈMES D'ALLOCATION DE TÂCHES MULTIROBOT

Les problèmes de MRTA ont été grandement étudiés et beaucoup de variations existent, par exemple par le nombre de robots auquel peut être attribuée une seule et unique tâche. Ces variations donnent lieu à des problèmes très divers nécessitant l'utilisation d'approches différentes. Il est donc nécessaire de pouvoir les identifier, c'est le travail qui a été relevé dans la taxonomie proposée par [GM04b] et [KSD13].

Dans cette section, nous présentons cette taxonomie et nous étendons par la suite afin de considérer la notion d'engagement des robots. Enfin, grâce aux éléments introduits nous classifions notre problème de chasse aux mines.

3.2.1 *Taxonomie*

Dans [GM04b], la taxonomie suivante est proposée :

- *single-task robots* (ST) ou *multi-task robots* (MT)
  - ST : Chaque robot est capable d'exécuter au plus une tâche à la fois ;
  - MT : Certains robots peuvent exécuter plusieurs tâches simultanément.
- *single-robot tasks* (SR) ou *multi-robot tasks* (MR)
  - SR : Chaque tâche n'a besoin que d'un seul robot pour être réalisée ;
  - MR : Certaines tâches peuvent nécessiter plusieurs robots.
- *instantaneous assignment* (IA) ou *time-extended assignment* (TA)
  - IA : Toutes les informations sur les robots, les tâches, et l'environnement ne permettent qu'une allocation sans composante temporelle future. Toutes les allocations sont considérées comme instantanées et il n'est pas attendu de devoir en allouer à nouveau ;
  - TA : L'affectation des tâches se fait sur une période de temps. Toutes les tâches ne sont pas assignées au même instant.

Par exemple un problème de MRTA peut être décrit comme *ST-MR-IA*.

Cette taxonomie est cependant incomplète. Le principal défaut concerne l'impossibilité de prendre en compte des tâches contraintes entre elles, pouvant mener à des blocages à l'exécution [GM04b]. De plus, le coût de certaines tâches peut être dépendant d'autres tâches devant être accomplies, c.-à-d. le problème comporte des *utilités corrélées* ("interrelated utilities"). Par exemple, c'est le cas dans le Multiple Traveling Salesman Problem (mTSP), où le coût de visite d'un sommet dépend des précédents sommets que le robot a visités. Il est ainsi maladroit de décrire un tel problème comme *ST-SR-TA* tant les dépendances entre tâches complexifie le problème.

Pour pallier ce défaut, cette taxonomie a été étendue dans [KSD13]. En sus de prendre en considération de dépendances complexes entre tâches, cette extension inclue également l'utilisation de réseaux de tâches hiérarchiques, appelés Hierarchical Task Network (HTN) [EHN94]. Au sein d'un HTN, une distinction est faite entre les *tâches primitives* ("primitive task"), c.-à-d. ne pouvant être décomposées, et les *tâches abstraites* ("coumpound task", le terme "complex task" est aussi utilisé), c.-à-d. pouvant être décomposées en d'autres tâches abstraites ou tâches primitives. Ces décompositions sont précisées par des *méthodes* ("method"). Ainsi, une tâche abstraite peut être décomposée par une ou plusieurs méthodes.

Dans la taxonomie, les auteurs considèrent des tâches abstraites, pouvant être décomposées de plusieurs manières. Ces distinctions sont illustrées par la Fig. 3.2.

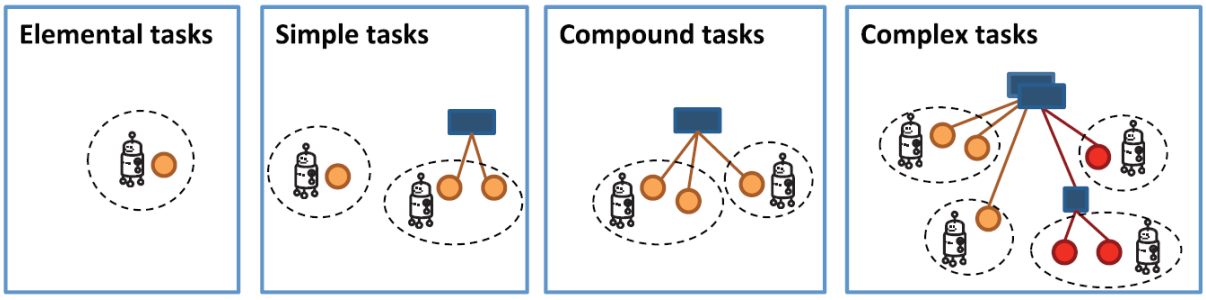


FIGURE 3.2 – Illustration des types de tâches [KSD13]. Les cercles pointillés indiquent des exemples de possibles allocations. Les cercles pleins représentent des tâches élémentaires et les rectangles des tâches abstraites.

Enfin, cette nouvelle taxonomie complète l'ancienne par la prise en compte de contraintes pouvant s'appliquer aux tâches. Pour ce faire, quatre marqueurs sont utilisés :

— *No Dependencies* (ND) : L'allocation et la décomposition peuvent être séparées car le problème est uniquement composé de tâches primitives et abstraites qui ont des *utilités indépendantes* quels que soient l'agent et la tâche. Ce type de problème est appelé "linear assignment problem" et peut être résolu en temps polynomial.

— *In-schedule Dependencies* (ID) : Un agent a des utilités dépendantes entre certaines tâches, c.-à-d. l'utilité d'un agent sur une tâche dépend d'au moins une autre tâche qu'il doit accomplir. La décomposition des tâches peut être séparée de l'allocation. Un problème de ce type est NP-hard.

— *Cross-schedule Dependencies* (XD) : Un agent a des utilités dépendantes entre certaines tâches effectuées par d'autres agents, en sus de *In-schedule Dependencies*. L'utilité d'un agent sur une tâche dépend de son planning, mais également de celui des autres, pour cette classe les dépendances sont prédéterminées, la décomposition peut être séparée de l'allocation. Un problème de ce type est NP-hard.

— *Complex Dependencies* (CD) : Un agent a des utilités dépendantes entre certaines tâches effectuées par d'autres agents pour des tâches abstraites, en sus de *In-schedule Dependencies* et *Cross-schedule Dependencies*. L'utilité d'un agent dépend de son planning et celui des autres ainsi que de la décomposition finale qui est retenue. Le problème de décomposition est donc couplé à celui d'allocation. Un problème de ce type est NP-hard.

Ces marqueurs sont illustrés par la Fig. 3.3.

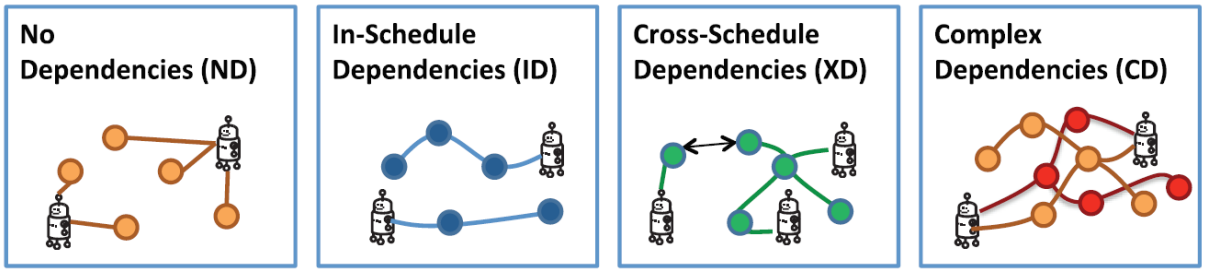


FIGURE 3.3 – Exemples illustrant les quatre catégories de la nouvelle taxonomie définies par l'interdépendance des tâches [KSD13].

Dans [KSD13], une définition de l'utilité en raisonnant sur des ensembles de robots et tâches est également proposée pour prendre en compte l'interdépendance des tâches et donc la notion de *interrelated utilities*. Elle reste cependant difficile à mettre en pratique en raison de la combinatoire des possibilités.

3.2.2 *Notion d'engagement et de coordination*

En sus des variations prises en compte par la taxonomie présentée ci-avant, il est possible d'ajouter à la classification du problème de MRTA la notion d'engagement des robots.

Dans [MSØ03], les auteurs mettent en lumière deux concepts intervenant dans la coopération multirobot, l'engagement (*commitment*) et la coordination :

— *Engagement* : Lorsqu'une contrainte d'engagement est placée, le robot ne peut abandonner sa tâche pour en accomplir une autre quand bien même l'utilité de ce geste serait profitable à l'équipe.

— *Coordination* : Les robots doivent s'assurer qu'ils ne font pas la même tâche. C'est-à-dire communiquer suffisamment pour confirmer qu'une tâche n'a été allouée qu'à un seul robot.

En ce qui concerne la coordination, [Loc10 ; LM09] illustrent par différentes expériences que les systèmes multirobots mettant en place une forte collaboration entre agents peuvent obtenir des gains substantiels par rapport à des robots ne coopérant pas et à des systèmes monorobots.

Pour ce qui est de la notion d'engagement, la présence ou non de contraintes d'engagement peut fortement impacter la résolution d'un problème de MRTA. Par exemple, on peut préférer attribuer une tâche à un robot moins à même de l'abandonner pour améliorer un profit local. Dans [MSØ03], quatre stratégies, reposant sur les couples possibles *engagement ou non* et *coordination ou non*, sont comparées. Cette comparaison est faite en simulation, mais également en expérimentations réelles avec prise en compte de pertes de communication.

Ces expérimentations montrent que les stratégies impliquant un engagement sont plus performantes pour le problème donné. Les auteurs relèvent cependant que les stratégies étudiées sont extrêmes et des compromis mériteraient d'être étudiés. Enfin, même si un lien entre simulation et résultats en expériences réelles semble se dessiner, les auteurs manquent de données pour confirmer la pertinence des résultats obtenus en simulation.

Cette notion d'engagement prend de l'importance suivant les aléas considérés, notamment dans le cas où un robot doit attendre l'accomplissement d'une tâche par un coéquipier [MSØ03]. Dans le cas où l'exécution du système serait considérée comme sans panne ou retard possible, un engagement laxiste est une première manière d'explorer les capacités de réparation d'un système. Ainsi, si un robot abandonne une tâche dont un coéquipier dépend, le coéquipier lésé doit trouver une manière de réparer son plan. Dans le cas où des aléas tels que des pannes ou retards sont considérés, l'importance de l'engagement est en retrait, car le robot ne peut avoir une confiance aveugle dans la bonne exécution de la tâche par son coéquipier. Il doit superviser son plan et, idéalement, celui de son coéquipier pour surveiller la bonne exécution de la tâche dont il est dépendant.

La notion d'engagement des robots est à considérer en amont du choix du processus de décision, elle fait partie de la description du problème. Par conséquent, nous proposons d'étendre la taxonomie présentée précédemment afin de spécifier si l'engagement des robots est requis (*Commitment*) ou non (*NoCommitment*).

3.2.3 *Classification du problème de chasse aux mines*

Selon la taxonomie définie dans les sections précédentes, notre problème de MRTA, défini à la Sec. 2.6.3, peut être caractérisé comme :

— *multi-task robots* (MT) : Un robot peut accomplir plusieurs tâches à la fois. Par exemple un *explorer* peut être amené à couvrir deux zones en même temps, c.-à-d. ne pas forcément finir de couvrir une zone avant d'en couvrir une autre, si cela lui permet d'optimiser sa trajectoire.

— *multi-robot tasks* (MR) : Certaines tâches peuvent nécessiter plusieurs robots pour être accomplies. Par exemple, afin d'identifier une mine tout en relayant des informations, des robots agissants comme relais de communication peuvent être nécessaires.

— *time-extended assignment* (TA) : Par la nature dynamique de la mission, il est impératif de considérer l'arrivée de nouveaux objectifs en ligne.

— *In-schedule Dependencies*, *Cross-schedule Dependencies*, et *Complex Dependencies* : La mission à accomplir étant partiellement ordonnée, le plan d'un robot peut comporter des contraintes internes mais également dépendre de celui d'un coéquipier. Enfin, il peut exister plusieurs plans pour accomplir un objectif donné, menant ainsi à la présence de dépendances complexes.

— *Commitment* : La mission étant par nature sensible, il est important de garder une certaine maîtrise sur la bonne réalisation des tâches. De plus, la présence de *Cross-schedule Dependencies* rend les robots dépendants les uns des autres, ils faut donc encourager le respect des engagements pris.

Notre problème de MRTA peut ainsi être catégorisé comme *MT-MR-TA with ComplexDependencies and Commitment*.

3.3 LES PRINCIPALES ARCHITECTURES

Une étape fondamentale dans la résolution du problème de MRTA consiste à définir la structure des processus de décision, de supervision, et de réparation ainsi que leurs interactions. L'agencement de ces processus est défini au sein de l'**architecture** du système. Les architectures sont à la base du développement d'un système autonome.

Il existe de nombreuses manières de concevoir une architecture visant à résoudre le problème d'allocation de tâches multirobot [Mos10]. Ces architectures sont variées, mais peuvent être regroupées suivant l'**organisation du processus de décision**.

On différenciera le plus souvent les approches où le système calcule un plan initial qui sera ensuite réparé par une entité responsable, des approches où les agents coopèrent autour de plans locaux sans résoudre un problème global. C.-à-d. les approches où le processus de décision ne repose que sur unique agent contre celles où chaque agent participe activement. La littérature s'accorde généralement à regrouper les premières dans la famille des architectures centralisées et les secondes dans celles des architectures distribuées, appelées également décentralisées [GM03b ; Dia+06].

Une ambiguïté subsiste cependant avec cette définition. En effet, cette seconde famille intègre comme un seul et même bloc un important groupe d'approches pourtant très hétérogènes selon le degré de coopération des robots. Par exemple, dans un schéma non centralisé, les robots peuvent ou non se concerter pour prendre une décision. Cette ambiguïté est renforcée par la définition variable entre des termes décentralisé et distribué selon les travaux.

Dans ce manuscrit et par souci de clarté, nous utilisons le critère de la coopération entre agents pour différencier les architectures décentralisées des architectures distribuées. Ainsi, nous considérons que dans une architecture décentralisée, les agents coopèrent pour prendre des décisions ensemble. À l'opposé, dans une architecture distribuée, les agents sont très autonomes et ne coopèrent que de manière opportuniste. Cette coopération n'est pas au coeur de leur processus de décision.

Nous classons ainsi les architectures dans un total de trois familles : **centralisées**, **décentralisées**, et **distribuées**. Nous pouvons notamment illustrer ces familles grâce au graphe de connexions par lequel transite la décision. Cette illustration se fait par une analogie des réseaux de communications, représentés sur la Fig. 3.4.

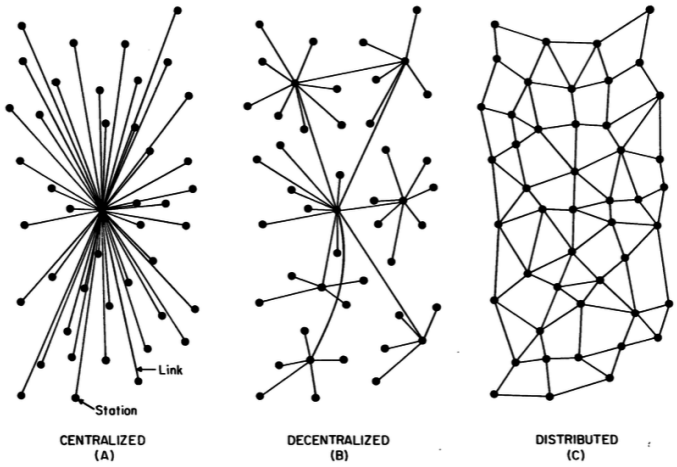


FIGURE 3.4 – Illustration des différents types de réseaux [Bar64].

Dans cette section, nous dressons une vue d'ensemble de ces familles d'architectures en en soulignant notamment les principaux avantages et limites. Puis, nous présentons comment ces architectures peuvent s'hybrider pour s'adapter à des situations différentes. Enfin, en lien avec l'importance des communications pour un système multirobot, nous soulignons comment ces architectures peuvent être utilisées au sein d'environnements contraignants.

3.3.1 *Architecture centralisée*

L'architecture centralisée est généralement la plus courante et simple à mettre en oeuvre. Tous les agents du réseau s'organisent autour d'un seul agent. Par exemple, dans le cas d'un système multirobot, le rôle de cet agent central est de :

1. Récupérer les informations de tous les agents ;
2. Résoudre le problème de MRTA, c.-à-d. planifier la mission pour l'ensemble de l'équipe ;
3. Transmettre les décisions prises, c.-à-d. le plan, à chaque agent.

La résolution du problème de MRTA peut ensuite se faire à l'aide d'un algorithme local à l'agent central.

Exemples d'approches

Une grande diversité d'approche existe pour résoudre le problème de MRTA de manière centralisée.

De nombreux travaux implémentent ces approches en modélisant le problème de MRTA comme un problème d'optimisation ou programmation sous contraintes [ABB17]. Dans [Ban+18], un problème d'optimisation linéaire en nombres entiers ("Mixed Integer Linear Programming (MILP)") est utilisé pour encoder le problème. Un plan global à l'équipe est ensuite trouvé grâce à un solveur spécialisé.

Il est également possible de représenter un problème de MRTA comme une variante de Traveling Salesman Problem (TSP) multirobot appelée mTSP. Par exemple, les sommets à visiter peuvent représenter

des tâches à allouer. C'est l'approche qui est utilisée dans [SM11] où les robots doivent surveiller régulièrement des positions. Des contraintes temporelles sont intégrées pour permettre d'assurer la régularité de visite des positions. La qualité des solutions obtenues est bonne, mais les temps de résolution trop longs pour envisager de déployer cette approche en ligne. Néanmoins, les auteurs soulignent qu'un tel algorithme peut toute fois être utilisé comme référence à des fins de comparaison. Dans [Ver18] un mTSP est étendu pour prendre en compte des contraintes temporelles ainsi que l'optimisation de l'énergie des agents. Avec les contraintes considérées, le problème devient très complexe et le passage à l'échelle impossible pour une utilisation en temps réel. Dans [Vid+13] des heuristiques spécifiques au problème de mTSP sont proposées pour réduire la complexité de la résolution.

Dans [Ban+22], la planification hiérarchique est utilisée pour résoudre le problème efficacement. La planification hiérarchique permet d'intégrer des connaissances expertes afin de guider la recherche d'une solution [Nau+99]. Le problème multirobot est décomposé en différents niveaux d'abstractions jusqu'à allouer des tâches individuelles aux robots. Cette méthode permet de réduire la complexité de la recherche de la solution en décomposant un problème complexe en plusieurs problèmes plus simples. Pour décomposer le problème un réseau de tâches hiérarchique est utilisé, il s'agit d'un arbre de tâches dont la racine peut être définie comme l'objectif à atteindre et les feuilles comme les tâches pouvant être exécutées par les agents. Ce principe est étendu dans [Gan+05] pour prendre en compte des aléas en reconstruisant le réseau de tâches.

Dans [SBD18b] une méthode basée sur la logique temporelle linéaire ("Linear Temporal Logic") est proposée. En s'appuyant sur cette logique, cette approche permet de mettre en place des contraintes temporelles finies au sein d'une tâche, mais sans pouvoir spécifier de dépendances entre plusieurs tâches.

Pour pouvoir réparer le plan en ligne avec ces approches il faut que tous les robots puissent communiquer avec l'agent central. Cela contraint fortement le système qui doit prendre en compte l'état futur de ses communications dans ses décisions. Par exemple dans [Ban+18 ; BBC18], le déploiement des robots est contraint de manière à maintenir les communications.

Avantages et inconvénients

Le principal atout des approches centralisées réside dans la qualité des solutions obtenues. En effet, sous réserve que l'algorithme utilisé intègre toutes les connaissances circulant dans l'équipe, il est possible d'atteindre une solution optimale. Cependant la centralisation des calculs se paye par un coût de calcul pour l'agent devant résoudre le problème de MRTA.

En pratique, plusieurs défauts rendent une approche centralisée difficile à utiliser pour un système multirobot. Ces défauts sont rappelés dans [Dia+06 ; GM03b] :

— Le passage à l'échelle : Avec un très grand nombre de tâches ou de robots, le coût calculatoire explose. Il devient alors difficile de proposer un plan dans des temps raisonnables, ce qui pose problème pour une prise de décision en ligne.

— La tolérance aux fautes : Ces approches intègrent un élément central d'échec. Toute rupture du lien de communication entre l'agent central et les autres rend impossible la suite de la mission.

Par construction, les architectures centralisées sont très sensibles aux aléas techniques telles que les pertes de communication et les pannes des agents. Certaines de ces approches contraignent le déploiement des robots pour limiter ces aléas, cependant avec ces approches il reste difficile d'adresser en ligne des problèmes de MRTA quand le nombre d'agents est grand ou le problème complexe. Par exemple, pour notre problème de chasse aux mines, classifié *MT-MR-TA with ComplexDependencies and Commitment*, l'utilisation d'une telle approche est difficile à envisager en raison du coût calculatoire.

3.3.2 *Architecture décentralisée*

Une architecture décentralisée permet de distribuer le processus de décision au sein du système multiagent. Ce processus n'est pas organisé autour d'un élément central fixe et donne plus de liberté aux agents qui s'organisent localement. Avec une approche décentralisée, les différents robots doivent échanger pour bâtir un plan global d'équipe.

Exemples d'approches

Dans l'architecture *HiDDeN* [GLB13 ; Les+16], les auteurs proposent une approche réparant le plan de mission de manière décentralisée. L'objectif de la mission est d'accomplir une mission de surveillance à l'aide d'un système multirobot hétérogène. Un premier plan est calculé hors-ligne grâce à la planification hiérarchique. L'originalité de l'approche repose sur l'encodage de ce plan global en plusieurs plans locaux encodés sous la forme de HTN. En cas d'aléas durant la mission, chaque robot appelle réactivement des méthodes de réparation modifiant son HTN. Par exemple, s'il n'est plus possible d'effectuer une tâche, une autre branche du HTN est utilisée comme alternative, si aucune alternative n'existe, l'algorithme essaye de réparer la tâche parente. Ce processus permet de ne réparer que des sous-problèmes et non pas le problème dans sa globalité. Les réparations sont locales tant que possible, si nécessaire les robots communiquent pour procéder à des réparations globales. Les auteurs soulignent qu'un des principaux avantages de cette approche est la réduction des communications nécessaires au système pour réparer.

Les *allocations par enchères* sont un cas d'école d'architecture décentralisée. Leur principe est fondé dans [Smi80]. Elles permettent à un agent de souscrire un contrat auprès d'un autre agent, par exemple pour lui acheter une tâche. De telles méthodes ne permettent pas, ou peu, de garantir les performances du système [Lag+05] mais, suivant le contexte et l'approche, elles démontrent empiriquement de bons résultats. Au cours des quarante dernières années elles ont reçues beaucoup d'attention de la part de la communauté multirobot [Dia+06].

Par exemple, dans [Dai+20], une méthode d'allocation par enchères est utilisée pour résoudre des problèmes de MRTA en ligne. Les auteurs soulignent de bons résultats tant en qualité d'allocation qu'en temps de calculs, cependant dans cette approche les tâches à réaliser sont simples. Il est également à noter que la qualité des allocations dépend du nombre de robots participant à l'enchère.

Le fort intérêt de la communauté multirobot pour les méthodes d'allocation par enchères est principalement dû à la grande flexibilité de leur principe de base. En effet, en conservant la logique d'un contrat entre agents mais en modifiant les algorithmes établissant ce contrat il est possible de faire des méthodes très différentes et adaptées à des situations tout aussi variées, aussi, nous reviendrons en profondeur sur ces méthodes dans une section ultérieure.

Avantages et inconvénients

Les approches décentralisées sont plus complexes à mettre en place et à étudier que les approches centralisées. Par l'absence d'agent central, ces architectures offrent en revanche une plus grande robustesse aux pertes de communication. Cependant, dans le cas où un élément serait totalement isolé, seul une autonomie suffisante peut lui permettre de continuer la mission à son échelle en résolvant ses problèmes locaux.

La qualité de la contribution d'un robot dépend directement de la quantité d'informations pertinentes disponibles sur l'état du système et son environnement. Ainsi, l'approche décentralisée, qui raisonne à une

échelle locale du système multirobot, peut mener à des plans sous-optimaux. Néanmoins, ce raisonnement à l'échelle locale facilite le passage à l'échelle, seul un nombre réduit d'agents doit être considéré.

Afin que chaque robot puisse apporter une contribution pertinente, il est important d'assurer un partage des connaissances suffisant. Les approches décentralisées doivent donc mettre en place des algorithmes de fusion de données efficaces, permettant de faire circuler les informations sans alourdir le système avec des redondances [Hol+11].

Différents travaux, comme dans [CLP15], explorent la quantité d'information à connaître sur les plans locaux des autres agents pour conserver cette cohérence.

3.3.3 *Architecture distribuée*

De même que pour l'architecture décentralisée, l'architecture distribuée permet aux agents d'évoluer seuls durant la mission et de coopérer avec leurs voisins afin d'améliorer leurs plans locaux. Cependant, leur autonomie est bien plus grande et leur coopération n'est pas au centre de leur prise de décision. Il n'y a pas de résolution d'un problème global qui soit commun à tous les agents. L'agent est généralement le seul à prendre des décisions pour lui.

Exemples d'approches

Ces approches peuvent utiliser des méthodes inspirées des sociétés d'insectes ou du comportement humain. On retrouve ainsi l'utilisation de techniques bioinspirées, telle que les phéromones pour se déplacer en groupe [VB01] et où la coopération est implicite. Avec une telle méthode, le passage à l'échelle est possible, l'ajout d'agent n'impacte pas la complexité des problèmes à résoudre. Cette coopération implicite peut également s'appuyer sur l'organisation du système par l'attribution de rôles à chacun des robots, cette solution est explorée par [VSH99] pour le cas du football multirobot.

Dans [ABB17], une stratégie distribuée et opportuniste est proposée. Les robots ne prévoient pas quand communiquer, c.-à-d. ils n'établissent pas de rendez-vous. Quand des robots finissent par être à portée de communication en accomplissant leurs plans, ils échangent des informations afin d'améliorer leur base de connaissance et pouvoir prendre de meilleures décisions. Avec une telle approche, les robots sont très peu encouragés à coopérer, leurs échanges sont uniquement opportunistes, mais le passage à l'échelle est possible. Dans le même esprit, [SB03] propose une approche où chaque robot partage ses observations et construit ses déplacements sur la connaissance actuelle et partagée de la position de cibles à traquer.

Dans [SE18], les auteurs présentent un algorithme de décision basé sur l'utilisation de cellules de Voronoi. Chaque agent calcule localement le découpage des cellules. Cet algorithme prend en compte l'hétérogéité des capteurs des coéquipiers, par conséquent les agents doivent s'échanger des informations sur leurs positions et capteurs afin de faire converger leur découpage.

Avantages et inconvénients

Dans une approche distribuée les agents sont très autonomes et n'ont à résoudre que des problèmes locaux. Un avantage immédiat d'une telle approche est le passage à l'échelle qui est grandement facilité par cette autonomie.

Cependant, au sein d'une approche distribuée, les robots ne sont pas dans l'obligation de communiquer pour prendre des décisions. La coopération résultante du système est faible, les robots ne se concertent que très peu pour prendre des décisions.

Par nature, les approches distribuées permettent beaucoup d'autonomie et de résilience au système, mais rendent extrêmement difficile l'émergence d'un consensus global. Par conséquent, l'équipe ne peut se coordonner autour d'objectifs complexes. Dans un problème de MRTA, ce type d'approche est le plus généralement utilisé pour des tâches peu complexes avec un grand nombre d'agents, on parlera alors le plus souvent d'essaim [Jev+12].

Un des problèmes résultants est l'accomplissement superflu d'une même tâche plusieurs fois par des robots qui ne se sont pas concertés. Dans le cas où les tâches sont coûteuses, p. ex. longues à réaliser, ce manque de coopération fait chuter les performances du système. En raison de la grande autonomie des agents, il est également difficile de garder une bonne compréhension des décisions du système [Jon+20].

3.3.4 *Des architectures mixtes*

Les architectures centralisées, décentralisées, et distribuées, ne sont classifiées que par l'organisation de leur processus de décision. Si cette organisation change durant la mission, alors la classification de l'architecture également. Par conséquent, il existe des architectures mixtes pouvant mettre en place des processus de décision différents selon les situations rencontrées.

Des approches où les robots peuvent évoluer de manière distribuée ou centralisée selon le contexte sont présentées dans [Ala+20] et [ABB17]. Ainsi, lorsqu'un critère comme la disponibilité des communications avec la station centrale est atteint, le système abandonne son autonomie au profit du recalcul d'un plan global. Cette bascule entre centralisé et décentralisé est également présente dans [DS03]. L'approche décrite permet aux robots d'évoluer de manière distribuée autour de tâches simples, mais également de se concerter de manière opportuniste pour centraliser la décision afin d'accomplir des tâches complexes. Cependant, il est difficile de réunir en ligne les conditions permettant au système de se centraliser afin d'améliorer ses performances.

Dans [Bec+14 ; Bec+18], les auteurs présentent *HiPOP*, un algorithme de planification hybride mêlant d'une part des contraintes opérationnelles hiérarchiques avec d'autre part des contraintes sur les préconditions, les effets, et les contraintes temporelles, des actions. Cet algorithme permet d'obtenir un plan initial hors-ligne pour un système multirobot. Ce plan est ensuite encodé sous une forme de Simple Temporal Network (STN) orienté multiagent et est exécuté par les robots. En cas d'aléas, tel qu'un retard, le plan est réparé de manière décentralisée. Par exemple, s'il n'est pas possible de respecter une contrainte temporelle, un agent est en charge de d'agréger les informations des coéquipiers à portée et de recalculer un plan de mission pour tous les coéquipiers à portée en relâchant les tâches à réaliser tant que nécessaire. Les modifications induites sont ensuite propagées de manière distribuée à tous les robots.

Dans [CLP15], les auteurs comparent d'autres stratégies pour procéder à la propagation de ces modifications. Une stratégie centralisée où les agents subissant un retard le communiquent à un agent central en charge de la réparation du plan global de mission et de la propagation à tous les agents. Et des stratégies distribuées, l'une où chaque agent a une vue complète du plan des coéquipiers, les agents s'échangent des informations sur les retards et chacun répare localement le plan global à partir de ces informations. D'autres où les agents n'ont qu'une connaissance partielle du plan global et doivent s'échanger des informations pour propager les réparations.

3.3.5 *Architectures multirobots sous contraintes de communication*

Dans un système multirobot, un élément fondamental impactant des notions comme l'architecture du système, la fiabilité des informations, le mécanisme de prises de décision, la supervision, ou encore la réparation réside dans la capacité à communiquer [ABB17]. Ainsi, une grande part de la littérature concerne l'étude des MRS sous contraintes de communications.

Par exemple dans [ABB17], les auteurs regroupent les approches suivant les contraintes imposées au graphe de connectivité. Ils définissent ainsi trois stratégies. Une stratégie sans communication, où chaque robot est autonome et ne peut que supposer ce que font les coéquipiers. Une seconde stratégie où la communication est basée sur des événements est proposée. Les robots sont dans l'obligation d'échanger quand un événement a lieu, comme l'obtention de nouvelles informations ou l'atteinte d'une deadline. Enfin, les auteurs proposent une dernière stratégie où le graphe de connectivité doit être maintenu tout le long de la mission. Dans ce cas, les robots sont contraints de ne prendre que des décisions qui ne rompront pas le graphe.

Ce maintien du graphe de connectivité peut être obtenu de manière plus ou moins stricte. Par exemple dans [QGL22b ; QGL22a], les auteurs proposent d'utiliser un mécanisme d'allocation de tâches par enchères incluant une prédiction de la connectivité future des enchérisseurs. Ainsi, il est possible d'optimiser l'allocation selon ce critère de manière à améliorer la connectivité globale du système.

Un des environnements les plus contraignants en termes de communication est l'environnement sous-marin. Dans [Zol+19], un état de l'art sur les moyens de communication sous-marins à disposition des systèmes multirobots est dépeint. Cet état de l'art met en avant l'utilisation générale d'ondes acoustiques comme vecteur de communication. Ce choix est principalement dû à la quasi-impossibilité de communiquer à des portées raisonnables avec des solutions électromagnétiques ou optiques. Les ondes acoustiques sont cependant impactées par les paramètres environnementaux. Par conséquent, les problèmes tels que les pertes de messages et les zones d'ombres sont nombreux (Sec. 2.6.2).

Pour pallier les difficultés de communications du monde sous-marin, certaines approchent reposent sur des processus de décision distribués et implicites exploitant une organisation générale du système. C'est par exemple la solution explorée par [Car13]. Dans cette approche les robots sont hétérogènes et organisés de manière à permettre une grande flexibilité selon un modèle "Agent-Groupe-Rôle". L'agent appartient à un groupe et joue un rôle au sein du système. Ce système est modulaire et reparamétrable en fonction des unités et de la mission considérée. À cette approche organisationnelle est combiné un modèle *satisfaction-altruisme* pour faire coopérer les robots entre eux. Ce modèle exploite des signaux d'insatisfaction peu coûteux en communication [Sim01]. Par exemple, si l'activité d'un coéquipier est néfaste pour un agent, l'agent concerné peut lui signaler en diffusant un signal quantifiant son insatisfaction. Le coéquipier peut alors prendre en considération l'information du signal et adapter son comportement si nécessaire.

Dans [SL10], des drones sous-marins utilisent également un réseau de tâches hiérarchique pour représenter le plan de mission. Ce réseau de tâches est mis dynamiquement à jour pour intégrer de nouveaux objectifs. Des communications broadcast sont utilisées périodiquement afin de faire circuler les informations sur l'avancement de la mission. Chacun des messages envoyés comporte l'identifiant de son émetteur, son état actuel, ainsi que toutes les informations connues par l'émetteur sur la mission, p. ex. les nouveaux objectifs à réaliser. Par la réception de ce message, chaque agent peut reconstruire l'intégralité de la mission. Ainsi, un robot arrivant dans le système peut se greffer à la réalisation de la mission. Chaque robot sélectionne localement l'objectif qu'il veut réaliser, le système est donc fortement distribué. Cette approche permet une grande autonomie et résilience du système, mais au prix de communications

importantes. Il est à noter que la difficulté d'obtention d'un consensus autour de tâches complexes est ici atténuée en partie par l'utilisation d'un réseau de tâches hiérarchique.

Dans [Bel11 ; BIL12], une architecture distribuée au sein de drones sous-marins devant traquer des cibles. Des rendez-vous sont fixés entre robots afin de partager les informations découvertes. Cette architecture permet de réactivement intégrer les informations reçues au plan de mission de l'agent. Cependant, pour améliorer leur plan les robots doivent communiquer le plus possible et il n'est pas possible de faire coopérer les robots autour de tâches complexes.

Dans [Fer+19 ; Fer+20], une architecture distribuée de fusion de donnée est utilisée. Cette architecture permet aux robots d'améliorer leurs connaissances sur la situation afin de gagner en autonomie. Le système développé a été testé en expérimentations réelles. Outre l'importance de coopérer pour gagner en efficacité, les résultats démontrent que des approches reposant sur des communications importantes sont possibles en milieu sous-marin tant qu'elles sont développées dans le respect de la bande passante et ne sont pas trop sensibles à la perte de messages.

3.4 ALLOCATION DE TÂCHES PAR ENCHÈRES

Les méthodes d'allocation par enchères ont été introduites dans Sec. 3.3.2 comme un cadre emblématique d'architecture décentralisée. Les allocations par enchères offrent une grande diversité d'approches pouvant s'adapter à des situations très diverses. Ces méthodes nous semblent d'un intérêt particulier pour résoudre un problème de MRTA, dans cette section nous présentons donc un état de l'art de ces méthodes.

Les méthodes d'allocation par enchères font partie des approches dites "Market-Based", qui doivent leur nom aux principes économiques dont elles s'inspirent. Leurs bases sont définies en grande partie dans [Smi80] qui présente le Contract Net Protocol. Ce protocole permet l'élaboration et l'exécution d'un contrat entre un agent manager et un agent contractant. L'obtention de ce contrat se fait grâce à des enchères. Dans le cas d'un système multirobot, le contrat peut être vu comme un objectif à accomplir.

Ce type de méthodes de résolution du problème de MRTA bénéficie d'un fort regain d'intérêt depuis ces vingt dernières années. Ces approches permettent d'attribuer selon un protocole simple une tâche ou un plan à un robot ou une sous-équipe [Kal+05].

Dans [Dia+06], les auteurs caractérisent une méthode market-based par :

— Une mission devant être accomplie par le système multirobot. Cette mission est décomposable en sous éléments accomplissables par un ou plusieurs robots. L'équipe a un nombre de ressources limité ;

— Une fonction objectif globale servant à quantifier les préférences à adopter sur toutes les solutions ;

— Une fonction individuelle d'utilité pour chaque robot ;

— Une correspondance entre la fonction objectif globale et les fonctions d'utilités individuelles ; Cette correspondance permet de définir comment les contributions de chacun font progresser l'objectif principal ;

— Un mécanisme d'enchère pour attribuer les ressources et tâches.

Dans le cas d'une allocation par enchères ("auction-based allocation"), les robots concernés sont en concurrence les uns avec les autres afin d'acheter des parties de la mission [Dia+06]. Lorsqu'une tâche doit être accomplie par le système multirobot, elle est mise en vente par un *commissaire-priseur* ("auctioneer"). Débute alors une phase d'enchère durant laquelle les robots *enchérisseurs* ("bidders") misent pour remporter la tâche. La valeur de la mise correspond le plus généralement à l'utilité de l'agent sur la tâche (Sec. 3.1.1). Le commissaire-priseur compare ensuite les mises reçues afin de pouvoir trouver la meilleure allocation et informe les gagnants des mises sélectionnées.

Ainsi, un tour d'enchère comporte quatre étapes essentielles :

1. **Annonce** : un commissaire-priseur ouvre le tour en diffusant l'information sur le ou les items en vente ;
2. **Estimation des mises** : les enchérisseurs analysent le ou les items en vente et estiment les mises qu'ils peuvent faire dessus ;
3. **Détermination des gagnants** : une fois que le commissaire-priseur a reçu toutes les mises, ou qu'une condition d'arrêt est atteinte, il compare les mises reçues et détermine le ou les items devant être alloués. Ce problème est également appelé Winner Determination Problem (WDP) ;
4. **Récompenses** : le commissaire-priseur informe le ou les gagnants et clôt l'enchère.

Ce protocole permet de distribuer la complexité de la décision, par exemple ce sont les enchérisseurs qui sont en charge d'évaluer leur utilité sur une tâche. Cette décision est ensuite centralisée par un commissaire-priseur. Ce rôle peut généralement être endossé par n'importe quel robot.

Ainsi, les approches par enchères peuvent être mises en place de manière centralisée, via un unique commissaire-priseur, ou décentralisée, où tous les robots peuvent être commissaire-priseur [BHK13].

Il est également possible de mettre en place des approches *Market-Based* distribuées qui se font sans recourir explicitement à des enchères. Dans ce dernier cas, les robots utilisent des principes économiques pour organiser leurs décisions autour de schémas distribués. Par exemple dans [HBH09], les robots obtiennent dans un premier temps leurs tâches grâce à des enchères, puis ils exploitent des méthodes de consensus pour résoudre les conflits apparaissant en ligne. Ainsi, si deux robots ont reçu la même tâche, ils peuvent s'accorder grâce au prix pour lequel ils ont acheté la tâche. L'utilisation de méthodes de consensus couplées à l'économie de marché en place permet ici de réparer les décisions prises pour atteindre une solution globale de meilleure qualité.

Les approches par enchères ont été grandement utilisées pour faire coopérer des robots. Les avantages que l'on retrouve le plus souvent pour les schémas les plus simples sont le passage à l'échelle et le coût de calcul faible. En revanche, les approches Market-Based ne sont presque jamais utilisées pour des tâches à trop forte coopération, comme porter un objet à plusieurs. Bien que ces approches montrent empiriquement de bons résultats, les règles théoriques et mathématiques permettant de garantir leurs performances sont encore un sujet de recherche très actif [GM04b].

3.4.1 *Principaux paramètres d'une approche par enchère*

De nombreux paramètres et stratégies sont à définir dans une approche par enchère, menant par conséquent à une grande diversité d'algorithmes. Parmi ces paramètres, on retrouve :

— La nature de l'item mis en vente. Par exemple, une tâche unitaire, un ensemble de tâches, ou encore une tâche abstraite décomposable.

— Le nombre d'items mis en vente ; On peut ainsi mettre en vente un item ou plusieurs à la fois. Dans le cas où plusieurs items sont en vente, il est nécessaire de définir des règles spécifiques sur le nombre d'items que peut remporter un enchérisseur ;

— La nature centralisée ou décentralisée des enchères, c.-à-d. qui peut tenir le rôle de commissaire-priseur ;

— L'utilisation d'une deadline. Par exemple, pour accorder un temps maximum d'enchérissement [Kal+05]. Cette deadline impacte le choix des algorithmes d'estimation des mises pour les enchérisseurs. C'est

un point très important quand les items à estimer sont complexes. En ligne les enchérisseurs doivent être aptes à soumettre une enchère tout en continuant l'exécution de leurs tâches en cours [Dia+06] ;

— La qualité des communications considérées. Selon la portée ou la perte des messages, tous les robots ne pourront pas participer aux enchères ;

— La disparité des connaissances entre robots. Avec les contraintes de communication, une diversité des connaissances sur la mission et l'environnement peut apparaitre en ligne. Les robots estimant leur mise selon leurs connaissances, il peut être nécessaire d'écarter les robots moins bien informés.

3.4.2 *Les principaux schémas d'enchères*

Les premiers algorithmes développés pour utiliser des approches par enchères sur des systèmes multirobots utilisent comme item en vente une unique tâche [GM03b]. La plupart des problèmes du monde réel sont complexes et constitués d'objectifs variés, cette approche doit donc est étendue. En conservant cette logique de tâche atomique, une première solution pour résoudre des problèmes complexes est de mettre en vente toutes les tâches élémentaires obtenues par décomposition du problème. La vente de toutes les tâches, si elle est atteignable, résout le problème.

Il faut cependant définir comment ces tâches sont mises en vente et obtenues. On retrouve généralement quatre stratégies différentes [KKT10] :

— *Single-Item (SI)* : Une unique tâche est mise en vente pour une phase d'enchère ;

— *Sequential Single-Item (SSI)* : Dans une même enchère, les tâches sont mises en vente et allouées une par une à chaque tour d'enchère ("auction round") ;

— *Parallel Single-Item auctions (PSI)* : Toutes les tâches sont en vente au même instant. Les enchérisseurs proposent une mise pour chacune d'entre elles. Dépendamment des mises reçues, l'enchère peut alors être résolue en un seul tour ;

— *Combinatorial* : Toutes les tâches sont en vente au même instant. Les enchérisseurs peuvent miser sur n'importe quel sous-ensemble de ces tâches.

Chacune de ces stratégies comporte ses forces et faiblesses. Par exemple, avec un schéma SSI, le résultat de l'allocation sera très dépendant des premiers tours d'enchère. Ainsi, la synergie pouvant lier les tâches entre elles, via les dépendances complexes, sera difficilement exprimable. Au fur et à mesure des tours, la qualité de solution divergera de l'optimal, dans certains cas il pourrait même être impossible de trouver un plan à cause des contraintes liant les tâches.

Les enchères combinatoires permettent d'acheter plusieurs tâches afin de laisser la possibilité aux robots de tenir compte des dépendances complexes. Les enchérisseurs peuvent miser, selon les algorithmes utilisés, sur un sous-ensemble ou plusieurs sous-ensembles de ces items. Ainsi, il est possible de faire des combinaisons de tâches au sein d'une même mise. Ces approches combinatoires sont cependant généralement coûteuses en communication et en complexité de résolution. En effet, le nombre de combinaisons à estimer augmentant de façon exponentielle avec le nombre de tâches considéré il devient extrêmement difficile d'estimer toutes les mises possibles et encore plus de déterminer les gagnants. Cette complexité peut être artificiellement jugulée pour des résultats variables en obligeant les robots à ne soumettre que leurs meilleures mises [VV03 ; Dia+06].

3.4.3 *Des approches nombreuses et variées*

Par les nombreuses variations du problème de MRTA et des approches par enchères, il difficile de comparer en détail l'efficacité d'algorithmes Market-Based entre eux. Ces algorithmes sont d'une part très spécifiques aux problèmes qu'ils résolvent, et d'autre part il n'existe pas de *benchmarks* de référence. Par exemple, les problèmes de planification [1] sont comparés sur des benchmarks adoptés comme référence dans une grande majorité de travaux. Néanmoins, quelques travaux ont été menés pour comparer des solutions d'allocation par enchères sur des problèmes spécifiques. Ces résultats permettent d'avoir un premier aperçu de l'efficacité d'une approche pour un problème donné.

Dans [OKS19], six algorithmes d'enchères sont comparés (*Sequential*, *parallel*, *combinatorial*, *G-prim*, et deux variations de *multi-rounds of a Repeated Parallel Auction*). Des problèmes de type *knapsack* et TSP sous contraintes de communication sont utilisés comme cas d'usage. Pour simuler les communications, un modèle de Bernoulli et un modèle de Gilbert-Elliot sont utilisés. Si le premier modèle permet de simuler la perte de message, le second permet également d'inclure la saturation de la bande passante. Dans ces travaux, des résultats favorables en faveur des algorithmes multirounds sont soulignés. Ces algorithmes permettent de réduire l'impact des pertes de communication.

Ces résultats font suite à [OKS17], où les mêmes auteurs avaient déjà comparé trois algorithmes d'enchères différents : SSI, PSI, et *G-Prim Auctions*, toujours sous contraintes de communication. L'étude concluait déjà en faveur des approches *G-Prim* et SSI, notamment en raison de la plus grande robustesse des enchères multirounds au risque de perte de messages.

Il est important de noter que dans leur implémentation de l'algorithme PSI, l'agent envoyait l'ensemble de ses mises dans un même message. Si ce message est perdu, l'agent ne peut obtenir aucune tâche. Ainsi, leur algorithme PSI réduit fortement le nombre de communications, mais augmente grandement la sensibilité du système aux erreurs de communication.

Une solution pour améliorer les performances des algorithmes PSI serait d'envoyer un message pour chaque item en vente. Cette solution augmente le nombre de messages, mais ces messages sont moins volumineux.

Dans [OKS19], la diffusion de la liste des précédents gagnants est également utilisée pour permettre aux enchérisseurs de reconstruire un historique des allocations. Ainsi, dans le cas où un robot ignore qu'il a remporté une tâche à cause d'un message perdu, il peut l'apprendre par la diffusion de cet historique. Cela permet de réduire les chances qu'un gagnant ne soit pas informé des items qu'il a remportés. Par ailleurs, cette méthode permet de réduire le nombre de messages échangés (en augmentant la taille de certains) et d'améliorer la Situation Awareness des robots. En effet, avec cette liste de gagnants diffusée à chaque tour d'enchère, les robots peuvent tenir à jour les tâches allouées à leurs coéquipiers pour améliorer leurs connaissances globales sur la situation. Ces informations permettent à un robot d'améliorer son plan en fonction des objectifs connus de ses coéquipiers. Cependant, c'est une solution complexe à mettre en place qui est peu explorée dans la littérature.

Une autre solution viable pour s'assurer de la bonne réception des tâches allouées pourrait être d'obliger le gagnant à envoyer une confirmation de réception à chaque message de récompense.

Des méthodes Market-Based distribuées sont explorées dans [Nay+20]. Cinq algorithmes distribués d'allocation de tâches avec communications imparfaites sont comparés. Parmi ces algorithmes plusieurs approches Market-Based avec consensus sont explorées. Ces approches s'appuient sur des enchères distribuées où chaque robot sélectionne localement une tâche à effectuer et diffuse sa mise pour la tâche en

1. Depuis 1998, La Compétition Internationale de Planification (IPC) a introduit ces benchmarks comme référence https://ipc.hierarchical-task.net/

question. La sélection de la tâche se fait également en intégrant les mises reçues. En plus des modèles de Bernoulli et de Gilbert-Elliot, les auteurs utilisent également le "Rayleigh Fading Model" pour prendre en compte les problèmes d'atténuations. Les résultats mettent en avant les compromis à établir en fonction de la qualité des communications et de leur modélisation. Le scénario utilisé est un problème de visite de points d'intérêt avec arrivée en ligne de nouveaux points. Les approches explorées démontrent de bons résultats, y compris sous contraintes de communications. Cependant, les tâches considérées sont simples et ne nécessitent pas de collaboration entre les robots.

Les études présentées précédemment sont réalisées en recourant à différents modèles de communication : modèles de Bernoulli, de Gilbert-Elliot et d'atténuation de Rayleigh. Ces modèles permettent de prendre en considération des éléments comme la perte de message, la saturation de la bande passante et l'atténuation. Ainsi, il est par exemple possible de comparer les architectures sur le volume de messages qu'elles nécessitent pour fonctionner. Dans les précédents travaux, les résultats font généralement consensus pour les modèles de Bernoulli et de Rayleigh mais, quand la bande passante est prise en compte, les approches distribuées chutent en performances. Cela peut s'expliquer par l'important échange d'informations qu'elles nécessitent.

Dans [SBD18a ; SBD18b], les auteurs proposent de raisonner sur des *options* probabilistes précalculées afin d'alléger le coût calculatoire. Les robots se concertent donc pour acheter ou non ces options. Pour ce faire, un formalisme Linear Temporal Logic (LTL) et des Markov Decision Process (MDP) sont mis en oeuvre. Ces travaux se destinent cependant à l'attribution de tâches simples et répétitives dans un contexte industriel. Par conséquent, les auteurs font l'hypothèse d'une communication parfaite. Dès qu'un robot finit une tâche, toutes les tâches non finies de tous les robots sont réunies et remises en vente au sein d'un nouveau tour d'enchères. La planification repose également sur une mécanique d'apprentissage des tâches répétitives.

Les approches par enchères sont également utilisées pour former dynamiquement des équipes. Ces équipes peuvent aussi bien servir à planifier la mission qu'à réparer le plan, par exemple en sollicitant l'aide d'un autre robot pour une tâche plus ardue que prévu. Par exemple dans [BA99], un robot peut recourir à une enchère SI pour créer une requête d'aide.

Dans [Jon+06], la formation d'équipes à l'aide d'enchères est également abordée. En parallèle de la mise en vente d'une tâche, les robots estiment des plans prédéfinis, appelés *Plays*, permettant d'accomplir la tâche. Chaque *Play* consiste en une composition de rôles pour exécuter la tâche. Pour estimer la valeur d'un *Play*, les robots mettent aux enchères les rôles. Une fois tous les rôles vendus ils peuvent proposer une mise pour la tâche auprès du commissaire-priseur initial. Cette formation d'équipe autour d'un objectif permet de planifier la mission. La plupart des approches présentées jusqu'ici allouaient des *single-robot tasks* où chaque tâche ne nécessite qu'un robot pour être accomplie, cependant avec l'approche présentée ici les robots s'organisent pour accomplir des *multi-robot task*.

Outre l'absence de benchmarks, les approches par enchères sont également difficiles à comparer par le manque de base commune pour définir le problème et le schéma d'enchère. Des travaux récents proposés dans [MP20a] définissent un langage pour uniformiser la définition des mécanismes d'enchères. Le langage développé permet de représenter un mécanisme d'enchère à partir d'un ensemble de règles, mais est malheureusement encore limité aux enchères SI.

3.4.4 *Écueils des approches par enchères*

Bien que les approches par enchères soient appréciées sur des problèmes peu complexes pour la simplicité de leur schéma, elles comportent plusieurs inconvénients. Ces défauts portent principalement

sur quatre éléments : la complexité des problèmes à résoudre, les besoins en communication, la divergence des connaissances entre participants, et la prise en compte de dépendances entre items.

Complexité de l'estimation des mises et du WDP

Un premier inconvénient réside dans la complexité de la résolution du problème de MRTA. Les approches par enchères sont par essence faites pour être utilisées en ligne, il faut donc que la résolution du problème de MRTA se fasse dans des temps acceptables pour la mission. La complexité de la résolution de ce problème apparait durant les phases d'estimation des mises et de détermination des gagnants.

Dans une approche par enchère, indépendamment de type de schéma retenu, la qualité de la solution est corrélée au nombre de mises possibles [GM04b]. En effet, plus le commissaire-priseur a de mises à comparer, plus grand est le champ des possibles. En étendant ce constat, on comprend que :

1. il est important pour un enchérisseur de pouvoir estimer une mise pour tous les items en vente ;
2. il est important pour le commissaire-priseur de recevoir une grande diversité de mises.

Cependant, une augmentation du nombre d'enchérisseurs et de mises se répercute sur la complexité des problèmes d'estimation des mises et de détermination des gagnants. Si le problème d'estimation des mises reste généralement surmontable, le vrai facteur limitant réside dans le problème de détermination des gagnants. Les coûts des calculs induits par les principales stratégies d'enchères sont représentés dans le Tab. 3.1. Comme attendu, la complexité du schéma SI est la plus faible et celle des enchères combinatoires la plus importante.

Auction type	Bid valuation	Winner determination	Number of auctions
Single-item	v	$O(r)$	n
Multi-item (greedy)	$O(n \cdot v)$	$O(n \cdot r \cdot m)$	$\lceil n/m \rceil$
Multi-item (optimal)	$O(n \cdot v)$	$O(r \cdot n^2)$[GM04b]	$\lceil n/m \rceil$
Combinatorial	$O(2^n \cdot V)$	$O((b+n)^n)$[San02]	1

TABLE 3.1 – Comparaison des complexités calculatoires de différents types d'enchères [Kal+05]. n est le nombre d'items, r le nombre de *bidders*, b le nombre de mises, et $m < r$ le nombre d'attributions maximum par tour d'enchère. v et V représentent la quantité de temps requise pour évaluer un item ou un ensemble d'items.

Dans le cas des enchères combinatoires, et dans une moindre mesure celui des enchères PSI, le problème de détermination des gagnants peut devenir extrêmement complexe à résoudre, notamment par le nombre de combinaison d'allocations possibles. Si ces combinaisons d'allocations permettent d'approcher une solution optimale, elles augmentent aussi la complexité du problème. Dans [Abr+07] les auteurs soulignent la nature *NP-hard* du WDP. La résolution du WDP en des temps raisonnables et sans sacrifier la qualité de la solution est un des défis les plus importants des approches par enchères.

Les besoins en communications

Le deuxième défaut inhérent aux enchères réside dans les communications à considérer. L'efficacité des enchères dépend en grande partie de la capacité des robots à entrer en communication avec un maximum d'enchérisseurs pour pouvoir comparer un plus grand nombre de mises. Par conséquent, elles sont sensibles à la portée et la qualité des communications.

Dans [MML08], la sensibilité au rayon de communication des algorithmes d'enchères est explorée. Plus le rayon de communication est réduit et les communications sont instables, moins grand sera le nombre de robots ayant pu participer aux enchères et moins l'allocation résultante sera pertinente.

Bien que dans beaucoup de travaux sur les systèmes multirobots des communications parfaites sont considérées, c'est loin d'être le cas pour les problèmes complexes du monde réel. Le rayon de communication peut être moindre que le théâtre d'évolution des robots, qu'il soit connu à l'avance ou variable, l'algorithme de décision doit être capable de le supporter.

En plus de la qualité et la portée des communications, les enchères sont également impactées par le volume de messages devant être envoyés. Dans les environnements où les solutions de communications peuvent être aisément saturées, comme c'est le cas en milieu sous-marin, il est important de limiter les messages échangés afin de rester le plus efficient possible.

Dans des approches par enchères, la majeure partie des communications est dédiée aux mises. Les coûts en communication des principales stratégies d'enchères sont représentés dans le Tab. 3.2. Ce sont les enchères combinatoires qui, comme pour la complexité, présentent le plus grand coût d'utilisation. Il est à noter que dans ce tableau les communications sont considérées *point-to-point* et la complexité porte sur leur nombre stricto sensu, l'usage de message broadcasté, c.-à-d. *one-to-many*, peut réduire ce nombre de communications.

Auction type	Auction call	Bid submission	Award	Award (+ losers)
Single-item	$O(r)$	$O(r)$	1	$O(r)$
Multi-item	$O(r \cdot n)$	$O(r \cdot n)$	$O(m)$	$O(r)$
Combinatorial	$O(r \cdot n)$	$O(r \cdot 2^n)$	$O(n)$	$O(r + n)$

TABLE 3.2 – Comparaison des complexités de communications de différents types d'enchères [Kal+05]. Les notations utilisées sont les mêmes que pour le Tab. 3.1.

Si les approches par enchères permettent une relative tolérance aux pertes de messages, elles doivent être développées en accord avec le milieu dans lequel elles s'intègrent.

Une première méthode pour réduire les communications réside dans la conception du schéma d'enchère. Par exemple, dans [Kal+05], une des approches proposées consiste à n'envoyer les items qu'aux enchérisseurs susceptibles de faire les meilleures mises.

Un des framework d'allocation par enchères ayant reçu le plus d'attention est *Hoplites*, présenté dans [KFS05]. Le framework repose sur deux principes de coordination :

— Une coordination passive qui produit rapidement des solutions locales. L'agent se repose sur sa perception de l'environnement et de ses coéquipiers pour prendre des décisions à l'aide de modèles locaux ;

— Une coordination active où l'agent communique avec ses coéquipiers pour améliorer son plan. Cette coordination est mise en place par des méthodes de négociations. Ce type de coordination est utilisée par l'agent pour résoudre des problèmes complexes.

L'architecture résultante alterne ces deux modes de coordination suivant les situations rencontrées.

Connaissances communes

Un inconvénient notable des approches par enchères réside dans le degré de connaissance globale des robots.

Par exemple, une tâche est mise en vente auprès de deux enchérisseurs. Considérons qu'un des enchérisseurs a connaissance d'une information sur l'environnement rendant plus difficile que supposée la réalisation de la tâche. La mise de cet enchérisseur reflétera cette information, par conséquent la tâche risque d'être remportée par le second enchérisseur, moins bien informé et plus optimiste.

Quelques travaux ont exploré l'utilisation de fonctions d'utilités intégrant un degré de connaissances sur l'environnement, mais beaucoup reste à accomplir dans ce domaine [Dia+06].

Ce problème de divergence des connaissances est le plus souvent éludé en faisant l'hypothèse que tous les robots partagent les mêmes connaissances sur l'environnement. Dans le cas d'environnements ne permettant pas un niveau de communication élevé, il est difficile de maintenir cette hypothèse.

Ainsi le problème de MRTA doit idéalement être adressé en prenant en compte la disparité des connaissances des robots, au risque de s'exposer à des replanifications constantes et des solutions sous-optimales. Afin d'éviter au maximum ces différenciations d'appréciations dans le coût d'une tâche il reste important de faire circuler les informations pertinentes dans le système lorsque cela est possible, que cela soit pendant ou en dehors de la phase d'enchère.

Le partage de connaissance ne vise cependant qu'à anticiper, quand cela est possible, des aléas pouvant être évités, car connus du système. Même avec un partage de connaissance efficace, les mises resteront une prédiction approximative de la réalité du terrain [Kal+05]. Il est donc vital d'implémenter des mécanismes permettant de réparer efficacement la mission afin d'être robustes aux aléas.

Enchérir sur des tâches dépendantes

Les bases d'un problème de MRTA sont de devoir considérer un ensemble de robots et un ensemble de tâches. Toute la difficulté du problème est de trouver des allocations de bonne qualité et dans un temps raisonnable.

Dans le cas des approches par enchère, une méthode naïve consiste à appliquer le schéma SSI pour allouer toutes les tâches. Dans ce schéma les tâches sont mises en vente une par une. Cette méthode à l'avantage d'être peu coûteuse en calculs, par exemple, la détermination du gagnant a chaque round peut se faire grâce à un algorithme glouton. Pour sa simplicité d'accès et ses bonnes performances sur des tâches simples, ce schéma a été grandement exploré [KFS05 ; MNG16 ; Rek+08].

Cependant, ce schéma perd en performance sur des problèmes complexes. Quand les tâches présentent des dépendances complexes, c.-à-d. quand les tâches sont contraintes entre elles et/ou comportent des utilitées corrélées, les premières allocations ont un impact fort sur le plan final qui sera obtenu. Ces utilités provoquent à l'échelle du robot des dépendances appelées *In-schedule Dependencies* (Sec. 3.2.1). Dans ces conditions il est très difficile d'obtenir des solutions de bonne qualité. De manière générale, chaque tour d'enchère est susceptible d'éloigner plus la solution de l'optimal.

Ce problème existe aussi dans le cas des enchères PSI. Par exemple, une variation possible des enchères PSI consiste à étaler les enchères sur plusieurs tours. Ainsi, les items sont mis en vente, mais les participants ne peuvent en remporter qu'un seul au maximum. Ils peuvent soit miser sur celui qu'ils désirent le plus ou tous [OKS17]. Si tous les items ne trouvent pas preneurs en un seul tour, un nouveau tour a lieu. En raison de la séquentialité des enchères, cette méthode peut mener à des solutions sous-optimales [GM03b].

Lors de la résolution du problème de MRTA, cette prise en compte des dépendances complexes est déjà un premier défi difficile à surmonter. Ce problème devient autrement plus complexe à résoudre lorsque sont considérées des contraintes entre tâches, pouvant résulter en *Cross-schedule Dependencies*, c.-à-d. lorsque le plan d'un robot dépend d'au moins un de ses coéquipiers. Les contraintes qui pouvant causer

ces dépendances peuvent être de différentes natures (Sec. 2.5.4). Par exemple, dans le cas d'une contrainte de précédence, il peut être requis d'effectuer une tâche avant une autre. Les contraintes entre tâches sont nécessaires pour pouvoir exprimer des problèmes complexes pouvant être partiellement ordonnés.

Dans le cadre d'une allocation par enchère, les robots achetant ces tâches contraintes doivent en tenir compte. Dans [Gin17], les challenges à surmonter pour prendre en compte ces contraintes avec une approche par enchères sont explicités. Ces contraintes et les allocations retenues doivent être répercutées à chaque phase de décision pour pouvoir être intégrées en ligne.

Le principal problème résultant de la prise en compte de ces *Cross-schedule Dependencies* reste la résolution, en des temps et qualités raisonnables, du WDP. Pour résoudre ce type de WDP, une première approche consiste à se reposer sur des enchères combinatoires, comme dans [BG06]. Cependant, ces enchères combinatoires sont trop coûteuses pour être utilisées en ligne. En effet, si l'allocation d'éléments dépendants est un des challenges que les enchères combinatoires peuvent théoriquement résoudre, cela se fait au prix d'une grande complexité, notamment pour le WDP.

Pour réduire cette complexité, [MNG16] utilisent des enchères itératives et PSI sur des tâches non contraintes entre elles. Cet ensemble de tâches est mis en vente au sein d'un même tour d'enchère. Ainsi, seuls les enchérisseurs doivent tenir compte des contraintes liant leurs tâches déjà acquises à celle sur laquelle ils souhaitent miser. La résolution des gagnants n'a donc pas à tenir compte de contraintes temporelles entre les tâches à allouer. Une fois ces tâches allouées, le commissaire-priseur détermine un nouvel ensemble de tâches non contraintes entre elles et le met en vente. Ce schéma permet de limiter la complexité du WDP mais pas de prendre en compte au sein d'un même tour des tâches contraintes entre elles. Par conséquent et à l'instar d'un schéma SSI, la solution finale sera extrêmement dépendante des premiers tours d'allocation.

Dans [NG15], les auteurs contraignent implicitement les tâches entre elles à l'aide de contraintes internes aux tâches. Ces contraintes sont de type fenêtres temporelles ("time windows"). Avec cette approche il est possible de synchroniser deux tâches, mais pas de mettre en place des contraintes de précédences ou causales. De plus, cette synchronicité est stricte sur une fenêtre décidée à l'avance, les robots ne peuvent se mettre d'accord sur l'intervalle de temps dans lequel exécuter les tâches.

Malgré les importants travaux menés dans le domaine des allocations par enchère, la vente d'items dépendants, la maîtrise des coûts calculatoires, et la réduction des communications sont encore autant de défis à relever.

3.4.5 *Enchérir sur des tâches complexes*

Un des principaux axes d'amélioration pour réduire la complexité des enchères tout en exprimant une certaine dépendance entre les tâches est de mettre en vente des ensembles prédéfinis de tâches, appelés bundles. L'utilisation de bundles prédéfinis est un compromis entre la trop grande complexité des enchères combinatoire, où les robots peuvent créer leurs propres bundles, et les faibles performances des enchères séquentielles.

Pour exprimer ces bundles plusieurs méthodes sont possibles [VV03]. Parmi elles, une approche particulièrement intéressante consiste à s'appuyer sur un réseau de tâches hiérarchiques [CLI12 ; KEK11]. Ces réseaux de tâches permettent de décomposer des tâches abstraites en d'autres tâches abstraites et/ou en tâches primitives.

Le problème de MRTA peut alors être vu comme un problème d'allocation de tâches primitives sur lesquelles vient s'ajouter une structure hiérarchique. Cette structure hiérarchique est une représentation abstraite de connaissances et est uniquement utilisée pour guider la décision. La définition d'un tel réseau

de tâches peut se faire soit par l'opérateur à l'aide de connaissances spécifiques à la mission, soit par les robots eux-mêmes. Par exemple dans [Kal+05], des bouts de plans prédéfinis sont vendus pour essayer de résoudre des problèmes complexes. Le robot ayant acheté un plan, c.-à-d. une décomposition, est libre de proposer sa propre décomposition et de la mettre en vente. Raisonner sur des plans permet d'alterner planification et allocation et ainsi de mettre en place une résolution hybride du problème de MRTA.

Ainsi, quelques recherches portant sur les approches par enchères ont exploré comment allouer, planifier, et réparer avec des tâches complexes décomposables.

Dans [ZS05 ; ZS06], un algorithme décentralisé d'allocation par enchères et basé sur un HTN est utilisé. Ce HTN est encodé sous la forme d'un arbre de noeuds *AND* et *OR*[2].

Dans un problème de MRTA représenté sous la forme d'un HTN, les tâches complexes sont décomposées en d'autres sous-tâches. Un exemple d'arbre utilisé pour une mission de couverture de zone est représenté sur la Fig. 3.5.

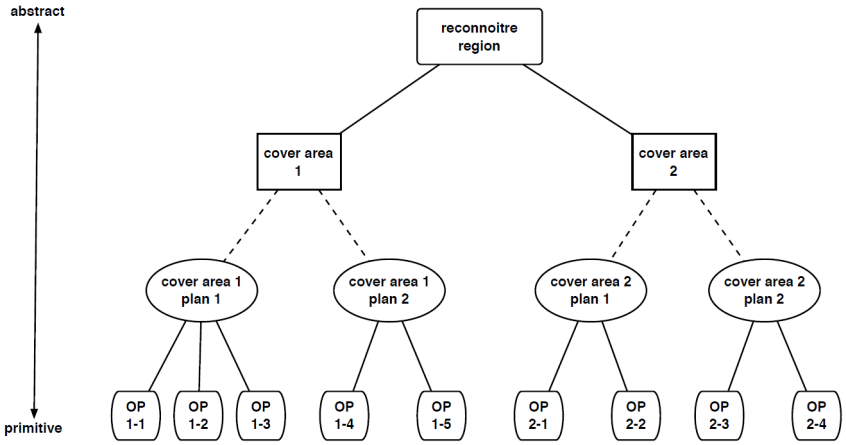


FIGURE 3.5 – Exemple d'arbre de tâches [ZS05]. Les lignes solides représentent les AND (toutes les tâches doivent être faites) tandis que les lignes pointillées représentent les OR (seule une tâche doit être faite). Les feuilles de l'arbre sont des tâches primitives. Les autres noeuds sont des tâches abstraites.

Dans [ZS05], les robots peuvent miser sur n'importe quel noeud de l'arbre. Si un enchérisseur remporte un noeud, alors il remporte tous les descendants de ce noeud. L'estimation de la mise d'un noeud abstrait est déduite de l'estimation de chacun des descendants. Les robots ayant obtenu un noeud ajoutent le sous-arbre correspondant à leur plan local afin d'exécuter les nouvelles tâches. Dans l'approche décrite les enchérisseurs sont également libres de revendre tout, ou une partie de l'arbre résultant de leurs allocations.

Bien que très intéressante, l'approche résout le WDP à l'aide d'un algorithme glouton couplé à un parcours en largeur de l'arbre. Seule la meilleure mise par noeud est retenue, privant la recherche des gagnants d'une grande partie des solutions possibles. Cet algorithme permet de réduire encore plus la complexité du WDP, mais en sacrifiant la qualité des allocations. Enfin, l'approche proposée ne permet pas de prendre en compte des contraintes entre les tâches.

Cette approche a également été étendue dans [KEK11 ; Liu+13] pour permettre aux enchérisseurs de décomposer eux-mêmes le noeud à estimer et miser avec leur décomposition s'ils estiment celle-ci meilleure. Cela permet de remplacer une décomposition peu efficace proposée par le commissaire-priseur et de raffiner le plan de mission. Dans ces travaux la résolution du WDP est également réalisée à l'aide d'un algorithme glouton réduisant la qualité de la solution.

2. Un noeud AND signifie qu'il faut effectuer tous les enfants de ce noeud. Un noeud OR signifie qu'un seul enfant est nécessaire.

Dans [KEK11], une méthode de remise en vente forcée est testée dans le but d'améliorer la résolution du problème de MRTA. Une enchère initiale est menée sur un arbre de tâche, puis chaque robot essaye à tour de rôle de revendre les items qu'il a remportés. Ces enchères successives se poursuivent jusqu'à ce qu'une condition d'équilibre soit atteinte, p. ex. une amélioration minime de la qualité du plan après plusieurs enchères. Cette méthode allonge grandement le temps de résolution du problème et augmente le nombre de communications nécessaires, ces deux facteurs remettent en question le choix d'utilisation d'un arbre de tâche et d'un algorithme glouton pour réduire la complexité des enchères. Une autre approche aurait été d'utiliser un algorithme permettant d'atteindre une meilleure solution au cours de l'enchère initiale, réduisant ainsi le besoin en communication.

Dans le même ordre d'idée, un système multirobot devant effectuer des tâches d'observation de zones, représentées hiérarchiquement par des *quadtrees*[3] est proposé dans [Cao+10 ; CLI12]. Ces quadtrees incluent les coûts associés à l'observation de chacune des portions de la zone. Chaque robot devant observer une zone peut ainsi la décomposer et mettre aux enchères des bouts de zones.

Les approches présentées ici sont une manière d'intriquer l'allocation et la décomposition de tâches (Sec. 3.1.2). Au fur et à mesure des tours d'enchères sur les arbres de tâches présentés, l'allocation peut prendre en compte les informations sur la décomposition et inversement.

3.4.6 *Enchérir en environnement sous-marin*

L'allocation de tâche par enchères pour les drones sous-marins est abordée dans [Fer+17a]. Le but de la mission est de couvrir des zones. L'allocation doit tenir compte d'un facteur d'urgence. Dans le but d'accomplir cette mission, les auteurs définissent l'algorithme *PADA* ("Periodic Auctions Distributed Algorithm"). Cet algorithme oblige les robots à réviser régulièrement leurs plans en misant sous forme d'enchères SI. L'originalité de l'approche provient du caractère périodique des enchères.

Quand un robot souhaite effectuer une tâche, il en devient commissaire-priseur, permettant ainsi à ses coéquipiers de miser sur cette tâche. Les mises se font grâce à des communications acoustiques broadcastées. Périodiquement, chaque robot, commissaire-priseur ou non, analyse les mises réceptionnées pour cette tâche et détermine le gagnant de l'enchère. Une fois un gagnant déterminé les robots agissent en conséquence. La périodicité de ces enchères permet d'assurer une réparation continue du plan. L'algorithme présenté requiert uniquement des communications locales et est robuste aux intermittences de communications. Cet algorithme périodique nécessite cependant un grand nombre de communications. Néanmoins, dans les simulations réalisées la bande passante n'a pas été saturée par le volume de données échangées.

Ces travaux ont été étendus dans [Fer+18] afin de prendre en compte des tâches infinies, c.-à-d. non délimitées dans le temps. Par exemple, la surveillance d'une zone de manière continue. En parallèle de ces tâches infinies, l'extension proposée vise également à intégrer de nouveaux objectifs durant la mission. Ces nouveaux objectifs, qui ne sont pas des tâches infinies et ne sont à réaliser qu'une fois, apparaissent suite à des aléas. Les nouveaux objectifs sont prioritaires par rapport aux tâches infinies.

Le caractère périodique de la solution utilisée permet de régulièrement inclure ces nouveaux objectifs dans le plan des robots. L'algorithme d'allocation de tâche proposé permet une réponse efficace à l'apparition de nouvelles tâches. Cependant, ces tâches sont simples et non contraintes entre elles, l'algorithme est donc limité à des problèmes ne nécessitant pas une forte coopération entre les drones.

3. Structure de données de type arbre dans laquelle chaque noeud a quatre fils.

Dans [Sar07 ; SBE08 ; SBS06], l'utilisation d'enchère pour un système multirobot hétérogène accomplissant une mission de chasse aux mines est explorée. Dans ces travaux les tâches considérées peuvent comporter des contraintes séquentielles. Les processus de décisions utilisés s'agencent en trois éléments principaux :

1. L'agent sélectionne localement une tâche à accomplir parmi celles connues ;
2. L'agent initie une enchère SI pour valider avec ses coéquipiers qu'il est le plus à même d'effectuer la tâche qu'il a sélectionnée ;
3. En cas d'erreur ou aléas, l'agent se repose sur des routines expertes, p. ex. en assumant les tâches d'un coéquipier présumé hors service car ne diffusant plus d'information.

Bien qu'intéressante, l'approche souffre du problème bien connu des enchères non combinatoires. Il est impossible pour les robots d'exprimer des dépendances entre tâches provoquant une divergence rapide avec la solution optimale [Dia+06].

Ici, la présence de contraintes de précédence entre les tâches cause des dépendances entre tâches. Cependant, en raison de la sélection locale d'une tâche et l'utilisation périodique d'enchère SI pour en choisir l'exécutant, il est difficile pour les robots de tenir compte de ces dépendances. Dans une telle configuration, la solution diverge rapidement avec l'optimal et les performances chutent. De plus, ce phénomène de divergence par des enchères SI est d'autant plus fort lorsque des contraintes existent entre les tâches.

3.5 SYNTHÈSE

Dans ce chapitre, nous avons défini le problème de MRTA et les principales architectures permettant de le résoudre. Ainsi, nous avons introduit l'architecture centralisée qui permet d'obtenir des solutions de très bonne qualité, mais au prix d'un coût calculatoire important et d'une grande sensibilité aux pertes de communications. À l'inverse, l'architecture distribuée repose sur des robots très autonomes, mais coopérants peu, rendant ainsi difficile l'accomplissement de tâches complexes. Enfin, l'architecture décentralisée, qui replace la coopération au centre du processus de décision tout en s'abstrayant d'élément central d'échec, nous a semblé un compromis intéressant.

Par conséquent, nous avons défini un état de l'art de la résolution du problème de MRTA grâce aux approches décentralisées d'allocations par enchères. En effet, ces méthodes nous semblent d'un intérêt particulier pour agencer le processus de décision du système. Ce type d'approche permet une certaine robustesse aux pertes de communication, une bonne réactivité face aux aléas, et une forte résilience du système.

Dans le chapitre suivant, nous exposons comment nous étendons les allocations par enchères avec des processus de planification hiérarchique afin de répondre aux défis de la chasse aux mines.

4

APPROCHE PROPOSÉE : ALLOCATION PAR ENCHÈRES ET PLANIFICATION HIÉRARCHIQUE

Aperçu

Dans ce chapitre, nous rappelons les défis de notre mission. Puis, sur la base de l'état de l'art précédemment établi, nous présentons et justifions les concepts utilisés au sein de notre approche en en décrivant une vue d'ensemble. Enfin, nous présentons le protocole général de notre approche.

Notre architecture, appelée HaucTioN, est conçue dans le but de résoudre le problème de MRTA d'une mission de chasse aux mines. Cette architecture permet l'allocation, la supervision et la réparation d'objectifs partiellement ordonnés pour un système multirobot sous contraintes de communication.

4.1 RAPPEL DES PRINCIPAUX DÉFIS

Dans le Chap. 2, nous avons détaillé le contexte de l'utilisation d'un système multirobot devant remplir une mission de chasse aux mines. L'analyse de ce contexte nous a permis de souligner les principaux défis entourant la coopération des robots au sein d'un tel système. Dans cette section, nous rappelons brièvement ces problématiques.

Allocation de tâches (Sec. 2.5.2) : La mission de chasse aux mines est initialement composée d'objectifs de couverture pouvant être accomplis selon différentes alternatives. Les robots doivent donc faire des choix sur la manière d'accomplir ces objectifs. Ces décisions doivent être prises collectivement afin de gagner en efficacité.

Il est impératif pour les robots d'être en mesure de déterminer ce qui doit être fait, comment le faire, et avec qui le faire. Ces questions sont au coeur du problème de MRTA. Pour résoudre ce problème, des modèles permettant d'estimer la valeur de la contribution d'un robot pour un objectif sont nécessaires. Les robots composant le système étant hétérogènes, ces modèles sont propres à chacun.

Réaction à des aléas (Sec. 2.5.3) : Une fois des décisions prises, les robots doivent superviser leur bonne exécution. Cette supervision est nécessaire afin de détecter des problèmes, comme des retards, obligeant les robots à réparer ces décisions. Par ailleurs, dans une mission de chasse aux mines, tous les objectifs ne sont pas connus à l'avance. La découverte d'un objet suspect mène par exemple à la création d'un nouvel objectif d'identification devant être intégré par le système. Par conséquent, le système doit être en mesure de réagir aux événements rencontrés durant la mission afin de réparer le plan. Ces réparations peuvent être locales à un agent ou globales, c.-à-d. concerner plusieurs agents. Dans le cas d'une réparation globale, la réparation doit être réalisée en considération de tous les agents touchés par la décision remise en question. Enfin, les réparations doivent être menées en accord avec le besoin de réactivité du système. Par exemple, une réparation globale peut prendre la forme d'un problème de MRTA apparaissant en ligne et devant être résolu dans un temps imparti.

Dépendance des tâches (Sec. 2.5.4) : Dans un problème de chasse aux mines, la contribution d'un robot à un objectif donné n'est pas indépendante des autres objectifs qu'il doit accomplir. La présence de ces dépendances imposent de planifier la mission avec prudence afin de ne pas causer de blocage à l'exécution. Ces contraintes peuvent être des contraintes causales, de précédence, de synchronicité, ou de périodicité. En plus de ces contraintes, il est nécessaire d'optimiser la résolution du problème de MRTA en tenant compte des utilités corrélées des tâches.

La résolution du problème de MRTA est grandement complexifiée par la présence de ces dépendances complexes. Ces dépendances doivent être prises en compte afin de ne pas faire chuter les performances du système.

Communications incertaines (Sec. 2.5.5) : Enfin, les mécanismes de coopération nécessaires pour relever ces défis doivent tenir compte des moyens de communication disponibles.

La modélisation des communications acoustiques utilisées en environnement sous-marin est extrêmement difficile. Bien que la finalité de notre approche soit de l'appliquer dans cet environnement, nous avons fait le choix de nous affranchir de certaines contraintes de modélisation afin d'adresser le coeur de notre problème : la coopération multirobot. Dans le but de simplifier le développement de notre architecture nous avons donc posé des hypothèses à la Sec. 2.6.2, qui prennent en compte un débit faible, des pertes de messages, et des communications bilatérales.

Ces hypothèses sont faites en adéquation avec l'environnement sous-marin, où les robots ne peuvent ni se reposer sur un lien de communication stable avec un agent central, ni échanger un trop grand volume d'information pour prendre des décisions. Ces contraintes de communication complexifient d'autant plus la résolution en ligne des problèmes de MRTA.

4.2 VUE D'ENSEMBLE DE L'APPROCHE

Dans notre analyse de l'état de l'art au Chap. 3, nous avons dressé une synthèse des travaux connexes et dégagé des pistes de recherche.

Au regard de cette analyse, nous proposons une approche reposant sur une architecture décentralisée mêlant allocation de tâches par enchères et planification hiérarchique. Un protocole de décision, basé sur des allocations par enchères de réseaux de tâches hiérarchiques, permet de réparer le plan en ligne. L'approche exploite la planification hiérarchique pour résoudre les problèmes issus de la tenue d'enchère et superviser le plan. Cette approche supporte des problèmes partiellement ordonnés comportant des dépendances complexes.

4.2.1 *Allocation par enchères pour la chasse aux mines*

L'organisation du processus de décision est au coeur de la conception du système. Si la probabilité de ruptures de communication permet d'écarter le choix d'une architecture centralisée, les architectures décentralisées et distribuées paraissent viables selon ce critère.

Les tâches composant la mission de chasse aux mines peuvent être très coûteuses en temps, p. ex. plusieurs heures pour une couverture. Par conséquent, une bonne délibération entre les robots est nécessaire afin d'éviter l'accomplissement multiple et superflu d'une même tâche par des robots qui ne se seraient pas concertés. De plus, étant donné la nature sensible de la mission, l'opérateur doit pouvoir garder une bonne compréhension des décisions et actions du système.

Enfin, la présence de *Cross-schedule Dependencies* définit un problème partiellement ordonné. Ces dépendances doivent impérativement être prises en compte afin de ne pas compromettre la mission. La planification de tâches aussi complexes nécessite un fort degré de coopération.

Pour ces raisons, une architecture décentralisée, robuste aux pertes de communication et où la coopération des robots est au centre du processus de décision, nous semble plus indiquée pour notre mission. Pour mettre en place un tel processus de décision décentralisé, une **approche par enchères** nous semble souhaitable.

Le choix des enchères

Les approches par enchères permettent aux robots de s'organiser localement pour prendre des décisions. Ils résolvent ainsi des problèmes locaux moins complexes qu'un problème global. Ces approches permettent également une grande réactivité et résilience du système. En effet, chaque robot peut ouvrir une enchère si la situation l'exige.

Durant la mission, des aléas comme des retards peuvent survenir en raison des incertitudes liées à l'environnement et aux actions des robots. Si un robot a planifié une action, un élément inattendu, tel qu'un courant plus fort que prévu, peut impacter la réalisation de l'action. Pour faire face à un environnement incertain, il est vital de pouvoir s'adapter à des divergences entre ce qui est prévu et ce qui est rencontré. Avec un processus de décision basé sur des enchères, il est possible de réparer les conséquences de ces incertitudes sans avoir à les considérer explicitement.

Grâce aux allocations par enchères, il est possible de réparer un plan de mission en revendant les tâches qu'un agent ne peut plus assumer, ou en mettant en vente de nouvelles tâches. Les allocations par enchères permettent donc une grande résistance aux fautes par une prise en compte réactive des aléas.

Enfin, les méthodes d'allocation par enchères, bien que dépendantes du nombre de coéquipiers pouvant communiquer, sont reconnues comme robustes à la perte de messages. Elles doivent cependant être conçues dans le respect de la bande passante du moyen de communication considéré.

Il est important de souligner qu'à notre connaissance, aucune de ces approches n'a encore été testée en expérimentations sous-marines réelles. Comme beaucoup de travaux sur les systèmes multirobots en environnement sous-marins, cela peut s'expliquer par la difficulté d'accès à ce genre d'expérimentations. Les moyens logistiques et financiers à mettre en oeuvre limitent généralement l'étude de tels systèmes au cadre de la simulation.

Néanmoins, les résultats obtenus en simulation par de telles approches [Sar07 ; Fer+18] nous semblent cohérents avec les expérimentations réelles réalisées autour de la coopération multirobot pour des environnements sous-marins [Fer+20]. De plus, dans une mission de chasse aux mines, le théâtre d'opérations entier peut théoriquement être couvert par la portée des moyens de communication acoustiques. Enfin, la coopération entre robots est nécessaire de manière locale, c.-à-d. avec des robots évoluant dans des zones proches.

Pour ces raisons, une approche par enchères en milieu sous-marin nous paraît adaptée pour définir les fondations de notre protocole de décision.

Tenir des enchères sur des tâches comportant des dépendances complexes

Bien que les approches par enchères permettent de surmonter les contraintes de communication du monde sous-marin, leurs performances chutent lorsqu'il faut considérer des problèmes difficiles comportant des dépendances complexes entre les tâches. Cela est dû à la résolution du WDP, qui devient, dans une telle situation, très complexe à résoudre en des temps et qualités raisonnables (Sec. 3.4.4). Or, les

problèmes de MRTA que notre architecture doit résoudre ont été classifiés comme *MT-MR-TA with ComplexDependencies and Commitment* selon une taxonomie de référence de la littérature (Sec. 3.2.3). Ces problèmes sont considérés comme difficiles à résoudre en raison de la présence d'alternatives pour accomplir une tâche, de contraintes entre tâches, et d'utilités corrélées. Par conséquent, nous devons lever un premier verrou pour mettre en place ce schéma de décision : Pouvoir prendre en compte des dépendances complexes lors d'allocations par enchères, notamment en résolvant le WDP en des temps et qualités convenables.

Pour pouvoir surmonter le défi posé par la présence d'alternatives, il est nécessaire d'alimenter le processus d'allocation d'information sur les décompositions possibles et d'alimenter la décomposition de la mission sur les choix d'allocation réalisés. L'entrelacement des allocations par enchères avec des structures hiérarchiques est un moyen d'y parvenir. De telles méthodes permettant de mettre en place un compromis avec des enchères combinatoires. Des travaux, comme [ZS05 ; CLI12], ont déjà exploré ces méthodes en procédant à des allocations de HTN. Les résultats de ces travaux mettent en avant la baisse de complexité du WDP et la prise en compte de choix dans les tâches allouables. Mais, ces travaux reposent sur des algorithmes gloutons sacrifiant la qualité de la solution. De plus, ces algorithmes sont incapables de traiter des tâches comportant des dépendantes complexes.

Il est important de souligner que ces approches n'utilisent les réseaux de tâches que comme des structures de données modélisant le problème de MRTA. **Notre intuition est que nous pouvons surmonter le défi de la prise en compte de dépendances complexes en entremêlant enchères et planification hiérarchique.** Selon nous, il est possible d'améliorer ces méthodes afin de gagner en qualité de solution tout en considérant de telles dépendances. Cette amélioration passe par une exploitation plus profonde du concept de HTN, notamment par l'utilisation de la planification hiérarchique [BAH19] pour estimer les mises et résoudre le WDP de manière efficace.

4.2.2 *Enchérir à l'aide de la planification hiérarchique*

Les tâches de chasse aux mines sont complexes à planifier mais peuvent être représentées avec des HTN pour ensuite tirer profit des performances de la planification hiérarchique.

Avantages de la planification hiérarchique

Les réseaux de tâches hiérarchiques existent depuis de nombreuses années, dans [EHN94] le formalisme HTN, encore largement utilisé, est présenté. Au sein d'un HTN, on distingue deux types de tâches : les *tâches primitives*, qui ne peuvent être décomposées ; et les *tâches abstraites*, pouvant être décomposées en d'autres tâches abstraites ou primitives. La Fig. 4.1 illustre la structure d'un HTN. Un réseau de tâche est construit à partir d'une tâche racine grâce à des *méthodes* de décompositions jusqu'à atteindre le niveau le plus bas du réseau, composé uniquement de tâches primitives.

Un HTN intègre des connaissances expertes en décomposant une tâche complexe en des tâches plus simples. Ainsi, des dépendances hiérarchiques peuvent être établies entre les tâches. Cet apport de connaissance permet de représenter de manière abstraite la mission à accomplir, mais également de guider la recherche d'une solution. Ainsi, la complexité du problème peut être réduite grâce à la spécification de la mission.

Bien que la planification hiérarchique ne soit pas prouvée comme plus performante que la planification classique, elle permet en pratique de résoudre des problèmes plus efficacement lorsqu'une modélisation adéquate est possible [GA15 ; BAH19]. La recherche d'une solution à un problème de planification

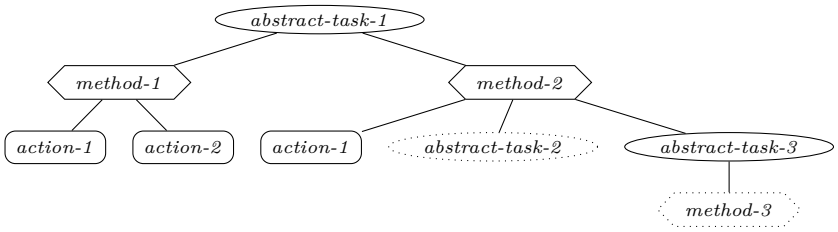


FIGURE 4.1 – Illustration de la structure d'un HTN. Les éllipses représentent des tâches abstraites, les hexagones des méthodes, et les rectangles des tâches primitives. Les tâches abstraites sont décomposées via des méthodes en d'autres tâches abstraites ou tâches primitives, également appelées actions. Les tâches et méthodes en pointillés ne sont pas décomposées par souci de clarté.

hiérarchique peut être faite par un solveur dédié comme le solveur PANDA [1] [Höl+21] ou Aries [2] [Bit18]. Cette formalisation du problème, couplée à un planificateur hiérarchique, permet d'obtenir de bonnes performances. Selon le solveur utilisé, il est possible d'optimiser la solution tout en prenant en compte des contraintes telles que des contraintes de précédences.

Représentation hiérarchique de la chasse aux mines

Toutes les missions ne se prêtent cependant pas à l'utilisation du formalisme HTN. Pour pouvoir construire ces réseaux de tâches, il faut une connaissance a priori sur la manière d'accomplir une mission. La mission de chasse aux mines peut intégrer de telles connaissance : bien que l'environnement soit incertain et la rencontre d'aléas probable, l'accomplissement des objectifs peut être découpé en plans génériques servant de base de connaissances au système multirobot. Par exemple, pour les objectifs de couverture, une décomposition a priori de la zone peut être obtenue à l'aide des données sur la mission. Enfin, il est possible de spécifier des contraintes entre les tâches grâce au formalisme HTN.

En usant de ces possibilités, il peut parfois être possible de décomposer une tâche coopérative, c.-à-d. une *multi-robot task* au sens de la taxonomie, comme plusieurs *single-robot tasks*. Par exemple dans le cas de la chasse aux mines, il est possible de décomposer une tâche d'identification avec transmission des données en direct, comme une tâche d'identification et une tâche de mise en place d'un relais de communication. Ainsi, un problème de coordination complexe entre robots peut être simplifié comme la résolution de plusieurs sous-problèmes locaux.

Utilisation de la planification hiérarchique pour les enchères

Pour résoudre notre problème de MRTA à l'aide d'enchères et de la planification hiérarchique nous modélisons les objectifs à accomplir au sein d'un HTN. Grâce au formalisme HTN, nous pouvons spécifier des choix dans la planification de la mission. Cette caractéristique, couplée à notre schéma d'allocation par enchères, nous permet d'entremêler décomposition et allocation. En se reposant sur notre formalisme, il est également possible d'inclure dans la modélisation du problème de MRTA des contraintes de précédences, des contraintes causales, ainsi que des contraintes de synchronicité.

Pour estimer les mises et résoudre le problème de détermination des gagnants nous exploitons la planification hiérarchique. En procédant ainsi, nous pouvons considérer les dépendances complexes du problème de MRTA tout en résolvant efficacement les problèmes d'estimation des mises et de détermination des gagnants.

1. https://panda-planner-dev.github.io/
2. https://github.com/plaans/aries

Modéliser le problème de détermination des gagnants comme un problème de planification hiérarchique

Si l'utilisation de la planification hiérarchique pour estimer les mises semble triviale, cela l'est beaucoup moins pour résoudre le WDP. Considérer les mises comme autant d'alternatives possibles pour accomplir une tâche respecte l'esprit de la planification hiérarchique, où chaque alternative correspond à une décomposition, mais la modélisation du WDP sous la forme d'un problème de planification n'est pas aisée.

Lors de la résolution du WDP, il est nécessaire d'intégrer deux impératifs propres à l'utilisation d'enchères : un enchérisseur peut remporter au plus une mise et une tâche peut être remportée par au plus un enchérisseur. Par conséquent, les mises d'un enchérisseur doivent être considérées de façon indépendantes, mais les mises retenues entre enchérisseurs doivent être compatibles entre elles, c.-à-d. elles sont dépendantes entre elles.

De plus, lors de l'estimation d'une mise, les enchérisseurs déterminent des nouvelles contraintes qu'ils soumettent au commissaire-priseur avec leur mise, si la mise est retenue ces contraintes doivent être intégrées de manière pérenne dans le système. Ces nouvelles contraintes sont nécessaires en raison des dépendances complexes liant les tâches, si elles ne sont pas intégrées au système des blocages peuvent apparaitre lors de l'exécution. Ces contraintes sont associées aux mises, elles doivent donc elles aussi être considérées de façon indépendante pour les mises d'un enchérisseur, et être compatibles pour les mises entre enchérisseurs. Mettre en place ces contraintes au sein du WDP constitue un verrou à lever. Dans notre approche, nous montrons qu'avec une modélisation adéquate reposant sur le formalisme HTN, il est possible modéliser le WDP de manière à inclure ces contraintes ainsi que les alternatives permettant d'accomplir une tâche. En procédant comme tel, les dépendances complexes de notre problème sont prises en compte.

4.2.3 *Supervison et réparation des décisions*

Notre protocole de décision reposant notamment sur l'utilisation de HTN, nous proposons de tirer profit de ces structures riches pour superviser l'exécution des décisions prises et réparer le plan lorsque nécessaire.

Superviser avec la planification hiérarchique

Pour s'assurer du bon déroulement de leurs plans, les robots doivent les superviser. Pour cela, les robots réutilisent la même décomposition de la mission que lors des allocations par enchères en y intégrant les informations obtenues en ligne. Puis, nous résolvons le problème ainsi formé en faisant une nouvelle fois appel à la planification hiérarchique. Ce cadre nous permet de procéder efficacement à des vérifications de plans en ligne tout en tenant compte des contraintes et des choix établis à la décision.

En procédant ainsi, il est possible de réparer le plan localement, c.-à-d. à l'échelle de l'agent, ou de détecter lorsqu'une réparation globale, c.-à-d. à l'échelle de plusieurs agents, est nécessaire. Si un plan est trouvé, il peut différer localement de celui prévu initialement par l'agent mais il respectera toujours les contraintes imposées lors de la résolution du MRTA grâce à l'exploitation que nous faisons de la structure hiérarchique de la mission. Si aucun plan n'est possible, alors une réparation globale est déclenchée.

Réparer à l'aide du protocole de décision

Notre approche permet au système de procéder à des réparations globales en ligne en conservant exactement le même protocole d'allocation de tâches que lors de la prise de décisions. Pour ce faire, nous

exploitons une nouvelle fois la représentation hiérarchique de la mission afin de permettre à un agent en difficulté de revendre des tâches qu'il ne pourrait plus assumer. Dans notre approche, l'agent n'est pas limité à la revente de la décomposition qu'il avait acquise, il peut également revendre toutes les décompositions alternatives. Ainsi, l'agent bénéficie d'une plus grande lattitude pour réparer le plan. Cela est possible grâce à la conservation de la structure hiérarchique tout au long de la mission.

L'intégration de nouveaux objectifs se fait également en suivant ce principe, les nouvelles tâches à accomplir sont intégrées à la structure hiérarchique de la mission et mises en vente grâce au protocole de décision.

4.2.4 *Synthèse*

Dans la Fig. 4.2, nous résumons les principales caractéristiques de notre approche. En premier lieu, dans le but de mettre en place un schéma de décision décentralisé et de prendre en compte des dépendances complexes nous utilisons un protocole de décision entremêlant enchères et planification hiérarchique. Puis, notre protocole de supervision exploite les structures hiérarchiques de la décision pour intégrer les informations d'exécution et contrôler l'état du plan de mission tout en procédant à des réparations locales grâce à la planification hiérarchique. Enfin, nous procédons à des réparations globales du plan de mission en faisant appel au protocole de décision. Par conséquent, l'approche décrite permet de gérer de manière uniforme la planification initiale, l'intégration en ligne de nouveaux objectifs et la réparation globale de décisions. Ainsi, le système peut faire face à la nature dynamique de la mission en résolvant en ligne plusieurs problèmes de MRTA.

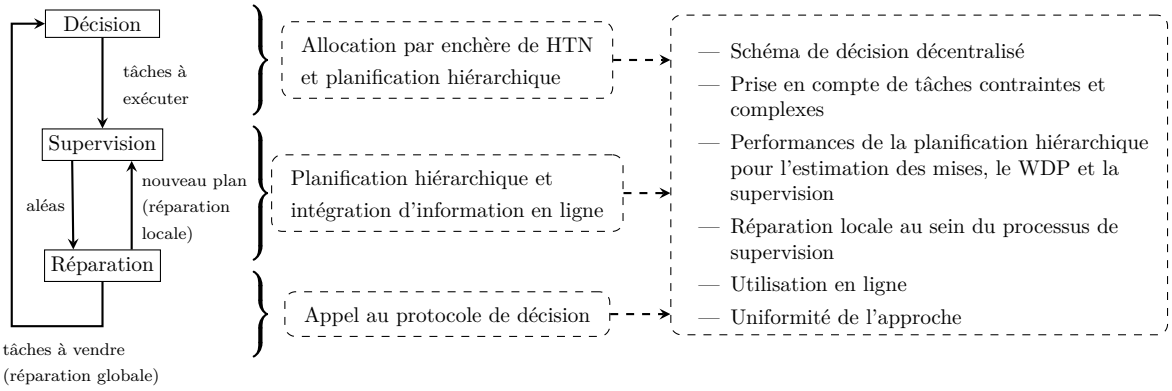


FIGURE 4.2 – Illustration des principales caractéristiques de notre approche. Sur le côté gauche les problématiques abordées, les processus de l'architecture sont encadrés, les flèches entre ces processus représentent des échanges de données. Au centre les concepts clés pour adresser ces problématiques. Sur le côté droit les principaux avantages de notre approche.

4.3 PROTOCOLE GÉNÉRAL

Dans cette section, nous décrivons le protocole général de notre approche mêlant enchères et planification hiérarchique. Ce protocole est illustré dans la Fig. 4.3. Nous attirons en premier lieu l'attention du lecteur sur les rectangles $\mathcal{H}$, ce sont des structures hiérarchiques appelées Multi-Robot HTN qui sont au coeur du fonctionnement de HaucTioN. Un Multi-Robot HTN regroupe toutes les informations sur les tâches devant être accomplies par les robots en encodant l'ensemble des tâches composant la mission et leurs décompositions. Chaque robot entretient un Multi-Robot HTN qui lui est propre et le met à jour en fonction

des informations générées et reçues. Par conséquent, un Multi-Robot HTN représente la décomposition hiérarchique de la mission ainsi que l'état du plan connu par un agent. Ces structures servent de base à la prise de décision des robots, mais également à la supervision de leur plan.

Les blocs de couleurs qui composent notre protocole sont des processus s'appuyant sur ces Multi-Robot HTN afin de rendre le système autonome en implémentant des algorithmes de décision, de supervision, d'exécution et de réparation. Dans les sections suivantes, nous présentons une vue d'ensemble de ces blocs et de leurs interactions.

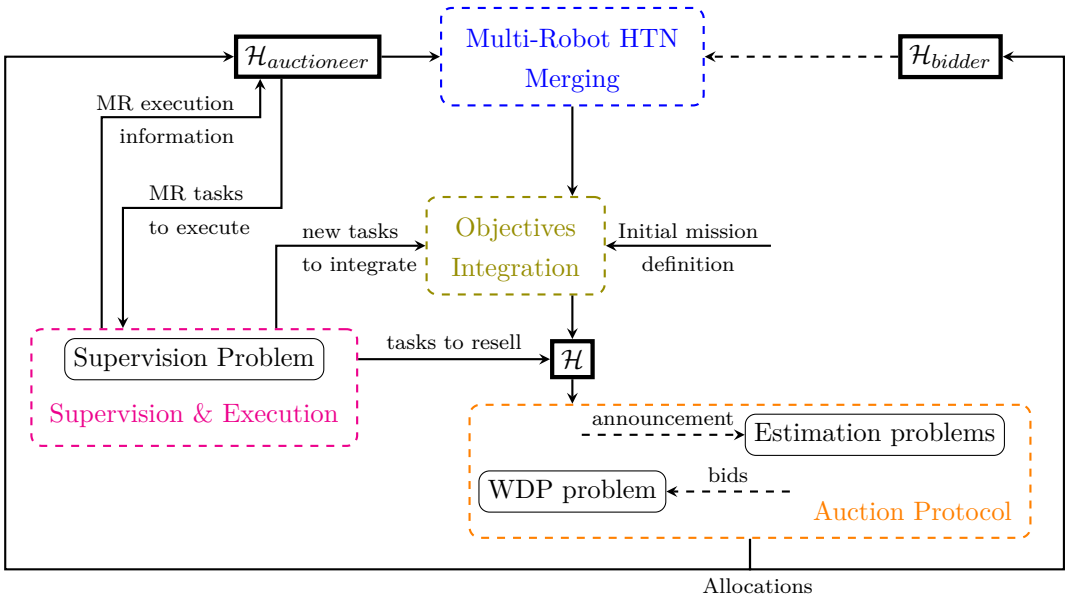


FIGURE 4.3 – Protocole général de HaucTioN. Les blocs de couleurs représentent des processus exécutés par les robots. Les rectangles représentent les structures de données hiérarchiques gérées par les robots, elles sont appelées Multi-Robot HTN et notées $\mathcal{H}$. Les rectangles arrondis indiquent des problèmes de planification hiérarchique. Les flèches en pointillés représentent des messages échangés entre commissaire-priseur et enchérisseurs. Les flèches continues indiquent les interactions entre les structures de données et les processus.

4.3.1 *Allocation et décomposition conjointes : Protocole d'enchère*

Le schéma de décision de HaucTioN correspond au bloc "Auction Protocol", en orange dans la Fig. 4.3. Une vue d'ensemble du schéma de décision de HaucTioN est présentée dans la Fig. 4.4. Ce schéma de décision permet d'allouer des objectifs de mission hiérarchiquement décomposés à l'aide d'enchères, il en suit donc les différentes étapes : 1. Annonce ; 2. Estimaton des mises ; 3. Détermination des gagnants ; 4. Récompense. Au sein de notre approche, l'allocation de tâches se fait en mettant en vente des tâches contenues dans le Multi-Robot HTN, noté $\mathcal{H}$, d'un agent. Dans cette illustration simplifiée du processus de décision, le Multi-Robot HTN dont les tâches sont en vente est l'arbre de tâche associé au message d'annonce. Les différents arbres associés aux autres messages illustrent les mises et décisions prises sur $\mathcal{H}$.

Annonce

Lorsqu'un robot a connaissance d'objectifs de mission non alloués, il devient commissaire-priseur d'une nouvelle enchère. Cette enchère débute par un message d'annonce dans lequel le commissaire-priseur diffuse son Multi-Robot HTN en précisant les tâches qui y sont en vente.

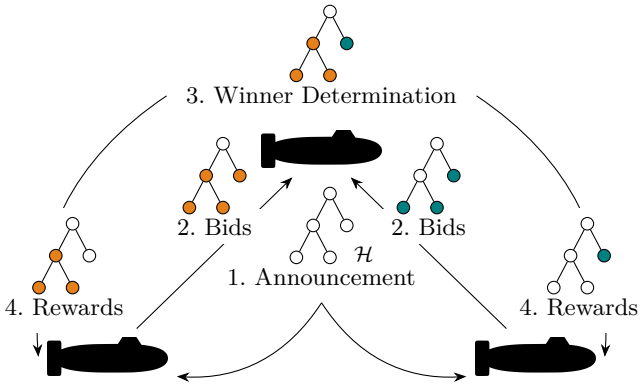


FIGURE 4.4 – Vue d'ensemble du schéma de décision. Les objectifs de mission, décomposés en un Multi-Robot HTN, sont mis aux enchères. Chaque robot enchérit sur les sous-tâches, un processus de détermination du vainqueur attribue les tâches et les envois aux robots ("rewards").

Estimation des mises

À la réception du Multi-Robot HTN correspondant à l'enchère, les enchérisseurs doivent estimer des mises sur les objectifs en vente. Pour réaliser cette estimation, ils estiment les tâches en vente du Multi-Robot HTN en fonction de leurs **capacités locales**. Ces capacités locales sont encodées dans des HTN décrivant comment accomplir les tâches contenues dans le Multi-Robot HTN.

L'estimation se fait donc en étendant le Multi-Robot HTN avec des HTN locaux afin de former un unique problème de planification hiérarchique. Une illustration du problème ainsi formé est représentée sur la Fig. 4.5. Dans cette illustration, le Multi-Robot HTN utilisé dans l'annonce est le même que celui de la Fig. 4.4.

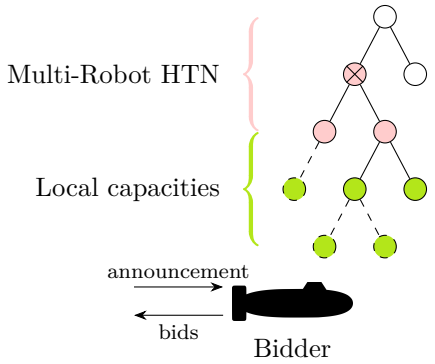


FIGURE 4.5 – Illustration simplifiée de la création du problème d'estimation des mises à partir du Multi-Robot HTN et des HTNs locaux de l'enchérisseur. La tâche à estimer et ses descendantes dans le Multi-Robot HTN sont en rose. Les décompositions locales sont en vert. Les noeuds en pointillés ne sont pas décomposés par souci de simplicité. La tâche a estimer est ⊗ .

Le problème d'estimation ainsi formé est résolu grâce un planificateur hiérarchique. La solution obtenue est composée d'un ensemble d'actions locales à réaliser pour accomplir les objectifs estimés. Ces actions locales incrémentent un coût qui est utilisé comme valeur de la mise.

La solution de l'estimation d'une mise permet également d'extraire le plan multirobot de l'agent pour les tâches du Multi-Robot HTN, c.-à-d. le plan qu'il a obtenu sans le raffinement en actions locales. Ce plan est soumis par l'agent au commissaire-priseur avec sa mise.

Nous considérons que les tâches contenues dans le Multi-Robot HTN et les tâches locales sont décorrélées, c.-à-d. que les effets des tâches contenues dans le Multi-Robot HTN n'ont pas d'impact sur ceux des tâches locales et inversement. Cette dualité nous permet de formuler un problème d'estimation cohérent dont les informations peuvent ensuite être utilisées pour résoudre le WDP.

Le plan multirobot de l'enchérisseur intègre les tâches du Multi-Robot HTN *souhaitées* ou *précédemment obtenues* par l'agent, mais également les tâches *ayant un impact* sur les tâches souhaitées ou déjà obtenues. Ces tâches multirobot impactant l'agent sont présentes dans son plan indépendamment de si elles ont déjà été allouées ou non à d'autres coéquipiers. À partir des informations contenues dans le plan multirobot, l'agent peut déterminer les contraintes qui lui sont nécessaires pour accomplir les objectifs concernés. Par exemple, si l'agent requiert une contrainte pour que sa mise soit valide, p. ex. "cette tâche doit être faite avant que je réalise celle-ci", et que cette contrainte n'est pas déjà présente dans le problème d'estimation, alors l'agent doit la communiquer avec sa mise. Ainsi, le commissaire-priseur peut intégrer cette contrainte en vue de sa résolution du WDP. Enfin, cette contrainte supplémentaire est ensuite intégrée de manière permanente au système si la mise est retenue, cela est réalisé en encodant la contrainte au sein des Multi-Robot HTN des participants de l'enchère. Il est important de noter que la prise en compte de ces contraintes supplémentaires est impérative pour supporter des problèmes partiellement ordonnés, sans cela, et comme discuté dans [Mil+21b], des blocages à l'exécution peuvent apparaitre.

Détermination des gagnants

Le commissaire-priseur intègre ensuite les mises réceptionnées dans le WDP. Le protocole utilisé pour inclure ces mises permet d'assurer que :

1. Un enchérisseur peut remporter au plus une mise ;
2. Une tâche peut être allouée à au plus un enchérisseur ;
3. Les mises retenues sont compatibles entre elles.

Enfin, si une tâche abstraite est allouée, alors ses descendants le sont également. Il n'est donc pas possible de mettre en vente une tâche abstraite si l'un de ses descendants a déjà été alloué.

Parfois, il peut être préférable d'espérer pouvoir allouer la tâche plus tard que de choisir une mise trop pénalisante. Pour mettre en place cette possibilité, le commissaire-priseur crée une décomposition correspondant à la revente à un tour ultérieur de la tâche. Suivant la fonction utilisée pour fixer le coût de revente, et comme expérimenté dans [Mil+21a] et le Chap. 8, il est possible d'optimiser des éléments comme la qualité d'allocation ou le nombre de tours d'enchères nécessaires (Chap. 6).

Une illustration simplifiée du problème de détermination des gagnants ainsi formé est représentée sur la Fig. 4.6. Avec la modélisation que nous faisons, le WDP est exprimé comme un problème de planification hiérarchique. Cette modélisation permet de prendre en compte la revente d'une tâche ou son allocation comme des alternatives possibles dans le plan.

Il est important de noter qu'une telle modélisation du WDP quand les tâches comportent des dépendances complexes n'est pas triviale. Aussi les défis soulevés par ces problèmes sont discutés dans le Chap. 6.

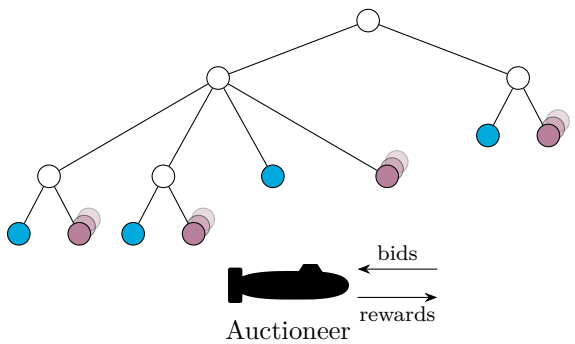


FIGURE 4.6 – Illustration simplifiée de la création du WDP à partir du Multi-Robot HTN, des mises reçues, et de la revente. Les noeuds de revente sont en bleu. Les noeuds d'allocation sont en violet. Par souci de simplicité les noeuds d'allocation sont superposés, ils correspondent à autant de décompositions possibles, c.-à-d. une pour chaque mise reçue.

Pour résoudre efficacement le WDP ainsi construit, notre approche utilise le même solveur hiérarchique que pour l'estimation des mises. La solution de ce problème permet d'obtenir un plan global correspondant à un ensemble d'allocations et de tâches à revendre à un prochain tour d'enchère. Ce plan est optimisé en tenant compte de la valeur des mises et des coûts de revente.

Récompense

Une fois un ensemble d'allocations trouvé, le commissaire-priseur annonce les récompenses.

En gagnant une mise, un agent s'engage à respecter le plan multirobot associé. Les contraintes de la mise sont donc directement intégrées dans le Multi-Robot HTN du commissaire-priseur qui diffuse à tous les participants les mises remportées ainsi que son Multi-Robot HTN mis à jour. Cette mise à jour est nécessaire en vue de prochains tours d'enchères ou à des fins de réparations ultérieures. Enfin, les agents concernés par une récompense planifient l'exécution des tâches remportées le cas échéant.

Si des tâches sont toujours non allouées à la fin de ce tour, le commissaire-priseur débute un nouveau tour d'enchère en envoyant le Multi-Robot HTN mis à jour et en y indiquant les tâches en vente.

4.3.2 *Exécution et supervision*

Une fois des tâches acquises, un agent doit les exécuter conformément au plan multirobot engagé dans la dernière mise qu'il a remportée, il doit également en superviser l'exécution afin de faire appel à des méthodes de réparation lorsque nécessaire. Le bloc violet "Supervision & Execution" dans la Fig. 4.3 correspond à ces processus.

Pour superviser le plan, notre approche tire profit de la conservation des structures hiérarchiques utilisées pour la décision en faisant une nouvelle fois appel à la planification hiérarchique. Similairement au processus d'estimation des mises, nous nous reposons sur l'extension du Multi-Robot HTN avec des capacités locales. L'agent étend avec ses décompositions locales les objectifs qu'il s'est engagé à réaliser dans son Multi-Robot HTN. Pour rappel, les contraintes engagées par l'agent lors de sa mise sont déjà intégrées à son Multi-Robot HTN lors de la phase de récompense, de plus, ces contraintes ne concernent que des tâches multirobots. Par conséquent, le plan de mission est encodé dans le Multi-Robot HTN de l'agent, et c'est la résolution d'un problème de planification hiérarchique basé sur son Multi-Robot HTN et ses capacités locales qui permet à l'agent de superviser son plan et déterminer les prochaines actions à exécuter.

Lors de la supervision, il est important de prendre en considération l'état actuel du plan. Sans information sur l'état du monde les robots peuvent très vite se retrouver bloqués par les contraintes portant sur les tâches. Ces informations sont perçues ou reçues par la couche fonctionnelle du robot. Au fur et à mesure de la mission, un agent va générer des informations sur les tâches de mission qu'il réalise, p. ex. l'agent a fini une tâche à un instant donné. Cet agent peut également recevoir des informations sur les tâches réalisées par ses coéquipiers, p. ex. tel coéquipier a commencé telle tâche à un instant donné. Pour tenir compte des informations dynamiques sur la mission, nous les intégrons au processus de supervision. Pour ce faire, nous utilisons une notion de passé et futur séparés par un temps courant correspondant au présent. Les informations connues appartiennent par nature au passé de l'agent. Ces informations sont intégrées aux tâches de mission auxquelles elles se rapportent dans le Multi-Robot HTN de l'agent. Par cette intégration, elles ancrent les éléments auxquelles elles se réfèrent dans le passé de l'agent. Les éléments pour lesquels aucune information n'est connue sont supposés comme non avenus, ils ne peuvent

appartenir au passé. Nous intégrons donc une contrainte avec le présent portant sur tous les éléments pour lesquels aucune information n'est connue : ces éléments doivent avoir lieu après le temps courant.

La Fig. 4.7 montre une représentation simplifiée du problème de supervision ainsi formé et un exemple de plan solution séquentiel. Nous utilisons un solveur hiérarchique sur ce problème de supervision pour déterminer si un plan est toujours possible. Si un plan est trouvé, alors il respecte les engagements de l'agent et l'agent peut exécuter les prochaines actions locales possibles, c.-à-d. les actions pouvant débuter au temps courant. Si aucun plan n'est trouvé, l'agent ne peut plus respecter ses engagements et une réparation est nécessaire.

Il est à noter que, si un plan est trouvé, le plan résultant peut être différent de celui obtenu lors de l'estimation de la mise. L'agent peut en effet utiliser des actions locales différentes, cependant, par la construction hiérarchique du problème, le plan obtenu respectera toujours les contraintes multirobot que l'agent s'est engagé à tenir.

Enfin, ce processus de supervision est appelé chaque fois qu'une nouvelle information sur le plan est connue ou que le robot remporte des tâches.

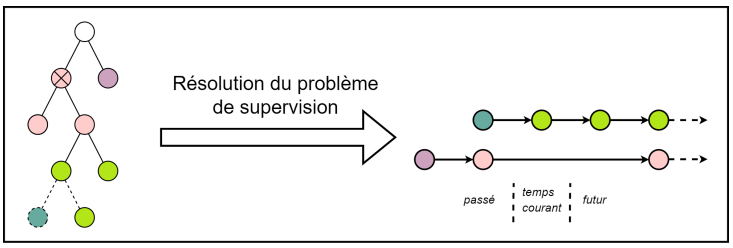


FIGURE 4.7 – Illustration simplifiée du problème de supervision. Les tâches multirobots à accomplir par l'agent sont en rose et les tâches locales sont en vert. Les tâches et représentent respectivement une tâche locale et une tâche multirobot finies. À droite est représenté le plan résultant de la résolution du problème de supervision avec son raffinement en tâches locales (en haut) et multirobots (en bas).

4.3.3 *Réparation locale et globale*

Les causes des réparations peuvent être multiples. Par exemple, une réparation est nécessaire lorsqu'un nouvel objectif est connu ou lorsqu'un agent ne peut plus respecter son plan. Ces réparations peuvent être locales à l'agent ou concerner plusieurs agents, nous parlons alors de réparation globale.

Les problèmes relevant de l'exécution du plan de l'agent sont détectés grâce au protocole de supervision introduit à la section précédente. Il est important de noter que, par sa nature itérative, et la possibilité de changer les actions locales à chaque itération, le problème utilisé pour la supervision permet à l'agent de régulièrement ajuster son plan. Une réparation globale est nécessaire lorsque les contraintes multirobot ne peuvent plus être satisfaites. Si un plan est trouvé, une réparation locale a pu avoir lieu.

Si aucun plan n'est trouvé lors de ce processus de supervision, l'agent peut faire appel à une réparation globale ou abandonner une partie de ses objectifs. Une des raisons à l'absence de plan lors de la supervision peut être un retard de l'agent sur son plan.

Dans ce cas l'agent peut déterminer un ensemble de tâches qu'il ne peut plus assumer et qu'il espère revendre. L'agent fait alors appel au bloc "Auction Protocol" pour ouvrir une enchère en diffusant son Multi-Robot HTN et en y indiquant les tâches qu'il revend. Ce processus de réallocation par enchères est réalisé de la même manière que pour la planification initiale (Sec. 4.3.1). Si l'agent arrive à revendre ses tâches, le plan est réparé. Si l'agent ne parvient pas à réparer son plan il est obligé d'abandonner tout ou une partie de ses objectifs.

Le logigramme de la réparation est illustré dans la Fig. 4.8.

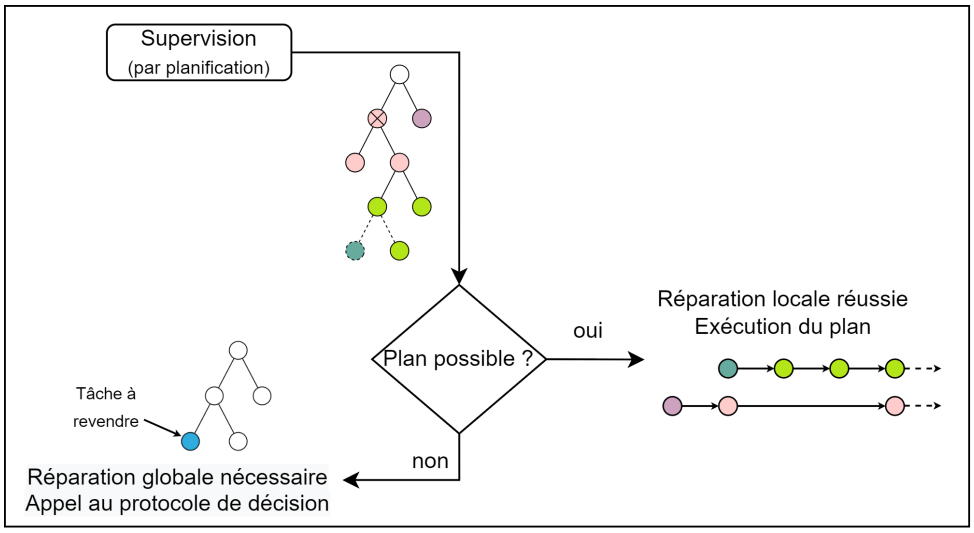


FIGURE 4.8 – Logigramme de la réparation.

Selon les situations, l'agent peut avoir la possibilité de choisir quels objectifs abandonner. Ce choix est fait au regard des éléments impactés par l'abandon, tels que les tâches dépendantes, le plan des coéquipiers concernés, ou la performance de la mission.

Enfin, il est nécessaire de pouvoir intégrer dans le système des nouveaux objectifs détectés en ligne. La couche fonctionnelle du robot, toujours dans le bloc "Supervision & Execution", est en charge de la détection de ces objectifs. Quand un nouvel objectif est détecté par un agent, les informations relatives à cet objectif sont transmises au bloc vert "Objectives Integration" en charge de l'intégration de ces objectifs au Multi-Robot HTN de l'agent. L'agent devient alors commissaire-priseur et met en vente les nouvelles tâches contenues dans son Multi-Robot HTN. Pour ce faire il fait appel au bloc "Auction Protocol".

Par conséquent, dans notre approche la réparation globale du plan multirobot passe par le processus de décision, c.-à-d. le bloc "Auction Protocol", que cela soit pour revendre des tâches ou pour intégrer des nouvelles tâches à accomplir. La réparation globale du plan à l'aide de notre protocole de décision a été présentée dans [Mil+22b ; Mil+22a] et le Chap. 7.

4.3.4 *Prévenir les divergences de connaissance*

En raison des erreurs de communication, les connaissances dynamiques des robots ne sont pas homogènes. Cette hétérogénéité des connaissances peut résulter à des incohérences en ligne. Par exemple, si un robot ne pouvant communiquer est supposé en panne par ses coéquipiers ses tâches peuvent être réallouées, mais si le robot revient dans le système les décisions prises par ses coéquipiers seront en incohérence avec celles qu'il connaît. Par conséquent, pour un robot à un instant donné, son plan multirobot peut toujours être valide en ne considérant que les contraintes imposées lors de la prise de décision, mais invalide lorsqu'il doit intégrer les informations obtenues en ligne. C'est d'autant plus courant lorsque l'exécution n'est pas totalement maitrisée, comme c'est le cas avec un système multirobot évoluant en environnement incertain.

Ces incohérences doivent être corrigées en encourageant les robots à communiquer lorsque possible. Dans notre approche, elles compromettent directement le processus d'enchères car les enchérisseurs et le commissaire-priseur doivent raisonner sur les mêmes informations pour formuler des décisions cohérentes.

Pour surmonter ce problème, l'ouverture du premier tour d'une enchère est précédée par une étape de fusion des informations détenues par les robots participant à l'enchère. Elle permet aux robots d'accorder leurs connaissances avant de prendre une décision. Par conséquent, nous imposons qu'un robot ne peut rejoindre une enchère en cours s'il n'a pas participé à cette étape de fusion des connaissances. Ce processus correspond au bloc "Multi-Robot HTN Merging" dans la Fig. 4.3.

Les connaissances nécessaires à la prise de décision sont contenues dans le Multi-Robot HTN de chaque agent, aussi ce sont ces structures de données qui doivent être comparées afin de faire émerger une cohérence globale. Il est important de souligner que cette fusion des connaissances constitue un défi d'envergure que nous n'avons pas encore pu explorer en profondeur. Dans notre approche, nous utilisons une solution simple où les connaissances du commissaire-priseur font foi. Cette méthode naïve peut résulter en de nombreuses réparations à accomplir ensuite par les enchérisseurs. De plus amples recherches sur ce défi sont donc une des perspectives principales de notre approche.

4.4 SYNTHÈSE DES CONNAISSANCES REQUISES PAR LES ROBOTS

Nous résumons ici les connaissances requises par les robots dans notre approche. Nous établissons une distinction entre ces deux types de connaissances :

— Connaissances *locales* : Elles sont propres à chaque agent et connues uniquement par l'agent. Elles ne sont pas partagées durant la mission. Par exemple, il peut s'agir des capacités d'un agent.

— Connaissances *multirobot* : Elles sont connues par un ou plusieurs agents et doivent idéalement être diffusées à tous. Par exemple, elles peuvent représenter les objectifs à accomplir ou les connaissances sur l'environnement.

Ces connaissances peuvent être dynamiques ou statiques. Elles sont représentées dans la Fig. 4.9.

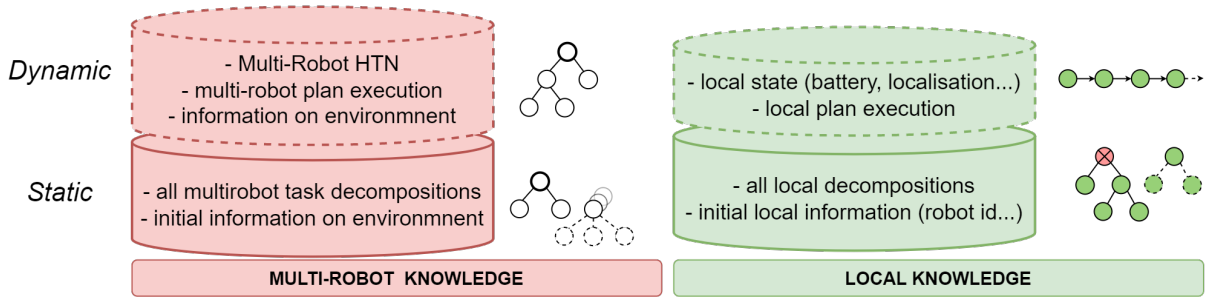


FIGURE 4.9 – Illustration des connaissances nécessaires à un robot.

Les objectifs de la mission relèvent des connaissances multirobots. Ils sont regroupés selon des décompositions préétablies dans le Multi-Robot HTN. Les objectifs à réaliser pouvant arriver en ligne, cette structure est dynamique. Par conséquent, chaque robot entretient son propre Multi-Robot HTN et le met à jour en fonction des informations sur la mission. Ces informations peuvent être générées localement par le robot ou réceptionnées suite à des messages de coéquipiers ou de l'opérateur.

Les capacités locales des robots décrivent leur aptitude à réaliser un objectif donné. Elles représentent des plans génériques permettant d'accomplir un objectif de mission. Ces plans sont des décompositions préétablies au sein de HTN locaux et statiques. Ces HTN constituent une bibliothèque de connaissances permettant à un robot de planifier efficacement la réalisation d'un objectif. Le choix de la décomposition à utiliser se fait lors de la planification de l'objectif en fonction de données dynamiques, p. ex. le niveau d'énergie. De même, les décompositions génériques des objectifs multirobots sont connues avant la mission

et partagées par tous, ce sont des connaissances statiques. Ces décompositions sont instanciées selon les paramètres de la situation rencontrée lorsqu'un robot doit créer un nouvel objectif.

Durant la mission, les robots peuvent planifier la réalisation du Multi-Robot HTN en appliquant leurs capacités locales aux objectifs le composant. Ce processus est au coeur de l'estimation des mises, de la supervision des plans, et des réparations locales. De cette planification découle un plan local que le robot exécute et supervise, ce plan local fait partie des informations dynamiques.

4.5 CONCLUSION

En conclusion, HaucTioN est articulée autour de deux protocoles principaux :

— Un protocole de décision ;

— Un protocole de supervision et d'exécution des décisions prises.

Ces protocoles permettent d'alterner en ligne les phases de décision et d'exécution.

Le protocole de décision mêle enchère et planification hiérarchique. Pour le mettre en place, il faut surmonter certains défis dont le principal est la modélisation du WDP sous la forme d'un problème de planification hiérarchique intégrant des dépendances complexes. Nous détaillons comment ces défis ont été relevés et nous formalisons ce protocole dans le Chap. 6.

Nous avons introduit comment le protocole de supervision réutilise les structures hiérarchiques de la décision pour superviser l'exécution du plan. Cette réutilisation des structures hiérarchiques permet également de réparer réactivement le plan en faisant appel au protocole de décision. Dans le Chap. 7, nous détaillons comment nous procédons pour superviser et réparer le plan à l'aide de ces structures.

Enfin, pour formaliser les structures hiérarchiques complexes qui sont exploitées par ces protocoles nous utilisons un formalisme HTN. Cependant, les formalismes HTN incluant des contraintes de synchronicité n'ont reçu que peu d'attention dans la littérature. Nous avons donc étendu un formalisme existant pour encoder de telles contraintes. Si le protocole de décision peut être perçu comme la clé de voûte de notre approche, ce formalisme en est la pierre angulaire. C'est donc le sujet du prochain chapitre, le Chap. 5.

5

FORMALISMES HTN ET HDDL TEMPORELS

Aperçu

Dans ce chapitre, nous décrivons les formalismes HTN et HDDL que nous utilisons au sein de notre approche. Nous présentons notamment comment ces formalismes ont été étendus afin de pouvoir intégrer les contraintes temporelles de notre problème.

Dans notre approche, nous modélisons la mission à accomplir sous la forme d'une structure hiérarchique comportant des dépendances complexes. Pour ce faire, nous exploitons deux formalismes :

— Un formalisme basé HTN [EHN94]. Il permet de **modéliser les structures hiérarchiques exploitées par les processus de décision, supervision, et réparation**, de notre architecture ;

— Un formalisme basé Hierarchical Domain Definition Language (HDDL) [Höl+20]. Il permet de **modéliser les informations en entrées** de notre système.

Ces formalismes définissent les fondations de notre approche.

Grâce au formalisme HTN présenté dans littérature, nous pouvons modéliser la mission à accomplir sous la forme d'une structure hiérarchique incluant des choix. Cette caractéristique, couplée à notre schéma d'allocation par enchères, nous permet d'entremêler décomposition et allocation. Avec un tel formalisme il est possible d'inclure des contraintes de précédences ainsi que des contraintes causales.

Cependant, le formalisme HTN est incomplet lorsqu'il s'agit de prendre en considération un cadre temporel. Il n'est par exemple pas possible d'inclure des contraintes de synchronicité. Il en est de même pour le formalisme HDDL. En effet, les problèmes de planification hiérarchique sous contraintes temporelles n'ont reçu que peu d'attention, certains travaux récents comme [Bit+20 ; Pel+22] explorent ces notions, mais aucun consensus n'existe autour d'un formalisme HTN ou HDDL capable d'intégrer un cadre temporel.

Pour pallier ce défaut, sur la base de la littérature, nous étendons les formalismes HTN et HDDL existants afin d'y intégrer une représentation temporelle. Cette extension temporelle permet par exemple aux robots de superviser l'exécution de leur plan en exploitant les informations générées en ligne et la structure hiérarchique de la mission.

Dans ce chapitre, nous introduisons les concepts manipulés par ces formalismes grâce à un exemple issu de notre mission. Puis, nous présentons ces formalismes et leurs extensions temporelles.

5.1 MODÉLISER DES RÉSEAUX DE TÂCHES COMPORTANT DES DÉPENDANCES COMPLEXES

Dans HaucTioN, les tâches composant la mission sont représentées grâce à des HTN. La conception de ces HTN se fait à l'aide de connaissances expertes de l'opérateur. Dans cette section, nous illustrons leur structure et caractéristiques à l'aide d'un objectif d'identification dont les décompositions possibles comportent des contraintes devant être considérées dans notre mission de chasse aux mines (Sec. 2.5.4). La Fig. 5.1 représente un HTN décomposant l'objectif d'identification.

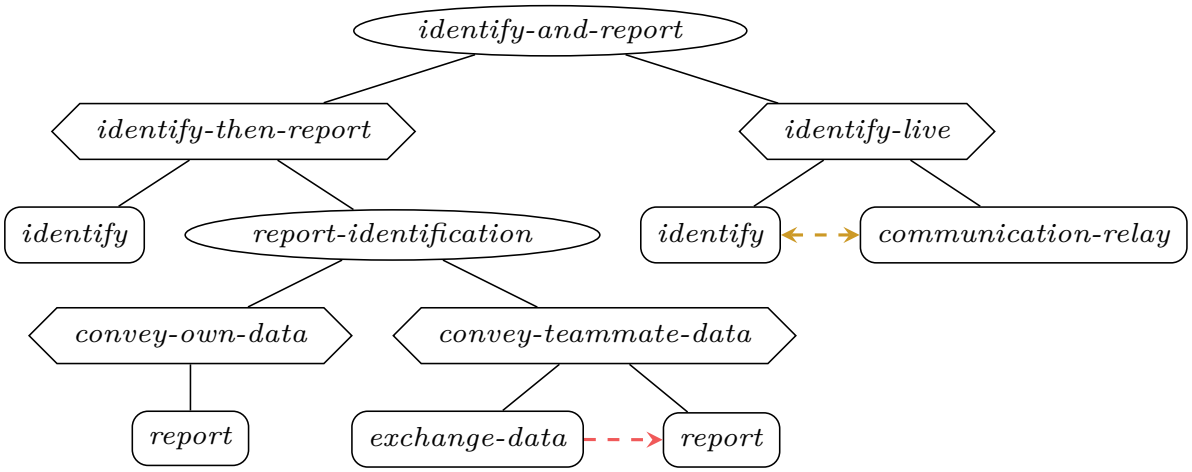


FIGURE 5.1 – Exemple de HTN correspondant à l'identification d'un objet suspect et la transmission à l'opérateur des données obtenues. Les éllipses représentent des tâches abstraites, les hexagones des méthodes, et les rectangles des tâches primitives. Les flèches - -> et <- -> représentent respectivement une contrainte de précédence et une contrainte de synchronicité.

La *tâche* identification doit être faite en rapportant les données à l'opérateur, soit via l'envoi d'un rapport post-identification, soit via un retour d'information en direct. Ces deux possibilités correspondent respectivement à deux *méthodes* : $identify\text{–}then\text{–}report$ et $identify\text{–}live$ du réseau de tâches.

La méthode d'identification avec retour d'information en direct nécessite d'établir un lien de communication entre le robot identifiant la mine et l'opérateur à la surface. Par conséquent, l'identification de la mine par l'AUV doit se faire conjointement à la mise en place d'un relai par un coéquipier, par exemple un USV *helper*. Ces *tâches primitives* d'identification ($identify$) et de relai ($communication\text{–}relay$) sont liées par une *contrainte* de synchronicité.

La méthode $convey\text{–}own\text{–}data$ permet au robot ayant identifié l'objet de rapporter par lui-même les informations de l'identification.

Pour qu'un robot puisse rapporter ces informations, il faut qu'il remonte à la surface, ce qui est un processus coûteux en temps et énergie. Pour optimiser les remontées nécessaires, il est également possible d'utiliser une mule de donnée pour rapporter les informations, ainsi ce n'est pas nécessairement le robot identifiant l'objet qui doit remonter à la surface. Cette méthode comporte une contrainte de précédence, il faut transmettre les données, puis rapporter.

Enfin, des *contraintes causales* lient également la tâche $identify$ aux tâches $exchange\text{–}data$ et $report$ car ces deux dernières nécessitent la disponibilité d'information sur l'identification de l'objet. L'identification doit donc se faire avant le rapport à l'opérateur.

En conclusion, cet HTN permet de spécifier trois alternatives pour identifier un objet suspect. Grâce à la structure abstraite de l'arbre, la recherche d'une solution est guidée pour réduire la complexité du problème.

Il est important de noter qu'avec les formalismes HTN existants, il est possible de spécifier des contraintes de précédence mais pas de synchronicité (respectivement les flèches - -> et <- -> dans Fig. 5.1). Dans les sections suivantes, nous introduisons donc un formalisme HTN et l'extension que nous en avons fait afin de pouvoir modéliser ces contraintes.

5.2 FORMALISME HTN

Le formalisme présenté ici est inspiré en grande partie du formalisme HTN utilisé dans [EHN94 ; Höl+20]. Tous les éléments introduits avec ce formalisme sont construits à partir d'un "Langage de planification", qui regroupe toutes les informations pouvant être utilisées. Une définition de ce langage est présenté dans la Def. 5.1.

Definition 5.1 (Langage de planification)

Par $\mathcal{L} = (\mathbb{V}, \mathbb{C}, \mathbb{P}, \mathbb{T}_p, \mathbb{T}_c, \mathbb{L})$*, nous désignons un langage du premier ordre où :*

- $\mathbb{V}$ *est un ensemble infini de variables ;*
- $\mathbb{C}$ *est un ensemble fini de constantes ;*
- $\mathbb{P}$ *est un ensemble fini de symboles de prédicats ;*
- $\mathbb{T}_p$ *est un ensemble fini de symboles de tâches primitives ;*
- $\mathbb{T}_c$ *est un ensemble fini de symboles de tâches complexes ;*
- $\mathbb{L}$ *est un ensemble infini de labels.*

De plus, nous définissons à partir de ces éléments des structures appelées *types*. Un *type* est un ensemble de constantes et/ou variables du langage. Les constantes et les variables peuvent appartenir à des types différents, par conséquent, les éléments d'ensemble peuvent être inclus dans un autre ensemble. Ou, dit autrement, un type peut hériter d'un autre type. Dans notre langage, tous les types héritent du type *Object*, qui est le type par défaut regroupant toutes les variables et constantes.

À partir de ce langage, il est possible de définir les tâches et la structure hiérarchique de notre problème de planification. Des exemples de comment les éléments du langage sont instanciés sont données dans Ex. 5.1.

Exemple 5.1 : Instances du langage de planification

AUVe1 et *AUVe2* sont deux constantes. L'ensemble formé par ces deux constantes définit le type *Explorer*, c.-à-d. $Explorer = \{ AUVe1, AUVe2 \}$. Le type *Explorer* hérite du type *Robot*, ce qui signifie que les objets *AUVe1* et *AUVe2* sont des robots et des explorers.
Un *prédicat* indique qu'un fait particulier est vrai. Il est utilisé pour représenter l'évolution d'un attribut spécifique de l'environnement pendant l'exécution du plan. Par exemple, $at(AUVe1, z_4)$ indique que le robot *AUVe1* se trouve dans la zone z_4. La plupart du temps, un prédicat est paramétré par des variables, dans ce cas le prédicat auquel il correspond dépend de la valeur prise par les variables. Par exemple, $at(r, z_4)$ fera référence à $at(AUVe1, z_4)$ ou $at(AUVe2, z_4)$ selon la valeur prise par la variable r de type *Robot*. La valeur des prédicats peut être modifiée par des *effets*.

Les *effets* sont définis au sein des *actions*. Une *action* représente une opération concrète réalisable. Elle est paramétrée par les éléments du langage de planification. La définition d'une action est présentée dans Def. 5.2.

Definition 5.2 (Action)

Par $\mathtt{T}_p = (\mathcal{S}, \mathrm{V}, \mathrm{C}, \mathsf{pres}, \mathsf{adds}, \mathsf{dels})$*, nous désignons une action (aussi appelée tâche primitive) où :*

- $\mathcal{S} \in \mathbb{T}_p$ *est le* ***symbole*** *de l'action, c.-à-d. son nom ;*
- V *et* C *sont respectivement les* ***variables*** *et* ***constantes*** *paramétrant l'action ;*

— pres *sont les* ***préconditions****. Chaque précondition indique que le prédicat associé doit être vrai pour effectuer l'action ;*

— adds *et* dels *sont respectivement des effets* ***positifs*** *et* ***négatifs****. Chaque effet e indique que le prédicat associé devient* ***vrai*** *si* $e \in$ adds *ou* ***faux*** *si* $e \in$ dels.

Une action est une opération concrète réalisable pouvant modifier l'état du monde.

Un exemple d'action est illustré dans Ex. 5.2

Exemple 5.2 : Instance d'une action

$move(r, from, to)$ désigne l'opération consistant à déplacer le robot r de l'emplacement $from$ à l'emplacement to. Il affectera les prédicats $at(r, from)$ et $at(r, to)$ afin d'interagir avec l'environnement.
Ici, *move* est un symbole correspondant à l'action de déplacement, $from$ et to sont des paramètres, $at(r, from)$ est une précondition devant être valide pour utiliser l'action, $at(r, from)$ est un effet négatif et $at(r, to)$ est un effet positif déclenchés par l'utilisation de l'action.

Les actions sont des éléments fondamentaux du problème, ce sont les seules opérations pouvant modifier l'état du monde. Pour bâtir une hiérarchie, il est nécessaire d'utiliser des *tâches*, qui peuvent être considérées comme un sous-type particulier des actions où adds = dels = $\varnothing$.

Une *tâche* représente une opération **abstraite** réalisable, c.-à-d. qu'elle peut être décomposée en d'autres tâches ou actions, mais ne peut pas changer la valeur des prédicats. Elle est également paramétrée par les éléments du langage de planification. La notion de tâche est définie dans Def. 5.3.

Definition 5.3 (Tâche)

Par $\mathtt{T}_c = (\mathcal{S}, \mathrm{V}, \mathrm{C}, \mathsf{pres})$*, nous désignons une tâche (aussi appelée tâche complexe) où* $\mathcal{S} \in \mathbb{T}_c$ *et* $\mathrm{V}, \mathrm{C}, \mathsf{pres}$ *correspondent aux mêmes éléments que pour une action (Def. 5.2).*

Une tâche est une opération abstraite. Elle peut être décomposée en d'autres tâches ou en actions.

Un exemple d'une tâche et de sous-tâches la décomposant est donné dans Ex. 5.3.

Exemple 5.3 : Tâche et sous-tâches

identify–and–report(m_1) désigne l'opération consistant à identifier un MILCO définit par la constante m_1 et communiquer les informations d'identification à l'opérateur (Fig. 5.1). Elle peut être décomposée en plusieurs actions et tâches. Une décomposition possible est d'identifier puis rapporter les informations, c.-à-d. *identify*(r, m_1) et *report–identification*(m_1).

Les sous-tâches d'une tâche sont encapsulées dans un *réseau de tâches*. Les réseaux de tâches sont définis dans la Def. 5.4.

Definition 5.4 (Réseau de tâches)

Par $\mathtt{tn} = (\mathtt{L}, \mathbb{T}, \prec, \alpha)$*, nous désignons un réseau de tâches où :*

— $L \subset \mathbb{L}$ *est un ensemble de* ***labels****;*

— $\mathbb{T}$ *est un ensemble de* ***tâches*** *ou* ***actions*** *présents dans le réseau de tâches ;*

— $\prec$ *est un* ***ordre partiel strict*** *sur* $\mathtt{L}$ *;*

— $\alpha : \mathtt{L} \to \mathbb{T}$ *établit une* ***correspondance*** *des labels vers une tâche ou une action.*

Un réseau de tâches définit un ensemble de tâches ou actions pouvant être liées par des contraintes de précédence, formant ainsi un ensemble de tâches ou actions partiellement ordonné. *Comme chaque tâche ou action peut apparaître plus d'une fois, elles sont associées à un label unique.*

Enfin, une *méthode* permet de spécifier une décomposition possible en associant une tâche à un réseau de tâche. Les méthodes sont définies dans la Def. 5.5.

Definition 5.5 (Méthode)

Par $\mathtt{M} = (\mathtt{T}_c, \mathtt{tn}, \mathtt{pres})$, *nous désignons une méthode. Cette méthode décrit qu'une façon de décomposer la tâche* $\mathtt{T}_c$ *est d'appliquer le réseau de tâches* $\mathtt{tn}$ *si les préconditions* $\mathtt{pres}$ *sont réunies.*

Un exemple de méthode est illustré dans Ex. 5.4.

Exemple 5.4 : Instance d'une méthode

En reprenant les éléments de Ex. 5.3, soient la tâche $\mathtt{T}_c$ correspondant à *identify–and–report*(m_1) et $\mathtt{tn}$ le réseau de tâches correspondant à l'application des tâches *identify*(r, m_1) et *report–identification*(m_1). Alors la méthode *identify–then–report*(m_1) définie comme $\mathtt{M} = (\mathtt{T}_c, \mathtt{tn}, \varnothing)$ décompose la tâche *identify–and–report*(m_1) par l'application du réseau de tâches $\mathtt{tn}$ comportant *identify*(r, m_1) et *report–identification*(m_1).

Les éléments introduits jusqu'ici sont encapsulés au sein d'un *domaine* et d'un *problème* permettant de regrouper les informations propres au problème de planification hiérarchique à résoudre. Domaine et problème sont respectivement définis dans les Def. 5.6 et Def. 5.7.

Definition 5.6 (Domaine de planification)

Par $\mathcal{D} = (\mathcal{L}, \mathcal{T}_p, \mathcal{T}_c, \mathcal{M})$, *nous désignons un domaine de planification où :*

- $\mathcal{L}$ *est le langage sous-jacent ;*
- $\mathcal{T}_p$ *est un ensemble d'actions (ou tâches primitives) ;*
- $\mathcal{T}_c$ *est un ensemble de tâches (ou tâches complexes) ;*
- $\mathcal{M}$ *est l'ensemble des méthodes de décomposition des tâches complexes.*

$\mathcal{D}$ *regroupe l'ensemble des tâches, des actions, et des méthodes disponibles, indépendamment du problème à résoudre.*

Definition 5.7 (Problème de planification)

Par $\mathcal{P} = (\mathcal{D}, \mathtt{s_i}, \mathtt{tn_i})$, *nous désignons un problème de planification où :*

- $\mathcal{D}$ *est le domaine sous-jacent ;*
- $\mathtt{s_i}$ *est l'état initial, c.-à-d. un ensemble d'effets positifs qui initialise les prédicats ;*
- $\mathtt{tn_i}$ *est le réseau de tâches initial. Il représente les tâches et/ou actions qui doivent être effectuées pour résoudre le problème.*

Cette structure représente le problème de planification à résoudre.

Résoudre un problème $\mathcal{P} = (\mathcal{D}, \mathtt{s_i}, \mathtt{tn_i})$ consiste à trouver un *réseau de tâches solution* $\mathtt{tn}$ tel que $\mathtt{tn}$ est *primitif* et *exécutable* dans $\mathtt{s_i}$, c.-à-d. qu'il existe une séquence des tâches de $\mathtt{tn}$, qui respecte les contraintes d'ordonnancement, dans laquelle les préconditions d'une tâche sont valides dans l'état résultant de l'application de la tâche précédente.

5.3 FORMALISME HDDL

Le formalisme HTN ne permet pas à un opérateur de modéliser aisément le problème. Il est difficile à manipuler et il serait fastidieux à un opérateur de l'utiliser pour modéliser les connaissances du système. Pour pallier ce défaut, nous utilisons le formalisme HDDL présenté dans [Höl+20].

Ce formalisme est un langage de modélisation de problème de planification hiérarchique. On y retrouve donc une syntaxe particulière correspondant à l'écriture des éléments définis pour le formalisme HTN.

Nous utilisons le formalisme HDDL pour modéliser les décompositions hiérarchiques des données d'entrée de notre problème.

Dans cette section, nous décrivons ce formalisme par quelques exemples et nous renvoyons le lecteur à [Höl+20] pour plus de détail.

Par exemple, Lis. 5.1 est la représentation de l'action $move(r, from, to)$ en HDDL. Dans cette représentation sont définis le nom, la liste des paramètres avec leur type, et la liste des préconditions. Les adds et dels sont regroupés dans la section des effets et sont différenciés par le mot-clé *not*.

Le Lis. 5.2 représente la méthode *identify–then–report* (m_1) en HDDL . Les paramètres de la méthode sont l'union entre les paramètres de la tâche exécutée et les paramètres des sous-tâches. Le réseau de tâches est défini directement dans la méthode avec les sections `subtasks` et `ordering`. À chaque sous-tâche est associé un label unique qui permet de l'ordonner par rapport aux autres sous-tâches.

```
(:action move
    :parameters (?r - robot ?from ?to - location)
    :precondition (and
        (at ?r ?from)
    :effect (and
        (not (at ?r ?from))
        (at ?r ?to)
```

Listing 5.1 – Action $move(r, from, to)$ en HDDL.

```
(:method identify-then-report
    :parameters (?r - robot ?m - milco)
    :task (identify-and-report ?m)
    :subtasks (and
        (t1 (identify ?r ?m))
        (t2 (report ?r ?m))
    :ordering (and
        (< t1 t2))
```

Listing 5.2 – Méthode *identify–then–report* (m_1) en HDDL.

À un domaine HTN correspond un domaine HDDL. Un exemple de domaine HDDL est représenté dans le Lis. 5.3.

```
(define (domain identify-and-report-domain)
    (:requirements :typing :hierarchy)
    (:types
        locatable location - object
        robot milco - locatable
```

```
        identifier - robot)
    (:predicates
        (at ?u - locatable ?l - location))
    (:task identify-and-report :parameters (?m - milco))
    (:method identify-then-report [...])
    [...]
    (:action identify
        :parameters (?r - identifer ?m - milco ?l - location)
        :precondition (and
            (at ?r ?l)
            (at ?m ?l))
        :effect (and
            (identified ?m))))
```

Listing 5.3 – Exemple de domaine en HDDL. Par souci de clarté certains éléments sont éludés et remplacés par le symbole `[...]`.

Pour définir le problème à résoudre, un domaine HDDL est spécifié. Il comporte les informations sur le réseau de tâches à accomplir, l'état initial, et les contraintes à considérer. En HDDL, il est également possible de déclarer des constantes dans le problème (et non pas uniquement dans le domaine) via le champ `objects`. Un exemple de domaine HDDL est représenté dans le Lis. 5.4.

```
(define (problem identify-and-report-problem)
    (:domain identify-and-report-domain)
    (:objects
        AUVi1 - identifier
        milco-1 - milco
        start-location milco-1-location - location)
    (:htn
        :subtasks (and
            (mr-root-task (identify-then-report milco-1)))
        :ordering () ;No ordering specified
        :constraints ())) ;No constraints specified
    (:init
        (at AUVi1 start-location)
        (at milco-1 milco-1-location)))
```

Listing 5.4 – Exemple de problème en HDDL. Ici, le réseau de tâche à accomplir est composé d'une unique tâche *identify–and–report*(*milco*–1)

Une solution au problème de planification décrit ici peut être trouvée en faisant appel à un solveur hiérarchique acceptant en entrée le formalisme HDDL.

5.4 EXTENSION TEMPORELLE DES FORMALISMES HTN ET HDDL

Dans les formalismes HTN et HDDL présentés, les actions, les effets, les préconditions, et les contraintes sont supposés instantanés.

Avec de tels formalismes, il n'est pas possible de modéliser des problèmes incluant un cadre temporel quantitatif, p. ex. il n'est pas possible de formuler une contrainte de synchronicité. Ce point est illustré dans Ex. 5.5.

Exemple 5.5 : Limite temporelle des formalismes HTN et HDDL

Écrire un problème "Couvrir la zone z_x avant la zone z_y avec $AUVe1$" est possible mais pas "Couvrir la zone z_x avec $AUVe1$ avant l'échéance t_d et faire rencontrer $AUVe1$ et $AUVe2$ à l'instant t_m".
Ce second problème nécessite la synchronisation de la présence des deux robots en un lieu et impose une date limite pour la fin de l'exécution de la couverture de z_x.

Notre mission de chasse aux mines comportant des contraintes temporelles, il est important de pouvoir spécifier un cadre temporel à nos domaines et problèmes.

Cette section montre comment les formalismes HTN et HDDL, présentés respectivement aux Sec. 5.2 et Sec. 5.3, ont été étendus pour traiter de telles contraintes temporelles. Cette extension a été le principal sujet du stage de Roland Godet. Il est à noter que, durant l'établissement de nos propres formalismes HTN et HDDL temporels, certains travaux, comme [Bit+20 ; Pel+22], ont exploré l'intégration d'un cadre temporel aux problèmes de planification hiérarchiques. Cependant, aucun consensus n'a été établi. Aussi, l'extension que nous proposons ici n'est pas conçue dans l'objectif de proposer une nouvelle référence au sein de la littérature scientifique, mais uniquement de servir notre cas d'usage.

Comme le HDDL permettant de représenter des problèmes hiérarchiques est inspiré du Planning Domain Description Language (PDDL), la syntaxe que nous proposons pour inclure un cadre temporel au formalisme HDDL s'inspire du PDDL2.1 [FL03] qui est lui-même une extension temporelle du PDDL. De plus, dans cette extension nous ajoutons également le concept de *variable d'état* afin de représenter plus aisément qu'avec des prédicats les valeurs que peut prendre un attribut de l'environnement.

5.4.1 *Actions temporelles*

Afin de gérer le temps, le type *timepoint* est ajouté à notre formalisme HTN. Il s'agit d'un ensemble de variables ou de constantes discrètes régulièrement espacées et représentant des temps absolus. Il est donc possible d'appliquer les opérations classiques d'addition, de soustraction et d'ordonnancement entre elles.

Les actions sont étendues avec trois nouveaux attributs : start, end, et constraints. Avec ces attributs elles deviennent des actions temporelles ("durative actions").

start et end représentent les instants *start* et *end* auxquels l'action est active. Il s'agit de variables ou de constantes de type *timepoint*. Dans ce manuscrit, quand l'intervalle d'une tâche doit être explicite nous notons la tâche comme ceci : "$[\mathsf{start}, \mathsf{end}]task\text{–}symbol(parameters)$".

constraints est un ensemble de *contraintes* entre les variables et les constantes de l'action. Par exemple, il est possible de spécifier une durée de 5 unités de temps à l'action $identify(r, m)$ avec la contrainte $\mathsf{end} = \mathsf{start} + 5$.

Il est également possible de créer des *délais*. Par exemple, si l'action $identify(r, m)$ doit être effectuée avant le temps 8, la contrainte correspondante sera $\mathsf{end} \leq 8$. Cette contrainte est ajoutée dans le champ constraints de la tâche.

La définition d'une action temporelle est donnée dans Def. 5.8.

Definition 5.8 (Action temporelle)

Par $\mathtt{T}_p = (\mathcal{S}, \mathrm{V}, \mathrm{C}, \mathsf{pres}, \mathsf{adds}, \mathsf{dels}, \mathsf{start}, \mathsf{end}, \mathsf{constraints})$, *nous désignons une action temporelle où :*

- $\mathcal{S}, \mathrm{V}, \mathrm{C}, \mathsf{pres}, \mathsf{adds}$ *et* dels *correspondent aux éléments introduits dans la définition Def. 5.2 ;*
- start *est l'instant de début de l'action, c'est une variable ou une constante de type timepoint ;*
- end *est l'instant de fin de l'action, c'est une variable ou une constante de type timepoint ;*
- $\mathsf{constraints}$ *est un ensemble de contraintes ne pouvant porter que sur les éléments composant la tâche.*

Les variables et constantes de type timepoint utilisées dans la tâche sont intégrés à V *et* C*, cependant ils ne sont pas explicitement représentés dans ce manuscrit par souci de simplicité.*

Afin de conserver le formalisme HDDL comme langage d'entrée, nous l'avons étendu avec les nouveaux attributs comme illustrés dans le Lis. 5.5 où l'action *identify*(r, m) a une durée de 5 unités de temps.

```
(:durative-action identify
    :parameters (?r - robot ?m - milco ?l - location)
    :constraints (= end (+ start 5))
    :precondition ([...])
    :effect ([...])
```

Listing 5.5 – Action temporelle *identify*(r, m). Ici, une durée minimum de 5 unités de temps a été spécifiée.

5.4.2 *Conditions et effets temporels*

Pour une modélisation plus complexe des effets et conditions (jusqu'ici appelées préconditions), nous associons des *variables d'état* aux conditions et aux effets plutôt que de simples prédicats.

Une variable d'état décrit un attribut spécifique de l'environnement, p. ex. $loc(r)$ dénote l'emplacement actuel du robot r. Les valeurs que peut prendre une variable d'état sont un ensemble fini. Par exemple, soit n localisations possibles, alors $loc(r) \in \{ l_1, \ldots, l_n \}$. Les prédicats sont un ensemble de variables d'état dont les valeurs possibles sont booléennes. Ainsi, plutôt que d'écrire $at(r, l_x)$, et implicitement avoir $n-1$ prédicats $(not(at(r, l_y)))$ (pour $y \in \{ l_1, \ldots, l_n \} \setminus \{ l_x \}$), on écrit uniquement $loc(r) = l_x$.

Une condition est maintenant définie par l'association d'un intervalle temporel, d'une variable d'état, et de la valeur souhaitée de la variable d'état. Elle est de la forme $[\tau_s, \tau_e] var(p_1, \ldots, p_n) = v$, c.-à-d. la variable d'état $var(p_1, \ldots, p_n)$ doit avoir la valeur v de τ_s à τ_e.

De la même manière, un effet est défini par l'association d'un intervalle temporel, d'une variable d'état, et de la nouvelle valeur à prendre pour la variable d'état. Il est de la forme $[\tau_s, \tau_e] var(p_1, \ldots, p_n) \leftarrow v$, c.-à-d. la variable d'état $var(p_1, \ldots, p_n)$ prend la valeur v au temps τ_e. Sur $]\tau_s, \tau_e[$, la variable d'état est dîte en transition de sa valeur précédente à v, durant cette période elle a une valeur indéterminée.

En général, l'intervalle temporel $[\tau_s, \tau_e]$ est exprimé en fonction du start et du end de l'opération associée.

Pour les conditions et les effets, il est possible de retrouver le comportement par défaut (c.-à-d. comportement instantané) avec $\tau_s = \tau_e$.

Nous avons étendu le formalisme HDDL pour inclure la notion d'intervalles temporels. De plus, des intervalles temporels couramment utilisés ont été définis pour les mots-clés suivants :

- at start : $[\mathsf{start}, \mathsf{start}]$

— at end : [end, end]

— after : [end, end + *timestep*]

— over all : [start, end]

L'adaptation de l'action *identify*(r, m) en intégrant ces notions de conditions et d'effets temporisés est représentée dans le Lis. 5.6. Il est à noter qu'avec la comparaison des variables d'état *loc*(r) et *loc*(m), le paramètre ?*l* n'est plus nécessaire.

```
(:durative-action identify
    :parameters (?r - robot ?m - milco)
    :constraints (= end (+ start 5))
    :condition (and
        (at start(= (loc ?r) (loc ?m))))
    :effect (and
        (at end (= (identified ?m))))
```

Listing 5.6 – Action temporelle *identify*(r, m) avec conditions et effets temporisés en HDDL.

5.4.3 *Contraintes temporelles*

Les ajouts présentés précédemment nous permettent de modéliser des opérations plus complexes incluant une notion de temporalité. Des derniers ajouts sont nécessaires pour pouvoir spécifier des contraintes entre différentes tâches afin de pouvoir les organiser et les synchroniser au besoin. Cela, dans le but d'encoder les contraintes d'une mission de chasse aux mines, et de manière plus générale, celles des intervalles d'Allen présentés à la Sec. 2.5.4.

Afin de pouvoir représenter les relations d'Allen, les attributs start, end, et constraints sont également ajoutés aux tâches. De plus, l'ordre partiel strict $\prec$ des réseaux de tâches est remplacé par un ensemble de contraintes temporelles entre les sous-tâches du réseau de tâches.

La déclaration de la méthode *identify–live*(m), qui inclue deux contraintes de synchronicité, est présentée dans le Lis. 5.7.

```
(:durative-method identify-live
    :parameters ( [...] )
    :task (identify-and-report ?m)
    :condition ()
    :subtasks (and
        (t1 (identify ?r1 ?m))
        (t2 (communication-relay ?r2 ?l)))
    :constraints (and
        (= t1.start t2.start)
        (= t1.end t2.end)))
```

Listing 5.7 – Exemple de contraintes temporelles entre sous-tâches. Ici, les deux sous-tâches doivent être exécutées en même temps.

5.4.4 *Résoudre des problèmes hiérarchiques incluant des contraintes temporelles*

Les extensions présentées dans cette section permettent de modéliser des actions et des tâches avec des contraintes temporelles portant sur leurs intervalles d'application mais également ceux de leurs conditions, effets et contraintes. Grâce à ces ajouts il est possible d'encoder les contraintes de la chasse aux mines.

Cependant, pour exploiter ces ajouts il faut utiliser un solveur capable de prendre en compte des problèmes de planification hiérarchique incluant des contraintes temporelles. Si plusieurs solveurs de problèmes de planification hiérarchique existent [1], comme PANDA [Höl+21], très peu sont capables de supporter nos besoins.

Dans nos travaux, nous utilisons le solveur LCP du projet Aries [2]. Ce solveur est capable de prendre en compte les aspects temporels et hiérarchiques de notre formalisme.

Le fonctionnement interne de LCP exploite en entrée des "chroniques"[GB22]. Ces chroniques sont des structures de données représentant des problèmes de planification hiérarchique incluant un cadre temporel. Les chroniques sont ensuite converties par le solveur en un problème de satisfaction de contraintes (Constraint Satisfaction Problem (CSP)).

Afin de travailler avec ce solveur, nous avons créé un convertisseur [3] faisant le lien entre notre formalisme HTN et le formalisme des chroniques. Le formalisme HDDL n'est donc utilisé qu'en entrée de la modélisation de nos problèmes, et non pas en entrée du planificateur.

À des fins d'illustration, la spécification HDDL d'une mission est donnée en An. A pour la partie multirobot et An. B pour les parties locales. Ces fichiers de spécifications sont par la suite utilisés pour évaluer l'approche dans le Chap. 8.

1. Ces solveurs ont notamment été mis en avant lors de l'édition 2020 de la Compétition Internationale de Planification
2. https ://github.com/plaans/aries
3. https ://github.com/Shi-Raida/temporal-htn (travail en cours)

6

PRENDRE DES DÉCISIONS AVEC DES ENCHÈRES ET DES RÉSEAUX DE TÂCHES HIÉRARCHIQUES

Aperçu

Dans notre approche, une enchère correspond à un problème de MRTA à résoudre à un instant donné. Ce problème prend en entrée un ensemble de connaissances multirobots et locales disponible à l'instant considéré. Les tâches en vente sont des objectifs de mission et la tenue d'une enchère permet de trouver un ensemble d'allocations permettant l'accomplissement de ces objectifs par les robots. Dans ce chapitre, nous formalisons le protocole de décision de HaucTioN permettant de résoudre un problème de MRTA. Nous présentons en premier lieu les données d'entrées du problème. Puis, nous décrivons le Multi-Robot HTN, la structure hiérarchique au coeur de nos protocoles. Enfin, nous détaillons les étapes de notre protocole d'enchère qui repose sur cette structure et la planification hiérarchique.

6.1 DONNÉES D'ENTRÉES DU PROBLÈME DE DÉCISION

Notre schéma d'allocation décentralisé vise à résoudre un problème de MRTA soumis par un robot à ses coéquipiers, dans notre cas un commissaire-priseur à des enchérisseurs. Ce problème est composé d'objectifs devant être alloués aux différents robots participant à l'enchère. Au sein de ce schéma, plusieurs prérequis sont nécessaires :

1. Le commissaire-priseur doit être capable de **construire** le problème, c.-à-d. déterminer les tâches à allouer et les **créer** si besoin ;
2. Les enchérisseurs doivent être aptes à **comprendre** les tâches en vente et **comment** ils peuvent les accomplir afin de les estimer.

Ainsi, pour que les agents puissent construire et résoudre ce problème de MRTA il est fondamental d'inclure au sein du système certaines connaissances. Ces connaissances doivent permettre au commissaire-priseur d'interpréter les informations reçues afin de pouvoir construire des tâches qui seront exploitables par lui et ses coéquipiers. C'est également à l'aide de ces connaissances qu'un enchérisseur doit comprendre comment il peut accomplir la tâche en vente.

Comme présenté à la Sec. 4.4, nous établissons une distinction entre les connaissances multirobots, connues par tous les agents, et les connaissances locales, propres à chaque agent. En raison de la nature dynamique de la mission, ces connaissances sont amenées à évoluer. Cependant, une certaine partie de ces connaissances multirobots et locales sont statiques. Ce sont elles qui permettent aux robots de construire et résoudre le problème de MRTA.

Ces connaissances statiques constituent une bibliothèque d'information utilisable par les robots pour répondre à une situation à un instant donné. Dans cette section, nous détaillons leur construction et intégration au sein du système.

6.1.1 *Objectifs multirobots*

Les connaissances statiques multirobots servent avant tout à spécifier la mission de chasse aux mines, c.-à-d. traduire la mission en des tâches pouvant être accomplies. En effet, si la finalité d'une mission de chasse aux mines est "Assurer qu'il n'y plus de mines dans la zone considérée", une manière pour le système multirobot d'atteindre cette finalité est de la convertir en un ensemble d'objectifs devant être réalisés, comme "couvrir cette partie de la zone" et "aller identifier cet objet suspect".

Dans notre approche, nous encodons au sein des robots des définitions génériques des tâches pouvant être accomplies durant une mission de chasse aux mines. Par exemple, qu'est-ce qu'une tâche de couverture, d'identification, de relais de communication ? Ces définitions génériques peuvent ensuite être instanciées par les robots afin de résoudre une situation précise. Enfin, comme justifié dans la Sec. 4.2, nous ajoutons une structure hiérarchique abstraite aux tâches à accomplir.

Dans les sous-sections suivantes, nous décrivons comment de tels modèles hiérarchiques peuvent être construits pour représenter la mission à l'aide d'objectifs de couverture. Puis nous généralisons notre raisonnement afin de montrer comment les autres objectifs de la mission peuvent être décomposés.

6.1.2 *Découpe initiale de la zone de mission et création des tâches de couverture*

La mission de chasse aux mines consiste en premier lieu en une mission de couverture. Une manière simple de couvrir efficacement la zone de mission est de pouvoir la diviser afin d'en attribuer des parties à nos différents robots.

Or, conformément aux hypothèses établies à la Sec. 2.6.2, nous considérons que la zone à couvrir est partiellement connue. Nous nous appuyons sur ces connaissances pour proposer une **découpe hiérarchique** de la zone de mission. Cette découpe est établie par l'opérateur, éventuellement assisté d'outils dédiés, sur des critères comme la nature des fonds attendus ou la direction du courant sous-marin. Par exemple, la couverture d'une zone sera facilitée par une forme géométrique simple, telle qu'un rectangle, et suivant le sens du courant.

Une représentation possible d'une découpe hiérarchique de la zone de mission est d'utiliser un arbre composé de noeuds ET et OU. Nous illustrons l'arbre résultant d'une telle découpe dans Ex. 6.1.

À partir de cet arbre de découpe et du formalisme HTN présenté au Chap. 5, l'opérateur, grâce à des connaissances expertes, peut représenter la couverture de la zone initiale comme un HTN de tâches de couvertures. Ces tâches de couvertures peuvent ensuite être transmises au système comme des tâches à accomplir. Lors de ce processus, l'opérateur peut également spécifier des contraintes entre les tâches. Une illustration d'un HTN de tâches de couverture est représentée dans Ex. 6.2.

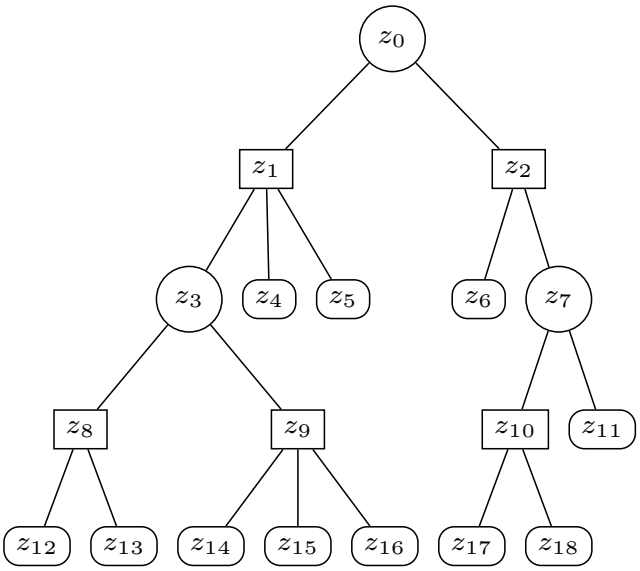
6.1.3 *Décomposition des autres tâches*

À l'instar des tâches de couvertures, il est possible de décomposer les autres tâches de la chasse aux mines, comme l'identification ou le rapport d'information. Tout comme pour les tâches de couverture, c'est à l'opérateur d'établir ces modèles. La spécification de ces modèles doit se faire à l'aide de connaissances expertes du domaine afin d'obtenir des décompositions exploitables par le système.

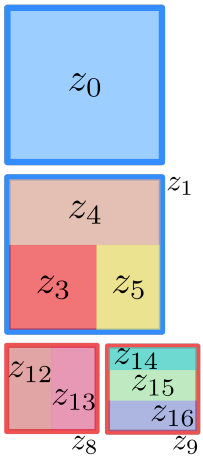
Pour notre mission de chasse aux mines nous considérons différentes décompositions possibles qui peuvent permettre aux robots d'accomplir leurs objectifs. Par exemple, dans le cas d'une tâche d'identification, une manière de l'effectuer est d'identifier puis de rapporter les données à l'opérateur ou

Exemple 6.1 : Découpe de la zone de mission

La zone de mission est appelée z_0. Cette zone peut être décomposée en sous-zones, plusieurs décompositions sont possibles. La Fig. 6.1 montre un arbre de noeuds ET et OU correspondant aux décompositions possibles de la zone de mission. La Fig. 6.1b illustre certaines de ces décompositions.



(a) Arbre des décompositions possibles de la zone de mission. Les noeuds rectangulaires et circulaires correspondent respectivement à des noeuds ET et OU. Les noeuds rectangulaires et arrondis sont des noeuds ne pouvant être décomposés.



(b) Illustration des décompositions de la moitié gauche de l'arbre.

FIGURE 6.1 – Représentation des décompositions de la zone de mission.

bien d'identifier tout en transmettant en direct les données. Afin de rapporter les données, il est possible pour un robot de remonter à la surface, ou de faire appel à un coéquipier agissant comme une mule de données, ou encore d'optimiser l'utilité de sa remontée en agissant lui-même comme une mule de données pour ses coéquipiers. Ces propositions sont autant de décompositions possibles pour la tâche à accomplir (Fig. 5.1).

Il est également possible pour l'opérateur d'ajouter des contraintes au sein des modèles de tâches établis. Par exemple, il peut spécifier qu'un objectif d'identification est urgent en le contraignant à être accompli au plus tard 10 unités de temps après sa création.

Notre approche se voulant générique, les modèles utilisés ne sont que des instanciations possibles des problèmes à résoudre. La pertinence de ces modélisations est à évaluer empiriquement et à discuter avec des opérateurs experts du domaine. Aussi, les modèles proposés ici n'ont pas pour but d'établir une référence, mais uniquement d'illustrer l'utilisation de notre approche.

6.1.4 *Capacités locales par robot*

De la même manière que pour les tâches multirobots, l'opérateur peut définir le domaine local de chaque robot. Ce domaine local décrit les capacités d'un robot, c.-à-d. *comment* il peut accomplir une tâche définie dans le domaine multirobot. Par exemple, l'opérateur peut définir qu'une manière d'accomplir

Exemple 6.2 : Tâches de couverture

En reprenant la découpe présentée dans Ex. 6.1, la couverture de la zone de mission peut être encodée par le HTN suivant :

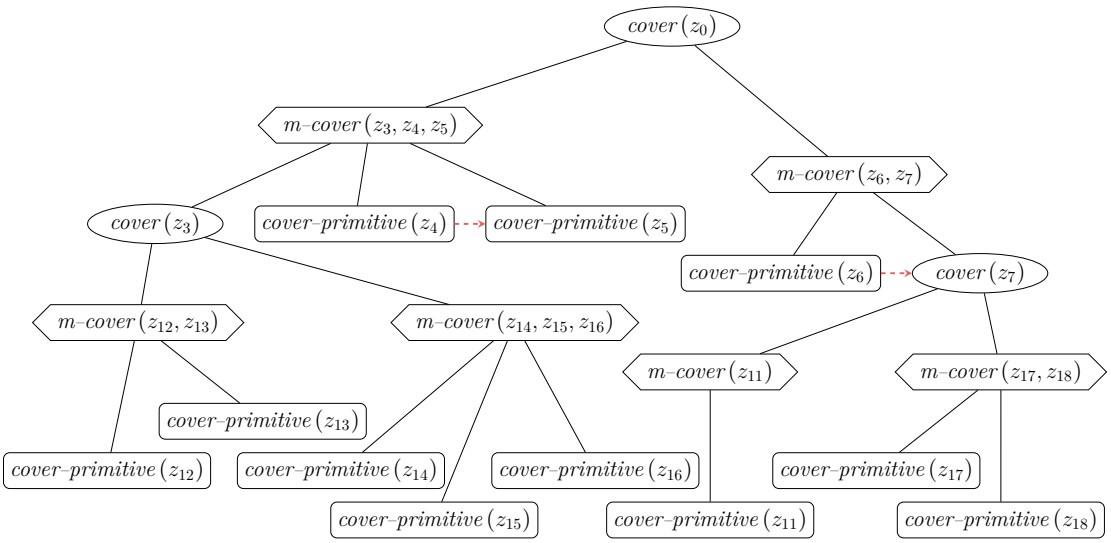


FIGURE 6.2 – HTN composés de tâches de couvertures. Les éllipses représentent des tâches abstraites, les hexagones des méthodes, et les rectangles des tâches primitives. Les flèches - -> représentent ces contraintes de précédence.

De plus, en raison de priorités strictes sur les zones à couvrir, l'opérateur impose que z_4 soit couverte avant z_5 ou que z_6 soit couverte avant z_7. Par la nature hiérarchique des découpes, les contraintes sont répercutées aux descendants des zones concernées, p. ex. z_{11} doit être couverte après z_6.

une tâche $cover(z_x)$ pour un robot r est de réaliser une série d'actions $move(r, from, to)$ sur la zone z_x. Ou de faire appel à un planificateur de trajectoire dédié, p. ex. de type boustrophédon. Ces capacités locales sont connues uniquement du robot et sont essentielles pour lui permettre d'estimer, exécuter et superviser les tâches multirobots.

Dans notre approche, nous regroupons ces compétences locales en fonction des unités spécialisées présentées à la Sec. 2.6.1. Par exemple, les agents *AUVe1* et *AUVi1* correspondent respectivement à un *Explorer* capable de décomposer une tâche de couverture et à un *Identifier* capable de décomposer une tâche d'identification.

6.1.5 *Spécification des décompositions initiales avec le formalisme HDDL*

Pour modéliser les décompositions en entrée des objectifs à accomplir, nous utilisons le formalisme HDDL présenté dans le Chap. 5. Grâce à ce formalisme, l'opérateur peut définir un domaine multirobot ainsi que des domaines locaux à chaque robot. Le domaine multirobot regroupe toutes les connaissances statiques que doivent connaître les robots sur les tâches à accomplir. Les domaines locaux encodent les capacités des robots pour déterminer comment ils peuvent accomplir une tâche. Un domaine multirobot et un domaine local sont respectivement représentés dans les Lis. 6.1 et Lis. 6.2.

Dans Lis. 6.1, l'opérateur encode la mission en spécifiant les décompositions des tâches multirobots à l'échelle du système. Par exemple, la tâche de couverture de z_0 peut être décomposée en tâches de couverture de z_3, z_4, et z_5. Et la couverture de z_4 doit précédée celle de z_5.

Dans Lis. 6.2, l'opérateur définit les capacités d'un robot de type *Explorer*. Le symbole de tâche *cover–primitive* est partagé avec le domaine multirobot, ainsi, l'opérateur peut spécifier que la réalisation d'une tâche de ce type consiste à exécuter une action locale de type *fl–cover–primitive*. Et le prédicat *free–token* permet de définir des tâches mutuellement exclusives, p. ex. le robot ne peut pas accomplir deux *fl–cover–primitive* en parallèle. L'opérateur spécifie également ce que le robot n'est pas capable de faire, cela permet de ne pas mettre en défaut les problèmes de planification utilisés dans l'architecture, p. ex. le robot est incapable d'accomplir une tâche de type *identify–primitive*.

```
(define (domain minehunting-mission)
    (:requirements :durative :typing :hierarchy
        )
    (:types
        zone milco - object)
    (:constants
        zone-0 zone-3 zone-4 - zone
        zone-5 zone-6 zone-7 - zone
        [...])
    (:predicates)
    (:task cover :parameters (?z - zone))
    (:task identify :parameters (?m - milco))
    (:durative-method m-cover-zone-0-1
        :parameters ()
        :task (cover zone-0)
        :subtasks (and
            (t1 (cover zone-3))
            (t2 (cover-primitive zone-4))
            (t3 (cover-primitive zone-5)))
        :constraint (and
            (< ?t2.end ?t3.start)))
    (:durative-action cover-primitive
        :parameters (?z - zone)
        :precondition ()
        :effect ()
        :constraint ())
    (:durative-method m-identify
        :parameters (?m - milco)
        :task (identify ?m)
        :subtasks (and
            (t1 (identify-primitive ?m))))
    (:durative-action identify-primitive
        :parameters (?m - milco)
        :precondition ()
        :effect ()
        :constraint (and
            (< end (+ start 30))))
            ;identify action must last less
                than 30 time units
```

Listing 6.1 – Exemple de domaine multirobot en HDDL.

```
(define (domain explorer-AUVe1-domain)
    [...] ;requirements
    (:types
        zone milco robot token - object
        explorer - robot)
    (:constants
        AUVe1 - explorer
        fl-token-AUVe1 - token)
    (:predicates
        ;used for mutally exclusive actions
        (free-token ?t - token))
    [...] ;task definition
    (:method m-cover-primitive
        :parameters (?z - zone ?r - explorer)
        :task (cover-primitive ?z)
        :ordered-subtasks (and
            (fl-cover-primitive ?r ?z)))
        ;"ordered-subtasks": define a strictly
            ordered task network
    (:durative-action fl-cover-primitive
        ;This action define a boustrophedon
            path
        :parameters (?r - explorer ?z - zone)
        :precondition (and
            (at start (free-token
                fl-token-AUVe1)))
        :effect (and
            (at start (not (free-token
                fl-token-AUVe1)))
            (after (free-token fl-token-AUVe1))
                ))
    (:method m-identify-primitive
        :parameters (?m - milco)
        :task (identify-primitive ?m)
        :ordered-subtasks (and
            (unable-to-do)))
    (:action unable-to-do)
    ;AUVe1 is unable to identify
```

Listing 6.2 – Exemple de domaine local en HDDL.

Nous considérons raisonnablement que tous les effets multirobots sont inclus dans la description du problème multirobot et que les actions locales n'ont pas d'impact sur eux. Cette hypothèse, séparant les parties multirobots des parties locales, nous permet de mettre en place des problèmes hiérarchiques cohérents et exploitables au sein d'un protocole d'enchère.

6.1.6 *Cas de la planification initiale*

En amont de la mission, un premier problème de MRTA est résolu afin de pouvoir déployer les robots avec un plan, c'est la *planification initiale*. Ce plan est ensuite réparé en ligne en fonction des aléas rencontrés. La planification initiale peut être réalisée en s'affranchissant de certaines contraintes techniques

telles que la complexité des calculs ou la disponibilité des communications entre robots. Le plus souvent, c'est une approche centralisée permettant d'obtenir une solution proche de l'optimale qui sera utilisée.

Par souci de simplicité, nous proposons de réaliser cette planification initiale à l'aide de notre approche. Notre protocole permet de résoudre de manière uniforme la planification initiale et les réparations en ligne. L'absence d'objectifs et de plans préexistants, propres à la planification initiale, nous permettent de décrire avec plus de clarté comment notre approche résout un problème de MRTA. Dans le chapitre suivant, nous détaillerons la prise de décision en ligne, c.-à-d. avec un plan préexistant.

Pour établir cette planification initiale, l'opérateur utilise une nouvelle fois le formalisme HDDL pour spécifier un problème multirobot. Ce problème multirobot permet de décrire les tâches devant être accomplies par les robots. Un exemple de problème multirobot initial est représenté dans le Lis. 6.3.

```
(define (problem initial-problem)
    (:domain minehunting-mission)
    (:objects)
    (:htn
        :subtasks (and
            (t1 (cover zone-0)))
        :constraints (and
            (< t1.end (+ t1.start 500))))
            ;mission must last less than 500 time units
    (:init))
```

Listing 6.3 – Exemple de problème multirobot en HDDL. Ici, l'objectif est de couvrir une zone.

6.2 LE MULTI-ROBOT HTN

Nous avons présenté dans la Sec. 6.1.1 comment la connaissance des tâches pouvant être accomplies par le système, c.-à-d. les tâches multirobot, était intégrée de manière statique au sein du système. Puis nous avons introduit comment ces modèles de tâches pouvaient être instanciés pour devenir des objectifs opérationnels concrets devant être réalisés par les robots.

Ces tâches sont regroupées sous forme de HTN et mises en vente par le biais d'enchères. Lorsque les robots enchérissent, ils enchérissent sur une instance spécifique d'une tâche. Cependant, comme le schéma de décision est décentralisé, nous devons nous assurer que les robots misent et achètent des instances de tâches identifiables, sans quoi des incohérences pourraient apparaitre. À l'instar de ce qui est fait dans un réseau de tâches au formalisme HTN, dans notre approche nous labélisons les tâches afin d'assurer leur unicité et surmonter ces problèmes.

Le rôle du **Multi-Robot HTN** est d'encoder les tâches à allouer sous la forme d'un problème HTN dont les tâches sont uniques et labélisées. Le Multi-Robot HTN est au coeur de tous les processus de notre architecture. Lors de la mission, les robots se réfèrent à leur Multi-Robot HTN durant les processus de décision, de supervision, et de réparation. Chaque robot met à jour son Multi-Robot HTN en fonction des décisions prises et, de manière plus générale, des informations reçues, p. ex. en y intégrant des contraintes ou des informations sur l'exécution d'une tâche.

Nous formalisons en détail le concept de Multi-Robot HTN dans les sous-sections suivantes. Nous détaillons d'abord le problème d'identification des tâches, puis nous définissons le Multi-Robot HTN. Enfin, nous présentons comment sont labélisées les tâches au sein d'une structure intermédiaire appelée Multi-Robot Graph qui est ensuite utilisée pour construire le Multi-Robot HTN.

6.2.1 *Problème d'identification des tâches*

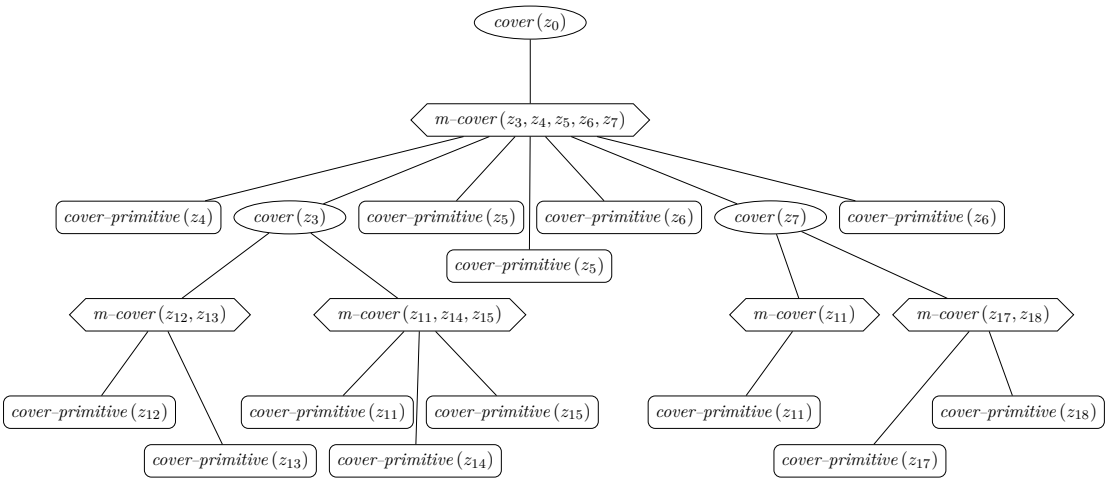
Tel que présenté dans la Chap. 5, les tâches d'une mission de chasse aux mines sont définies par une instance de paramètres, p. ex. $cover(z_x)$ est une instance de tâche correspondant à "Couvrir la zone x". Cependant, il est possible que le système soit amené à accomplir plusieurs fois une même instance de tâche. Par exemple, afin d'améliorer le niveau de confiance dans la couverture d'une zone, il peut être nécessaire de la couvrir deux fois, c.-à-d. l'opérateur souhaite que le système accomplisse deux fois $cover(z_x)$.

Dans notre schéma d'allocation décentralisé, les robots sont amenés à miser pour acquérir ces instances de tâches. Ils peuvent également ajouter des contraintes issues de leur plan. Ces contraintes doivent ensuite être globalement intégrées par le système afin que la suite des décisions reste cohérente. Or, le formalisme HTN ne permet pas de différencier ces instances. Nous illustrons ce point dans Ex. 6.3.

Exemple 6.3 : Problème d'identification de multiples instances d'une tâche

Soit la situation introduite dans Ex. 6.1 et Ex. 6.2 avec les trois changements suivant :

- z_0 se décompose en z_2 ET z_3 (remplacement du OU) ;
- z_{16} et z_{11} sont équivalentes, par conséquent nous ne conservons que la définition de z_{11} ;
- z_5 et z_6 sont de grande importance et doivent être couvertes deux fois.



En mettant directement en vente cet HTN, il n'est pas possible d'identifier les instances de tâches souhaitées par les robots. Par exemple, si *AUVe1* et *AUVe2* obtiennent chacun une instance de $cover(z_5)$ et $cover(z_6)$, il est possible que :

- *AUVe1* planifie : $cover(z_5)$ puis $cover(z_6)$;
- *AUVe2* planifie : $cover(z_6)$ puis $cover(z_5)$.

Rien ne s'oppose à la proposition de tels plans et les contraintes soumises par ces robots doivent être intégrées aux connaissances multirobots en vue de futures allocations.
De plus, il est implicite que $cover(z_{11})$ doit être effectuée deux fois. Cela mène aux mêmes problèmes que pour $cover(z_5)$ et $cover(z_6)$ sans la connaissance explicite du nombre de couverture exigées par l'opérateur.

Afin d'identifier les tâches devant être accomplies par les robots nous les labélisons. Labéliser les instances des tâches permet de s'assurer que les robots raisonnent sur les mêmes éléments et que la mission est entièrement remplie. Ce sont ces tâches labélisées qui sont encodées dans le Multi-Robot HTN.

6.2.2 *Un HTN de tâches multirobots labélisées*

Grâce à cette labélisation, nous définissons une structure de données appelée le Multi-Robot HTN. Cette structure de donnée est un HTN de tâches labélisées devant être accomplies par les robots.

Formellement, le Multi-Robot HTN est basé sur un problème au formalisme HTN (Def. 5.7). Au sein du Multi-Robot HTN, nous distinguons les tâches *virtuelles*[1] des tâches *opérationnelles*. Les tâches opérationnelles correspondent aux tâches multirobots définies par l'opérateur, p. ex. $cover(z_x)$. Les tâches virtuelles ne sont qu'une représentation des tâches opérationnelles après labélisation. **Toutes les tâches virtuelles comportent en paramètre un label unique permettant de les identifier**, ces labels sont appelés *labels multirobots*. Au sein de HaucTioN, toutes les informations échangées par les robots, que cela soit pour prendre des décisions ou pour transmettre une information sur l'exécution d'une tâche, se font à l'aide du label multirobot permettant d'identifier la tâche.

Comme à un label multirobot correspond une tâche opérationnelle, nous définissons une tâche virtuelle comme $mr\text{–}task(l)$, où l est un label unique, p. ex. $cover(z_x)$ devient $mr\text{–}task(l)$. Le symbole *mr–task* est volontairement générique, il est utilisé pour les tâches virtuelles. Une définition formelle des tâches virtuelles est présentée dans Def. 6.1.

Definition 6.1 (Tâche virtuelle primitive et abstraite)

Soit l, un label multirobot, et $\mathtt{T}_{p\text{–}op} = (\mathcal{S}_{op}, \varnothing, \mathrm{C}_{op}, \mathsf{pres}_{op}, Adds_{op}, \mathsf{dels}_{op}, \mathsf{start}_{op}, \mathsf{end}_{op}, \mathsf{constraints}_{op})$, une action opérationnelle instanciée (c.-à-d. qui ne comporte pas de variables).

En s'appuyant sur le formalisme HTN présenté dans Chap. 5, nous définissons une tâche virtuelle primitive comme $\mathtt{T}_p = (\mathcal{S}, \varnothing, \mathrm{C}, \mathsf{pres}, \mathsf{adds}, \mathsf{dels}, \mathsf{start}, \mathsf{end}, \mathsf{constraints})$, où :

- *$\mathcal{S} = mr\text{–}task$, un symbole de tâche volontairement générique et utilisé pour les tâches virtuelles (primitives et abstraites) ;*
- *$\mathrm{C} = \{l\}$, c.-à-d. la tâche n'est paramétrée que par l ;*
- *$\mathsf{pres} = \mathsf{adds} = \mathsf{dels} = \varnothing$*
- *start est une variable de type timepoint représentant l'instant de début de la tâche ;*
- *end est une variable de type timepoint représentant l'instant de fin de la tâche ;*
- *$\mathsf{constraints}$ est l'ensemble de contraintes obtenu à partir de $\mathsf{constraints}_{op}$ après conversion de start_{op} vers start et end_{op} vers end, p. ex. si $(\mathsf{end}_{op} = \mathsf{start}_{op} + 5) \in \mathsf{constraints}_{op}$ alors $(\mathsf{end} = \mathsf{start} + 5) \in \mathsf{constraints}$.*

Lors de la résolution d'un problème de MRTA, des nouvelles contraintes peuvent être déterminées, ces contraintes sont encodées dans les champs pres, adds, dels, et $\mathsf{constraints}$. Par exemple, une contrainte causale peut être spécifiée en ajoutant un prédicat partagé dans le champ pres d'une première tâche et le champ adds d'une autre tâche.

Nous définissons une tâche virtuelle abstraite comme $\mathtt{T}_c = (\mathcal{S}, \varnothing, \mathrm{C}, \mathsf{pres}, \mathsf{start}, \mathsf{end}, \mathsf{constraints})$, où $\mathcal{S}, \mathrm{V}, \mathrm{C}, \mathsf{pres}, \mathsf{start}, \mathsf{end}, \mathsf{constraints}$ correspondent aux mêmes éléments que pour une tâche virtuelle primitive.

Enfin, nous notons $\mathrm{mr\text{–}task}(l)$ la tâche virtuelle (primitive ou abstraite) résultante.

Ces tâches virtuelles composent le Multi-Robot HTN, dont les propriétés sont définies dans Def. 6.2.

1. Attention : Les tâches virtuelles ne sont pas à confondre avec les tâches abstraites (également appelées complexes) du formalisme HTN. Si elles portent le nom de virtuelles, c'est uniquement parce que leur existence est une abstraction des tâches opérationnelles.

Definition 6.2 (Multi-Robot HTN)

Par $\mathcal{H} = (\mathcal{P}, \mathcal{D}_{op}, \mathtt{M}_t)$, nous désignons un Multi-Robot HTN où :

- $\mathcal{P} = (\mathcal{D}, \mathtt{s_i}, \mathtt{tn_i})$, *où :*
  - $\mathcal{D}$ *est un domaine regroupant les connaissances multirobots* ***virtuelles****. Le langage du domaine comporte notamment un ensemble fini de constantes de type mr–label appelé* $\mathtt{L}_{mr}$ *(ou labels multirobots) ;*
  - $\mathtt{s_i}$ *est l'état initial multirobot de la mission, c.-à-d. un ensemble d'effet initialisant les variables d'état multirobots ;*
  - $\mathtt{tn_i}$ *est le réseau de tâches initial Multi-Robot HTN. Il n'est composé que de sa racine.*
- $\mathcal{D}_{op}$ *est un domaine regroupant les connaissances multirobots* ***opérationnelles*** *;*
- $\mathtt{M}_t$ *est une fonction de correspondance des labels multirobots de* $\mathcal{D}$ *vers les tâches de* $\mathcal{D}_{op}$*, c.-à-d.* $\mathtt{M}_t : \mathtt{L}_{mr} \rightarrow \mathcal{T}_{p-op} \cup \mathcal{T}_{c-op}$.

Par définition, le domaine $\mathcal{D}$ *ne comporte que deux symboles de tâches, car* $\mathbb{T}_p = \{mr\text{–}task\}$ *et* $\mathbb{T}_c = \{mr\text{–}task\}$*. De plus, la fonction de correspondance* $\mathtt{M}_t$ *permet aux robots d'accéder à la définition opérationnelle des tâches lorsque nécessaire. Enfin, pour tout* $l \in \mathtt{L}_{mr}$*, il existe une instance de tâche* $mr\text{–}task(l) \in \mathcal{T}_p \cup \mathcal{T}_c$*. Par construction, une solution de* $\mathcal{P}$ *ne peut comporter au plus qu'une seule instance de tâche "*mr–task (l)*".*

Il est important de noter que le Multi-Robot HTN comporte l'entière décomposition de la mission, et ce qu'importe les choix retenus, c.-à-d. aucune décomposition n'est retirée lorsqu'une une décision est prise. Par exemple, si pour une même tâche plusieurs alternatives existent et que l'une d'entre elles a été retenue lors d'une allocation, alors les autres alternatives sont toujours présentes au sein du Multi-Robot HTN. La conservation de ces décompositions "caduques" au sein du Multi-Robot HTN offre notamment plus de possibilités pour la réparation en ligne.

6.2.3 *Construction du Multi-Robot HTN*

La construction d'un Multi-Robot HTN prend en entrée un problème au formalisme HDDL contenant des tâches opérationnelles. Ce problème, appelé *auction problem*, est noté $\mathtt{P}_{auc}$. Le passage du *auction problem* au Multi-Robot HTN comporte trois étapes clés :

1. La conversion de $\mathtt{P}_{auc}$ en un problème au formalisme HTN noté $\mathcal{P}_g$;
2. La labélisation des tâches opérationnelles contenues dans $\mathcal{P}_g$;
3. La création de $\mathcal{H} = (\mathcal{P}, \mathcal{D}_{op}, \mathtt{M}_t)$, un Multi-Robot HTN contenant uniquement des tâches virtuelles.

La première étape consiste à convertir $\mathtt{P}_{auc}$ en un problème HTN instancié noté $\mathcal{P}_g$. Un problème instancié ("grounded") est un problème où toutes les variables ont été remplacées par des constantes. Notre algorithme de labélisation ne peut être utilisé que sur des tâches dont les paramètres sont fixés, nous imposons donc l'utilisation de problèmes instanciés comme prérequis à la labélisation. La seconde étape consiste à labéliser les tâches de $\mathcal{P}_g$ et à encoder cette labélisation au sein d'une structure intermédiaire appelée le **Multi-Robot Graph**. Enfin, des tâches virtuelles sont créées à partir du Multi-Robot Graph afin de former le Multi-Robot HTN. La Fig. 6.3 résume ce protocole que nous formalisons dans les paragraphes suivants.

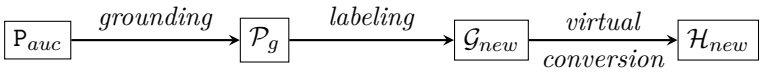


FIGURE 6.3 – Protocole de création d'un Multi-Robot HTN. Le problème HDDL $\mathtt{P}_{auc}$ est converti en un problème instancié $\mathcal{P}_g$. Puis $\mathcal{P}_g$ est labélisé pour créer $\mathcal{G}_{new}$, le Multi-Robot Graph. Enfin, le Multi-Robot HTN $\mathcal{H}_{new}$ est créé.

Définition du Multi-Robot Graph

Le Multi-Robot Graph nous sert à encoder nos tâches opérationnelles labélisées tout en conservant la structure hiérarchique du problème. Nous le définissons dans Def. 6.3.

Definition 6.3 (Multi-Robot Graph)

Par $\mathcal{G} = (V_T, V_M, E)$, nous désignons un Multi-Robot Graph où :

- *V_T est un ensemble de tâches labélisées ;*
- *V_M est un ensemble de méthodes labélisées ;*
- *E est un ensemble de paires dirigées reliant "une tâche vers une méthode" ou "une méthode vers une tâche".*

Le Multi-Robot Graph forme un graphe dirigé dont ***chaque sommet possède au plus un parent****.*

Du problème HDDL $\mathtt{P}_{auc}$ vers un problème HTN $\mathcal{P}_g$

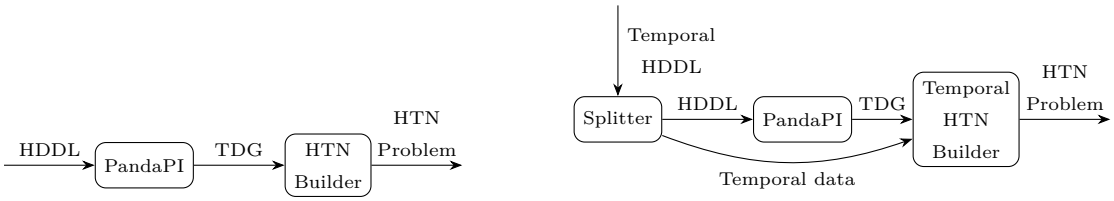
Le problème opérationnel devant être traité par notre architecture est donné au format HDDL, soit par un opérateur, soit par un robot. Ce problème opérationnel, appelé *auction problem*, comporte des tâches opérationnelles qui doivent être allouées aux robots afin d'être accomplies.

Ce problème est composé d'un réseau de tâche initial dont l'accomplissement est l'objectif. Ce réseau de tâche initial est instancié, c.-à-d. tous les paramètres de ses tâches sont des constantes. Mais les méthodes (et sous-tâches) le décomposant ne sont pas forcément instanciées. Leurs paramètres peuvent être des variables qui seront fixées si un solveur de problème hiérarchique est appelé. Cependant, afin de pouvoir allouer les tâches composant ce problème hiérarchique il est nécessaire de les labéliser, et donc de les instancier. Nous convertissons donc le *auction problem* en $\mathcal{P}_g$, un problème HTN instancié.

Dans notre approche, cette conversion est réalisée grâce à un "grounder" du nom de PandaPI. C'est un outil du solveur Panda [2] qui permet d'efficacement instancier un problème au formalisme HDDL. La sortie de PandaPI permet de reconstruire le Task Decomposition Graph (TDG) du problème. Le TDG est une représentation d'un problème de planification hiérarchique sous forme de graphe, il est définit dans [Ber+17]. À partir de ce TDG, nous construisons notre problème au formalisme HTN. Cependant, PandaPI ne supporte pas l'extension temporelle du formalisme HDDL, nous avons donc mis en place un processus spécifique permettant d'extraire les données temporelles d'un problème HDDL puis de les réintroduire dans un problème HTN.

Le processus complet de conversion est représenté dans la Fig. 6.4b, afin d'en refléter les particularités, le même processus de conversion sans données temporelles est représenté dans la Fig. 6.4a. Dans notre approche, le problème en entrée de ce processus est *auction problem* et le problème HTN en sortie est $\mathcal{P}_g$.

2. https ://panda-planner-dev.github.io/



(a) Sans données temporelles.

(b) Avec données temporelles.

FIGURE 6.4 – Processus de conversion d'un problème HDDL avec et sans données temporelles vers un problème HTN instancié. Les processus "Splitter" et "Temporal HTN Builder" permettent d'extraire les données temporelles d'un problème au formalisme HDDL et de les réintroduire dans un TDG pour former un problème au formalisme HTN.

Du problème HTN $\mathcal{P}_g$ vers le Multi-Robot Graph$\mathcal{G}_{new}$

Pour construire le Multi-Robot Graph nous utilisons le TDG correspondant à $\mathcal{P}_g$. Pour obtenir ce TDG il faut partir du réseau de tâche initial de $\mathcal{P}_g$ et appliquer systématiquement toutes les décompositions possibles. Par exemple, dans Ex. 6.3, la structure hiérarchique présentée correspond à un TDG.

Le Multi-Robot Graph est une structure représentant les tâches et méthodes du problème de manière similaire au TDG. Cependant, le TDG ne permet pas de différencier les multiples instances d'une tâche, alors que dans le Multi-Robot Graph chaque instance de tâche est étiquetée.

Nous notons ξ le TDG obtenu à partir de $\mathcal{P}_g$. L'algorithme de construction du Multi-Robot Graph, représenté dans Al. 1 parcourt en largeur les tâches de ξ depuis sa racine.

— Pour chaque tâche t du TDG, un sommet (l, t) est créé et ajouté aux sommets des tâches labélisées V_T du Multi-Robot Graph, avec l un label unique de type *mr–label* (Ligne 14) ;

— Pour chaque méthode $\mathtt{M}$ du TDG correspond à une méthode $m = (\mathtt{T}_c, \mathsf{tn}, \mathsf{pres})$, un sommet $l, \mathtt{M}$ est créé et ajouté aux sommets des méthodes labélisées V_M du Multi-Robot Graph, avec l le label de la tâche décomposée par la méthode (Ligne 10) ;

— Pour chaque liaison entre une tâche et une méthode (et inversement pour une méthode et une tâche), une arête e est créée et ajoutée aux arêtes E du Multi-Robot Graph(Ligne 15).

Pour pouvoir appliquer cet algorithme il faut que le TDG :

— soit instancié ("grounded"), c.-à-d. que tous les paramètres soient des constantes (et non pas des variables) ;

— ne comprennent pas de cycles, c.-à-d. qu'aucune méthode ne puisse renvoyer vers l'une de ses tâches ascendantes. Ce prérequis est vérifié grâce aux propriétés du TDG.

Du Multi-Robot Graph $\mathcal{G}_{new}$ vers le Multi-Robot HTN $\mathcal{H}_{new}$

Le Multi-Robot Graph fait le lien entre les tâches opérationnelles et les labels multirobots et regroupe toutes les informations nécessaires au problème de MRTA, cependant il ne prend pas la forme d'un problème HTN exploitable par un solveur. Le Multi-Robot Graph est donc converti afin de revenir à un problème au formalisme HTN.

Dans notre approche, le résultat de cette conversion est un Multi-Robot HTN, c.-à-d. un problème HTN avec des informations supplémentaires pour faire le lien avec les tâches opérationnelles.

Avec $\mathcal{G} = (V_T, V_M, E)$, un Multi-Robot Graph, et $\mathcal{H} = (\mathcal{P}, \mathcal{D}_{op}, \mathtt{M}_t)$, un Multi-Robot HTN, nous accomplissons cette conversion comme suit :

Algorithm 1: Algorithme de construction du Multi-Robot Graph

Input: ξ
Output: $\mathcal{G} = (V_T, V_M, E)$, *labels*

if ξ *has cycles* **then return** error;
Let X be an empty First-In-First-Out list
Let l_{top} be a new unique label
$V_T \leftarrow \{(l_{top}, top)\}$, where *top* is the root task of ξ
$labels \leftarrow \varnothing$; $V_M \leftarrow \varnothing$; $E \leftarrow \varnothing$
$X.push((l_{top}, top))$
while X *is not empty* **do**
  $(l, t) \leftarrow X.pop()$
  forall *method* $\mathtt{M} \in \xi$ *such that* $\mathtt{M} = (\mathtt{T}_c, \mathsf{tn}, \mathsf{pres})$ *, and* $\mathsf{tn} = (\mathtt{L}, \mathbb{T}, \prec, \alpha)$ **do**
    $V_M \leftarrow V_M \cup \{(l, \mathtt{M})\}$
    $E \leftarrow E \cup \{((l, t), (l, \mathtt{M}))\}$
    forall $u \in \mathtt{L}$ **do**
      Let v be a new unique label
      $V_T \leftarrow V_T \cup \{(v, \alpha(u))\}$
      $E \leftarrow E \cup \{((l, \mathtt{M}), (v, \alpha(u)))\}$
      $labels \leftarrow labels \cup \{l, v)\}$
      $X.push((v, \alpha(u)))$

— **Création des tâches virtuelles** : Pour chaque $v_t = (l, t_{op}) \in V_T$, une tâche virtuelle *mr–task*(l) est créée. Si t est une tâche primitive alors *mr–task*(l) est ajouté aux tâches primitives de $\mathcal{D}$, sinon aux tâches abstraites ;

— **Correspondance vers les tâches opérationnelles** : Pour chaque $v_t = (l, t_{op}) \in V_T$, t_{op} est intégrée à $\mathcal{D}_{op}$ et la correspondance $l \rightarrow t$ est enregistrée dans $\mathtt{M}_t$;

— **Initialisation des méthodes virtuelles** : Pour chaque $e = ((l, t_{op}), (l, m_{op})) \in E$, une méthode virtuelle $\mathtt{M} = (\mathtt{T}_c, \mathsf{tn}, \mathsf{pres})$ est créée et ajoutée aux méthodes de $\mathcal{D}$. Ici :

  — $\mathtt{T}_c$ est la tâche virtuelle correspondant à l ;

  — tn est un réseau de tâche vide ;

  — pres est l'ensemble de conditions de m.

  Puis, pour chaque $e = ((l_p, m_{op}), (l_c, t_{op})) \in E$:

  — La méthode virtuelle $\mathtt{M} = (\mathtt{T}_c, \mathsf{tn}, \mathsf{pres})$ correspondant à (l_p, m_{op}) est récupérée dans $\mathcal{D}$;

  — La tâche virtuelle correspondant à l_c est intégrée à tn.

— **Correspondance des méthodes opérationnelles** : Pour chaque $v_m = (l, m_{op}) \in V_M$, $m_{op} = (\mathtt{T}_{c–op}, \mathsf{tn}_{op}, \mathsf{pres}_{op})$ est intégrée à $\mathcal{D}_{op}$;

— **Application des contraintes des méthodes opérationnelles** : Pour chaque $v_m = (l, m_{op}) \in V_M$, $m_{op} = (\mathtt{T}_{c–op}, \mathsf{tn}_{op}, \mathsf{pres}_{op})$, les contraintes portant sur les tâches opérationnelles sont appliquées sur les tâches virtuelles, pour cela :

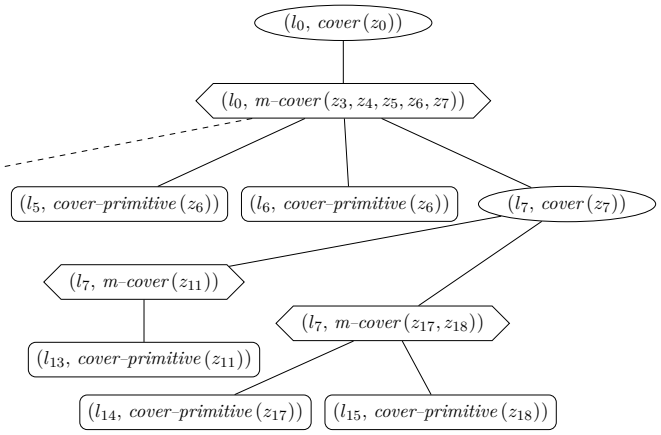
  — La méthode virtuelle $\mathtt{M} = (\mathtt{T}_c, \mathsf{tn}, \mathsf{pres})$ correspondant à (l_p, m_{op}) est récupérée dans $\mathcal{D}$;

  — Les contraintes contenues dans $\prec_{op}$ (de tn_{op}) sont ajoutées à celles de $\prec$ (de tn) après conversion des timepoints des tâches opérationnelles vers les tâches virtuelles correspondantes. Par exemple, soient $t_{x–op}$ et $t_{y–op}$ deux tâches opérationnelles correspondant respectivement à *mr–task*(l_a) et *mr–task*(l_b), si $(\mathsf{start}_{y–op} > \mathsf{end}_{x–op} + 5) \in \prec_{op}$ alors $(\mathsf{start}_{a–l} = \mathsf{start}_{b–l} + 5) \in \prec$.

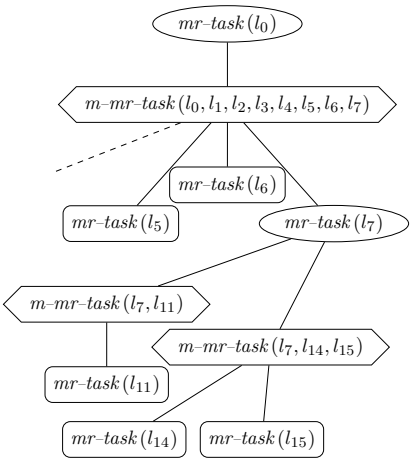
Un exemple de Multi-Robot Graph et de sa conversion en Multi-Robot HTN sont représentées dans Ex. 6.4. Il est à noter que la structure du Multi-Robot HTN est exactement la même que celle du Multi-Robot Graph.

Exemple 6.4 : Le Multi-Robot Graph

En reprenant la situation de Ex. 6.3, il est possible de construire le Multi-Robot Graph correspondant aux tâches de couverture, puis de le convertir en un Multi-Robot HTN. Des extraits du Multi-Robot Graph et du Multi-Robot HTN correspondant sont montrés respectivement dans Fig. 6.5a et Fig. 6.5b.



(a) Exemple de Multi-Robot Graph. Les tâches opérationnelles sont transformées en sommets labélisés.



(b) Exemple de Multi-Robot HTN. Les sommets labélisés sont remplacées par des virtuelles.

FIGURE 6.5 – Exemple de Multi-Robot Graph et Multi-Robot HTN. Par souci de clarté, la racine n'a pas été décomposée entièrement.

6.3 VUE D'ENSEMBLE DU PROTOCOLE DE DÉCISION

Le protocole de décision de HaucTioN est basé sur un schéma d'allocation par enchère, il en suit donc les étapes. Au sein d'un tour d'enchère, on retrouve les étapes présentées dans la Sec. 4.3.1. Ces étapes font intervenir les éléments introduits jusqu'ici et plus particulièrement le Multi-Robot HTN et les capacités locales des enchérisseurs. Dans cette section et sur la base de ces éléments, nous formalisons les différentes étapes de ce protocole.

Le protocole de décision est présenté dans sa globalité dans la Fig. 6.6, nous y reviendrons tout au long des sections suivantes afin d'en expliquer le contenu.

6.4 ANNONCE

La toute première étape d'un tour d'enchère consiste à transmettre un message d'annonce contenant des informations sur les objets en vente.

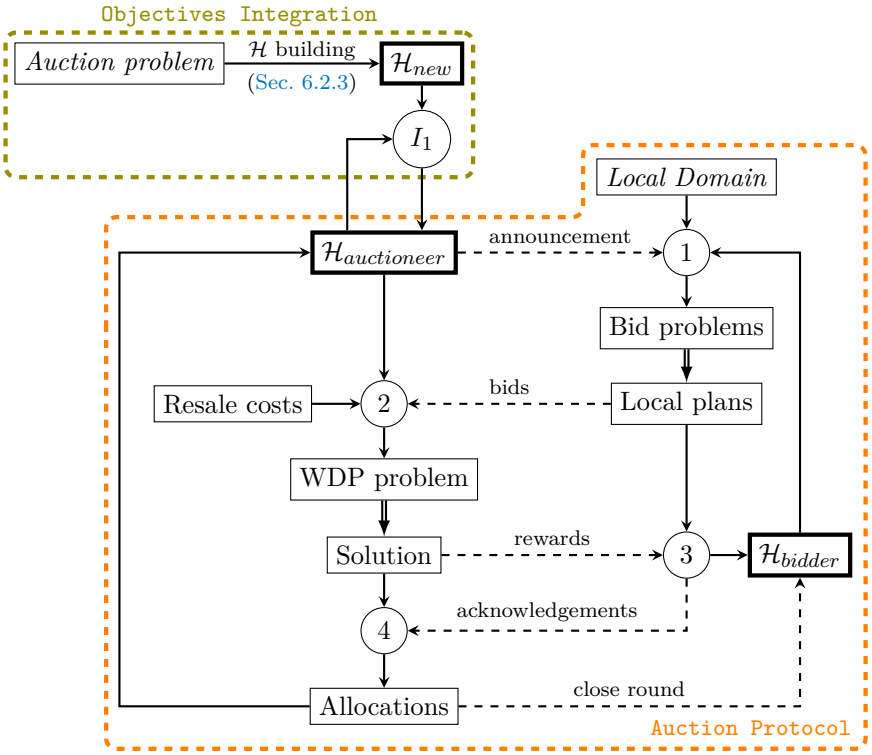


FIGURE 6.6 – Protocole de décision de HaucTioN. Les rectangles représentent des structures de données gérées par les robots. Les blocs de gauche sont gérés par le commissaire-priseur, les blocs de droite par chaque enchérisseur. Les flèches en pointillés représentent les données échangées entre le commissaire-priseur et les enchérisseurs. Les cercles indiquent les processus qui agrègent les informations pour construire de nouvelles structures. Les doubles flèches indiquent les appels à un planificateur HTN. Les flèches continues indiquent les interactions entre les structures de données et les processus.

6.4.1 *Détermination des labels disponibles à la vente*

Dans notre approche, n'importe quel agent peut ouvrir une enchère en devenant commissaire-priseur. Une enchère est automatiquement déclenchée quand un agent a connaissance de tâches à vendre. Les raisons qui peuvent causer l'ouverture d'une enchère sont :

— L'intégration de nouveaux objectifs, p. ex. suite à un aléa tel que la découverte d'un objet suspect. La création de ces objectifs correspond au bloc vert "Objectives Integration" dans la Fig. 6.6 ;

— Une réparation globale, l'agent doit revendre certaines de ses tâches.

Dans notre approche, la planification initiale est déclenchée comme "L'intégration de nouveaux objectifs" en utilisant un agent comme premier commissaire-priseur. Dans un tel cas, l'agent intègre ces objectifs à son propre Multi-Robot HTN grâce au protocole décrit dans Sec. 6.2 (processus I_1 dans Fig. 6.6). Au début de la mission, son Multi-Robot HTN est vide, il est donc initialisé par l'intégration des objectifs initiaux.

Quand un agent doit vendre des tâches, il est donc en possession :

— De son Multi-Robot HTN, qui contient notamment les instances des tâches concernées ainsi que toutes les informations connues par l'agent sur les tâches multirobots ;

— D'un ensemble de labels correspondant aux tâches à vendre, appelé *labels for sell*.

C'est avec ces deux informations qu'il peut ouvrir une enchère en diffusant un message d'annonce.

6.4.2 *Formalisation du message d'annonce*

Un message d'annonce signifie l'ouverture d'un tour d'enchère, que cela soit le premier tour de l'enchère concernée ou non. La contenu d'un message d'annonce, est défini dans Def. 6.4.

Definition 6.4 (Message d'annonce)

Un message d'annonce ("announcement") comporte les informations suivantes :

- auctioneer id *: L'identifiant du commissaire-priseur de l'enchère, p. ex. "*AUVe1*";*
- auction id *: L'identifiant de l'enchère, p. ex. "auction–*AUVe1*–*1*";*
- round id *: L'identifiant du tour de l'enchère, p. ex. "round–*1*";*
- t_a *: Une échéance imposée par le commissaire-priseur comme critère d'arrêt de l'estimation des mises, p. ex. "*$t = 100$*". Passé cette échéance, le commissaire-priseur commence la résolution du WDP et n'accepte pas de mises supplémentaires ;*
- δ *: L'objet mis en vente, appelé "item for sale".*

Le champ *item for sell* d'un message d'annonce contient la spécification du problème posé par le commissaire-priseur. Ce champ est défini dans Def. 6.5.

Definition 6.5 (Article en vente)

Par $\delta = (\mathcal{H}_\delta, \lambda_\delta)$*, nous désignons un article à vendre ("item for sale") où :*

- $\mathcal{H}_\delta$ *est le Multi-Robot HTN intégrant toutes les données du problème de planification hiérarchique des tâches multirobots virtuelles. Dans notre approche,* $\mathcal{H}_\delta$ *est toujours le Multi-Robot HTN du commissaire-priseur ;*
- λ_δ *est l'ensemble des labels des tâches en vente, c.-à-d. les* labels for sell.

L'état initial de $\mathcal{H}_\delta$*, c.-à-d.* $\mathsf{s_i}_{-\delta}$*, est déterminé par le commissaire-priseur selon les informations a sa disposition. Pour rappel, cet état initial ne porte que sur le domaine multirobot, il ne peut pas concerner des éléments locaux aux agents.*

6.5 ESTIMATION DES MISES

Une fois qu'un message d'annonce est reçu, l'enchérisseur doit calculer une mise pour chaque tâche labélisée réalisable parmi *labels for sell*, c.-à-d. chaque tâche exécutable par l'enchérisseur. La valeur de la mise correspond au coût d'exécution de cette tâche.

Par conséquent, pour chaque $\chi \in \lambda_\delta$, nous construisons un *problème d'estimation*, noté $\mathcal{P}_\chi$, que nous résolvons grâce à un solveur de problème hiérarchique. La solution du solveur est ensuite encodée au sein d'une mise qui est renvoyée au commissaire-priseur.

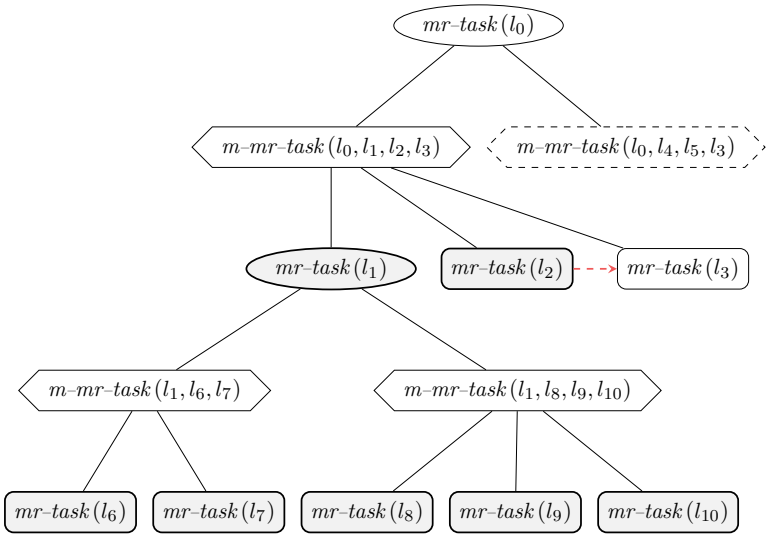
La création d'un problème d'estimation $\mathcal{P}_\chi$ correspond à l'étape ① de la Fig. 6.6. Pour formuler ce problème, nous suivons le processus suivant :

1. Obtention de $\mathcal{H}_\Gamma$, une version réduite de $\mathcal{H}_\delta$ où les décompositions alternatives à l'utilisation de la tâche correspondant à χ ou les tâches déjà obtenues par l'agent sont supprimées ;
2. Instanciation d'un problème local $\mathcal{P}_\sigma$ grâce aux éléments contenus dans $\mathcal{H}_\Gamma$;
3. Création d'un problème d'estimation $\mathcal{P}_\chi$;

Nous décrivons ci-après ces étapes.

Exemple 6.5 : Un article en vente

Soit δ l'article à vendre composé de $\mathcal{H}_\delta$, le Multi-Robot HTN représenté ci-dessous et $\lambda = (\, l_1, l_2, l_6, l_7, l_8, l_9, l_{10} \,)$ l'ensemble des labels en vente. Dans la figure, sont représentées en gris les tâches en vente. La contrainte --> est une contrainte de précédence.



6.5.1 *Réduction du problème multirobot aux labels à estimer - De $\mathcal{H}_\delta$ à $\mathcal{H}_\Gamma$:*

Dans notre approche, afin de tenir compte de l'ensemble des dépendances liant les tâches multirobots au sein d'un Multi-Robot HTN, nous résolvons nos problèmes hiérarchiques en utilisant la racine du Multi-Robot HTN comme réseau de tâches à accomplir. Cependant, dans le cas de l'estimation des mises, il est nécessaire de contraindre la planification à utiliser la décomposition dans laquelle se trouve la tâche que l'agent souhaite estimer. Sans cela, la planification d'une mise pourrait ignorer la décomposition qu'il faut estimer, car le Multi-Robot HTN comporte l'entière décomposition de la mission.

De plus, l'enchérisseur doit également tenir compte des labels qu'il a déjà obtenus, notés Φ. En effet, lors de l'estimation d'une mise un agent peut replanifier son plan local dans le respect des contraintes multirobots mais ne peut abandonner ses anciens engagements. Par conséquent, il est également nécessaire de contraindre la planification à utiliser les décompositions dans lesquelles se trouvent les tâches qu'un agent doit accomplir. Nous illustrons ces notions dans Ex. 6.6.

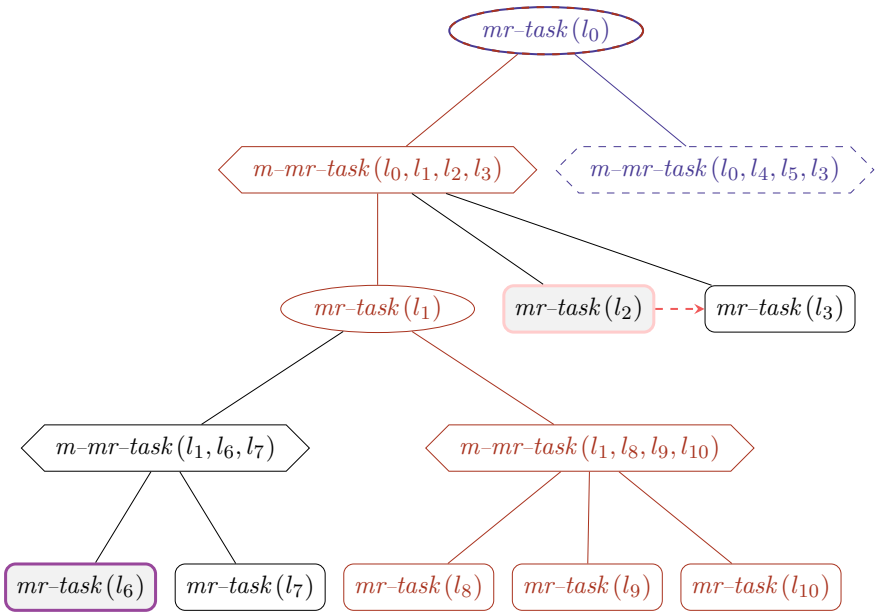
Enfin, il est nécessaire de tenir compte de la nature hiérarchique du problème pour estimer χ. Si χ correspond au label d'une tâche abstraite, alors il faut que l'enchérisseur puisse décomposer n'importe quel label de tâche primitive dans $desc(\chi)$, où $desc(\chi)$ est l'ensemble des labels descendants de χ. L'ensemble des labels devant être estimés est donc noté $\Theta = \{\chi\} \cup desc(\chi) \cup \Phi$. Dans notre approche, nous contraignons la planification en éliminant toutes les alternatives qui n'incluent pas un label présent dans Θ.

6.5.2 *Instanciation des capacités locales - Création de $\mathcal{P}_\sigma$:*

Pour estimer une mise, les robots doivent déterminer *comment* ils planifient d'effectuer les tâches multirobots. Ils peuvent en effet devoir effectuer des actions spécifiques, p. ex. se déplacer jusqu'à l'emplacement de la tâche et activer des capteurs ou des actionneurs particuliers. Nous considérons que les

Exemple 6.6 : Contraindres le Multi-Robot HTN aux labels à estimer

Soient $\mathcal{H}_\delta$, le Multi-Robot HTN représenté ci-dessous et *AUVe1* ayant déjà obtenu la tâche de label l_6 (ou *AUVe1* • (l_6)). *AUVe1* doit estimer une mise pour l_2 Dans la figure, sont représentées en rose la tâche en vente devant être estimée et en violet la tâche déjà acquise.
La contrainte - -> est une contrainte de précédence.
Si rien ne contraint la recherche d'une solution à utiliser les décompositions menants à l_2 et l_6, alors les chemins représentés en bleu et rouge sont possibles.



descriptions des décompositions locales à chaque robot sont définies dans une bibliothèque de connaissance appelée *capacités locales*. Ces connaissances sont statiques, elles contiennent des décompositions génériques pouvant être instanciées avec les paramètres des tâches multirobots à estimer. Au sein de notre approche, nous utilisons le formalisme HDDL pour représenter ces capacités locales, un exemple de modèles locaux est disponible en An. B.

L'instanciation de ces modèles permet de générer un *problème local* $\mathcal{P}_\sigma = (\mathcal{D}_\sigma, \mathsf{s_i}_\sigma, \varnothing)$, où :

— $\mathcal{D}_\sigma$ est un domaine regroupant les connaissances locales de l'agent. Il est composé de tâches opérationnelles multirobots ainsi que de tâches abstraites et primitives locales ;

— $\mathsf{s_i}_\sigma$ est l'état initial local, c.-à-d. un ensemble d'effet initialisant les variables d'état locales. Par exemple, sa position actuelle et son énergie.

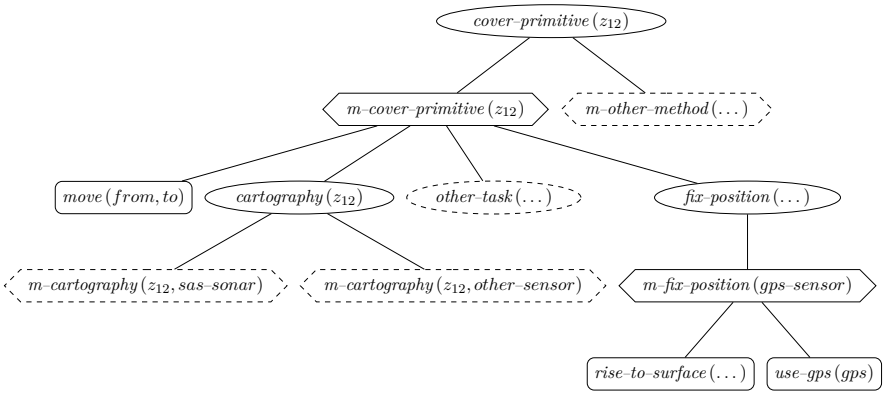
Chaque action locale au sein de $\mathcal{D}_\sigma$ possède un coût.

Les tâches multirobots utilisées pour créer ce problème sont les tâches opérationnelles correspondantes aux labels primitifs à estimer. C'est à dire que, pour chaque $l \in \Theta$, la tâche opérationnelle obtenue par application de $\mathtt{M}_t(l)$ est ajoutée aux tâches instanciant les capacités locales de l'enchérisseur.

Une illustration d'une décomposition contenue dans le domaine local $\mathcal{D}_\sigma$ est représentée dans Ex. 6.7.

Exemple 6.7 : Domaine local

La figure ci-dessous représente une instanciation des modèles de décompositions locaux d'un agent sur la tâche opérationnelle $cover(z_{12})$. Cet HTN est intégré à $\mathcal{D}_\sigma$ et peut servir à estimer n'importe quelle tâche d'un Multi-Robot HTN dont la tâche opérationnelle correspondante est $cover(z_{12})$.



6.5.3 *Structure d'un problème d'estimation*

Afin de créer un problème d'estimation, l'agent fusionne les éléments de $\mathcal{H}_\Gamma$ avec ceux de son problème local $\mathcal{P}_\sigma$. Ce processus de fusion nécessite la mise en place de décompositions mixtes permettant d'établir un pont entre le domaine multirobot et le domaine local. De plus, il faut également tenir compte des conditions, des effets, et des contraintes associées aux tâches opérationnelles primitives mais qui ne sont pas encodées dans les tâches virtuelles.

Pour construire le problème d'estimation, il est nécessaire d'étendre les tâches multirobots contenues dans $\mathcal{H}_\Gamma$ avec les capacités locales, les tâches primitives dans $\mathcal{H}_\Gamma$ sont donc soit :

— Des tâches primitives devant être décomposées localement, c.-à-d. des tâches à estimer ;

— Des tâches primitives ne devant pas être décomposées localement, c.-à-d. des tâches qui ne sont pas à estimer.

En ce qui concerne le premier point, c.-à-d. la décomposition locale de tâches primitives multirobots, le formalisme HTN ne permet que la décomposition de tâches abstraites. De plus, il faut tenir compte des contraintes multirobots qui sont encodées dans la tâche virtuelle. Par conséquent, pour un label l, il faut considérer dans le problème d'estimation à la fois $mr\text{–}task(l)$ et la tâche opérationnelle résultante de l'application de $\mathtt{M}_t(l)$. Nous résolvons ces défis en encodant des décompositions mixtes, liant l'aspect multirobot du problème à l'aspect local, ces décompositions sont appelées *link decompositions* et sont encodées grâce au formalisme HTN.

La création de ces décompositions est nécessaire pour chaque label l de tâche primitive dans Θ. Nous modélisons les *link decompositions* comme suit :

1. $mr\text{–}task(l)$ est copiée dans une tâche primitive notée $mr\text{–}task\text{–}impl(l)$;
2. $mr\text{–}task(l)$ est transformée en une tâche abstraite notée $mr\text{–}task\text{–}abs(l)$. Par conséquent, les effets et contraintes de $mr\text{–}task(l)$ ne sont plus compris que dans $mr\text{–}task\text{–}impl(l)$;
3. La tâche opérationnelle primitive $\mathtt{M}_t(l)$ est transformée en tâche abstraite. Dans le même temps ses conditions, effets et contraintes sont transférés à $mr\text{–}task\text{–}impl(l)$;

4. Une méthode *m–estimation*$(l, \dots)$ est créée pour décomposer *mr–task*(l) en deux sous-tâches :
   — *mr–task–impl*(l). Elle permet de tenir compte des contraintes multirobots initialement présentes dans *mr–task*(l) ;
   — La tâche opérationnelle abstraite $\mathtt{M}_t(l)$. C'est cette tâche qui est également inclue dans le domaine local.
5. Des contraintes de synchronicité sont ajoutées au sein de *m–estimation*$(l, \dots)$ pour que les tâches *mr–task–impl*(l), et $\mathtt{M}_t(l)$ commencent et se terminent en même temps. Ces contraintes permettent de maintenir la cohérence du problème en liant notamment *mr–task–impl*(l) à $\mathtt{M}_t(l)$

Par cette adaptation, nous établissons un lien entre le problème multirobot et le problème local tout en conservant la cohérence de l'ensemble.

Néanmoins, le processus de construction du problème d'estimation n'est pas terminé, car il n'est pas possible d'utiliser les tâches primitives ne devant pas être décomposées localement telles quelles. En effet, il faut tenir compte des conditions, des effets, et des contraintes multirobots associées aux tâches opérationnelles. Ces éléments sont pris en compte dans le cas des tâches à estimer, car les tâches opérationnelles sont ajoutées au problème par les *link decompositions*, mais dans le cas des tâches qui ne sont pas à estimer, les tâches virtuelles ne sont pas décomposées en tâches opérationnelles.

Par conséquent, nous reportons ces éléments dans des tâches virtuelles spécifiquement créées à cette intention. Ces tâches sont appelées *mr–task–impl* et viennent remplacer les *mr–task* correspondantes. La définition de ces tâches est présentée dans Def. 6.6.

Definition 6.6 (Tâche virtuelle primitive de type "*mr–task–prim*")

Soit $\mathtt{T}_{p\text{–}l} = (\mathcal{S}_l, \varnothing, \mathrm{C}_l, \mathsf{pres}_l, \mathsf{adds}_l, \mathsf{dels}_l, \mathsf{start}_l, \mathsf{end}_l, \mathsf{constraints}_l)$, *une tâche virtuelle primitive et* $\mathtt{T}_{p\text{–}op} = (\mathcal{S}_{op}, \varnothing, \mathrm{C}_{op}, \mathsf{pres}_{op}, \mathsf{adds}_{op}, \mathsf{dels}_{op}, \mathsf{start}_{op}, \mathsf{end}_{op}, \mathsf{constraints}_{op})$, *l'action opérationnelle correspondante.*

Par $\mathtt{T}_p = (\mathcal{S}, \varnothing, \mathrm{C}, \mathsf{pres}, \mathsf{adds}, \mathsf{dels}, \mathsf{start}_l, \mathsf{end}_l, \mathsf{constraints}_l)$, *nous définissons une tâche virtuelle primitive de type "mr–task–prim", où :*

— $\mathcal{S}$ *est mr–task–prim ;*
— $\mathrm{C} = \{l\} \cup \mathrm{C}_{op}$ *est l'ensemble des constantes paramétrant la tâche virtuelle ;*
— $\mathsf{pres} = \mathsf{pres}_l \cup \mathsf{pres}_{op}$ *sont les conditions de la tâche virtuelle ;*
— $\mathsf{adds} = \mathsf{adds}_l \cup \mathsf{adds}_{op}$ *sont respectivement les effets positifs de la tâche virtuelle ;*
— $\mathsf{dels} = \mathsf{dels}_l \cup \mathsf{dels}_{op}$ *sont les effets négatifs de la tâche virtuelle ;*

Ainsi, une tâche de ce type est notée mr–task–prim$(l, \dots)$, *où "…" sont les paramètres de la tâche opérationnelle correspondante. Par souci de simplicité,* ***dans ce manuscrit nous conservons une notation de type*** **mr–task**(l), *cependant, chaque fois qu'un problème de planification comporte une tâche primitive* mr–task(l), *nous sous-entendons implicitement qu'il s'agit d'une tâche* mr–task–prim$(l, \dots)$.

Pour construire le domaine d'estimation nous mettons donc en oeuvre ces *mr–task–prim*, *mr–task–impl*, et *mr–task–abs*. Un exemple de domaine d'estimation résultant de ce processus de construction est présenté dans Ex. 6.8. Enfin, la Def. 6.7 définit un problème d'estimation.

Definition 6.7 (Problème d'estimation)

Soit δ, *un article en vente et* $\chi \in \lambda_\delta$.

Par $\mathcal{P}_\chi = (\mathcal{D}_\chi, \mathsf{s_i}_\chi, \mathsf{tn_i}_\chi)$, *nous désignons le problème d'estimation de* χ *où :*

— $\mathcal{D}_\chi$ *est le domaine résultant de la fusion des éléments de* $\mathcal{D}_\Gamma$ *avec ceux de* $\mathcal{D}_\sigma$ *après mise en place de décompositions mixtes et conversion des tâches primitives non décomposées en mr–task–prim.*

Les décompositions mixtes ne sont mises en place que pour les labels primitifs des tâches à estimer, c.-à-d. tous les labels primitifs dans Θ ;

— $\mathsf{s_i}_\chi = \mathsf{s_i}_\Gamma \cup \mathsf{s_i}_\sigma$ *est l'état initial, c.-à-d. un ensemble d'effets positifs qui initialise les variables d'état multirobots et locales ;*

— $\mathsf{tn_i}_\chi$ *est le réseau de tâches initial. Il est composé uniquement de la racine de* $\mathcal{H}_\Gamma$.

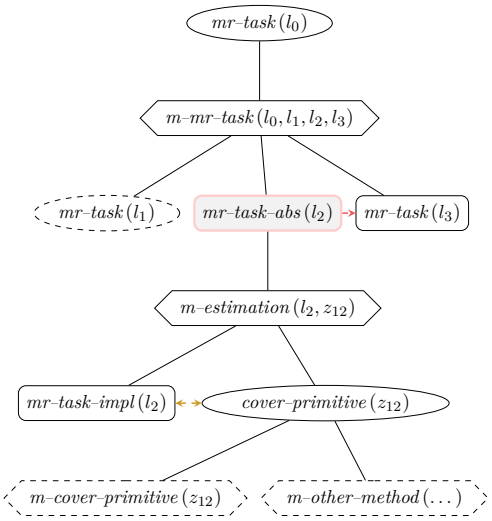
Il est important de noter qu'un problème d'estimation comporte les propriétés suivantes :

— *Pour pouvoir décomposer une tâche du Multi-Robot HTN, il faut que la signature de la tâche opérationnelle correspondante soit également dans* $\mathcal{P}_\sigma$, *c.-à-d. l'union des tâches de* $\mathcal{D}_{op-\delta}$ *avec celles de* $\mathcal{D}_\sigma$ *est non-vide ;*

— *Les problèmes locaux et multirobots ne partagent aucune variable d'état. Ainsi, une tâche multirobot dans le problème d'enchère ne peut pas dépendre d'un prédicat qui ne pourrait être validé que par l'action locale spécifique d'un robot. Inversement, les actions locales d'un robot ne peuvent pas dépendre des effets des tâches multirobot.*

Exemple 6.8 : Domaine d'estimation

Soient les structures des Ex. 6.6 et Ex. 6.7. Le HTN ci-contre représente la décomposition permettant d'estimer la tâche de label l_2. La tâche primitive *mr–task* (l_2) a été remplacée par une tâche abstraite *mr–task–abs* (l_2), ses conditions, ses effets et ses contraintes propres, ont été déplacées dans une tâche primitive *mr–task–impl* (l_2). La contrainte de précédence du réseau de tâches dont faisait partie *mr–task* (l_2) a été reportée sur *mr–task–abs* (l_2) (- - →).
Une décomposition s'appliquant à *mr–task–abs* (l_2) avec comme sous-tâche *mr–task–impl* (l_2) et la tâche opérationnelle obtenue par application de $\mathtt{M}_t l_2$ a été créée. Enfin, *mr–task–impl* (l_2) et $\mathtt{M}_t l_2$ sont contraintes à être exécutées en même temps (↔) .



6.5.4 *Estimation de la mise à l'aide de la planification hiérarchique*

Une fois le problème d'estimation $\mathcal{P}_\chi$ construit, l'enchérisseur peut estimer la tâche en vente de label χ en appelant un solveur hiérarchique pour résoudre $\mathcal{P}_\chi$. Ainsi, l'estimation de la tâche en vente se fait de la même manière que si l'enchérisseur devait planifier son exécution. Pour rappel, par la construction du problème d'estimation, cette planification prend en compte les précédents engagements de l'enchérisseur.

Dans notre approche, les problèmes de planification sont modélisés avec notre formalisme HTN puis ils sont résolus par le solveur LCP [3] (Sec. 5.4.4). Ainsi, nous résolvons $\mathcal{P}_\chi$ à l'aide de LCP pour déterminer la mise de l'enchérisseur.

3. https ://github.com/plaans/aries

Une mise peut être proposée si une solution existe à ce problème d'estimation. Dans le cas d'un problème d'estimation, le plan solution est composé d'un ensemble d'actions multirobots virtuelles et d'actions locales. Les actions multirobots virtuelles dans le plan permettent de déterminer les contraintes multirobots que l'enchérisseur doit soumettre avec sa mise. Les actions locales permettent de déterminer le montant de la mise.

Utilisation d'un coût comme fonction d'utilité

Dans notre approche, le montant de la mise est déterminé en sommant les coûts des actions locales dans le plan solution. Les tâches primitives virtuelles sont comptabilisées à un coût de 0. Enfin, le solveur que nous utilisons optimise la solution trouvée en minimisant le coût total du plan.

Il est à noter qu'une optimisation monocritère sur la valeur d'utilité telle que le coût d'une action reste la manière la plus générale d'aborder une approche par enchères. La simplicité de ce critère permet de se concentrer sur l'exploration d'autres éléments de l'approche quand le réalisme de la modélisation des mises n'est pas au coeur de la contribution. Mais il est possible d'utiliser d'autres critères d'utilités, p. ex. l'état estimé du réseau de communication ou le niveau d'énergie des robots après une action [QGL22a]. Une telle modification est facilitée par la généricité du schéma de décision.

Nécessité de transmettre les intentions de l'enchérisseur sur les tâches multirobots

Avec la structure hiérarchique utilisée, miser sur une tâche abstraite permet de miser sur tout le sous-arbre associé. Cette caractéristique permet aux enchérisseurs d'acheter directement des sous-arbres. Mais pour maintenir un problème cohérent, l'enchérisseur doit fournir des informations sur les choix qu'il a faits au sein de ce sous-arbre. Cela est nécessaire en raison des dépendances complexes pouvant lier les tâches. Par exemple, une contrainte causale peut n'être satisfaite que dans une alternative, si cette alternative n'est pas retenue, le commissaire-priseur ne doit pas allouer des tâches dépendant de cette contrainte. Un tel cas est illustré dans la Fig. 6.7. Dans cet exemple, l'agent estime la tâche $mr\text{–}task(l_1)$ et détermine un plan impliquant l'action $mr\text{–}task(l_3)$. L'agent ne planifie pas d'utiliser l'action $mr\text{–}task(l_4)$ et donc d'activer l'effet C_x. Or, la tâche $mr\text{–}task(l_6)$ est liée par une contrainte causale sur C_x à $mr\text{–}task(l_4)$, son allocation n'est donc pas compatible avec la mise de l'agent.

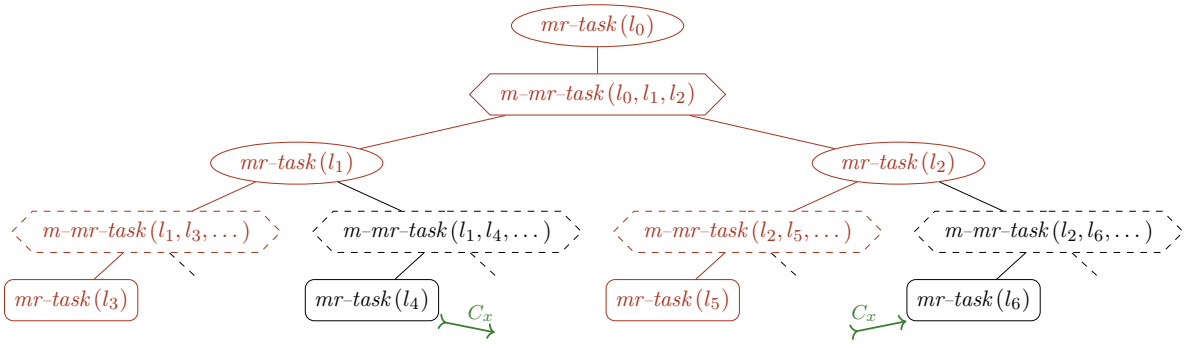


FIGURE 6.7 – Exemple d'estimation dont les choix remettent en cause des décompositions impliquant une contrainte causale. La tâche estimée est $mr\text{–}task(l_1)$. En rouge sont représentées les décompositions retenues dans la résolution du problème d'estimation. Les flèches vertes représentent une contrainte causale, une condition de $mr\text{–}task(l_6)$ est satisfaite par un effet de $mr\text{–}task(l_4)$.

En sus d'information sur les décompositions retenues, les enchérisseurs doivent également transmettre le plan associé aux tâches multirobots qu'ils vont réaliser. Cela est encore une fois dû à la présence de dépendances complexes.

Les intentions des enchérisseurs sur les tâches multirobots sont encapsulées au sein des plans que les enchérisseurs déterminent à chaque mise. Ces plans contiennent à la fois le choix des décompositions et les contraintes supplémentaires nécessaires à la validité des mises. Si ces plans ne sont pas transmis, la résolution du WDP ne dispose pas des informations suffisantes pour obtenir une solution cohérente, résultant à terme à des impossibilités d'exécution. Nous détaillons ces problèmes dans [Mil+21b].

Ainsi, il est nécessaire de transmettre le plan multirobot déterminé lors de l'estimation, ce plan intègre :

— Les tâches multirobots que l'enchérisseur compte accomplir, c.-à-d. les tâches soit déjà acquises, soit sur lesquelles il mise ;

— Les tâches multirobots que l'enchérisseur souhaite que d'autres accomplissent, car nécessaires à ses objectifs.

Plans solutions et liens causaux

Soit Π_χ le plan solution de ce problème d'estimation. Au sein du plan solution Π_χ, nous définissons deux sous-plans :

— Un plan local Π_σ composé uniquement des actions locales à effectuer ;

— Un plan multirobot Π_μ composé uniquement des actions multirobots à effectuer, c.-à-d. des actions de types *mr–task* et *mr–task–impl*.

La définition d'un plan Π solution d'un problème de planification est présentée dans la Def. 6.8. Formellement, $\Pi_\chi = \Pi_\mu \cup \Pi_\sigma$, où Π_μ est le plan multirobot et Π_σ est le plan local. Ainsi, Π_μ est un extrait du plan retourné par LCP ne contenant que les éléments multirobots, de même pour Π_σ avec les éléments locaux.

Definition 6.8 (Plan solution d'un problème de planification)

Par Π, nous désignons un plan solution d'un problème de planification.

$\Pi = (\mathtt{T}_p, \mathsf{CL}, \mathsf{c})$, *où :*

— $\mathtt{T}_p$ *est un ensemble de tâches primitives dans le plan ;*

— CL *est un ensemble de liens causaux, un lien causal est un couple entre deux deux tâches* $\mathtt{T}_p$ *;*

— c *est la valeur associée au plan, c.-à-d. son coût.*

Un exemple de plan solution d'un problème de planification retourné par le solveur LCP est donné en An. D.

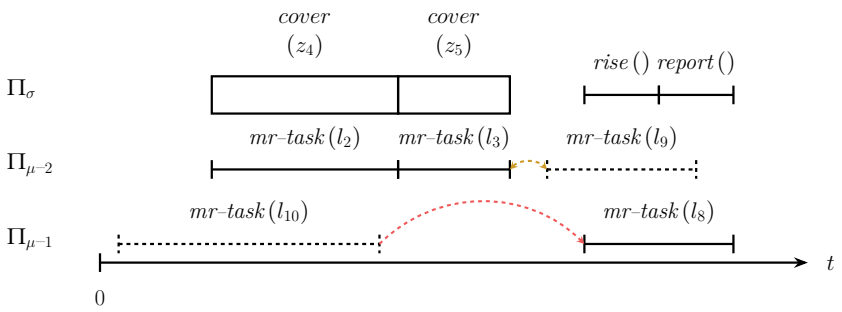
Le plan multirobot est utilisé pour transmettre au commissaire-priseur les choix retenus et les contraintes soumises par l'enchérisseur. Cependant, la résolution du problème de planification passe par une instanciation de toutes les variables du problème, le plan solution ne comporte par conséquent que des tâches dont les intervalles d'exécution sont définis, p. ex. si $[3, 8]\,mr\text{–}task(l) \in \Pi_\mu$, l'exécution de la tâche $mr\text{–}task(l)$ est planifiée sur l'intervalle $[3, 8]$, ce qui se traduit par deux contraintes de synchronicité impactant fortement les décisions futures.

Nous relâchons donc les contraintes non nécessaires de ce plan afin de ne pas surcontraindre les tâches multirobots. Pour ce faire, nous utilisons les liens causaux qui unissent les tâches entre elles. Les liens causaux sont déterminés par le solveur lors de la planification. Deux tâches sont unies par un lien causal si l'exécution d'au moins une de ces tâches n'est pas possible sans l'autre. Par exemple, si une condition d'une tâche est déclenchée par un effet d'une autre tâche, alors un lien causal existe. Si aucun lien causal ne lie une tâche à au moins une autre tâche, alors les contraintes ajoutées lors de l'estimation sont relâchées. Si le lien causal permet d'identifier une contrainte de précédence, alors les contraintes de synchronicité

sont relâchées et remplacées par une contrainte de précédence. Sinon, les contraintes sont ajoutées lors de l'estimation sont conservées. Les tâches liées entre elles par des liens causaux forment une *chaine de causalité*. Un exemple de chaine de causalité est donnée en An. C

Un exemple de solution d'un problème d'estimation est donné dans Ex. 6.9 et en An. D. Dans cet exemple, une contrainte de synchronicité a été relâchée au profit d'une contrainte de précédence.

Exemple 6.9 : Solution du problème d'estimation



Plan 6.1

Extrait du plan solution d'une estimation. Les boîtes représentent des tâches abstraites tandis que les lignes noires représentent des actions. Pour des raisons de lisibilité, les tâches ne sont pas décomposées.
Les flèches - -> et <- -> représentent respectivement une contrainte de précédence et un contrainte de synchronicité. Les contraintes représentées ici sont les contraintes supplémentaires déterminées par l'enchérisseur et nécessaires à la validité de sa mise, c.-à-d. après relâchement des contraintes grâce aux liens causaux.
Dans ce plan, $\Pi_{\mu-1}$ et $\Pi_{\mu-2}$ représentent deux chaines de causalités du plan multirobot Π_μ.

6.5.5 *Formalisation du message de mise*

Pour finir, l'enchérisseur retourne au commissaire-priseur un message de mise, dont le contenu est défini dans Def. 6.9.

Definition 6.9 (Message de mise)

Un message de mise ("bid"), comporte les informations suivantes :

- r *: L'identifiant de l'enchérisseur, p. ex. "*AUVe2*";*
- auction id *et* round id *: Les identifiants de l'enchère et du tour d'enchère ;*
- χ *: Le label de la tâche souhaitée ;*
- ψ *: La valeur de la mise, c.-à-d. La valeur* c *du plan solution de* Π_χ *;*
- Π_μ *: Le plan multirobot associé à la mise. Ce plan contient notamment les nouvelles contraintes multirobots soumises par l'enchérisseur ;*
- handled labels *: Les labels que l'enchérisseur s'engage à accomplir avec la mise, c.-à-d.* Θ*. Dans notre approche, cette information est nécessaire au commissaire-priseur pour modéliser le WDP.*

6.5.6 *Critères d'arrêts de l'estimation des mises*

Le critère d'arrêt principal de la phase d'estimation des mises est l'échéance t_a fixée par le commissaire-priseur. Quand cette échéance est atteinte, les enchérisseurs arrêtent leurs estimations et le commissaire-priseur débute la détermination des gagnants.

En sus de cette échéance, dans notre implémentation nous utilisons également le nombre de mises comme critère d'arrêt. Nous contraignons chaque enchérisseur à estimer une mise pour chaque label en vente. Si un enchérisseur est incapable d'accomplir une tâche, un message de mise est tout de même envoyé pour le signaler. Ce comportement permet au commissaire-priseur d'utiliser le nombre de mises reçues comme critère d'arrêt. Ainsi, si le nombre de mises réceptionnées est suffisant, le commissaire-priseur n'a pas a attendre jusqu'à t_a pour débuter la résolution du WDP.

Le nombre de mises attendues est égal au nombre d'enchérisseurs multiplié par le nombre de labels en vente. [4] Par exemple, le commissaire-priseur peut commencer à résoudre le WDP lorsque le nombre de mises reçues est égal au nombre maximum de mises attendues ou qu'une certaine proportion des mises attendues a été reçue.

6.5.7 *Synthèse du processus d'estimation*

Dans la Fig. 6.8, nous présentons une synthèse du processus d'estimation décrit dans cette section. Les données d'entrées du processus d'estimation sont :

- L'article en vente δ, composé de l'ensemble des labels en vente λ_δ et du Multi-Robot HTN de l'enchère $\mathcal{H}_\delta$;
- L'ensemble des labels précédemment obtenus par l'enchérisseur, noté Φ ;
- Les modèles génériques des capacités locales de l'enchérisseur, notés "*Local Domain*" ;
- L'état initial de l'enchérisseur, noté $\mathsf{s_{i}}_\sigma$.

À partir de ces données d'entrées sont construites ces différentes structures intermédiaires :

- L'ensemble des labels à estimer, noté Θ ;
- Le Multi-Robot HTN réduit du problème d'estimation, noté $\mathcal{H}_\Gamma$;
- Le problème local au formalisme HTN, noté $\mathcal{P}_\sigma$;
- Le problème d'estimation à résoudre, noté $\mathcal{P}_\chi$;
- Le plan solution du problème d'estimation, noté Π_χ, ainsi que son coût et ses parties locales et multirobots, notés respectivement c, Π_σ et Π_μ ;

Enfin, la donnée de sortie du processus est la mise de l'enchérisseur.

6.6 DÉTERMINATION DES GAGNANTS

À la fin d'un tour d'enchères, le commissaire-priseur doit résoudre le WDP pour déterminer les allocations de tâches. Étant donné λ_δ l'ensemble des labels des tâches à vendre et $R = (r_1, \ldots, r_n)$ l'ensemble des enchérisseurs participant à l'enchère, nous notons B_{r_i} l'ensemble des mises reçues de l'enchérisseur r_i portant sur des labels dans λ_δ. L'ensemble de toutes les mises reçues est désigné par $\mathcal{B} = \bigcup_{r_i \in R} B_{r_i}$. La résolution du WDP consiste à trouver un ensemble de mises gagnantes $\mathcal{B}_w \subset \mathcal{B}$ tel que :

4. Dans notre approche, les participants à une enchère sont déterminés lors de l'ouverture de l'enchère par une phase initiale qui est présentée dans le chapitre suivant, le Chap. 7.

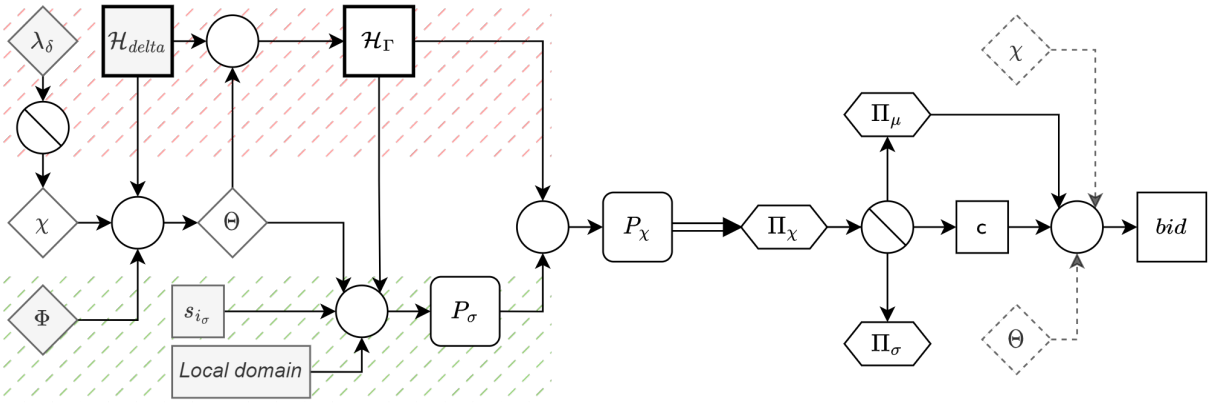


FIGURE 6.8 – Processus d'estimation d'une mise. Les blocs dans les parties hachurées en rose et en vert représentent respectivement des éléments propres à l'article en vente et à l'enchérisseur. Les rectangles sont des structures de données gérées par l'enchérisseur. Les losanges représentent des ensembles de labels multirobots. Les blocs grisés sont des données d'entrée du problème d'estimation. Les cercles pleins indiquent les processus qui agrègent les informations pour construire de nouvelles structures et les cercles barrés indiquent les processus qui déconstruisent une structure en plusieurs autres. Les rectangles arrondis représentent des problèmes de planification hiérarchique. La double flèche indique un appel à un planificateur HTN. Les hexagones représentent des plans solutions d'un problème de planification. Enfin, les éléments en pointillés allègent la figure en faisant référence à leurs blocs d'origine.

— Chaque enchérisseur gagne au plus une mise, car un enchérisseur estime ses mises de façon indépendante ;

— Chaque $\chi \in \lambda_\delta$ soit gagné par au plus un enchérisseur.

Afin de déterminer cet ensemble de mises gagnantes, nous utilisons une nouvelle fois la planification hiérarchique. Ainsi, le commissaire-priseur doit construire $\mathcal{P}_{wdp} = (\mathcal{D}_{wdp}, \mathsf{si}_{wdp}, \mathsf{tni}_{wdp})$ un problème de planification dédié à la résolution du WDP.

Pour ce faire, nous agrégeons les informations de δ avec les mises reçues et un coût de revente (étape ② dans la Fig. 6.6).

6.6.1 *Revendre afin d'allouer à un prochain tour d'enchères*

Il est possible de ne pas pouvoir allouer toutes les tâches en un seul tour d'enchères. En effet, le commissaire-priseur peut ne pas avoir reçu de mise pour une tâche, ou il n'est juste pas possible de tout allouer parce que chaque enchérisseur peut remporter au plus une mise. Cependant pour résoudre le WDP, il faut pouvoir trouver une solution globale au problème de planification des tâches multirobots. Cette solution doit donc pouvoir inclure des tâches qui n'ont pas encore pu être allouées. Ainsi, à chaque tâche allouable doit être associée une alternative de *revente* ("resell") correspondant à la non-allocation de cette tâche à un enchérisseur.

Comme nous ne pouvons pas forcer le WDP à allouer exactement une tâche à chaque robot, nous ne pouvons pas fixer le coût dans le plan d'une revente à 0 : ces tâches seraient systématiquement revendues lors de l'optimisation du plan d'allocations. De plus, dans certaines situations il peut être plus intéressant de revendre une tâche que de l'allouer, p. ex. parce que la valeur de la mise est trop pénalisante. Par conséquent, nous devons définir des *coûts de revente* ("resale costs"), c.-à-d. un prix d'appel, pour avoir un comportement d'allocation sain et efficace.

La valeur du coût de revente permet de contrôler le comportement de l'allocation. Il est par exemple possible de le fixer de manière à allouer le plus de tâches possible en un seul tour, ou au contraire de

manière à favoriser uniquement les meilleures mises en revendant au prochain tour les tâches pour lesquelles les mises reçues étaient trop pénalisantes.

Bien que ces coûts puissent être définis en fonction du domaine d'application, nous avons proposé dans [Mil+21a] deux stratégies génériques pour les définir, basées sur les mises reçues pour une tâche. Soit $\Psi = \bigcup_{\chi \in \lambda_\delta} \Psi_\chi$, l'ensemble des valeurs des mises reçues pour une tâche de label $\chi \in \lambda_\delta$, ces stratégies sont détaillées dans Def. 6.10 et Def. 6.11.

Definition 6.10 (Stratégie de revente *maxB*)

La stratégie de revente maxB est une stratégie pessimiste quant aux mises qui seront reçues lors des prochains tours. Pour cette stratégie :

- *Le coût de revente de chaque tâche est fixé comme strictement supérieur à la valeur maximale des mises reçues pour cette tâche, c.-à-d.* $resale\text{–}cost = max(\Psi_\chi) + 1$;
- *S'il n'y a pas de mise pour une tâche, le coût de revente est strictement supérieur à la somme des coûts de revente des descendants, si plusieurs décompositions sont possibles, la décomposition la moins avantageuse est retenue, c.-à-d.* $resale\text{–}cost = max\{sum(\Psi_l) + 1 : l \in desc(\chi)\}$;
- *Si aucune mise n'a été faite sur la tâche ou ses descendants, la valeur par défaut est strictement supérieure à la somme de toutes les mises, c.-à-d.* $resale\text{–}cost = sum(\Psi) + 1$.

Avec cette stratégie, c'est l'allocation d'autant de tâches que possible en un tour d'enchère qui est favorisée.

Definition 6.11 (Stratégie de revente *minB*)

La stratégie de revente minB est une stratégie optimiste qui favorise la revente au prochain tour lorsqu'une meilleure mise est espérée à un tour ultérieur. Pour cette stratégie :

- *Le coût de revente de chaque tâche est fixé comme strictement supérieur à la valeur minimale des mises reçues, c.-à-d.* $resale\text{–}cost = min(\Psi_\chi) + 1$.
- *S'il n'y a pas de mise pour la tâche, le coût de revente est strictement supérieur au à la somme des coûts de revente des enfants plus un, si plusieurs décompositions sont possibles, la décomposition la plus favorable est retenue, c.-à-d. ie* $resale\text{–}cost = min\{sum(\Psi_l) + 1 : l \in desc(\chi)\}$;
- *Si aucune mise n'a été faite sur la tâche ou ses descendants, la valeur par défaut est strictement supérieure à la somme de toutes les mises, c.-à-d.* $resale\text{–}cost = sum(\Psi) + 1$.

Avec cette stratégie, c'est la qualité des allocations qui est favorisée au détriment du nombre de tours d'enchères.

Pour prendre en compte le coût de revente d'une tâche, il faut ajouter au problème de planification du WDP un mécanisme de revente permettant d'utiliser les tâches revendues dans le plan tout en guidant l'optimisation de la solution.

6.6.2 *Modélisation du problème de détermination des gagnants*

Dans le but d'assurer la cohérence de l'allocation, il est nécessaire d'encoder dans le WDP les plans que les enchérisseurs associent à leurs mises afin de tenir compte des contraintes multirobots que les enchérisseurs soumettent. L'encodage de ces contraintes au sein du problème de détermination des gagnants n'est pas trivial, cela pour deux raisons :

— Les mises d'**un** enchérisseur, et donc ses contraintes, doivent être considérées de façon indépendante. Car chaque enchérisseur peut remporter au plus une mise.

— Les mises **des** enchérisseurs ne sont pas indépendantes entre elles, c.-à-d. pour être retenues les mises de deux enchérisseurs doivent être compatibles.

Ainsi, il est nécessaire de considérer des plans optionnels, mais dépendants entre eux.

Modélisation à l'aide de structures hiérarchiques dédiées à la revente et l'encodage des mises

Pour résoudre le problème d'encodage des mises et de la revente nous procédons comme suit :

— Nous associons aux tâches primitives virtuelles de $\mathcal{H}_\delta$ des prédicats dédiés à la vérification des plans ;

— Pour chaque enchérisseur, nous construisons un nouvel HTN dédié à l'encodage de ses mises et de son plan préexistant, c.-à-d. une décomposition pour chaque mise et une, le cas échéant, pour le plan engagé au tour précédent. À partir de chacun de ces HTN et pour chaque enchérisseur $r_i \in R$, nous construisons un problème de planification hiérarchique $\mathcal{P}_{r_i}$ dédié à l'encodage des mises de l'enchérisseur r_i ;

— Nous construisons un nouvel HTN dédié à la revente des tâches. Pour chaque tâche en vente, deux décompositions sont créées, une pour revendre la tâche, une pour la signaler comme non revendue (car allouée). À partir de cet HTN, nous construisons un problème de planification hiérarchique $\mathcal{P}_{resell}$ dédié à l'encodage des informations de la revente.

Pour n enchérisseurs, le problème comporte donc $n+2$ structures hiérarchiques.

Grâce à ces nouveaux prédicats et HTN nous assurons la cohérence des allocations entre elles. Les nouveaux prédicats permettent de lier ces structures hiérarchiques, car l'encodage des mises et la revente y font référence afin de vérifier la validité du plan. Dans Ex. 6.10, nous illustrons les structures hiérarchiques résultant de ce processus.

Formalisation du WDP

En se reposant sur ces nouvelles structures, il est possible de modéliser le problème de détermination comme un problème de planification hiérarchique. Une définition du problème résultant est présentée dans Def. 6.12.

Definition 6.12 (Problème de détermination des gagnants)

Par $\mathcal{P}_{wdp} = (\mathcal{D}_{wdp}, \mathsf{si}_{wdp}, \mathsf{tni}_{wdp})$, nous désignons un problème de détermination des gagnants. Ce problème est construit en intégrant les structures complexes définies par l'article en vente, le problème de revente, et les problèmes encodant les mises, c.-à-d. $\mathcal{H}_\delta$, $\mathcal{P}_{resell}$, et $\mathcal{P}_{r_i}$ pour chaque $r_i \in R$. Ainsi, nous avons :

— $\mathcal{D}_{wdp} = \mathcal{D}_\delta \cup \mathcal{D}_{resell} \bigcup_{r_i \in R} \mathcal{D}_{r_i}$;

— $\mathsf{si}_{wdp} = \mathsf{si}_{-\delta}$;

— $\mathsf{tni}_{wdp} = \left(\mathtt{L}_{wdp}, \mathbb{T}_{wdp}, \prec_{wdp}, \alpha_{wdp}\right)$ *est défini par :*

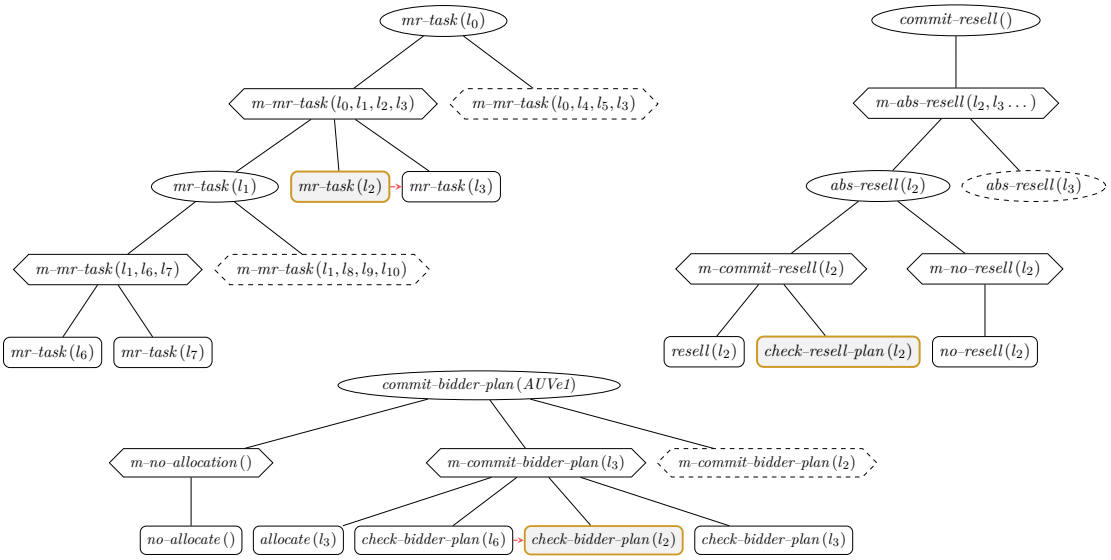
  — $\mathtt{L}_{wdp}$ *est un ensemble de labels ;*

  — $\mathbb{T}_{wdp} = \mathbb{T}_\delta \cup \mathbb{T}_{resell} \bigcup_{r_i \in R} \mathbb{T}_{r_i}$. *C'est-à-dire les tâches racines de $\mathcal{H}_\delta$, du HTN de revente et de chaque HTN dédiés à l'encodage des mises des enchérisseurs ;*

  — $\prec_{wdp} = \varnothing$;

  — $\alpha_{wdp} : \mathtt{L}_{wdp} \to \mathbb{T}_{wdp}$ *établit une correspondance des labels de $\mathtt{L}_{wdp}$ vers une tâche de $\mathbb{T}_{wdp}$.*

Exemple 6.10 : Domaine du WDP



Le Multi-Robot HTN de l'enchère, $\mathcal{H}_\delta$, est en haut à gauche. Par souci de simplicité, les noeuds en pointillés ne sont pas décomposés. Les flèches - -> représentent des contraintes de précédence.
En haut à droite, un HTN encodant la revente des tâches est représenté. La racine de cet HTN se décompose en une tâche abstraite par label en vente, pour accomplir ces tâches abstraites deux décompositions sont possibles : Revendre la tâche ou ne pas revendre la tâche. Chacune des décompositions associées à la revente possède une action *resell* incrémentant le coût du plan de la valeur de la revente.
En bas au milieu, un HTN encodant les plans possibles pour un enchérisseur. La racine de cet HTN possède une décomposition par plan, c.-à-d. une décomposition d'allocation par mise reçue ainsi qu'une décomposition de non-allocation. Chacune des décompositions associées aux mises possède une action *allocate* incrémentant le coût du plan de la valeur de la mise.
Enfin, chacune des décompositions de revente et d'allocation possède des tâches $\mathit{check}\text{–}\ldots\text{–}\mathit{plan}(l_x)$ permettant de vérifier le plan multirobot associé à la décompositon.
Par exemple, les actions $\mathit{check}\text{–}\mathit{bidder}\text{–}\mathit{plan}(\ldots)$ représentées dans la figure encodent le plan multirobot associé à la mise de $\mathit{AUVe1}$ sur la tâche $\mathit{mr}\text{–}\mathit{task}(l_2)$, dans ce plan $\mathit{AUVe1}$ soumet une contrainte de précédence entre $\mathit{mr}\text{–}\mathit{task}(l_6)$ et $\mathit{mr}\text{–}\mathit{task}(l_2)$. Chacune des tâches $\mathit{check}\text{–}\ldots\text{–}\mathit{plan}(l_x)$ est contrainte à être exécutée de manière synchrone avec $\mathit{mr}\text{–}\mathit{task}(l_x)$ dans le plan, c.-à-d. avec les mêmes intervalles temporels. Les actions grisées représentent un lot de ces actions synchrones.

6.6.3 *Résolution du problème de détermination des gagnants à l'aide de la planification hiérarchique*

Nous avons précédemment détaillé comment, à partir de $\mathcal{H}_\delta$ et des mises reçues, un nouveau problème de planification $\mathcal{P}_{wdp}$ correspondant au WDP de ce tour d'enchères était créé par le commissaire-priseur. Ce problème, encodé dans notre formalisme HTN, est ensuite résolu par un planificateur hiérarchique. La solution obtenue est un plan noté Π_{wdp}, ce plan intègre notamment :

— Les actions multirobots à accomplir, c.-à-d. les actions précédemment allouées, les actions venant d'être allouées, et les actions nécessaires qui devront être revendues au prochain tour ;

— Pour chaque action multirobot dans le plan, si cette action était en vente alors une action d'allocation ou une action de revente lui est associée dans le plan.

Enfin, le coût de ce plan est optimisé en minimisant le coût des allocations et des reventes. Un extrait du plan solution obtenu en sortie de LCP est donné en An. E.

À partir de ce plan, le commissaire-priseur définit les messages de récompense associé aux mises gagnantes et les transmet aux enchérisseurs concernés.

6.6.4 *Synthèse du processus de détermination des gagnants*

Dans la Fig. 6.10, nous présentons une synthèse du processus de détermination des gagnants décrit dans cette section. Les données d'entrées du processus de détermination des gagnants sont :

— L'article en vente δ, composé de l'ensemble des labels en vente λ_δ et du Multi-Robot HTN de l'enchère $\mathcal{H}_\delta$;

— L'ensemble des mises reçues, noté $\mathcal{B}$.

À partir de ces données d'entrées sont construites ces différentes structures intermédiaires :

— L'ensemble des mises de chaque enchérisseur, noté B_{r_i} pour chaque enchérisseur $r_i \in R$;

— Les problèmes HTN dédiés à l'encodage des mises des enchérisseurs, noté $\mathcal{P}_{r_i}$ pour chaque enchérisseur $r_i \in R$;

— Le problème HTN dédié à l'encodage de la revente, noté $\mathcal{P}_{resell}$;

— Le problème de détermination des gagnants, noté $\mathcal{P}_{wdp}$;

— Le plan solution du problème de détermination des gagnants, noté Π_{wdp}.

Enfin, la donnée de sortie du processus est l'ensemble des allocations, c.-à-d. les mises gagnantes $\mathcal{B}_w$.

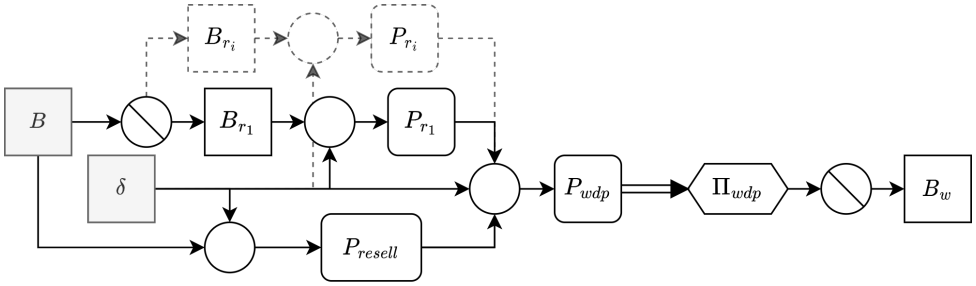


FIGURE 6.10 – Processus de détermination des gagnants. Les rectangles sont des structures de données gérées par le commissaire-priseur. Les blocs grisés sont des données d'entrée du problème de détermination des gagnants. Les cercles pleins indiquent les processus qui agrègent les informations pour construire de nouvelles structures et les cercles barrés indiquent les processus qui déconstruisent une structure en plusieurs autres. Les rectangles arrondis représentent des problèmes de planification hiérarchique. La double flèche indique un appel à un planificateur HTN. L'hexagone représente un plan solution d'un problème de planification. Enfin, les éléments en pointillés représentent une répétition du processus d'encodage des mises n'importe quel enchérisseur r_i, avec $r_i \in R = (r_1, \ldots, r_n)$.

6.7 TRANSMISSION DES RÉCOMPENSES ET CLÔTURE DU TOUR D'ENCHÈRE

En raison des pertes de communication possibles, nous imposons un protocole de confirmation des récompenses aux robots. Ce protocole permet de garantir que le plan associé à une récompense ne sera encodé de manière permanente au sein des Multi-Robot HTN des agents que si l'enchérisseur concerné

a confirmé avoir reçu sa récompense. Un enchérisseur victorieux doit donc transmettre un accusé de réception de sa récompense. Par conséquent, la fin d'un tour d'enchère se fait en trois étapes :

— La réception des récompenses par les enchérisseurs (processus ③ dans la Fig. 6.6) ;

— La réception des accusés de réception des récompenses par le commissaire-priseur (processus ④ dans la Fig. 6.6).

— La transmission d'un message de clôture de l'enchère par le commissaire-priseur. Ce message contient notamment un Multi-Robot HTN intégrant les plans associés aux récompenses qui ont été confirmées.

Réception d'une récompense

Lors de la réception d'une récompense, un enchérisseur victorieux doit intégrer les tâches qu'il s'est vu allouer à l'ensemble des tâches qu'il doit accomplir. Pour intégrer ces tâches, les contraintes contenues dans le plan multirobot associé à la mise sont encodées de manière permanente dans le Multi-Robot HTN de l'enchérisseur. Ces tâches seront ensuite supervisées et exécutées par les processus dédiés de l'agent et dont le fonctionnement est détaillé dans le Chap. 7. Enfin, l'enchérisseur doit transmettre un accusé de réception de la récompense.

Réception des accusés de réception

Lors de la réception de l'accusé de réception, le commissaire-priseur valide l'allocation des tâches concernées et intègre lui aussi les contraintes de la mise au Multi-Robot HTN utilisé comme article en vente. Nous notons $\mathcal{H}_\Delta$ ce Multi-Robot HTN intégrant des confirmations de récompense, c.-à-d. $\mathcal{H}_\Delta$ correspond à $\mathcal{H}_\delta$ avec des contraintes supplémentaires. Le processus de formulation d'un accusé de réception étant rapide, le commissaire-priseur utilise une échéance courte comme critère d'arrêt de l'attente des confirmations des réceptions des récompenses. Si aucun accusé de réception n'est reçu lorsque l'échéance est rencontrée, le commissaire-priseur considère que les tâches qui n'ont pas été confirmées doivent être revendues lors d'un prochain tour d'enchères.

Grâce à ce processus, les contraintes associées à une récompense ne sont encodées que si l'agent concerné a bien reçu sa récompense. Par conséquent, les Multi-Robot HTN des agents ne comportent que des contraintes qui ont été confirmées par leurs coéquipiers. De plus, les tâches qui n'ont pas été confirmées sont réallouées tant qu'un enchérisseur victorieux ne confirme pas leur réception, permettant ainsi de s'assurer de leur bonne intégration au plan de mission.

Clôture du tour d'enchère

En raison du protocole d'accusé de réception des récompenses, il n'est pas possible de diffuser à tous les robots les mises remportées et d'arrêter là le tour d'enchère, car seules les récompenses confirmées doivent être intégrées par les agents. Par conséquent, après réception des accusés de réception, le commissaire-priseur diffuse à tous les agents un message de clôture d'enchère. Ce message est composé de :

— *auction id* et *round id* : Les identifiants de l'enchère et du tour d'enchère ;

— $\mathcal{H}_\Delta$: Le Multi-Robot HTN obtenu après intégration des plans associés aux récompenses confirmées ;

— *close auction* : Un booléen indiquant si l'enchère est finie ou si un nouveau tour va débuter. Ce champ permet par exemple aux enchérisseurs de déterminer s'ils doivent commencer à exécuter les tâches qu'ils ont acquises.

Une fois un message de clôture d'enchère réceptionné, les agents mettent à jour les décisions connues sur la mission en remplaçant leur propre Multi-Robot HTN par $\mathcal{H}_\Delta$.

Enchère sur plusieurs tours

Les étapes décrites jusqu'ici composent un unique tour d'enchère. Cependant, comme vu précédemment avec les concepts de coût de revente et d'accusé de réception des récompenses, un tour d'enchère peut se terminer avec des tâches qui n'ont pas été allouées. Dans ce cas, le problème de MRTA n'a pas pu être résolu en un seul tour d'enchère et un nouveau tour doit être déclenché par le commissaire-priseur.

Pour ouvrir un nouveau tour, le commissaire-priseur crée un nouveau message d'annonce composé de son Multi-Robot HTN et des labels à revendre. Puis, les étapes se succèdent à nouveau. Dans Ex. 6.11, nous illustrons une résolution possible de tours d'enchères successifs.

Exemple 6.11 : Allocations par tour

Soient la structure présentée dans Ex. 6.6 comme $\mathcal{H}_\delta$ et $\lambda = (\, l_0, l_1, l_2, l_3, l_6, l_7, l_8, l_9, l_{10}, \dots\,)$ les labels en vente, les labels en pointillés correspondent aux tâches non décomposées de la figure.

Dans le tableau ci-contre, l'allocation de la tâche l_1 au robot *AUVe2* correspond à la cellule "*AUVe2* - Tour 1" du tableau (note : l'allocation de l_1 implique l'allocation de $(\, l_6, l_7\,)$, ou $(\, l_8, l_9, l_{10}\,)$). Les tâches devant être renvendues au tour *n+1* sont représentées dans la cellule "Revente - Tour *n+1*".

	Tour 1	Tour 2
AUVe1		l_3
AUVe2	l_1	l_2
Revente	l_2 l_3	

Dans notre approche, les tours d'enchères se succèdent ainsi jusqu'à ce qu'une des conditions suivantes soit satisfaite :

— Il n'est plus nécessaire de vendre des tâches pour accomplir le Multi-Robot HTN du commissaire-priseur. Car la résolution du WDP n'a mené a aucune revente et toutes les récompenses ont été confirmées ;

— Un critère d'arrêt est atteint, p. ex. la tenue d'un tour sans aucune allocation. Dans un tel cas il est par exemple possible pour le commissaire-priseur d'ouvrir une nouvelle enchère lorsqu'il rentre en contact avec de nouveaux robots afin de réessayer d'allouer ces tâches.

6.8 VERS UNE UTILISATION EN LIGNE

Dans ce chapitre, nous avons détaillé le fonctionnement de notre protocole de décision. La structure de ce protocole est basée sur des allocations par enchères, les objets en vente sont des réseaux de tâches hiérarchiques, et la planification hiérarchique est utilisée pour résoudre les étapes d'estimation des mises et de détermination des gagnants.

Avec ce protocole de décision, il est possible de résoudre le problème de MRTA correspondant à la planification initiale, mais également de procéder à des réparations en ligne. En effet, une des grandes forces des allocations par enchères réside dans la résilience qu'elles offrent au système face à des aléas. Dans notre approche, chaque agent peut devenir commissaire-priseur et faire appel à ce protocole de décision pour résoudre en ligne un nouveau problème de MRTA. Ainsi, n'importe quel agent peut essayer de réparer le plan de mission en intégrant réactivement de nouveaux objectifs ou en revendant des tâches.

La finalité du protocole qui a été décrit dans ce chapitre s'inscrit donc dans une utilisation en ligne. Aussi, dans le chapitre suivant, le Chap. 7, nous détaillons comment les structures complexes issues de ce protocole de décision sont réutilisées pour superviser et exécuter les décisions prises, mais également pour réparer le plan de mission en faisant appel à ce même protocole de décision.

7

SUPERVISER ET RÉPARER AVEC DES RÉSEAUX DE TÂCHES HIÉRARCHIQUES ET DES ENCHÈRES

Aperçu

Durant l'accomplissement de notre mission, de nombreux aléas peuvent perturber le plan des robots. Dans ce chapitre, nous décrivons les aléas inhérents à notre mission. Puis, nous détaillons les processus qui sont mis en oeuvre au sein de notre architecture dans le but de surmonter ces perturbations.

L'utilisation d'un système robotisé en situation réelle implique de faire face à des situations dont les paramètres ne sont pas maitrisés. En effet, de nombreux inattendus peuvent apparaitre durant le déploiement d'un robot dans le monde réel. Par exemple, lors de l'exécution d'un plan calculé avant la mission, un agent peut ne pas réussir une action, ou ne pas parvenir à la réaliser à l'instant exact qui était prévu, ou encore rencontrer une panne d'un capteur. Ces aléas peuvent remettre en cause la bonne exécution de la mission.

Pour ne pas compromettre la mission, l'autonomie du système doit être suffisante pour percevoir ces aléas et réagir en conséquence. La prévision de tous les aléas est une chimère, car dans le monde réel tout peut arriver. Mais la mise en place de méthodes de réparation adaptées à la mission lorsque cela est possible est essentielle pour améliorer la tolérance du système aux fautes et sa résilience globale.

Dans les sections suivantes, nous présentons les aléas considérés dans notre mission de chasse aux mines. Puis, nous détaillons les processus de supervision et d'exécution de HaucTioN permettant de réaliser les décisions prises tout en réagissant aux aléas perçus. Enfin, nous présentons comment notre architecture réexploite le protocole de décision pour formuler un nouveau problème de MRTA en ligne et réparer le plan.

7.1 ALÉAS DU DÉPLOIEMENT EN LIGNE

Les aléas sont des perturbations dans la mission initialement prévue, ils peuvent apparaitre à la suite d'événements internes au système ou externes. Les conséquences de ces perturbations peuvent être positives ou négatives, mais ce sont généralement les perturbations négatives dont il faut tenir compte dans la conception du système. En raison de perturbations négatives, même un plan initial optimal peut être mis en défaut. Pour les applications du monde réel, la question de l'optimalité du plan est donc souvent moins importante que celle de la résilience du système aux perturbations.

Pour mettre en place un comportement résilient, il est nécessaire de reconnaitre ces perturbations et d'y réagir. Usuellement, un ensemble de connaissances et méthodes expertes est fourni aux robots dans ce but. Il n'est cependant pas envisageable d'avoir de telles connaissances pour toutes les perturbations possibles du monde réel. Par conséquent, une première étape vers l'autonomie du système est d'isoler les perturbations les plus courantes pour la mission considérée.

7.1.1 *Aléas internes*

Nous considérons les aléas internes comme ceux propres aux moyens techniques mis en oeuvre dans le système, que ces moyens soient logiciels ou physiques. Par exemple, le dysfonctionnement d'un capteur ou la perte d'un message sont des aléas internes.

Panne et échec détectés par l'agent : Durant la mission, un premier aléa auquel un agent doit faire face est celui d'une panne interne à ses composants, tels qu'un capteur, compromettant ainsi sa contribution à la mission.

Plus généralement, un agent peut également rencontrer des échecs dans l'exécution de ses actions, par exemple en raison de la nature non déterministe de son exécution. De tels échecs doivent être perçus à la source, c.-à-d. par l'agent, et réparés lorsque cela est possible. Suivant la nature des aléas rencontrés, l'agent peut réussir à réparer son plan localement, ou devoir faire appel à ses coéquipiers pour essayer de le réparer globalement.

Panne et échec détectés par un coéquipier : Dans le cas idéal, l'agent rencontrant un échec dans son plan doit en informer les coéquipiers, l'agent peut cependant être dans l'impossibilité de transmettre ces informations, p. ex. en raison de ruptures de communication ou d'une panne majeure.

Lorsque les plans des robots sont dépendants entre eux, le manque d'information sur cette erreur peut compromettre la mission. Il est donc nécessaire de mettre en place des méthodes de supervision du plan des coéquipiers lorsque cela est possible. Par exemple, un agent peut devoir communiquer régulièrement un message pour signaler son état actif, si ses coéquipiers ne reçoivent pas le message, ils peuvent supposer que l'agent est hors service et réallouer ses tâches.

Pertes de communication : Dans notre architecture, les agents doivent communiquer à minima pour prendre des décisions et transmettre des informations sur l'exécution de leurs plans. Ainsi, des pertes de messages au sein de notre protocole d'enchère ou concernant l'exécution des tâches sont autant d'aléas internes devant être traités.

7.1.2 *Aléas externes*

Les aléas pouvant impacter notre système multirobot ne sont pas qu'internes au système. Le système multirobot peut également subir des perturbations externes.

Incertitudes sur l'environnement Les principaux aléas externes susceptibles d'être rencontrés par le système sont dûs à l'environnement incertain de la mission. En effet, les robots interagissent en permanence avec leur environnement, par conséquent, l'exécution de leurs actions ne saurait être déterministe si tous les paramètres liés à leur environnement ne sont pas connus. Dans un environnement dynamique tel que le monde sous-marin, l'environnement est au mieux partiellement connu.

Ces incertitudes sur l'environnement peuvent être à l'origine de nombreuses perturbations dans la bonne exécution de la mission. Par exemple, une divergence entre le courant attendu et le courant réel peut induire des retards dans l'exécution d'une tâche.

Pour faire face à un environnement incertain, il est possible de se reposer sur une perception active de cet environnement [Bes+18]. En prenant en compte les écarts entre prévision et réalité, les agents peuvent nourrir en permanence leur modèle de connaissance par les observations réalisées et ainsi adapter leurs décisions. Cependant, peu importe le degré de raffinement des informations captées sur l'environnement, cette mise à jour dynamique des connaissances ne peut garantir que le système n'aura pas à traiter les conséquences d'une divergence entre ce qui est attendu et ce qui rencontré. Ce qui est primordial pour

l'agent est donc d'être en capacité de réparer les décisions prises en conséquence des divergences observées entre prévision et réalité.

7.1.3 *Modification de la mission*

Le système multirobot est déployé dans le but de réaliser une mission. Si la mission est par nature dynamique, les changements de sa spécification, qu'ils soient volontaires ou non, sont autant de perturbations internes et externes dont les robots doivent tenir compte afin d'adapter le plan en cours.

Changement volontaire de la mission : L'objectif initial d'une mission de chasse aux mines est fixé par un opérateur en amont du déploiement du système, p. ex. "Couvrir la zone de mission avant l'instant t_x". Bien que notre système ne soit pas conçu dans le but d'être téléopéré, l'opérateur peut intéragir en ligne avec certains paramètres de la mission. En effet, la nature sensible de la mission implique de devoir réagir à des changements volontaires dans sa spécification, p. ex. "Couvrir la zone de mission avant l'instant $t_y < t_x$".

Nous considérons que ces changements volontaires sont issus de l'opérateur et répercutés au système par l'intermédiaire d'un robot à portée de communication de l'opérateur, p. ex. un USV *Helper*. La propagation de ces changements peut ensuite se faire de façon similaire à la propagation des informations dynamiques générées par les agents. En procédant ainsi, l'opérateur est susceptible de changer n'importe quelle information sur le contenu de la mission. Pour les robots réceptionnant ces changements, ces nouvelles informations sont des perturbations dont ils doivent tenir compte en adaptant leur plan si nécessaire.

Introduction de nouveaux objectifs : Comme détaillé dans le Chap. 2, la mission même de chasse aux mines implique l'apparition non contrôlée de nouveaux objectifs à accomplir. Ainsi, les robots doivent être aptes à intégrer en ligne des nouvelles tâches telles que l'identification d'un objet suspect.

Introduction de nouveaux agents ou retour d'un agent : Le déminage d'une zone par des moyens conventionnels peut durer de plusieurs dizaines d'heures à plusieurs jours, la mission à considérer est donc particulièrement longue. Pour une mission d'une telle durée, il est envisageable d'introduire volontairement une nouvelle unité dans un système en cours d'opération.

Plus généralement, durant l'accomplissement d'une mission les robots sont susceptibles de rencontrer des aléas internes les mettant temporairement hors service. Par exemple, un robot avec une panne de capteur peut être réparé et réintroduit. Ou un robot présumé hors service par ses coéquipiers peut revenir à portée de communication.

Ces agents doivent être intégrés dans le schéma de décision par les coéquipiers à portée afin de pouvoir prendre en charge une partie du plan.

7.2 SUPERVISER AVEC LA PLANIFICATION HIÉRARCHIQUE

Dans HaucTioN, les agents se reposent sur leur Multi-Robot HTN pour intégrer toutes les décisions prises (Chap. 6). Une fois qu'un agent est concerné par une décision, c.-à-d. une tâche lui est allouée, il peut commencer à l'exécuter. L'agent doit alors déterminer les actions locales à exécuter et superviser leur bonne exécution. Pour ce faire, nous exploitons une fois de plus la planification hiérarchique.

Pour déterminer les tâches à exécuter, nous aurions pu réutiliser directement le plan calculé lors de l'estimation, mais la dynamique de la mission impose d'être apte à le superviser et le replanifier si besoin. Dans notre approche, nous proposons de tirer profit des structures hiérarchiques complexes utilisées à la

décision pour à la fois superviser le plan et déterminer les prochaines actions à exécuter. Plus formellement, **nous mettons en place un processus similaire à celui utilisé pour l'estimation d'une mise afin de résoudre la supervision comme un problème de planification hiérarchique**. Ce processus est au coeur du protocole de supervision et réparation, il permet aux robots de superviser leur plan et déclencher des méthodes de réparation lorsque nécessaire.

Dans cette section, nous détaillons le fonctionnement de ce processus et son intrication dans le protocole de supervision et réparation de HaucTioN, qui est présenté dans la Fig. 6.6.

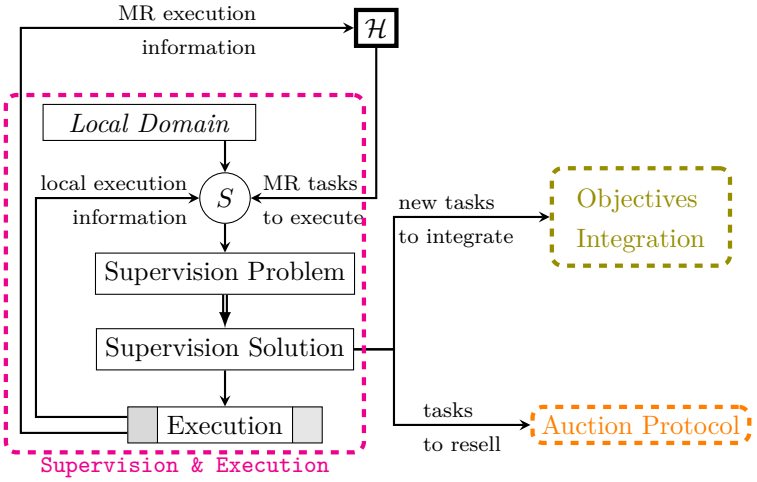


FIGURE 7.1 – Protocole de supervision et de réparation de HaucTioN. Les rectangles représentent des structures de données gérées par les robots. Le cercle indique un processus d'agrègation des informations pour construire le problème de supervision. La double flèche indique un appel à un planificateur HTN. Les flèches continues indiquent les interactions entre les structures de données et les processus. Le rectangle "Execution" correspond à la couche fonctionnelle du robot, en charge d'exécuter les actions locales et de faire remonter les informations perçues et reçues. Enfin, les blocs vert et orange représentent les appels aux autres protocoles de HaucTioN.

7.2.1 *Superviser, c'est estimer*

Dans la Sec. 6.5.1 nous avons mis en avant la nécessité pour un agent de décomposer localement non pas uniquement la tâche sur laquelle il mise, mais également les tâches qu'il aurait déjà acquises. En procédant ainsi, l'estimation est un problème de planification de la totalité de la mission dont une partie est à exécuter localement par le robot, c.-à-d. nous considérons dans le problème toutes les tâches multirobots connues et nous décomposons localement celles estimées et celles déjà acquises.

Dans notre approche, nous formulons un problème de supervision similaire à celui de l'estimation, au détail près que le problème de supervision n'inclut pas les décompositions locales de tâches à estimer, mais uniquement celles des tâches déjà acquises.

Par conséquent, et en reprenant les notations introduites à la Sec. 6.5.1, un problème de supervision :

— porte sur la planification des tâches multirobots acquises, notée Φ ;

— exploite $\mathcal{H}_\Gamma$, une version réduite du Multi-Robot HTN de l'agent où l'utilisation dans le plan des tâches acquises est forcée, c.-à-d. les décompositions alternatives à l'utilisation de tâches correspondant aux labels de Φ sont supprimées.

La création d'un problème de supervision (processus $\textcircled{S}$ dans Fig. 7.1) se fait donc presque directement en suivant le processus de création d'un problème d'estimation. La seule différence vient d'un élément qui

n'est pas considéré dans l'estimation lors de la planification initiale : l'intégration en ligne d'information sur l'exécution des tâches.

7.2.2 *Intégration des informations d'exécution au problème de supervision*

Lors de l'exécution de leurs tâches les agents génèrent des informations associées telles que "La tâche de label l_x a été débutée à l'instant 10". Nous nommons ces informations des *achievements*. En incluant ces informations au problème de supervision, les agents peuvent superviser le déroulement de leurs tâches multirobots.

Structure des achievements

Dans notre approche, les *achievements* sont représentent les informations d'exécution. Dans la Def. 7.1, la définition d'un *achievement* est donnée.

Definition 7.1 (Achievement)

Un achievement *est une structure de données décrivant une information relative à l'exécution d'une tâche multirobot virtuelle. Il est noté* ρ.

À un achievement *sont toujours associés les éléments suivants :*

- *Un label de tâche multirobot, noté* l_ρ ;
- *Une référence à l'élément concerné par* ρ. *Cet élément est toujours un* timepoint *issu de notre formalisme (Sec. 5.4.1) et peut soit correspondre au début ou à la fin de :*
  - *La tâche multirobot ;*
  - *Une condition, un effet ou une contrainte de la tâche.*
- *Un instant correspondant à la réalisation de* ρ, *noté* t_ρ.

Un exemple d'un *achievement* est donné dans Ex. 7.1.

Exemple 7.1 : Instance d'un *achievement*

Soient $AUVe1 \bullet (\, l_x \,)$. Pour accomplir la tâche $[\mathsf{start}, \mathsf{end}]mr\text{–}task(l_x)$, $AUVe1$ doit exécuter un ensemble d'actions locales. À l'instant t_s, $AUVe1$ commence une première action locale, la réalisation de $mr\text{–}task(l_x)$ vient donc de débuter et un *achievement* ρ_1 est généré.
Les données de cet *achievement* sont $\rho_1 = (l_x, \mathsf{start}, t_s)$.

Lors de la création d'un problème de supervision, les *achievements* sont intégrés de manière à refléter les connaissances de l'agent sur l'état du plan.

Intégration des achievements

Lors de la création d'un problème de supervision, chaque agent intègre au sein des tâches multirobots les *achievements* dont il a connaissance. Pour intégrer un *achievement* ρ, nous ajoutons une contrainte de synchronicité entre l'élément auquel se rapporte ρ et l'instant t_ρ. Par exemple, si ρ concerne le début d'une tâche $[\mathsf{start}, \mathsf{end}]mr\text{–}task(l_x)$: nous ajoutons une contrainte "$\mathsf{start} = t_\rho$".

Les timepoints compris dans une tâche multirobot sont donc fixés au fur et à mesure des *achievements* reçus. En procédant ainsi, la recherche de la solution est contrainte à respecter l'état connu du plan.

Exemple 7.2 : Intégration d'un *achievement* au problème de supervision

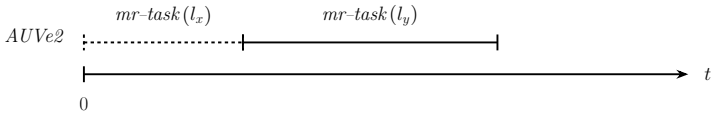
Soit la situation décrite dans Ex. 7.1. Pour rappel, dans notre formalisme HTN la tâche $mr\text{–}task(l_x)$ est de la forme $\mathtt{T}_p = (\, mr\text{–}task, \varnothing, \{l_x\}, \mathsf{pres}, \mathsf{adds}, \mathsf{dels}, \mathsf{start}, \mathsf{end}, \mathsf{constraints}\,)$ (Def. 5.8).
L'intégration de ρ_1 au Multi-Robot HTN de *AUVe1* se fait en ajoutant une nouvelle contrainte, c.-à-d. $(\mathsf{start} = t_s) \in \mathsf{constraints}$.

Gestion des dépendances complexes avec les achievements

Les problèmes de MRTA que nous devons résoudre comportent des dépendances complexes, les plans des agents obtenus à la résolution de tels problèmes peuvent donc être dépendants entre eux. Par conséquent, un agent doit superviser l'exécution des plans dont il dépend afin de s'assurer de la bonne exécution du sien. Nous illustrons ces dépendances dans Ex. 7.3.

Exemple 7.3 : Intégration d'information et vérification de plan en ligne

Soient $AUVe1 \bullet (\, l_x \,)$ et $AUVe2 \bullet (\, l_y \,)$ et une contrainte de précédence entre la fin de $mr\text{–}task(l_x)$ et le début de $mr\text{–}task(l_y)$.
Un plan possible pour *AUVe2* est : $mr\text{–}task(l_x)$ (par *AUVe1*) puis $mr\text{–}task(l_y)$ par *AUVe2* (Pl. 7.1). *AUVe2* ne peut rien faire tant que $mr\text{–}task(l_x)$ n'est pas finie.



Plan 7.1

Dans notre approche, notre supervision prend en compte le plan des coéquipiers grâce au Multi-Robot HTN. En effet, le rôle du Multi-Robot HTN est d'encoder toutes les connaissances connues par un agent sur les tâches multirobots, par conséquent, un agent supervise implicitement le bon déroulement des plans de ses coéquipiers lorsqu'il supervise le bon déroulement du sien à l'aide de son Multi-Robot HTN.

La mise à jour du Multi-Robot HTN d'un agent repose sur les partage d'information que font ses coéquipiers. Les agents doivent donc communiquer afin de transmettre leurs *achievements*. Sans cela, des blocages peuvent apparaitre en ligne.

Cependant, les contraintes pesant sur les communications sont fortes dans notre mission, notamment en raison des pertes de messages et de la bande passante limitée. Dans notre approche, nous faisons le choix d'imposer à chaque agent de renvoyer un historique des *achievements* dont il a connaissance dès qu'il génère un nouvel *achievement*. Par rapport à ne transmettre qu'une fois un *achievement*, c.-à-d. uniquement lorsqu'il est généré, cette méthode permet de réduire les risques de blocages mais implique un poids plus élevé par communication. Le choix d'une telle solution est notamment motivée par ces raisons :

— Le poids en données d'un *achievement* est faible, p. ex. environ $400B$ dans notre implémentation ;

— Les conséquences de la non réception d'un *achievement* peuvent être lourdes, p. ex. des blocages dans le plan d'un coéquipier ;

— Par la structure hiérarchique de la mission, le nombre de *achievements* est relativement faible. Les *achievements* ne concernent que les tâches multirobots, pas les actions locales.

Nous soulignons cependant que l'envoi de cet historique réduit uniquement les risques de blocages mais ne garantit pas leur absence.

7.2.3 *Gestion du temps courant dans le problème de supervision*

Nous formulons donc un problème de supervision sous la forme d'un problème de planification hiérarchique et nous contraignons certains des timepoints de ce problème avec des *achievements*. En sus de ce procédé, nous devons considérer le temps auquel se déroule la supervision. La raison en est très simple : si nous ne contraignons pas les timepoints pour lesquels aucun *achievement* n'est connu, alors la planification peut utiliser n'importe quelle valeur pour ces timepoints, y compris une valeur antérieure au temps auquel se déroule la supervision.

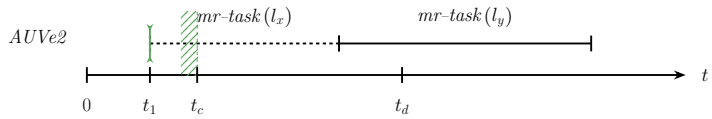
Dans notre approche, nous appelons le temps auquel se déroule la supervision le temps courant, noté t_c. Les temps associés aux *achievements* connus par l'agent sont par nature antérieurs à ce temps courant, ils appartiennent au passé de l'agent.

Contraintes avec le temps courant

Nous imposons aux agents qu'en l'absence d'informations sur un élément, c.-à-d. un *achievement*, **ils ne peuvent le supposer que comme supérieur ou égal au temps courant**. Pour ce faire, nous ajoutons une contrainte entre t_c et chaque timepoint non connu, c.-à-d. soit $t_{\neg\rho}$ un timepoint non connu, nous ajoutons la contrainte "$t_c <= t_{\neg\rho}$" Par conséquent, les timepoints pour lesquels aucun *achievement* n'est connu ne peuvent appartenir qu'au présent ou au futur de l'agent. Ces notions sont illustrées dans Ex. 7.4.

Exemple 7.4 : Contraintes avec le temps présent

Soient $AUVe1 \bullet (\, l_x \,)$ et $AUVe2 \bullet (\, l_y \,)$ et t_d une échéance imposée par l'opérateur sur la fin de $mr\text{–}task(l_y)$. Nous représentons dans Pl. 7.2 le plan de $AUVe2$.
Le temps courant t_c est représenté par la ligne verticale hachurée en vert. À gauche du temps actuel, les flèches verticales vertes représentent des timepoints connus correspondant au temps associé à leurs *achievements*.
À l'instant courant t_c, $AUVe2$ n'a reçu qu'un *achievement* de la part de $AUVe1$ l'informant du début de $mr\text{–}task(l_x)$ a l'instant t_1. Par conséquent, lors de la vérification de son plan, $AUVe2$ ajoute deux contraintes sur $[\mathsf{start}_x, \mathsf{end}_x] mr\text{–}task(l_x)$: $\mathsf{start}_x = t_1$ et $\mathsf{end}_x >= t_c$. De même pour $[\mathsf{start}_y, \mathsf{end}_y] mr\text{–}task(l_y)$ avec le temps courant, $\mathsf{start}_y >= t_c$.



Plan 7.2

Ces contraintes peuvent directement permettre à $AUVe2$ de vérifier la validité de son plan. Si $AUVe2$ estime qu'il lui faudra 10 unités de temps pour accomplir $mr\text{–}task(l_y)$ et que $t_c + 10 > t_d$ alors le plan n'est plus valide. Par conséquent, $AUVe2$ détermine qu'une réparation est nécessaire.

Le défi de la gestion du temps courant, des dépendances complexes, et des contraintes de communications

Il est important de noter que les contraintes imposées entre les éléments inconnus et le temps courant sont très restrictives en matière de supervision des plans. En effet, si un *achievement* existe mais qu'un agent le reçoit en retard, cela peut mettre en défaut l'existence d'une solution au problème de supervision de l'agent.

Une telle hypothèse peut sembler en contradiction avec la nature des communications sous-marines, qui peuvent induire des retards et des pertes de communication. Cependant, l'hypothèse inverse "les agents peuvent faire des suppositions sur l'état des tâches des coéquipiers", rentre en contradiction avec la nature de la mission, qui comporte des dépendances complexes que les agents doivent traiter avec prudence.

Dans la mise en place de notre architecture, nous avons fait le choix de prioriser une supervision stricte. Nous soulignons cependant quelques perspectives d'évolution du protocole de supervision dans le Chap. 9. Ces pistes d'amélioration peuvent rendre le protocole de supervision plus adapté aux contraintes de communication tout en conservant la même architecture et en respectant les dépendances complexes dans le plan.

7.2.4 *Résolution du problème de supervision à l'aide de la planification hiérarchique*

Soit $\mathcal{P}_\Phi$ un problème de supervision obtenu par un agent en intégrant les *achievements* connus par l'agent à son Multi-Robot HTN et en étendant, tel que pour l'estimation d'une mise, son Multi-Robot HTN avec ses capacités locales. Nous résolvons ce problème de planification hiérarchique en faisant appel au même solveur que pour l'estimation d'une mise et la résolution du WDP.

Détermination et exécution des actions locales

Si une solution à $\mathcal{P}_\Phi$ est obtenue, alors elle respecte les contraintes multirobot de l'agent. Le plan de mission n'est pas compromis et l'exécution des tâches par l'agent peut se poursuivre. De même que pour l'estimation, ce plan solution est composé de deux sous-plans (Sec. 6.5.4) :

— Π_μ qui regroupe les tâches multirobots ;

— Π_σ qui regroupe les tâches locales.

Dans notre approche, nous considérons qu'à une tâche multirobot de Π_μ est associé un ensemble d'actions locales à réaliser. Cet ensemble est déterminé grâce à la structure hiérarchique du problème de supervision. Chaque action locale dans le plan solution est associée à la première tâche multirobot ascendante dans le problème de supervision. Soit χ un label de tâche multirobot dans Π_μ, nous notons l'ensemble des actions locales associées à ce label comme σ_χ. Les actions locales sont ensuite exécutées selon Π_σ.

Dans notre approche, nous faisons le lien avec les *achievements* des tâches multirobots grâce à ces ensembles d'actions locales. Pour chaque action locale $\mathtt{T}_{p\text{–}\sigma}$ dans σ_χ :

— Si $\mathtt{T}_{p\text{–}\sigma}$ débute et qu'aucune autre action dans σ_χ n'a débuté, alors un *achievement* est généré pour χ (début de $mr\text{–}task(\chi)$) ;

— Si $\mathtt{T}_{p\text{–}\sigma}$ se termine et que toutes les autres actions dans σ_χ sont finies, alors un *achievement* est généré pour χ (fin de $mr\text{–}task(\chi)$).

Réparation locale

Il est important de noter que, comme le problème de supervision ne contraint pas les actions locales, le plan obtenu peut différer de celui prévu lors de l'estimation, ou même de celui trouvé lors de la dernière supervision. Ces variations permettent à l'agent de localement réparer son plan lorsque cela est nécessaire. Dans Ex. 7.5, une réparation locale est illustrée.

Exemple 7.5 : Réparation locale

Dans cet exemple, *AUVi1* doit identifier deux objets suspects. Pour faciliter la compréhension des labels multirobots et des tâches opérationnelles, le Multi-Robot Graph d'un objectif d'identification est représenté dans la Fig. 7.2. Les tâches du second objectif d'identification sont labélisées comme suit : l_0 devient $l_{0'}$, l_1 devient $l_{1'}$, ...
Soient *AUVi1* • $(l_0, l_1, l_2, l_4, l_{0'}, l_{1'}, l_{2'}, l_{4'})$ et une contrainte d'échéance spécifiant que le résultat d'une identification doit être rapporté au plus tard 15 unités de temps après la réalisation de l'identification (↔ dans le graphe).

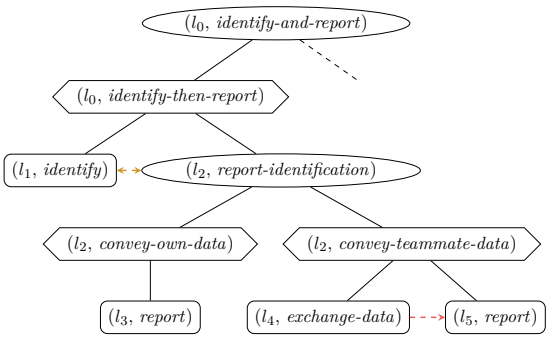
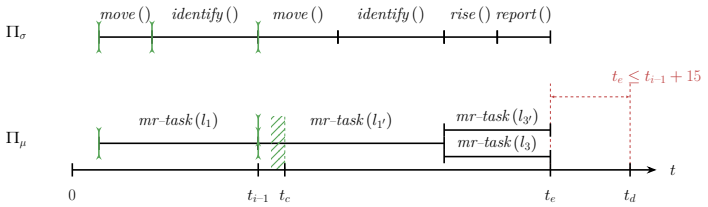


FIGURE 7.2 – Multi-Robot Graph d'un objectif d'identification. Ce graphe est obtenu en reprenant la décomposition illustrée dans la Fig. 5.1.

Par exemple, pour $[\mathsf{start}_1, \mathsf{end}_1]mr\text{–}task(l_1)$ et $[\mathsf{start}_2, \mathsf{end}_2]mr\text{–}task(l_2)$, est ajoutée la contrainte suivante $\mathsf{end}_2 <= \mathsf{end}_1 + 15$. Par la structure hiérarchique de la mission, cette contrainte est héritée aux descendants de l_2, c.-à-d. (l_3, l_4, l_5).
AUVi1 a optimisé son plan local en prévoyant de ne remonter à la surface qu'une fois, c.-à-d. de rapporter les données de la première identification en même temps que celles de la seconde. Son plan est représenté dans Pl. 7.3.



Plan 7.3

AUVi1 a terminé la première identification (l_1) au temps t_{i-1} et a estimé une durée de : 3 unités de temps pour se déplacer de la première identification à la seconde, 6 pour réaliser la seconde action d'identification, et 5 pour remonter à la surface et transmettre les données. La contrainte d'échéance portant sur l'identification est respectée, car $3 + 6 + 5 < 15$.
Durant son déplacement vers la seconde identification, *AUVi1* prend 2 unités de temps de retard. Lors de la vérification de son plan il détermine qu'il ne peut plus respecter la contrainte d'échéance car $5 + 6 + 5 > 15$, le plan prévu n'est plus possible.
Cependant, son processus de supervision peut initier une réparation locale. Au regard de ses informations, *AUVi1* peut trouver un plan solution au problème de supervision consistant à arrêter le déplacement en cours pour directement remonter transmettre les informations de la première identification. Un tel plan est possible car aucune contrainte n'oblige à réaliser l_3 après $l_{1'}$, ce plan respecte donc les contraintes multirobots.

Dans le cas d'une réparation locale, si l'ensemble σ_χ a été modifié, les tâches locales déjà accomplies dedans ne sont pas répétées.

Appel à une réparation globale

Si aucune solution n'est trouvée à $\mathcal{P}_\Phi$, alors cela signifie que les contraintes multirobots ne peuvent plus être respectées, une réparation globale est nécessaire. Pour essayer de réparer son plan, l'agent devient commissaire-priseur et remet en vente les tâches qu'il ne peut plus assumer en faisant appel au processus de décision. S'il n'arrive pas à les réallouer, l'agent abandonne les tâches jusqu'à pouvoir reprendre la mission. Le choix des tâches à abandonner peut se faire selon des critères comme l'impact de la tâche sur le plan des coéquipiers ou l'impact sur les performances de la mission.

7.3 RÉPARER LE PLAN À L'AIDE DU PROTOCOLE DE DÉCISION

Dans la version actuelle de notre approche, un agent peut devoir faire appel à une réparation globale pour ces deux raisons :

— L'intégration de nouveaux objectifs ;

— La réallocation d'objectifs que l'agent ne peut plus assumer, c.-à-d. une modification de précédentes décisions.

Les causes de ces réparations sont des événements imprévisibles, tels que des perturbations dans le plan ou la détection d'un objet suspect. Ces aléas sont à l'origine d'un nouveau problème de MRTA devant être résolu en ligne afin de pouvoir poursuivre la mission.

Pour résoudre ces problèmes de MRTA, nous faisons appel au protocole de décision décrit dans le Chap. 6. Dans cette section, nous décrivons comment ce protocole est utilisé en ligne.

7.3.1 *Variations spécifiques à l'utilisation en ligne du protocole de décision*

Afin d'en faciliter la compréhension, dans le Chap. 6 nous avons présenté le protocole de décision HaucTioN sous l'angle de la planification initiale et n'avons qu'implicitement abordé les spécificités de son utilisation en ligne. Cependant, certaines précisions quant à son fonctionnement en ligne doivent être explicitées afin de pouvoir l'exploiter durant la mission.

Dépendances des enchères

En premier lieu, il est important de souligner que la résolution de multiples problèmes de MRTA implique la tenue de multiples enchères. Or, ces problèmes ne peuvent être traités de façon indépendante, car notre mission comporte des dépendances complexes. Par conséquent, les enchères utilisées pour résoudre ces problèmes ne sont pas indépendantes les unes des autres.

La résolution d'une enchère doit se faire en considérant les décisions prises lors d'une autre enchère. Ainsi, il est nécessaire d'intégrer les décisions antérieures à l'ouverture d'une nouvelle enchère. Un des avantages directs de notre approche est de pouvoir surmonter ce problème, pour ce faire nous nous reposons sur le Multi-Robot HTN.

En effet, chaque agent entretient un Multi-Robot HTN dont le rôle est d'encoder toutes les connaissances connues par l'agent sur les tâches multirobots, c.-à-d. les objectifs de la mission. Cette structure de données est donc mise à jour dynamiquement au fur et à mesure des décisions prises et de l'exécution des tâches.

Dans notre schéma de décision, le commissaire-priseur utilise son Multi-Robot HTN pour définir le problème de MRTA à résoudre dans un tour d'enchère. Par conséquent, les décisions prises à des enchères antérieures, et donc les dépendances à d'autres enchères, sont considérées.

Des enchères en parallèle

Le second point faisant directement suite à la tenue de multiples enchères concerne la possibilité pour le système de générer plusieurs enchères en même temps. Dans notre approche, n'importe quel agent peut devenir commissaire-priseur pour procéder à une réparation globale. Aussi, des aléas proches dans le temps et subis par des agents différents peuvent résulter en l'ouverture parallèle d'enchères.

Pour rappel, au sein d'un tour d'enchère, un agent peut remporter au plus une mise, car ses mises sont estimées de façon indépendante. Or, la participation d'un agent à des tours d'enchères différentes en même temps peut casser cette contrainte d'indépendance des mises. Par exemple, si l'agent participe à deux tours d'enchères à la fois, il est susceptible de remporter deux mises. Mais ces deux mises ont été formulées de façon indépendante, par conséquent, l'estimation que l'agent a faite pour chacune des mises peut ne plus être possible lorsque l'agent doit considérer les éléments de l'autre mise.

Pour pallier ce problème, une approche usuelle des schémas d'enchères en ligne est de se reposer sur un jeton unique de commissaire-priseur. Ce jeton est acquis lorsqu'un agent souhaite ouvrir une enchère et permet de garantir que le système ne peut avoir à traiter plusieurs enchères en même temps [VMO07 ; LAL04]. Cependant, cette approche n'est possible qu'avec des communications parfaites, ce qui n'est pas envisageable dans le cadre de notre mission.

Dans notre approche, nous surmontons ce problème en imposant aux robots de réguler leurs participations aux enchères avec les conditions suivantes :

— Un agent ne peut pas participer à une enchère s'il participe déjà à une autre, cette condition est nécessaire pour assurer l'indépendance des mises ;

— Un agent ne peut pas ouvrir une enchère s'il a connaissance d'une autre enchère en cours, cette condition limite le nombre d'enchères parallèles.

Avec ces sécurités, le système peut avoir à traiter plusieurs enchères en même temps, mais l'indépendance des mises est respectée.

Intégration des informations d'exécution

Dans la Sec. 7.2.1, nous avons décrit les similitudes entre un problème de supervision et un problème d'estimation de la planification initiale. Puis, dans la Sec. 7.2.2 nous en avons en souligné les différences, notamment dues à la nécessité d'intégrer les informations générées en ligne au problème de supervision. Enfin, nous avons conclu dans Sec. 7.2.4 sur comment le problème de supervision était formulé en exploitant les processus utilisés pour la modélisation du problème d'estimation tout en intégrant les informations de l'exécution. Or, un nouveau parallèle entre problème d'estimation et problème de supervision peut être fait dans le cas d'une estimation en ligne.

En effet, lors de la résolution d'un problème de MRTA en ligne, il est nécessaire de considérer les décisions antérieures, mais également l'état du plan en cours. Comme vu précédemment, ces informations sont encodées dans le Multi-Robot HTN de l'agent au fur et à mesure de l'avancée de la mission. Ainsi, l'estimation d'une mise en ligne doit se faire en intégrant les décisions antérieures, mais également les informations d'exécution. En ce qui concerne les décisions antérieures, le processus d'estimation décrit au Chap. 6 permet déjà de les supporter, car une décision au tour d'enchère précédent est assimilable à une décision antérieure. Cependant, l'intégration des informations d'exécution au processus d'estimation des mises nécessite de considérer le temps courant et les *achievements*. Nous le faisons en les intégrant au problème d'estimation de la même manière que pour le problème de supervision (Sec. 7.2.2). Le problème d'estimation en ligne ainsi formé est donc quasiment égal au problème de supervision à ceci près que

le problème de supervision ne décompose localement que les tâches déjà acquises, là où le problème d'estimation décompose également les tâches sur lesquelles le robot mise.

Divergence des connaissances

Pour résoudre les problèmes de MRTA apparaissant en ligne, les robots se reposent sur leurs connaissances des décisions prises et de l'état du plan. Dans un environnement où les communications sont contraintes, ces connaissances peuvent diverger. Par exemple, un message d'exécution diffusé par un agent peut avoir été reçu par un second agent et perdu pour un troisième agent.

Mais la présence de robots moins informés que d'autres n'est pas le seul problème dont il faut tenir compte. Le principal défi dans la prise en compte de ces divergences de connaissance réside dans la résolution des incohérences auxquelles elles peuvent mener. En effet, les décisions prises par des robots ayant des connaissances différentes peuvent être incompatibles. Dans Ex. 7.6, nous illustrons une telle incohérence.

Exemple 7.6 : Décisions et incohérences en ligne

Soient $AUVe1 \bullet (\, l_x \,)$ et $AUVe2 \bullet (\, l_y \,)$. Durant la mission $AUVe1$ ne répond plus et sa tâche est réallouée à $AUVe2$ qui prévoit de faire $mr\text{–}task(l_y)$ puis $mr\text{–}task(l_x)$. Par conséquent, une contrainte de précédence $(l_y \rightarrow l_x)$ est ajoutée par $AUVe2$, cette contrainte est intégrée par les coéquipiers coopérant avec $AUVe2$.
Puis, $AUVe3$ obtient $mr\text{–}task(l_z)$ avec comme plan : $mr\text{–}task(l_y)$ (par $AUVe2$) puis $mr\text{–}task(l_z)$ (par $AUVe3$) puis $mr\text{–}task(l_x)$ (par $AUVe2$). Ici, $AUVe3$ respecte la contrainte ajoutée par $AUVe2$ $(l_y \rightarrow l_x)$ et il ajoute deux autres contraintes $((l_y \rightarrow l_z)$ et $(l_z \rightarrow l_x))$
Enfin, $AUVe1$ entre en contact avec $AUVe3$ et l'informe qu'il a terminé l_2. Si $AUVe3$ n'a pas encore terminé $mr\text{–}task(l_z)$, alors son plan n'est plus valide. La contrainte de précédence $l_z \rightarrow l_x$ ne peut pas être respectée.

Nous considérons que le processus de décision de HaucTioN débute par une phase de concertation des agents souhaitant participer à une enchère. Durant cette phase de concertation, les agents comparent et fusionnent leurs connaissances jusqu'à pouvoir faire émerger une connaissance commune à tous les participants de l'enchère. Ensuite, le premier tour d'enchère peut commencer.

Comme introduit dans Sec. 4.3.4, la résolution des divergences de connaissance n'est pas un problème trivial. Par exemple, si deux robots possèdent deux informations différentes se rapportant à une même tâche, ce n'est pas parce que l'une des informations est plus récente que l'autre qu'elle doit supplanter l'autre. Toutes deux peuvent être vraies au regard de la vérité de terrain du système. Nous illustrons ce point dans Ex. 7.7.

Exemple 7.7 : Informations divergentes

En reprenant la situation de Ex. 7.6, soit $AUVe4$ un robot étant rentré en contact avec $AUVe1$ puis $AUVe2$.
$AUVe4$ a deux informations concernant l'allocation de $mr\text{–}task(l_x)$: $AUVe1 \bullet (\, l_x \,)$ et $AUVe2 \bullet (\, l_x \,)$. Si $AUVe4$ reçoit un message de $AUVe1$ indiquant que la tâche $mr\text{–}task(l_x)$ est finie puis un autre de $AUVe2$ indiquant que la tâche $mr\text{–}task(l_x)$ a débuté, il ne peut trancher sur quelle information doit être gardée ou rejetée.

Dans notre approche, les connaissances dynamiques d'un agent sur les décisions prises et l'état du plan sont encodées au sein du Multi-Robot HTN de l'agent. C'est donc sur cette structure que doit s'appuyer le processus d'unification des connaissances des agents.

Dans la version actuelle de nos travaux, nous n'avons pas encore eu le temps d'adresser en profondeur ces problèmes. Aussi, nous nous appuyons sur une méthode simple, mais peu efficace, pour surmonter ce défi. Nous considérons que les connaissances du commissaire-priseur supplantent toutes les autres. Chaque fois qu'un tour d'enchère débute, les enchérisseurs à portée adoptent le Multi-Robot HTN utilisé pour ouvrir l'enchère, c.-à-d. le Multi-Robot HTN du commissaire-priseur. Dans le Chap. 9, nous discutons plus en profondeur des conséquences d'une solution aussi radicale et des perspectives pour surmonter d'une meilleure façon ce problème.

Robustesse aux pertes de message

L'utilisation en ligne du protocole de décision implique de devoir se soumettre aux contraintes techniques affectant le système durant la mission. Une des principales contraintes qui pèsent sur le système réside dans la capacité à communiquer des robots. Dans le cadre de notre mission, nous considérons que les communications sont bidirectionnelles et offrent une grande portée, mais leur débit est faible et les pertes de messages probables.

En ligne, ces contraintes de communication affectent les agents dans l'exécution de leur plan, mais également lors de leurs prises de décision. Avec l'étape initiale d'une enchère, consistant à adopter le Multi-Robot HTN du commissaire-priseur comme référence, les pertes de communication causant une divergence des informations d'exécution n'affectent pas les enchères. Cependant, il reste possible de perdre des messages associés au processus de décision, p. ex. la mise d'un robot n'est pas réceptionnée par le commissaire-priseur.

Le Tab. 7.1 résume les messages échangés dans le protocole de décision et les conséquences directes de leur perte.

Type de message	Émetteur	Destinataire	Conséquences de la perte
Annonce du 1er tour (Ouverture de l'enchère)	CP	E	Le destinataire ne participera pas à cette enchère qu'importe les messages d'annonces reçus par la suite.
Annonce d'un nouveau tour (Prolongation de l'enchère)	CP	E	Le destinataire ne participera pas à ce tour, mais pourra réagir à tous les messages reçus par la suite.
Mise	E	CP	La mise ne sera pas considérée dans la résolution du WDP.
Récompense	CP	E	La récompense est annulée par le CP, les labels remportés sont remis en vente.
Confirmation d'une récompense	E	CP	La récompense est annulée par le CP, les labels remportés sont remis en vente. L'enchérisseur exécute tout de même ses tâches, possibilité de double allocation.
Clôture du tour	CP	E	L'enchérisseur n'a pas connaissance des décisions prises dans ce tour pour les autres agents. Ce manque d'information peut être comblé par la réception d'un message d'annonce ou de clôture d'un nouveau tour.

TABLE 7.1 – Récapitulatif des messages utilisés au sein du processus de décision et des conséquences de leur perte. Le commissaire-priseur et l'enchérisseur sont respectivement notés "CP" et "E".

Dans ce tableau, il est important de noter le cas particulier de la perte d'une confirmation de récompense. En effet, il est possible qu'un enchérisseur réceptionne une récompense, mais que le commissaire-priseur ne

réceptionne jamais l'accusé de réception. Dans un tel cas, l'enchérisseur victorieux exécutera les tâches de la récompense comme convenu, mais le commissaire-priseur remettra en vente ces tâches au tour d'enchère suivant. Les performances de la mission sont donc impactées par la possibilité d'effectuer plusieurs fois la même tâche.

Néanmoins, dans notre mission des aléas tels que des pannes peuvent survenir. Il est donc toujours nécessaire d'utiliser des processus de supervision des actions des coéquipiers. Par exemple pour réallouer les tâches d'un coéquipier ne communiquant plus depuis trop longtemps, dans une telle situation il est également possible d'effectuer plusieurs fois une même tâche. Les conséquences de la perte de la confirmation d'une récompense sont similaires, elles doivent donc être traitées de la même manière par des méthodes de réparation adaptées à la mission.

Importance des critères d'arrêts

Pour finir, l'utilisation en ligne du protocole de décision implique que les robots doivent établir des sécurités pour faire face aux conséquences des ruptures de communication. Par exemple, dans notre implémentation le commissaire-priseur impose une échéance pour estimer les mises, plutôt qu'attendre tant que toutes les mises des robots à portée ne sont pas réceptionnées.

De plus, si aucune mise n'est reçue, ou que la résolution d'un tour d'enchère ne mène à aucune allocation, le commissaire-priseur met fin à l'enchère. Il pourra réessayer d'allouer les tâches restantes en ouvrant plus tard une nouvelle enchère. Cette nouvelle enchère est débutée soit lorsqu'une échéance temporelle est atteinte, soit lorsqu'il rentre en contact avec de nouveaux agents.

7.3.2 *Réparer en intégrant de nouveaux objectifs*

Un des besoins les plus importants de notre mission est l'intégration de nouveaux objectifs en cours de mission. Cela est principalement dû à la découverte de MILCO, des objets suspects devant être identifiés.

Dans notre approche, le robot qui détermine de nouveaux objectifs est en charge de les allouer. Il devient alors commissaire-priseur d'une nouvelle enchère. Dans notre protocole d'enchère, c'est le Multi-Robot HTN du commissaire-priseur qui est utilisé pour spécifier le problème de MRTA à résoudre. Cette structure étant unique à chaque robot, le robot en charge des nouveaux objectifs doit les intégrer à son propre Multi-Robot HTN.

Pour procéder à cette intégration, le commissaire-priseur construit un nouveau Multi-Robot HTN composé de ces nouveaux objectifs, puis il le fusionne à son propre Multi-Robot HTN. Enfin, il met en vente l'article résultant en indiquant les labels des nouveaux objectifs comme labels en vente.

Intégration des nouveaux objectifs au Multi-Robot HTN de l'agent

Dans la Sec. 6.2.3, nous avons décrit comment un Multi-Robot HTN pouvait être créé à partir d'un *auction problem*. Ici, nous adressons le cas particulier de la création d'un Multi-Robot HTN à partir de deux autres, c.-à-d. la fusion de deux Multi-Robot HTN. C'est un processus exclusif à l'intégration de nouveaux objectifs.

Lorsque des objectifs sont créés par un agent, ils doivent être décrits sous la forme d'un Multi-Robot HTN, cependant, en ligne un plan peut déjà être préexistant à cette création, c.-à-d. le Multi-Robot HTN de l'agent n'est pas vide. Pour intégrer ces nouvelles tâches au Multi-Robot HTN de l'agent, nous :

1. Construisons un nouvel Multi-Robot HTN tel que décrit dans la section précédente, noté $\mathcal{H}_{new}$;
2. Fusionnons ce nouvel Multi-Robot HTN à celui préexistant, noté $\mathcal{H}_{old}$.

Cette fusion se fait très simplement à l'aide du formalisme HTN. $\mathcal{H}_{new}$ et $\mathcal{H}_{old}$ étant non-vides, ils comportent respectivement une racine $root_{new}$ et $root_{old}$. Pour créer une continuité entre les éléments de $\mathcal{H}_{new}$ et $\mathcal{H}_{old}$, nous créons une nouvelle racine *root* à l'aide d'un nouveau label. Puis, nous ajoutons une nouvelle décomposition spécifiant que la tâche *root* se décompose en $root_{new}$ et $root_{old}$. Ainsi, la fusion $\mathcal{H}_{new}$ et $\mathcal{H}_{old}$ se résume à un nouvel Multi-Robot HTN, noté $\mathcal{H} = (\mathcal{P}, \mathcal{D}_{op}, \mathtt{M}_t)$, où :

- $\mathcal{P} = (\mathcal{D}, \mathtt{s_i}, \mathtt{tn_i})$, où :
  - $\mathcal{D} = \mathcal{D}_{new} \cup \mathcal{D}_{old}$, la nouvelle tâche *root* et la nouvelle méthode décomposant *root* en sous-tâche $root_{new}$ et $root_{old}$ est également intégrée à $\mathcal{D}$;
  - $\mathtt{s_i} = \mathtt{s_i}_{new} \cup \mathtt{s_i}_{old}$;
  - $\mathtt{tn_i}$ est composé uniquement de la nouvelle racine, c.-à-d. *root*.
- $\mathcal{D}_{op} = \mathcal{D}_{op\text{–}new} \cup \mathcal{D}_{op\text{–}old}$;
- $\mathtt{M}_t = \mathtt{M}_{t\text{–}new} \cup \mathtt{M}_{t\text{–}old}$.

Une illustration du Multi-Robot HTN résultant de ce processus est montrée dans Fig. 7.3.

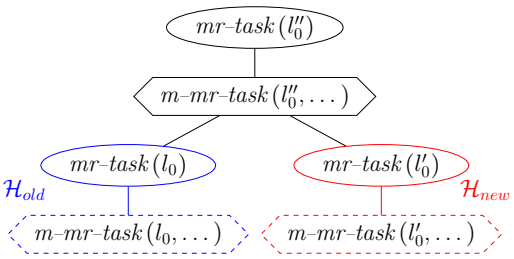


FIGURE 7.3 – Représentation de la fusion de deux Multi-Robot HTN. $\mathcal{H}_{new}$ est fusionné à $\mathcal{H}_{old}$. Pour ce faire une nouvelle racine $mr\text{–}task\left(l''_0\right)$ est créée et décomposée en la racine de $\mathcal{H}_{old}$, $mr\text{–}task\left(l_0\right)$ et $\mathcal{H}_{new}$, $mr\text{–}task\left(l'_0\right)$.

Mise en vente des nouveaux objectifs

Une fois les nouveaux objectifs intégrés, le protocole d'enchère est suivi en commençant par diffuser un message d'annonce contenant le Multi-Robot HTN de l'agent et une liste de labels en vente au sein de ce Multi-Robot HTN. Les labels de cette liste correspondent aux labels des nouveaux objectifs. Dans Ex. 7.8, nous illustrons une réparation du plan par intégration et mise en vente de nouveaux objectifs.

7.3.3 *Réparer en revendant*

Lorsque qu'un agent termine la résolution du problème de supervision sans trouver de plan, cela signifie qu'il n'est plus possible de respecter les contraintes multirobots au sein de son Multi-Robot HTN. L'agent déclenche alors une réparation en essayant de revendre les tâches qu'il ne peut plus assumer jusqu'à ce qu'un plan solution au problème de supervision soit trouvé. Nous illustrons un tel cas de réparation dans Ex. 7.9.

Détermination des tâches à revendre

Dans notre approche, nous nous appuyons sur le processus de supervision pour déterminer les tâches à revendre. Lorsque le problème de supervision n'a pas de solution et qu'une réparation par revente est nécessaire, l'agent en difficulté créer un nouveau problème de supervision en enlevant une tâche parmi les tâches qu'il n'a pas encore réalisé, c.-à-d. un label parmi les labels remportés Φ.

Exemple 7.8 : Réparation globale

Soit *AUVe1*, un AUV possédant le Multi-Robot HTN suivant et accomplissant la tâche de label l_x qui est une tâche de couverture. Durant sa couverture, *AUVe1* détecte un objet suspect nommé m_1 pour MILCO.

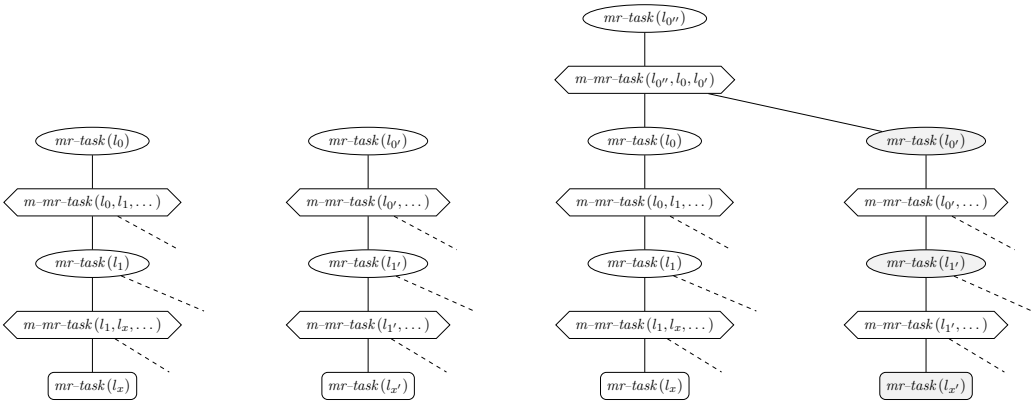


FIGURE 7.4 – Intégration et mise en vente de nouveaux objectifs par un agent. À gauche, le Multi-Robot HTN d'origine de l'agent (racine l_0) et un nouveau Multi-Robot HTN composé des nouveaux objectifs (racine $l_{0'}$). À droite, le Multi-Robot HTN de l'agent après fusion avec les nouveaux objectifs, dans ce processus, une nouvelle racine ($l_{0''}$) est créée. Les tâches mises en vente par l'agent sont représentées en gris.

AUVe1 exploite alors le processus d'instanciation de ses connaissances multirobots pour créer $\mathcal{H}_{new}$, un Multi-Robot HTN comportant uniquement les nouveaux objectifs (Sec. 6.2.3).
Ensuite, *AUVe1* fusionne $\mathcal{H}_{new}$ à son propre Multi-Robot HTN.
Enfin, *AUVe1* ouvre une nouvelle enchère en transmettant son Multi-Robot HTN comme $\mathcal{H}_\delta$ et *labels for sell* $= (\,0', 1', x', \dots)$.

Exemple 7.9 : Réparation globale

Soit la situation décrite dans Ex. 7.5. Si *AUVi1* n'a pas d'alternative lui permettant de rapporter lui-même les données de la première identification à temps, p. ex. parce qu'il n'a pas assez d'énergie pour remonter deux fois, une contrainte multirobot ne peut être respectée. Le processus de supervision de *AUVi1* ne peut alors pas déterminer de plan. *AUVi1* ne peut plus assumer l_3 et donc l_2, une réparation globale est nécessaire.
AUVi1 ouvre une enchère pour revendre ces objectifs. Pour ce faire, il s'appuie sur la structure hiérarchique utilisée pour mettre en vente l_2, par conséquent, l'action l_3 est mise en vente mais également les autres tâches multirobots correspondant à la décomposition *convey–teammate–data*, c.-à-d. l_4 et l_5.
A priori, personne ne pourra acquérir l_3, car cette décomposition n'est utilisable que par le robot possédant les données d'identification et *AUVi1* ne peut plus accomplir cette tâche, mais, grâce à la structure hiérarchique, le plan peut être réparé en allouant la seconde décomposition. Si la seconde décomposition n'est pas achetée, *AUVi1* est contraint d'abandonner l_2 et l_3, ou d'essayer de réparer en revendant le rapport de la seconde identification, c.-à-d. $l_{2'}$ et $l_{3'}$.

Si le nouveau problème peut être résolu, alors l'agent procède à une réparation globale en essayant de réallouer le label qu'il a retiré de ses obligations. Sinon, l'agent réitère la procédure avec une autre tâche, puis avec deux tâches, etc., jusqu'à épuisement des possibilités.

Il est important de noter que les tâches sont uniquement retirées des obligations de l'agent, c.-à-d. que l'agent ne les décompose plus localement. Elles font cependant toujours partie du Multi-Robot HTN de l'agent, la résolution d'un tel problème de supervision se fait donc en considérant toujours ces tâches. Ainsi, si le problème est résolu cela signifie qu'un plan théorique existe où les tâches retirées (ou celles d'une décomposition alternative) sont accomplies par un coéquipier.

Dans ce processus d'essai erreur, le choix des tâches à retirer à l'agent est aléatoire parmi les tâches qui n'ont pas encore été réalisées. Cette approche naïve nous permet de mettre en place les bases nécessaires à des méthodes plus évoluées que nous présentons en perspectives dans le Chap. 9.

7.3.4 *Gestion des échecs de la réparation*

Lorsqu'un agent rencontre un aléa, il exploite des méthodes spécifiques pour *essayer* de réparer le plan. Dans le cadre opérationnel que nous considérons, rien ne garanti qu'une nouvelle solution puisse exister pour faire face à un aléa. Par exemple, la perte du seul robot capable d'accomplir un type de tâche en particulier signifie l'impossibilité pour le système de réaliser ces tâches, p. ex. tous les *identifiers* rencontrent une panne.

De plus, la résolution d'un problème de MRTA en ligne grâce à des enchères peut ne pas être possible, p. ex. aucun enchérisseur n'est capable d'accomplir une tâche en vente.

Des méthodes spécifiques sont donc nécessaires pour permettre au système de réagir à ces revers.

Report d'une enchère

Les processus de réparation mis en oeuvre s'appuient sur les coéquipiers à portée, ce qui limite les possibilités. Par exemple lors d'une enchère, il est possible qu'aucun participant ne puisse formuler une offre adéquate. Dans un tel cas, l'enchère est arrêtée grâce aux critères d'arrêts présentés dans la Sec. 6.7.

Le commissaire-priseur conserve les tâches non-allouées en vue d'essayer à nouveau de les allouer lors d'une prochaine enchère. La tenue de cette prochaine enchère est déclenchée soit par :

— Une échéance, p. ex. l'agent attend 30 unités de temps avant d'ouvrir une nouvelle enchère ;

— Un aléa, p. ex. de nouveaux objectifs sont détectés, ils sont mis en vente en même temps que les tâches non-allouées.

Abandon d'une tâche

Lors d'une tentative de réparation par revente, l'agent essaye de résoudre des problèmes de supervision par essai/erreur pour déterminer les tâches à revendre, mais il est que cette recherche soit un échec, c.-à-d. que l'agent a épuisé toutes ses possibilités.

Cela signifie que même en ne conservant qu'une tâche, le problème de supervision n'est jamais résolu. Un tel cas est possible en raison des contraintes pesant sur les tâches et des aléas techniques que peuvent rencontrer les agents. Par exemple, si un agent doit réaliser une tâche avant une échéance donnée qui est dépassée, il n'existe aucune solution au problème de supervision. De même, si l'agent rencontre une panne, alors il peut ne plus pouvoir accomplir un type de tâche opérationnelle, p. ex. la perte d'un capteur lié à l'identification signifie l'impossibilité pour l'agent d'accomplir des tâches d'identification.

Dans un tel cas, les tâches non réalisées par l'agent sont abandonnées. Dans la version actuelle de notre approche, nous nous servons d'un encodage particulier des *achievements* pour signaler l'abandon d'une tâche et ainsi, diffuser cette information grâce au protocole usuel d'échange d'information sur l'exécution. Lorsqu'une tâche est abandonnée, un *achievement* avec un temps négatif est créé. Ensuite, lorsqu'un

agent intègre ses *achievements* à un problème de planification, si le temps est négatif alors la tâche est neutralisée en supprimant la totalité de ses conditions, effets, et contraintes. En procédant ainsi, la mission peut se poursuivre sans les tâches neutralisées tout en conservant la structure hiérarchique du Multi-Robot HTN. Dans notre approche, l'abandon d'une tâche est considéré comme définitif, ce point est notamment discuté dans le Chap. 9.

7.4 PROTOCOLE COMPLET DE HAUCTION

Dans ce chapitre, nous avons décrit les principaux aléas qui peuvent être rencontrés par notre système chasseur de mines. Puis, nous avons décrit notre protocole de supervision et réparation dont le rôle principal est de réagir aux perturbations causées par ces aléas. Nous avons notamment décrit comment les méthodes de réparation exploitent le processus de décision pour réparer le plan de mission. Ainsi, le processus de décision et les processus de supervision et de réparation sont liés au sein de HaucTioN. Le protocole général résultant est représenté dans la Fig. 7.5. Les clés de compréhension de ce protocole reposent notamment sur la Fig. 4.3, qui en présente une vision simplifiée, et les Fig. 6.6 et Fig. 6.6, qui présentent les protocoles de décision et de supervision. Dans ce protocole, les blocs de couleurs correspondent aux processus de :

— Concertation des participants d'une enchère (Sec. 7.3.1) pour le bloc bleu "Multi-Robot HTN Merging" ;

— Intégration de nouveaux objectifs au sein du Multi-Robot HTN du commissaire-priseur (Sec. 7.3.2) pour le bloc vert "Objectives Integration" ;

— Résolution d'un problème de MRTA par la tenue d'enchère (Sec. 6.3 à Sec. 6.7) pour le bloc orange "Auction Protocol" ;

— Exécution, supervision, et réparation du plan (Sec. 7.2 et Sec. 7.3) pour le bloc violet "Supervision & Execution". Par souci de clarté, ce bloc n'a été représenté dans le protocole que pour le robot commissaire-priseur.

7.5 SYNTHÈSE DE L'UTILISATION EN LIGNE ET DES TOLÉRANCES AUX FAUTES

Bien que le protocole complet de HaucTioN place l'architecture dans une dynamique d'utilisation en ligne, il ne garantit pas une exécution sans faille de la mission. Comme vu précédemment, les robots peuvent par exemple n'avoir d'autres choix que l'abandon d'une tâche. De plus, la réparation du plan de mission par un agent est intrinsèquement liée à l'état de son réseau de communication, plus le nombre de coéquipiers participant à la réparation sera grand, meilleures seront les chances de réparer le plan. Cependant, même sous conditions de communication parfaites une réparation peut s'avérer impossible, ne laissant d'autres choix aux agents que l'abandon de tâches et l'échec d'une partie de la mission.

Dans notre approche, les données de la mission sont encodées dans des structures hiérarchiques complexes à partir desquelles sont créés des problèmes de planification hiérarchique. Ces problèmes hiérarchiques sont résolus à des étapes clés de nos différents protocoles, ce sont donc eux qui composent les principaux points d'échecs. Dans le Tab. 7.2, nous résumons les échecs que peuvent rencontrer ces processus de planification hiérarchiques et les réactions induites par le système.

Les causes de ces échecs peuvent être multiples, p. ex. l'échec de la réalisation d'une action locale ou l'incapacité pour un robot d'accomplir une action. Cependant, une des principales causes est celle

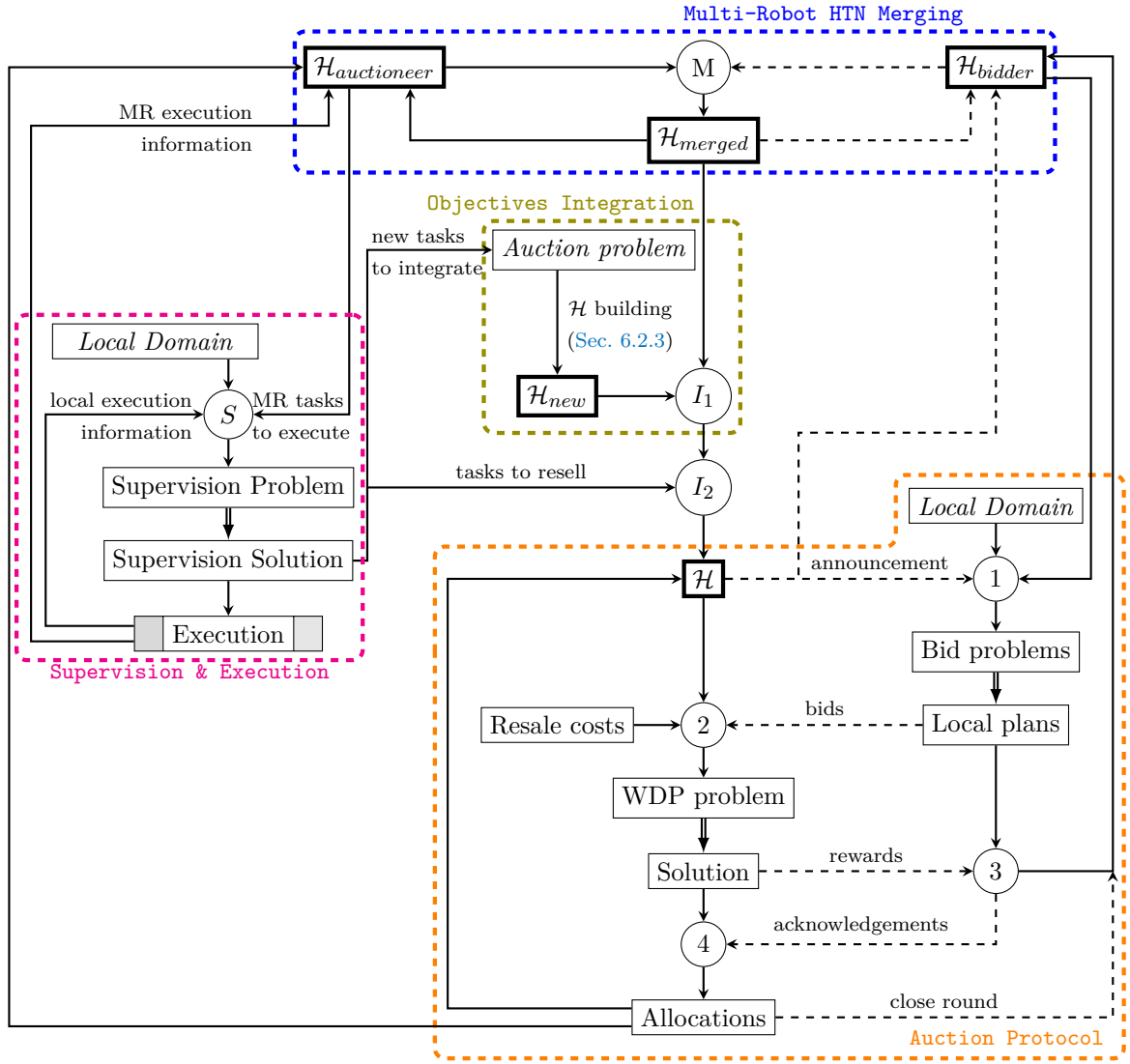


FIGURE 7.5 – Protocole complet de HaucTioN. Les rectangles représentent des structures de données gérées par les robots. Les blocs les plus à droite sont gérés par chaque enchérisseur, le reste par le commissaire-priseur, ces deux ensembles sont séparés par les flèches en pointillés qui représentent les données échangées entre le commissaire-priseur et les enchérisseurs. Les cercles indiquent les processus qui agrègent les informations pour construire de nouvelles structures. Les doubles flèches indiquent les appels à un planificateur HTN. Les flèches continues indiquent les interactions entre les structures de données et les processus. Le rectangle "Execution" correspond à la couche fonctionnelle du robot, en charge d'exécuter les actions locales et de faire remonter les informations perçues et reçues.

des pertes de communication. Par exemple, tous ces processus doivent tenir compte de l'état actuel du plan, qui n'est connu que par la génération ou la réception d'information d'exécution. Sans réception de ces informations, contenues dans des *achievements*, il peut ne pas être possible de résoudre le problème de planification de la mission, à cause des dépendances complexes qui peuvent lier les tâches. Dans le Tab. 7.3, nous résumons l'impact de la perte de ces messages de supervision de l'état du plan.

Ainsi, le bon déroulement de la mission ne dépend pas uniquement de l'allocation d'une action et de sa réussite, mais également de la connaissance qu'ont les robots de la réussite de l'action. Sans cette connaissance, la mission peut être compromise.

Processus	Échec	Causes	Actions / Méthodes de réparations
Estimation	Pas de solution	Les contraintes multirobots ne peuvent être respectées et/ou E ne peut contribuer à la tâche	E signale à CP qu'il ne peut pas formuler de mise
WDP	Pas d'allocations	Absence de mises acceptables	CP reporte l'enchère
WDP	Pas de solution	Les contraintes multirobots ne peuvent être respectées, y compris en revendant des tâches	CP abandonne des tâches
Supervision	Pas de solution	Les contraintes multirobots ne peuvent être respectées	L'agent ouvre une enchère pour revendre des tâches et/ou l'agent abandonne des tâches

TABLE 7.2 – Récapitulatif des échecs pouvant être rencontrés par les processus utilisant la planification hiérarchique et des réactions induites par le système. Le commissaire-priseur et l'enchérisseur sont respectivement notés"CP" et "E".

Type de message	Destinataire(s)	Impératif de réception	Conséquences de la perte	Actions de réparation
Achievements	Tous	Aucun	Manque d'information dans le Multi-Robot HTN de l'agent Peut mener à des blocages si des dépendances existent	Envoi systématique de l'historique des *achievements*

TABLE 7.3 – Récaptitulatif du message de supervision du plan et des conséquences de sa perte.

Enfin, par la nature non-déterministe des actions des robots, un problème peut tout simplement ne pas avoir de solution. Par exemple, un retard trop important sur une tâche peut être impossible à réparer, qu'importe les coéquipiers à portée.

Pour ces raisons, il est difficile d'apporter des garanties quant au fonctionnement en ligne de notre système. Cette question des garanties est un des écueils principaux des systèmes de systèmes, qui sont complexes à étudier en raison des nombreux processus qui les composent. Aussi, l'utilisabilité des systèmes de systèmes reposent usuellement sur des méthodes apportant de la tolérance aux fautes et des résultats empiriques plutôt que sur des preuves formelles. Dans nos travaux, nous avons priorisé cette approche en intégrant à notre système la capacité d'identifier la cause des erreurs et y réagir lorsque cela est possible.

8 ÉVALUATION DE L'APPROCHE

Aperçu

Dans ce chapitre, nous dressons une première évalution en simulation des capacités de HaucTioN à répondre du déploiement d'un système multirobot pour accomplir une mission de chasse aux mines. Nous introduisons le contexte de cette évaluation, puis, nous établissons une preuve de concept de notre approche. Enfin, nous présentons nos résultats.

Dans les chapitres précédents, nous avons introduit notre approche, HaucTioN, qui a pour objectif l'accomplissement d'une mission de chasse aux mines avec un système multirobot hétérogène. Nous avons notamment détaillé le fonctionnement de l'architecture et des processus de décision, supervision et réparation qui la composent. Dans ce chapitre, nous évaluons en simulation cette approche.

Le système que nous considérons doit notamment être apte à :

— Résoudre **en ligne** des problèmes de MRTA comportant des **dépendances complexes** ;

— Superviser l'exécution du plan, **détecter les erreurs**, et les **réparer**, notamment en incluant de nouveaux objectifs de mission ou en modifiant de précédentes décisions ;

— Faire face aux **contraintes de communication**, notamment par des débits faibles et des pertes de message.

— Le tout en maximisant les **performances** de la mission, p. ex. nombre de tâches réalisées et temps d'exécution ;

Ainsi, le système doit résoudre et optimiser des problèmes complexes tout en faisant preuve d'une certaine réactivité et tolérance aux fautes.

Pour évaluer l'aptitude du système à surmonter ces défis, nous avons mis en place une implémentation de notre architecture permettant de simuler des pertes de communication, de relever les métriques nécessaires, et de comparer nos résultats à une solution centralisée.

En premier lieu, nous introduisons une vue d'ensemble de notre implémentation d'évaluation et des métriques auxquelles nous nous intéressons. Nous illustrons ensuite les capacités du système en détaillant le déroulement d'une mission type d'évaluation. Enfin, nous présentons nos résultats, d'abord sous l'angle de notre contribution principale, la résolution en ligne de problème de MRTA grâce au protocole de décision, puis dans le contexte global de la décision, de la supervision, et de la réparation.

8.1 IMPLÉMENTATION DE L'APPROCHE

Le contexte de notre évaluation est défini par l'implémentation que nous utilisons. Dans cette section, nous dressons une vue d'ensemble de cette implémentation en introduisant les solutions techniques, l'architecture d'évaluation, et les métriques relevées.

8.1.1 *Solutions techniques*

L'implémentation de l'architecture repose sur le middleware ROS2, qui permet de mettre en place des processus informatiques parallèles pouvant s'échanger des données. Ainsi dans notre implémentation, à chaque agent correspond un ensemble de processus indépendants.

Dans notre approche, trois problèmes de planification hiérarchique sont au coeur des décisions et réactions du système. Ces problèmes de planification hiérarchique sont résolus en faisant appel au planificateur LCP [1]. LCP est un solveur compilant le problème à résoudre comme un problème de satisfaction de contrainte ("Constraint Satisfaction Problem"). Ce planificateur supporte notamment l'utilisation de contraintes temporelles et permet d'optimiser le plan suivant un coût des actions.

Les interactions des agents avec l'environnement sont simulées au sein d'un processus dédié. Ce processus permet notamment de simuler la détection de MILCO en notifiant l'agent concerné.

Nous avons évalué l'approche sur une machine virtuelle disposant de 6 coeurs, d'un processeur intel® core™ i7-9750h CPU @ 2.60GHz et de 8 GB de RAM.

8.1.2 *Architecture de simulation*

L'architecture résultante de notre implémentation est représentée dans la Fig. 8.1. Il est important de noter que toutes les communications des agents transitent par un processus dédié permettant de simuler les contraintes de communication.

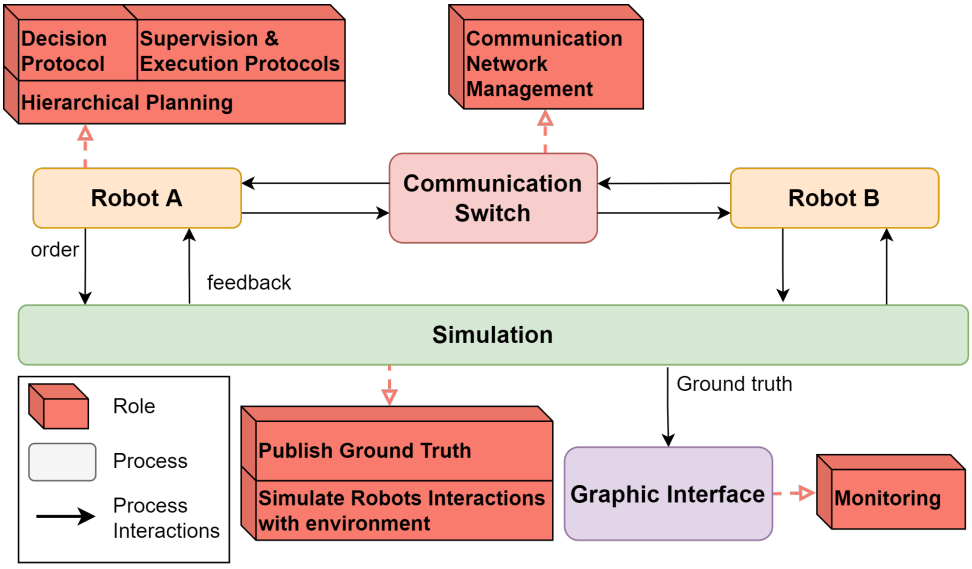


FIGURE 8.1 – Schéma de l'architecture de simulation.

Simulation des agents

Les robots sont simulés par des processus indépendants. Ce sont ces processus qui sont en charge de :

— Suivre les protocoles de décision, supervision et exécution ;

— Faire appel au solveur de problème hiérarchique ;

1. https ://github.com/plaans/aries

— Envoyer des ordres à la simulation de la couche fonctionnelle, p. ex. des actions à réaliser comme des déplacements ;

— Envoyer des communications aux autres agents ;

— Traiter les communications reçues ;

Simulation des pertes de communication

Au sein du processus *Communication Switch* de la Fig. 8.1, les contraintes de communication sont simulées grâce à un modèle de Bernoulli. Ce modèle prend en entrée la distance entre les agents et une probabilité de rupture de communication.

Lorsqu'une communication est envoyée par un agent à un autre agent, ce processus est appelé. Si le processus valide la communication, la totalité du message est reçue par le destinataire, sinon le message est perdu. Sauf utilisation d'un protocole de réception spécifique, l'émetteur n'a pas connaissance de la bonne réception ou non de son message.

Simulation des interactions des agents avec l'environnement

La couche *Simulation* représentée sur la Fig. 8.1 est en charge de simuler les couches fonctionnelles des robots et leurs interactions avec l'environnement. Par exemple, elle simule la possible détection de MILCO lorsqu'un robot est en train de couvrir une zone. Enfin, elle transmet l'état des robots à une interface graphique permettant de superviser le déroulement de la simulation.

Paramètres de l'architecture

Notre architecture permet de spécifier certains paramètres. Dans notre évaluation, nous considérons les valeurs ci-après.

Stratégie de revente : Nous utilisons les stratégies *maxB* et *minB* pour fixer le coût de revente d'une tâche (Sec. 6.6.1). Nous utilisons par défaut la stratégie *minB*.

Critères d'arrêts de l'estimation : Pour nos simulations, nous considérons les deux critères suivants pour arrêter la phase d'estimation des mises :

— Réception de toutes les mises attendues, c.-à-d. le nombre de participants à l'enchère fois le nombre de labels en vente ;

— Atteinte d'une échéance, en l'occurrence 60 secondes.

Optimisation du plan : Le coût des actions est minimisé par le solveur. Ce coût est par exemple incrémenté par les actions locales lors de l'estimation d'une mise ou par une action d'allocation. Le coût des actions locales est fixé en amont de la mission pour chaque couple $(agent, action)$.

8.1.3 *Missions d'évaluation*

Notre approche a pour but la résolution en ligne de problème de MRTA sous contraintes de communication. Le cas d'usage considéré est une mission de chasse aux mines de type *Port allié* (Sec. 2.2.1).

Les missions utilisées pour tester l'approche sont décomposées en deux phases :

1. La planification initiale, où seul le protocole de décision est exploité ;
2. L'exécution, où le plan est supervisé, les tâches exécutées, et le protocole de décision est appelé pour réparer en intégrant de nouveaux objectifs ou corriger des décisions antérieures.

Pour accomplir cette mission, les robots peuvent se voir allouer ces différents types de tâches opérationnelles :

— *cover* : Tâche de couverture d'une zone. Accomplissable uniquement par un robot de type *Explorer* ;

— *identify* : Tâche d'identification d'un objet suspect, c.-à-d. un MILCO. Accomplissable uniquement par un robot de type *Identifier* ;

— *report* : Tâche de rapport à l'opérateur. Accomplissable uniquement par un robot de type *Identifier*.

— *communication–relay* : Tâche de relai de communication, une tâche de ce type est toujours exécutée en parallèle d'une tâche *report*. Accomplissable uniquement par des robots de type *Explorer* ou *Identifier* ;

Il n'est pas possible de prédire des MILCO avant le début de la mission, aussi, le problème de planification initiale n'est composé que de tâches de couvertures.

Lors de la spécification de la mission, ces tâches sont modélisées en un HTN intégrant des contraintes complexes, telles des contraintes de précédence, des contraintes causales, et des contraintes de synchronicité.

Les différents paramètres de nos missions portent sur :

— Le nombre de robots *Explorer* ;

— Le nombre de robots *Identifier* ;

— Le nombre de tâches dans le problème initial ainsi que le nombre de contraintes spécifiées dans ce problème ;

— Le nombre d'objets suspects susceptibles d'être trouvés par les robots.

8.1.4 *Critères d'évaluation*

Métriques relevées

Chacun des processus de l'architecture de simulation génère des métriques permettant d'analyser le comportement et les performances du système. Dans notre évaluation, nous relevons les métriques suivantes :

— Temps de résolution d'un problème de planification hiérarchique, c.-à-d. pour l'estimation d'une mise, la résolution du WDP, et la supervision du plan ;

— Coût total du plan par solution à un problème de planification hiérarchique, p. ex. plan solution du WDP, plan solution du problème de supervision, … ;

— Nombre et volume des communications envoyées par type de communications, p. ex. *announcement*, *achievement*, … ;

— Nombre de communications perdues par type de communications.

Avec ces métriques, il est par exemple possible de comparer la résolution d'une enchère à l'approche centralisée en fonction du coût total des allocations, ou d'étudier l'impact du nombre de mises perdues sur un tour d'enchère.

Comparaison avec une solution centralisée

Les performances du protocole de décision de notre approche sont comparées avec une solution centralisée. Cette solution centralisée permet d'obtenir une référence optimale pour chaque problème de MRTA à résoudre, c.-à-d. chaque enchère.

Cette solution est basée sur l'utilisation du Multi-Robot HTN du premier tour d'enchère et les décompositions locales des participants. En étendant ce Multi-Robot HTN avec les décompositions de tous les agents, le problème de MRTA est modélisé sous la forme d'un problème de planification hiérarchique. Le processus permettant de combiner le Multi-Robot HTN aux décompositions locales est similaire à celui décrit pour l'estimation d'une mise.

Le problème ainsi formé est ensuite résolu par le planificateur LCP. Le coût de ce problème établit une référence avec la solution obtenue à la clôture du dernier tour d'enchère.

8.1.5 *Complexité de l'évaluation des systèmes de systèmes*

Dans de précédents travaux, nous avions présenté des résultats portant sur une première version de notre protocole de décision [Mil+21a]. Les résultats de ces travaux nous ont encouragés à persévérer dans la mise en place de notre approche jusqu'à présenter l'architecture décrite dans ce manuscrit. Cependant, cette précédente expérience a été pour nous édifiante sur **la grande difficulté entourant l'évaluation de systèmes de systèmes**.

Dans ces travaux, nous avions mis en place une évaluation statistique mettant en jeu des problèmes de MRTA générés aléatoirement. Mais les performances du protocole de décision peuvent grandement varier suivant les paramètres des problèmes à résoudre. Par exemple, un problème de planification hiérarchique peut être défini par le nombre de tâches primitives, de tâches abstraites, et de méthodes, et à ces éléments viennent s'ajouter les contraintes pouvant porter sur les tâches. Chaque variation de l'un de ces paramètres établit un nouveau problème de planification hiérarchique dont la complexité de résolution peut être très différente d'un autre problème "proche" en termes de paramètres. Bien qu'une évaluation statisque permette d'obtenir des résultats complets, pour qu'une telle évaluation soit viable il est nécessaire d'être exhaustif en considérant une très grande diversité de problèmes. Cette diversité nécessite de simuler le système à grande échelle en investissant beaucoup de temps et de ressources matérielles.

Par ailleurs, les problèmes utilisés dans [Mil+21a] ne comportaient que des contraintes de précédence et non pas des contraintes de synchronicité. Enfin, la résolution de ces problèmes se faisait sous hypothèse de communications parfaites et en absence de plan déjà établi. Les éléments dont nous devons tenir compte dans la version actuelle de notre approche complexifient d'autant plus l'évaluation du système. S'il est possible d'explorer les dimensions composant nos problèmes une à une, faire des tests exhaustifs représente un coût important. Par exemple, dans notre approche un problème de MRTA correspond à une enchère et peut être défini par :

- le nombre de robots dans le système, p. ex. de 1 à 10 robots ;
- la qualité des communications, p. ex. de 10% (1 chance sur 10 de réceptionner un message) à 100% ;
- l'article en vente :
  - Nombre total de tâches primitives, p. ex. de 2 à 20 ;
  - Nombre total de tâches abstraites, p. ex. de 1 à 10 ;
  - Nombre de méthodes, p. ex. de 1 à 20 ;
  - Contraintes intrinsèques à chaque tâches, p. ex. pas de contrainte, contrainte sur `start`, contrainte sur `end`, contrainte sur `start` et `end`, … ;
  - Contraintes entre tâches, p. ex. entre le `start` de deux tâches, ou par des contraintes causales ;
  - Nombre de tâches en vente, peut être compris entre 1 et le nombre total de tâches ;

En ne considérant que le nombre restreint de 100 articles en vente différents (p. ex. pour uniquement 15 à 30 tâches en tout et de 1 à 15 labels en vente), un pas de 10 pour la qualité des communications, et de 1 à 10 enchérisseurs, il existe environ 450 000 variations d'un problème de MRTA définit par le tuple {*nb. enchérisseurs, article en vente, qualité de communication*}. De plus, les résultats n'étant pas déterministes, car les communications sont imparfaites, il est nécessaire de répéter un certain nombre de fois la résolution d'un problème de MRTA pour déterminer un comportement moyen. Par exemple, en considérant 10 itérations de chacun des problèmes il est nécessaire de résoudre 4 500 000 enchères. En supposant de manière très optimiste que la résolution d'une enchère ne dure qu'une seconde, il serait nécessaire de consacrer 1250 heures, soit 52 jours, à la génération de tels résultats. Enfin, rappelons que le nombre de tests nécessaires est ici déterminé sans prendre en considération les variations des contraintes portant sur les tâches.

Nos ressources étant limitées, nous avons ici fait le choix inverse d'une évaluation statistique. Nous utilisons un nombre restreint de problèmes de MRTA dont nous maitrisons les paramètres afin d'établir un premier diagnostic de notre approche en faisant varier ces paramètres. Par cette méthode, nous présentons des résultats plus qualitatifs que quantitatifs, aussi, nous attirons l'attention du lecteur sur l'importance de traiter ces résultats avec prudence.

8.2 SIMULATION D'UNE MISSION TYPE

Afin d'illustrer au mieux les éléments auxquels se rapportent les résultats des sections suivantes, nous détaillons ici le déroulement d'une des missions d'évaluation.

Le système est caractérisé par l'utilisation de 4 robots *explorers* et de 2 robots *identifiers*. Avec une qualité de communication de 70%, c.-à-d. la probabilité qu'une communication arrive à son destinataire est de $0,7$. Les fichiers HDDL de spécification de la mission sont disponibles en An. A et An. B.

Pour pouvoir illustrer le déroulement de la mission, des captures d'écran des étapes clés sont présentées dans Fig. 8.2 et Fig. 8.3. Ces captures d'écran proviennent de l'interface graphique de l'architecture de simulation (bloc *Graphic Interface* de la Fig. 8.1), cette interface a été développée au cours d'un stage par Jean-Hugues Cointot. Les captures d'écran montrent l'état du système à un instant donné, tels que la position des robots et leur statut (commissaire-priseur, enchérisseur, en exécution, en panne ou au repos), les MILCO détectés et les mines identifiées.

8.2.1 *Déroulement de la mission*

Planification initiale

Initialement, le système est à l'instant t_0 dans la configuration montrée dans la Fig. 8.2a. La première étape pour le système consiste à résoudre le problème de MRTA initial, c.-à-d. procéder à la planification initiale. Pour cette mission, c'est le robot *AUVe1* qui est en charge de cette étape en devenant le premier commissaire-priseur.

AUVe1 initialise son Multi-Robot HTN, noté $\mathcal{H}_{AUVe1}$, à partir des fichiers HDDL spécifiant les objectifs initiaux de la mission. Pour cela, il suit le protocole définit dans la Sec. 6.2.3 en construisant en premier lieu un problème instancié au formalisme HTN, puis un Multi-Robot Graph, et enfin un Multi-Robot HTN. À des fins d'illustration, le Multi-Robot Graph utilisé pour encoder les objectifs initaux est disponible en An. F sur la Fig. F.4. Une fois son Multi-Robot HTN initialisé, *AUVe1* ouvre une enchère en diffusant un message d'annonce. Le contenu de ce message est représenté dans Tab. 8.1. Les communications n'étant

pas parfaites, les robots *AUVi1* et *AUVe4* ne réceptionnent pas ce message, ils ne participent donc pas à l'enchère initiale. À la réception du message d'annonce, les robots *AUVe2*, *AUVe3*, et *AUVi1* initialisent leur Multi-Robot HTN avec celui en vente.

announcement	*auctioneer id*	*auction id*	*round id*	t_a	δ	message size (kB)
message	*AUVe1*	*auction–AUVe1–1*	*round–1*	$t = t_0 + 20$	$(\mathcal{H}_{AUVe1}, (l_1, \ldots, l_{25}))$	9.5

TABLE 8.1 – Message d'annonce de l'enchère initiale.

Les robots *AUVe1*, *AUVe2*, *AUVe3*, et *AUVi1* commencent ensuite à estimer une mise pour chaque label dans $(l_1, \ldots, l_{25})$. Comme les tâches en vente ne sont que des tâches de couverture le robot *AUVi1* ne peut rien obtenir. Les robots *AUVe1*, *AUVe2* et *AUVe3* estiment chacun 25 mises en suivant le protocole décrit à la Sec. 6.5. Tous enchérisseurs confondus, la durée moyenne d'estimation d'une mise est de 0.14 secondes et la pire durée est de 0.29 secondes.

En raison des communications imparfaites seules 38 des 50 mises de *AUVe2* et *AUVe3* sont réceptionnées par le commissaire-priseur. En comptant ses propres mises, le commissaire-priseur doit résoudre un WDP contenant en tout 63 mises. Une fois l'instant $t_0 + 20$ atteint, le problème de planification hiérarchique correspondant à ce WDP est modélisé en suivant les étapes décrites à la Sec. 6.6. Il est résolu par le commissaire-priseur en 78 secondes.

La résolution mène à l'allocation de la tâche abstraite de label l_4 à *AUVe1* et des tâches primitives l_{11} et l_{13} respectivement à *AUVe2* et *AUVe3*. *AUVe1* diffuse leurs récompenses à *AUVe2* et *AUVe3* qui en accusent bonne réception et mettent à jour $\mathcal{H}_{AUVe2}$ et $\mathcal{H}_{AUVe3}$ avec leur récompense respective. Puis, *AUVe1* met à jour $\mathcal{H}_{AUVe1}$ avec sa propre récompense et celles confirmées, c.-à-d. avec les allocations de l_4, l_{11} et l_{13}, Lors de l'estimation de l_4, *AUVe1* a déterminé le plan $l_{14} \rightarrow l_{15} \rightarrow l_{16}$, or, en raison du prédicat *free–token* (Voir An. B) une chaine de causalité existe sur les tâches primitives l_{14}, l_{15}, et l_{16}, par conséquent, les contraintes de la mise de *AUVe1* sur l_4 sont intégrées à $\mathcal{H}_{AUVe1}$. Ensuite, *AUVe1* détermine si l'enchère doit continuer, c.-à-d. si la solution du WDP comporte au moins une tâche à revendre. Avec ces informations, le commissaire-priseur diffuse le message de clôture de tour représenté dans le Tab. 8.2.

close round	*auction id*	*round id*	*close auction*	$\mathcal{H}_\Delta$	message size (kB)
message	*auction–AUVe1–1*	*round–1*	False	$\mathcal{H}_{AUVe1}$	9.5

TABLE 8.2 – Message de clôture du premier tour de l'enchère initiale.

Le commissaire-priseur détermine alors quelles tâches peuvent être revendues et débute à l'instant t_1 un nouveau tour avec le message d'annonce représenté dans le Tab. 8.3. Un nouveau tour se déroule

announcement	*auctioneer id*	*auction id*	*round id*	t_a	δ	message size (kB)
message	*AUVe1*	*auction–AUVe1–1*	*round–2*	$t = t_1 + 20$	$(\mathcal{H}_{AUVe1}, (l_{12}, l_5))$	9.5

TABLE 8.3 – Second message d'annonce de l'enchère initiale.

alors similairement au premier avec la participation de *AUVe1*, *AUVe2*, *AUVe3*, et *AUVi1*. À la fin de ce tour les deux tâches en vente l_{12} et l_5 sont respectivement allouées à *AUVe2* et *AUVe3*. Les principales métriques de cette enchère sont résumées dans le Tab. 8.4. Il est à noter que le problème comporte dès le premier tour des contraintes de précédence et de synchronicité, il s'agit d'un problème partiellement ordonné. Les deux contraintes de précédence supplémentaires dans le message d'annonce du second tour proviennent de l'allocation de l_4 à *AUVe1* durant le premier tour.

	Nombre de tâches en vente		Nombre de contraintes de		Nombre de tâches primitives allouées ou revendues				Temps d'estimation d'une mise (s)		Temps de résolution du WDP (s)
	Abstraites	Primitives	Précédence	Synchronicité	AUVe1	AUVe2	AUVe3	Revendues	Pire temps	Total moyen par robot	
Tour 1	7	18	3	22	3	1	1	2	< 0.3	3.5	78
Tour 2	0	2	5	22	0	1	1	0	< 0.05	< 0.1	< 0.1

TABLE 8.4 – Récapitulatif des paramètres et métriques de l'enchère initiale.

La durée totale de la planification initiale a été d'environ 120 secondes, cela en raison de la durée du premier WDP (78 secondes) et des échéances utilisées pour clôturer la phase d'estimation des mises (deux phases de 20 secondes). Enfin, le coût total des allocations est de 700, et celui de la référence centralisée pour le même problème est de 600. La qualité d'allocation de cette enchère est donc d'environ 83% relativement à la référence centralisée.

Exécution et premier aléa

Après la première enchère, les robots *AUVe1*, *AUVe2*, et *AUVe3* résolvent chacun un problème de supervision. Un exemple de la structure du problème de supervision formé pour *AUVe1* est donné en An. F sur la Fig. F.3. La résolution de ces problèmes permet de déterminer les prochaines actions locales que les robots peuvent déclencher pour exécuter les tâches multirobots qu'ils ont acquises. Par exemple, les tâches multirobots étant des tâches de couvertures, ils les accomplissent en exécutant des trajectoires de type boustrophédon grâce à l'action locale $fl\text{–}cover\text{–}primitive$. Ces trajectoires consistent en une séquence d'actions *move*, à chaque fois qu'une action *move* débute ou termine un problème de supervision est résolu pour déterminer la suite du plan.

En débutant chacun une tâche, les robots ont généré 3 *achievements*. Le message de *achievement* diffusé par *AUVe1* en débutant la tâche l_{14} à l'instant t_2 est donné dans le Tab. 8.5.

achievement	l_ρ	ref_ρ	t_ρ	message size (kB)
message	l_{14}	$\mathsf{start}_{l_{14}}$	t_2	0.08

TABLE 8.5 – Message de *achievement* de *AUVe1* pour la tâche de label l_{14}.

Lors de l'accomplissement d'une de ses tâches de couvertures, le robot *AUVe2* détecte un MILCO. La Fig. 8.2b montre l'état du système à cet instant. *AUVe2* ouvre alors une seconde enchère pour allouer les nouvelles tâches correspondant à l'identification de l'objet. En ouvrant cette enchère, et comme représenté dans la Fig. 8.2c, le statut des robots évolue, chacun continue d'exécuter ses tâches, mais passe également en mode commissaire-priseur ou enchérisseur.

AUVe2 utilise comme article en vente son Multi-Robot HTN, noté $\mathcal{H}_{AUVe2}$, après intégration des nouveaux objectifs d'identification en suivant le processus décrit dans la Sec. 7.3.2, le Multi-Robot HTN résultant est donné en An. F sur la Fig. F.4. L'article en vente est notamment composé de :

— 5 labels en vente, uniquement des labels relatifs à l'objectif d'identification ;

— 3 nouvelles tâches primitives, elles correspondent aux tâches opérationnelles $identify\text{–}primitive$, $report\text{–}sync\text{–}primitive$, et $relay\text{–}sync\text{–}primitive$ dans An. A ;

— 1 nouvelle contrainte causale, en raison de l'usage du prédicat $mr\text{–}sync\text{–}flag$ dans An. A. Cette contrainte permet notamment de procéder à la synchronisation des tâches $report\text{–}sync\text{–}primitive$ et $relay\text{–}sync\text{–}primitive$.

Et le message d'annonce correspondant est représenté dans le Tab. 8.6, il est envoyé à l'instant t_3.

Tous les robots réceptionnent le message d'annonce de *AUVe2* puis misent en suivant le même protocole que pour la première enchère. Cette enchère est résolue en un tour. Les robots *AUVi2* et *AUVe4*, qui

announcement	*auctioneer id*	*auction id*	*round id*	t_a	δ	message size (kB)
message	*AUVe2*	*auction–AUVe2–*1	*round–*1	$t = t_3 + 20$	$(\mathcal{H}_{AUVe2}, (l_{1'}, l_{2'}, l_{3'}, l_{4'}, l_{5'}))$	10.7

TABLE 8.6 – Message d'annonce de la seconde enchère.

étaient jusque là au repos, obtiennent respectivement les tâches opérationnelles d'identification et de relais de communication. La durée totale de cette seconde enchère a été d'environ 21 secondes, avec un WDP résolu en < 0.1 secondes, car le nombre de mises à considérer est très inférieur à celui de la première enchère, et une phase d'estimation de 20 secondes.

Le robot *AUVi2* débute alors son identification du MILCO tandis que le robot *AUVe4* commence à relayer les données, Fig. 8.2d. La synchronisation entre *AUVe4* et *AUVi2* pour accomplir respectivement les tâches de *relay–sync–primitive* et *report–sync–primitive* est ici réalisée en diffusant successivement les *achievements* présentés dans le Tab. 8.7.

	l_ρ	ref_ρ	t_ρ	message size (kB)
achievement message from *AUVe4*	$l_{4'}$	$\mathsf{start}_{l_{4'}}$	t_3	0.08
achievement message from *AUVi2*	$l_{5'}$	$\mathsf{start}_{l_{5'}}$	t_4	0.08
achievement message from *AUVi2*	$l_{5'}$	$\mathsf{end}_{l_{5'}}$	t_5	0.08
achievement message from *AUVe4*	$l_{4'}$	$\mathsf{end}_{l_{4'}}$	t_6	0.08

TABLE 8.7 – Messages de *achievements* de *AUVe4* et *AUVi2* pour les tâches *relay–sync–primitive* ($l_{4'}$) et *report–sync–primitive* ($l_{5'}$), avec $t_4 > t_5 > t_6 > t_7$.

Détection d'un second MILCO et identification du premier

Sur la Fig. 8.2e, *AUVi2* et *AUVe4* ont fini l'identification du MILCO et sont au repos. Le robot *AUVe3* vient de détecter un second MILCO.

AUVe3 ouvre une troisième enchère de manière similaire à ce qu'à fait *AUVe2*. La résolution de cette enchère mène à l'allocation de l'identification à *AUVi1* et celle du relais à *AUVe3*, ces robots résolvent alors un problème de supervision pour déterminer les actions locales à effectuer. *AUVi1* et *AUVe3* débutent alors l'identification du MILCO (Fig. 8.2f). Comme le relais de communication peut être fait en parallèle d'un déplacement, *AUVe3* n'a pas eu à interrompre sa couverture.

Panne de AUVe1

Sur la Fig. 8.3a, les robots accomplissent leurs tâches respectives, mais *AUVe1* a rencontré une panne. Cette panne l'empêche de trouver une solution à son problème de supervision, il ouvre alors une nouvelle enchère pour mettre en vente les tâches qu'il n'a pas encore terminées. Le message d'annonce de cette enchère est représenté dans le Tab. 8.8, il est envoyé à l'instant t_8.

announcement	*auctioneer id*	*auction id*	*round id*	t_a	δ	message size (kB)
message	*AUVe1*	*auction–AUVe1–*2	*round–*1	$t = t_8 + 20$	$(\mathcal{H}_{AUVe1}, (l_{15}, l_{16}))$	10.7

TABLE 8.8 – Message d'annonce de la quatrième enchère.

Cette quatrième enchère est résolue en deux tours. La durée totale de cette enchère a été d'environ 41 secondes, avec deux WDP résolus en < 0.1 secondes et deux phases d'estimation de 20 secondes. La résolution de cette enchère mène à la réallocation des tâches l_{15} et l_{16} à *AUVe4* (Fig. 8.3b).

Identification du second MILCO et détection d'un troisième

Sur la Fig. 8.3c, *AUVi1* a terminé son objectif d'identification du second MILCO et *AUVe2* et *AUVe3* sont proches de la fin de leurs couvertures.

Puis, sur la Fig. 8.3d, *AUVe4* a détecté un troisième MILCO. Toujours en suivant le même processus que précédemment, *AUVe4* ouvre une cinquième enchère pour allouer les nouveaux objectifs d'identification. Cette enchère est résolue en un tour pour une durée totale d'environ 21 secondes. De même que pour le premier MILCO, ces objectifs sont acquis par *AUVi2* et *AUVe4*. Après la résolution d'un problème de supervision, les robots commencent à accomplir ces nouvelles tâches (Fig. 8.3e).

Fin de la mission

Enfin, le système est montré dans son état final sur la Fig. 8.3f. Tous les MILCO ont été identifiés comme des mines et toute la zone de mission a été couverte. Chacun des robots est au repos.

8.2.2 *Discussion*

Avec cette mission d'illustration nous établissons une preuve de concept des algorithmes présentés dans ce manuscrit. En premier lieu, le protocole de décision est exploité afin de résoudre le problème de planification initiale. Puis les tâches sont exécutées en suivant le protocole de supervision et en générant des *achievements*. Enfin, quatre aléas rencontrés obligent les robots à réparer globalement le plan grâce au protocole de décision en allouant de nouveaux objectifs et en modifiant de précédentes décisions.

Ce sont en tout 5 problèmes de MRTA comportant des dépendances complexes qui ont été résolus avec notre schéma de décision. Puis, les plans issus de la résolution de ces problèmes ont été exécutés et supervisés en respectant les contraintes des problèmes et les contraintes supplémentaires établies par les robots lors de la phase de décision.

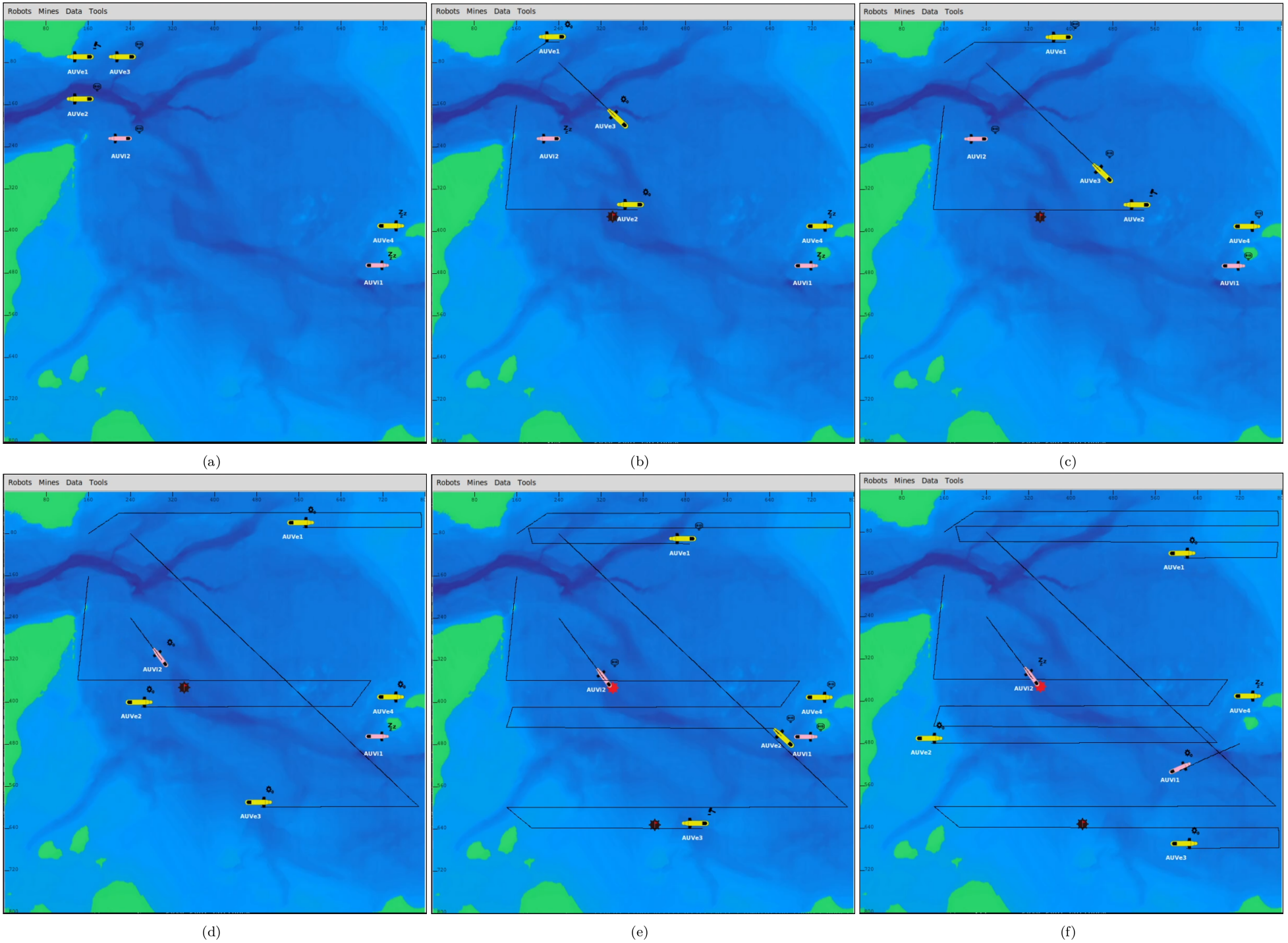


FIGURE 8.2 – Captures d'écran de la mission - (captures 1 à 6).

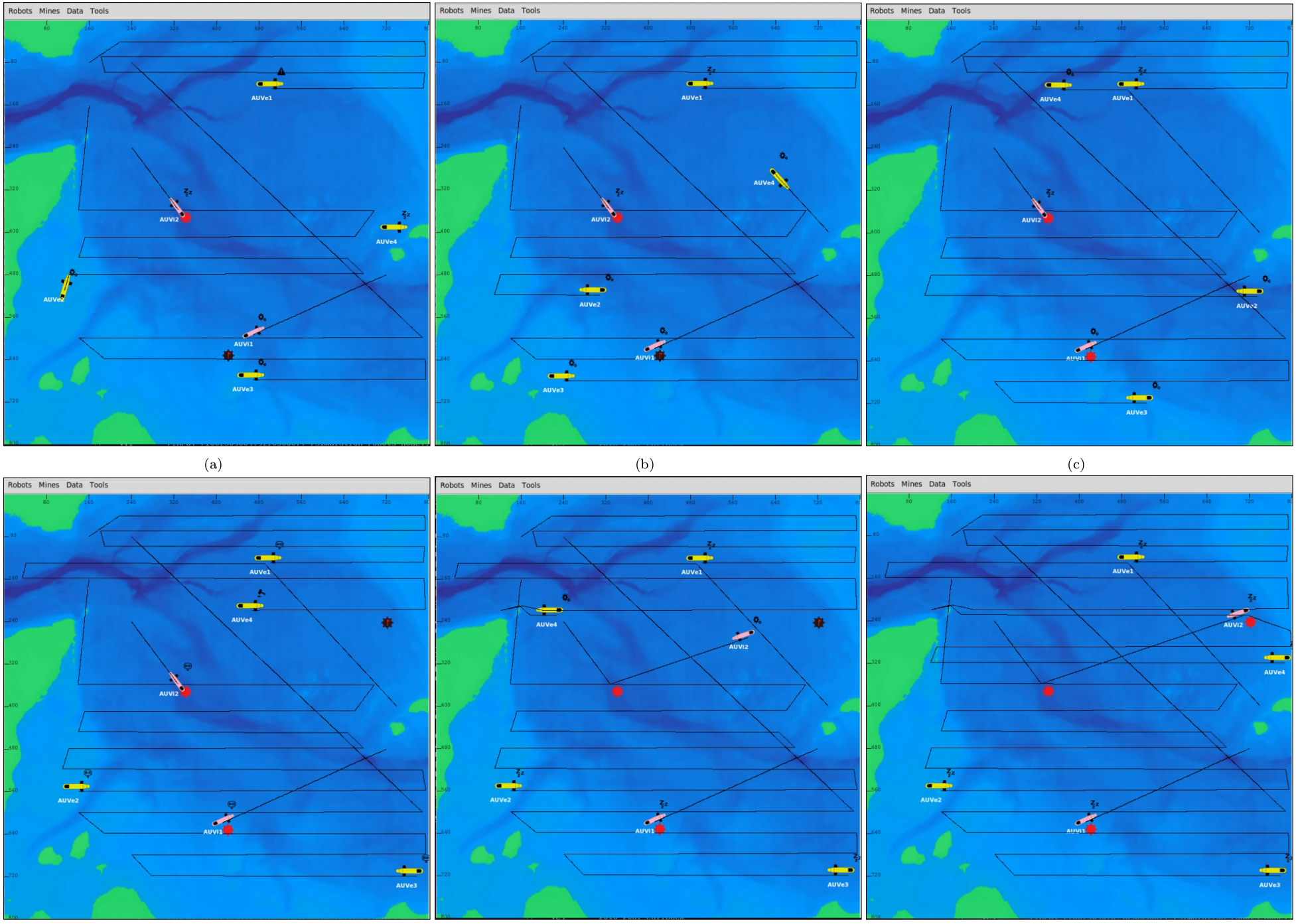


FIGURE 8.3 – Captures d'écran de la mission - (captures 7 à 12).

8.3 ÉVALUATION DU PROTOCOLE DE DÉCISION

Dans cette section, nous évaluons l'approche sous l'angle de notre principale contribution, la résolution en ligne de problèmes de MRTA grâce à des enchères et la planification hiérarchique.

Dans HaucTioN, c'est le protocole de décision qui est en charge de la résolution de ces problèmes. Outre la résolution d'un problème de MRTA comportant des dépendances complexes, nos principaux critères d'intérêts portent sur :

— **La qualité de l'allocation**, notée $\mathbb{Q}$: Les allocations obtenues à l'issue du protocole de décision sont-elles d'une qualité satisfaisante ? La solution d'un problème de MRTA, si elle existe, se traduit par un ensemble d'allocations incrémentant un coût total, noté cost. Nous comparons ce coût à celui de la référence centralisée, noté cost_{ref}, afin d'établir la qualité de l'allocation comme une différence relative à la valeur de référence, c.-à-d. $\mathbb{Q} = (1 - (\mathsf{cost} - \mathsf{cost}_{ref})/\mathsf{cost}_{ref}) \times 100$. Cette différence est bornée entre 0 (qualité minimale) et 100% (qualité maximale). Il est à noter que cost est toujours supérieur ou égal à cost_{ref} ;

— **Le coût calculatoire** : La complexité des calculs menés est-elle en adéquation avec l'utilisation en ligne du protocole de décision ? Comment évolue-t-elle en fonction du contexte de la mission ? Pour répondre à ces questions, nous étudions avec un intérêt particulier le problème de détermination des gagnants ;

— **La robustesse aux pertes de communication** : Quel est l'impact des communications sur le protocole de décision ? Pour étudier ce critère, nous faisons varier la qualité des communications ;

— **La bande passante** : Quel est le coût du protocole de décision en termes de bande passante ? Ce coût est-il en adéquation avec les contraintes du monde sous-marin ? Pour cela nous relevons le nombre de messages échangés, indépendamment de leur bonne réception ou non.

Pour répondre à ces questions, nous nous étudions la résolution d'une enchère où :

— L'article en vente est initialement composé de 5 à 30 tâches multirobots, en pas de 5. Ces tâches sont uniquement des tâches de couvertures partiellement ordonnées. Initialement tous les labels sont en vente ;

— Les enchérisseurs sont uniquement des robots *Explorers*. De 1 à 6 enchérisseurs participent à l'enchère, en pas de 1 ;

— La qualité de communication varie de 10 à 100% en pas de 10, p. ex. une qualité de communication de 10% signifie qu'un message à 1 chance sur 10 d'être reçu.

Ainsi, nous désignons les paramètres d'un problème de MRTA par un tuple {*nb. enchérisseurs, nb. tâches, qualité de communication*}.

De plus, un seul commissaire-priseur est utilisé et nous considérons qu'il ne peut pas miser, c.-à-d. participer également en tant qu'enchérisseur. Cela est nécessaire afin que toutes les mises soient dépendantes de la qualité des communications. En effet, dans notre approche n'importe quel robot peut devenir commissaire-priseur et miser sur les propres tâches qu'il met en vente. Si le commissaire-priseur peut miser, l'intégralité de ses mises est considérée, qu'importe la qualité des communications. Dans cette évaluation cependant nous avons souhaité séparer les rôles de commissaire-priseur et d'enchérisseur afin de ne pas biaiser les résultats avec les mises du commissaire-priseur.

Enfin, chaque mission d'évaluation est réitérée 10 fois afin d'en lisser les performances, car la résolution de ces problèmes n'est pas déterministe, p. ex. en raison des pertes de communication.

8.3.1 *Qualité d'allocation en fonction du nombre de tâches en vente et de la qualité des communications*

La Fig. 8.4 montre la qualité finale d'allocation, $\mathbb{Q}$, obtenue avec notre approche pour différents problèmes de MRTA. Chaque figure correspond à un nombre de tâches dans le problème.

L'axe des abscisses représente le nombre de robots dans le système, et donc un nombre d'enchérisseurs potentiels. L'axe des ordonnées indique la qualité de communication considérée pour le système. Chaque case représente donc la qualité d'allocation finale pour problème de MRTA définit par {*nb. enchérisseurs, nb. tâches, qualité de communication*} où le nombre de tâches est fixé par figure.

Résultats

Ces données montrent en premier lieu que les performances du protocole de décision sur ces problèmes sont globalement bonnes pour des qualités de communication et un nombre d'enchérisseurs élevés.

Le deuxième élément ressortant de ces résultats concerne les faibles performances pour les problèmes où la qualité de communication est de 10 ou 20%, c.-à-d. les deux lignes du bas.

Les cases où la qualité finale d'allocation est très proche de 0% (en violet foncé) signifient qu'un tour d'enchère s'est terminé sans mises gagnantes. Dans le fonctionnement nominal de l'architecture, les tâches sont remises en vente dans une autre enchère, mais dans l'évaluation de cette enchère cela se traduit par un score de 0%. C'est notamment le cas pour les problèmes avec nombre important de tâches, plus de tours d'enchères sont nécessaires, augmentant ainsi les chances de faire un tour sans réceptionner de mise. C'est ce phénomène que l'on peut notamment observer pour les systèmes de 1 à 6 enchérisseurs pour le problème à 30 tâches où la qualité de communication est de 10%.

Ces données semblent également confirmer que l'impact de la qualité des communications est graduel. Par exemple pour les problèmes à 5, 10 et 30 tâches, la qualité d'allocation semble augmenter avec la qualité de communication. Cependant, des tests à plus grande échelle sont nécessaires pour confirmer cette tendance, car les problèmes de communications impliquent des résultats variables. En effet, quelques résultats semblent plus aléatoires, par exemple pour le problème à 25 tâches.

Dans la grande majorité des cas, les performances sont bonnes avec des qualités d'allocation variant de 80 à 100%.

Enfin, la qualité des solutions pour le problème de MRTA avec 30 tâches est satisfaisante pour les systèmes de 2 à 6 robots, mais semble décroitre avec le nombre d'enchérisseurs.

Discussion

Certains de nos résultats peuvent s'expliquer par la nature combinatoire du problème de détermination des gagnants.

Dans les approches par enchères, la solution du WDP est dépendante des mises reçues. D'autant plus lorsqu'il est possible d'allouer plusieurs mises en une seule résolution du WDP. Dans notre approche, la résolution du WDP ne se fait pas en allouant la meilleure mise, mais en cherchant la meilleure combinaison de mises. Par conséquent, il est attendu que la qualité d'allocation soit intrinsèquement liée à la diversité des mises reçues. Moins de tâches, moins de communications passantes, et moins d'enchérisseurs impliquent moins de mises, et donc un espace de solution moins grand. Il est également possible que ce phénomène soit exacerbé par la structure hiérarchique, qui tend à réduire le nombre de tours d'enchères en permettant d'allouer un ensemble de tâches, faisant ainsi rapidement diverger la solution de l'optimale.

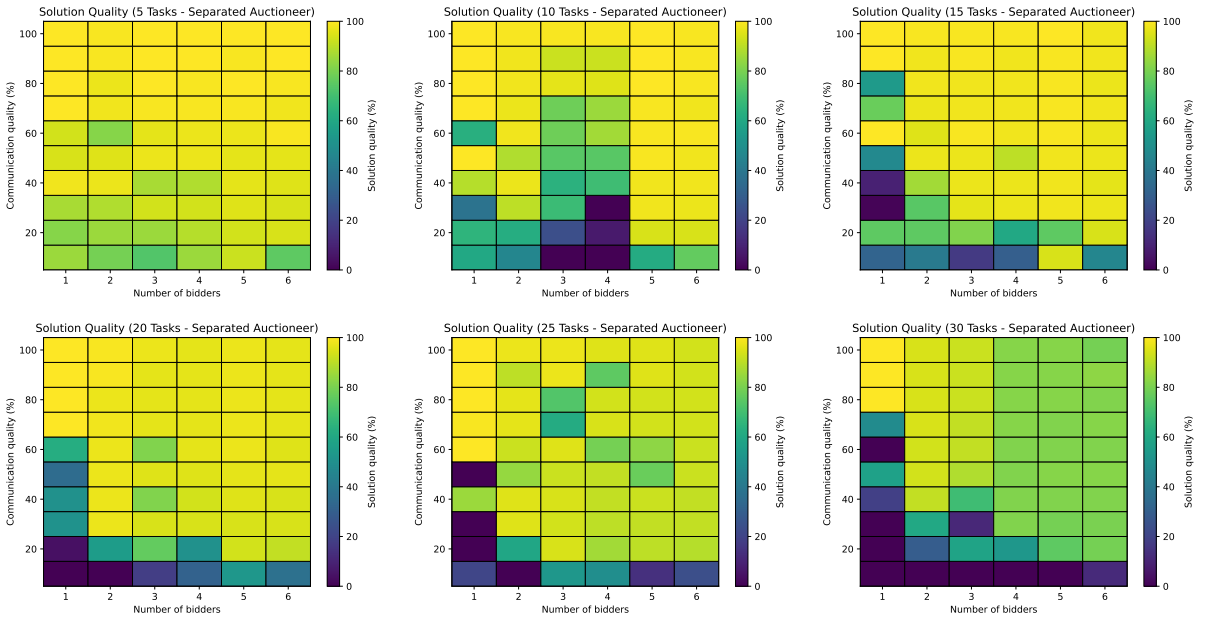


FIGURE 8.4 – Qualité d'allocation d'une enchère en fonction du nombre de robots dans le système, du nombre de tâches en vente, et de la qualité des communications.

De plus, la qualité d'allocation plus faible pour le problème de 30 tâches pour un nombre d'enchérisseurs élevé pourrait s'expliquer par un nombre plus important de récompenses par tour et ainsi un nombre faible de tours.

Enfin, il est important de rappeler que les tendances montrées ici ne sont que des résultats préliminaires sur un nombre réduit de problèmes. Une évaluation à plus grande échelle est nécessaire pour les confirmer, nous discutons notamment ce point dans le chapitre suivant.

8.3.2 *Impact du nombre de mises sur le temps de résolution du WDP*

Dans cette section, nous présentons nos résultats au regard du coût calculatoire du WDP.

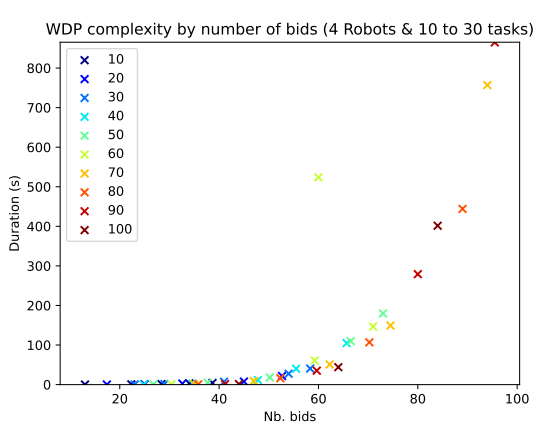
Résultats

La Fig. 8.5b montre l'évolution du temps de résolution d'un problème de détermination des gagnants en fonction du nombre de mises reçues. Pour ce faire, la moyenne courante et l'écart-type sont affichés avec une fenêtre de taille 10. Cette moyenne courante a été calculée en prenant les données obtenues sur l'enchère initiale de chaque mission d'évaluation, c.-à-d. chaque tuple {*nb. enchérisseurs, nb. tâches, qualité de communication*} possible pour 1 à 6 enchérisseurs, 5 à 30 tâches en vente, et 10 à 100% en qualité de communication. À des fins d'illustration, la Fig. 8.5a montre une fraction des données utilisées dans ce processus.

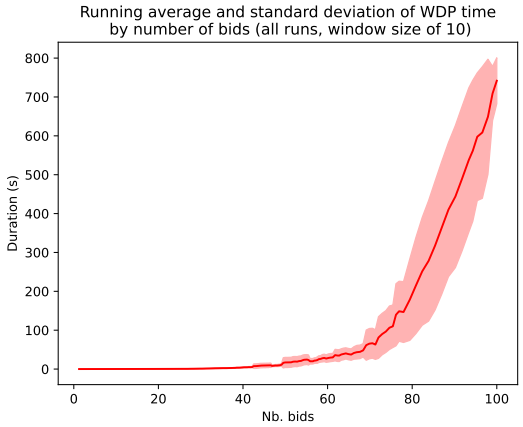
Dans cette figure, nous pouvons observer une augmentation exponentielle du temps de résolution du WDP en fonction du nombre de mises.

Discussion

Comme évoqué dans le Chap. 4, notre schéma d'allocation par enchères sur des réseaux de tâches hiérarchiques s'apparente à une forme d'allocation par enchères combinatoires où les combinaisons sur



(a) Exemple de données brutes du temps de résolution du WDP en fonction du nombre de mises. Ici sont représentées les données pour un système composé de 4 enchérisseurs avec chaque problème allant de 5 à 30 tâches (6 problèmes) pour des qualités de communication de 10 à 100%. Une seule itération de chacune de ces missions est représentée, pour un total de 60 missions (6×10).



(b) Évolution du temps du WDP en fonction du nombre de mises. La moyenne courante est représentée avec fenêtre glissante de taille 10.

FIGURE 8.5 – Évolution du temps du WDP en fonction du nombre de mises.

lesquelles il est possible de miser sont restreintes. Cette restriction des combinaisons permet de limiter le nombre maximum de mises possibles. Cependant, dans notre approche il est possible d'allouer plusieurs mises en un seul tour (au plus une par enchérisseur). Par conséquent, la recherche d'une solution au WDP passe par la considération des combinaisons de mises possibles.

L'ensemble des combinaisons de mises grandit de manière exponentielle avec le nombre de mises reçues. La recherche et l'optimisation d'une solution étant intrinsèquement liée aux combinaisons de mises possibles, cette recherche suit elle aussi une tendance exponentielle.

Cette dépendance au nombre de mises doit être prise en considération afin de pouvoir utiliser l'approche en ligne, où les ressources en temps et puissance de calculs sont généralement limitées.

8.3.3 *Évolution du pire temps d'estimation*

Dans cette section nous présentons l'évolution du pire temps d'estimation en fonction du nombre de tâches dans le problème de MRTA. Le pire temps d'estimation d'une mise correspond au temps le plus important pour une estimation sur un tour d'enchère, tous enchérisseurs confondus. La Fig. 8.6 montre la moyenne et l'écart-type du pire temps d'estimation sur l'ensemble des missions réalisées.

Le pire temps d'estimation semble évoluer linéairement avec le nombre de tâches dans le problème de MRTA. De plus, le temps de résolution est très rapide, y compris pour un nombre élevé de tâches. Enfin, le temps d'estimation d'une mise est très inférieur à celui du WDP. Même en considérant le temps total d'estimation, c.-à-d. dans le pire cas $30 \times 0.2 = 6$ secondes, ce temps reste généralement négligeable devant celui de résolution du WDP.

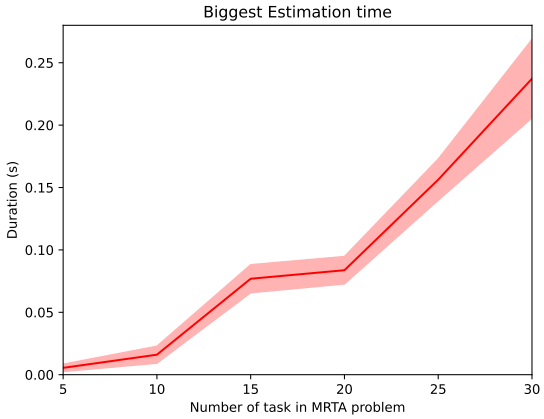


FIGURE 8.6 – Évolution du pire temps d'estimation en fonction du nombre de tâches.

8.3.4 *Coûts en communications du protocole de décision*

Dans le but d'évaluer les coûts en communications de notre protocole de décision nous avons relevé le nombre et la taille des messages échangés au cours d'une enchère, indépendamment de la réception ou non de ces messages.

Résultats

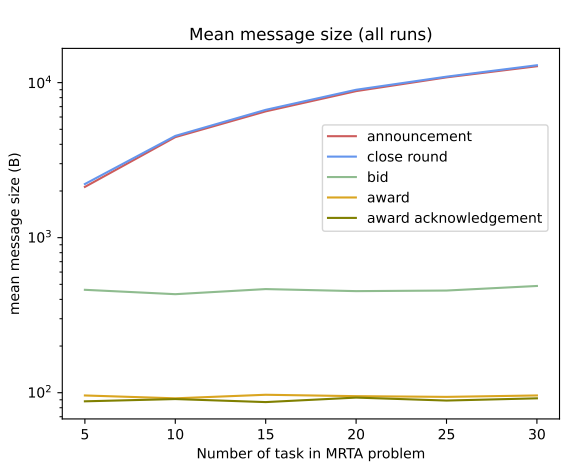
Afin de déterminer l'impact de la taille du problème de MRTA sur le poids d'une communication, nous nous intéressons en premier lieu à l'évolution du poids moyen d'un message. Ce poids moyen est représenté dans la Fig. 8.7a. Il est important de noter l'échelle logarithmique de la figure. Les poids des messages de mise, de récompense, et de confirmation de récompense sont bien moindres que ceux des messages d'annonce ou de clôture de tour. De plus, les poids moyens des messages de mise, de récompense, et de confirmation de récompense restent stables alors que ceux des messages d'annonce et de clôture d'un tour d'enchère augmentent linéairement. Ces deux derniers sont tous deux composés en majorité Multi-Robot HTN du commissaire-priseur, leur évolution linéaire s'explique par l'évolution de la taille du problème de MRTA. La taille moyenne d'un message de clôture d'enchère est très légèrement supérieure à celle d'un message d'annonce, cela est dû à l'ajout d'information dans le Multi-Robot HTN pour encoder les contraintes des mises retenues.

Nous présentons ensuite dans la Fig. 8.7b le poids total des données échangées lors du processus de décision. Le poids total est affiché par nombre de robots dans le système et par nombre de tâches en vente, ainsi, à chaque nombre de robots est associé 6 poids totaux correspondant de gauche à droite à 5, 10, 15, 20, 25, et 30 tâches en vente.

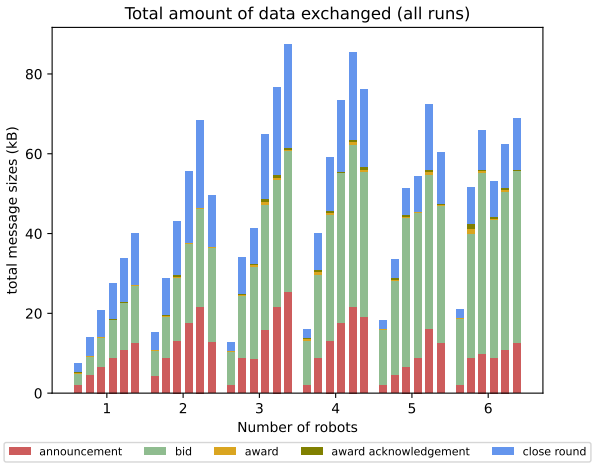
Sur cette figure, les poids des messages de récompenses et de confirmation de récompenses semblent négligeables devant ceux des mises, d'annonces, et de clôtures d'un tour d'enchère. Pour chaque taille de problème, le poids total des communications augmente, entrainé par l'augmentation du poids d'un message d'annonce et de clôture, mais également par l'augmentation du nombre de mises.

Au regard du nombre de robots dans le système, le poids total des communications n'évolue pas de manière linéaire. Par exemple, le poids total des communications pour un système composé de 5 à 6 enchérisseurs est plus faible que pour un système composé de 3 à 4 pour les problèmes de 15 à 30 tâches. Ce résultat contre intuitif souligne l'impact du nombre de tours d'enchères sur le poids total des

communications, notamment par l'envoi, pour chaque tour, de volumineux messages d'annonce et de clôture. Une enchère avec un plus grand nombre d'enchérisseurs peut mener à un plus grand nombre d'allocation par tour et donc réduire à terme le nombre de tours nécessaires. Comme moins de tours sont nécessaires, le nombre total de messages est réduit.



(a) Poids moyen d'un message du protocole de décision (en bits). L'échelle est logarithmique.



(b) Poids total des communications issues du protocole de décision. Pour chaque nombre de robots, les résultats par taille du problème sont affichés dans cet ordre : 5, 10, 15, 20, 25, et 30 tâches en vente.

FIGURE 8.7 – Poids en bits par type de communication et poids total par problème de MRTA. Les données affichées représentent la moyenne de toutes les itérations effectuées.

Discussion

Dans notre implémentation, la taille des messages n'est pas optimisée, aussi, de nombreuses données sont redondantes. Par exemple, le Multi-Robot HTN envoyé dans le message de clôture d'un tour correspond à celui du message d'annonce avec uniquement quelques nouvelles données. C'est ce que l'on peut observer sur la Fig. 8.7a. L'optimisation du message de clôture pour n'inclure que ces nouvelles données permettrait une économie significative du nombre de bits à envoyer. De même, une enchère pouvant se dérouler en plusieurs tours, ce même type d'économie est possible pour les messages d'annonce ultérieurs à celui du tour initial.

Enfin, les poids totaux obtenus ici semblent en adéquation avec l'hypothèse de $3kbps$ que nous considérons pour les modems acoustiques des robots, mais doivent être traités prudemment. Par exemple, dans le pire cas montré dans la Fig. 8.7b, $90kbits$ doivent être échangés, $90/3 = 28.33secondes$ seraient donc théoriquement nécessaires à un tel échange. Cependant, cette première approximation du coût des communications ne prend pas en compte la saturation de la bande passante à un instant donné. Si ces résultats semblent encourageants, de plus amples expérimentations sont encore à mener.

8.3.5 *Impact de la stratégie de revente*

Dans notre approche, la stratégie de revente permet de fixer le coût de revente d'une tâche au sein du processus d'enchère, c.-à-d. un prix d'appel.

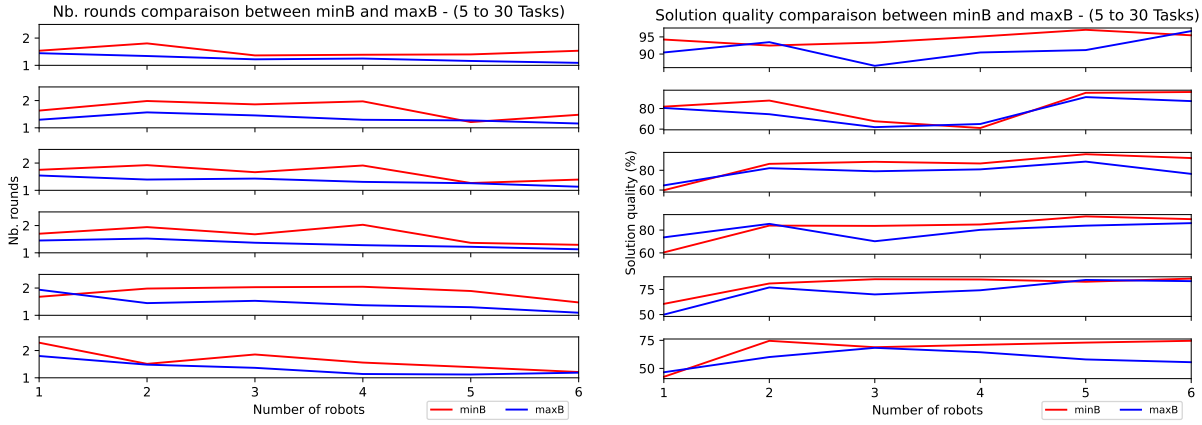
Dans de précédents travaux, nous avions comparé les stratégies *minB* et *maxB* (Def. 6.11 et Def. 6.10) sur un problème de planification initiale [Mil+21a]. Nos résultats mettaient en avant une meilleure qualité

de solution avec *minB* au détriment d'un nombre de tours d'enchères plus important, et inversement pour *maxB*. Cependant, ces résultats ont été obtenus sur une preuve de concept du protocole de décision qui ne supportait que les contraintes de précédence. Notre implémentation actuelle du schéma de décision ne partage que peu d'éléments avec la version utilisée dans ces travaux. Nous avons donc reconduit cette comparaison similaire avec la version actuelle du protocole de décision afin de déterminer l'impact des stratégies *minB* et *maxB*.

Résultats

La Fig. 8.8a compare le nombre de tours d'enchères nécessaires pour chaque stratégie. Pour chaque problème de MRTA de notre évaluation, c.-à-d. chaque tuple {*nb. enchérisseurs, nb. tâches, qualité de communication*} où *nb. tâches* est fixé, la moyenne obtenue est affichée dans le graphe correspondant, p. ex. le graphe du haut représente le nombre de tours d'enchères moyen pour un système allant de 1 à 6 enchérisseurs et une qualité de communication de 10 à 100% pour 5 tâches en vente. Ainsi, il est possible d'observer un nombre de tours généralement plus important avec la stratégie *minB*.

De même, la Fig. 8.8b compare la qualité d'allocation pour chaque stratégie. Il est ainsi possible d'observer une qualité d'allocation généralement meilleure avec *minB*.



(a) Nombre de tours d'enchères.

(b) Qualité d'allocation.

FIGURE 8.8 – Comparaison des stratégies *minB* et *maxB* en fonction du nombre de tours d'enchères et de la qualité d'allocation. Ici, les données représentent la comparaison du résultat moyen de chaque stratégie de revente sur chaque problème de MRTA pour un nombre fixé de tâches en vente. À chaque graphe correspond un nombre de tâches, de haut en bas : 5, 10, 15, 20, 25 et 30 tâches.

Discussion

Les résultats obtenus ici semblent confirmer ceux de nos précédents travaux [Mil+21a]. La stratégie *minB*, qui peut être vue comme une stratégie optimiste sur les futures mises, à tendance à favoriser la revente afin d'améliorer la qualité d'allocation. La stratégie *maxB*, qui est une stratégie pessimiste sur les futures mises, à tendance à favoriser l'allocation de plus d'éléments et ainsi réduire le nombre de tours d'enchères nécessaires.

La stratégie de revente apparait donc comme un paramètre permettant de contrôler le comportement du protocole de décision.

8.3.6 *Impact de la taille du Multi-Robot HTN sur le temps de résolution du WDP et de l'estimation*

Dans notre approche, chaque robot conserve un Multi-Robot HTN qui ne fait que grandir au fur et à mesure de la mission avec l'intégration de nouvelles tâches. Usuellement, chaque fois que le Multi-Robot HTN augmente une enchère est menée pour allouer les nouvelles tâches, ainsi, seule une sous-partie du Multi-Robot HTN est à allouer et le reste des tâches contenues peut être considéré comme "inerte" car il n'est pas nécessaire de miser dessus. La résolution d'un problème de planification, que cela soit pour l'estimation ou le WDP, nécessite cependant de résoudre la totalité du Multi-Robot HTN, il est donc nécessaire d'évaluer quel impact la taille de ce Multi-Robot HTN peut avoir sur la complexité des problèmes de planification. Pour cela, nous nous sommes servis du protocole de décision en considérant un article en vente dont la taille augmente linéairement, mais le nombre de labels en vente reste stable.

Pour mener cette expérience, nous avons considéré :

— Un Multi-Robot HTN de référence noté $\mathcal{H}_{ref}$ et composé de 5 tâches ;

— Trois robots, dont un seul commissaire-priseur ;

— Des communications parfaites, c.-à-d. une qualité de communication de 100%.

Pour cette expérience, le commissaire-priseur mène une enchère initiale avec $\mathcal{H}_{ref}$ comme article à vendre. Le nombre total de tâches dans l'article à vendre est donc de 5 tâches pour 5 labels en vente.

Puis, $\mathcal{H}_{ref}$ est passé au commissaire-priseur comme un ensemble de nouveaux objectifs, il intègre donc par fusion $\mathcal{H}_{ref}$ a son propre Multi-Robot HTN avant de mettre en vente 5 nouveaux labels. Le nombre total de tâches dans le nouvel article à vendre est donc de 11 tâches (5 pour l'ancien Multi-Robot HTN, 5 pour les nouveaux objectifs et 1 nouvelle tâche racine issue du processus de fusion) pour 5 labels en vente.

À l'issue de cette seconde enchère, l'opération est réitérée en intégrant une nouvelle fois $\mathcal{H}_{ref}$ comme de nouveaux objectifs en vente. Nous procédons ainsi en résolvant successivement 10 enchères.

Résultats

La Fig. 8.9 montre l'évolution du pire temps d'estimation d'une mise et du pire temps de résolution du WDP pour chaque enchère. À chaque enchère, le nombre de tâches augmente de façon linéaire par construction du problème.

Sur la figure, les pires temps d'estimation et de résolution du WDP augmentent eux aussi de façon linéaire avec la taille de l'article en vente.

Discussion

Ces résultats montrent que la taille du Multi-Robot HTN impacte de façon linéraire les temps de résolution de nos problèmes de planification. Cette influence du nombre de tâches, s'il ne remet pas en question l'approche, doit être pris en compte dans la modélisation de le mission, car un Multi-Robot HTN de très grande taille pourrait poser problème même si ces tâches sont allouées petit à petit. Dans notre contexte d'une mission de chasse aux mines, les nouveaux objectifs à intégrer font principalement suite à des MILCO, pouvant raisonnablement être considérés comme ponctuels et moins nombreux que les objectifs initiaux, par conséquent, l'augmentation de la taille du Multi-Robot HTN ne semble pas poser de réels problèmes.

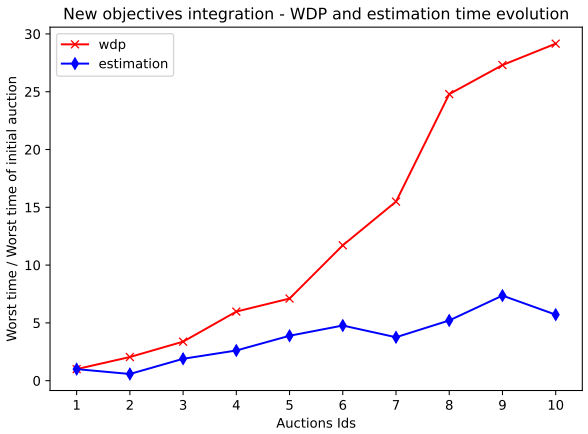


FIGURE 8.9 – Évolution du pire temps du WDP et de l'estimation en fonction du nombre de tâches. Pour chaque enchère le plus grand temps est rapporté à celui de l'enchère initiale. De même, le nombre de tâches est rapporté à celui de l'enchère initiale. Ces résultats sont donnés pour 3 enchérisseurs et 10 itérations.

8.4 CONCLUSION

L'évaluation que nous avons menée nous a permis d'établir une preuve de concept de notre approche. Au cours d'une mission d'évaluation de type chasse aux mines, nous avons montré comment les différents protocoles de HaucTioN permettaient de résoudre en ligne des problèmes de MRTA comportant des dépendances complexes, de superviser la bonne exécution des décisions prises, et de réparer le plan lorsque nécessaire.

Concernant le protocole de décision, qui constitue notre principale contribution et le coeur de HaucTioN, nous avons présenté des résultats qui montrent que ce protocole permet d'obtenir, sur les problèmes considérés, des solutions de bonne qualité tout en bénéficiant d'une certaine robustesse aux pertes de messages. Nous avons également souligné comment la stratégie de revente pouvait permettre de contrôler le comportement d'allocation en optimisant la qualité d'allocation ou le nombre de tours d'enchères nécessaires. De plus, le poids total des communications échangées dans le cadre de ce protocole semble en adéquation avec les hypothèses de bande passante du monde sous-marin. Enfin, nous avons observé que la principale limitation de ce protocole réside dans le temps de résolution du WDP qui évolue de façon exponentielle avec le nombre de mises à considérer.

Ces résultats sont encourageants pour la poursuite de nos travaux, mais de plus amples expérimentations sont nécessaires pour confirmer les tendances observées. C'est un des sujets du prochain chapitre, qui fait état des perspectives de notre approche.

9 CONCLUSIONS ET PERSPECTIVES

Aperçu

Dans ce chapitre, nous concluons sur les avantages et limites de notre approche et introduisons des perspectives en vue de son amélioration.

9.1 BILAN

Nous avons présenté HaucTioN, une architecture visant à relever certains des défis de la coopération multirobot. Les principaux défis que nous avons abordés sont :

— La résolution en ligne de problèmes de MRTA comportant des dépendances complexes ;

— La supervision et l'exécution des plans obtenus à la résolution de ces problèmes ;

— La réparation des plans lorsqu'un aléa est rencontré ;

— La robustesse de ces processus aux problèmes de communication.

Et, dans le Chap. 8, nous avons établi en simulation une preuve de concept de cette architecture et présenté quelques analyses de résultats.

Dans le Chap. 4, nous avons introduit le fonctionnement de l'architecture dans sa globalité et notamment comment les algorithmes de HaucTioN reposent sur l'exploitation de structures hiérarchiques complexes pour accomplir la mission. Ces structures permettent d'exprimer des dépendances entre les tâches, et notamment des contraintes de précédence, de synchronicité et de causalité. Nous modélisons ces structures sur des formalismes HTN et HDDL intégrant un cadre temporel et présentés dans le Chap. 5.

Dans le Chap. 6, nous avons détaillé comment le protocole de décision utilise ces structures hiérarchiques en entremêlant allocations par enchères et planification hiérarchique afin de résoudre un problème de MRTA. En procédant ainsi, nous permettons aux robots d'obtenir efficacement des ensembles de tâches tout en considérant leurs dépendances complexes. En basant ce protocole sur un schéma d'allocation par enchères, nous mettons en place un schéma de décision décentralisé dont la robustesse aux pertes de communication a été éprouvée dans notre évaluation. De plus, les résultats obtenus sur le coût en communication d'un tel schéma semblent en adéquation avec les hypothèses du monde sous-marin. Enfin, nous avons souligné comment ce protocole intègre un levier de contrôle de la décision grâce aux stratégies de revente.

Dans le Chap. 7, nous avons présenté le fonctionnement de l'architecture durant la mission en détaillant les protocoles de supervision et de réparation. Le protocole de supervision fait usage des structures hiérarchiques issues de la décision en y intégrant les informations générées en ligne lors de l'exécution des tâches. Dans ce chapitre, nous avons expliqué comment la supervision du plan était réalisée grâce à la planification hiérarchique, et comment le protocole de supervision interagissait avec des méthodes de réparation en cas d'aléas.

Nous avons notamment détaillé comment les méthodes de réparations de HaucTioN permettent de corriger de précédentes décisions ou d'intégrer de nouveaux objectifs à la mission en réexploitant les mêmes structures hiérarchiques que pour la décision et la supervision. Enfin, nous avons formalisé comment ces structures étaient réinjectées dans le protocole de décision afin de corriger le plan en résolvant en ligne un problème de MRTA.

Bien que les résultats obtenus dans le Chap. 8 soient encourageants, de nombreux défis restent à relever, notamment sur les problématiques de supervision et réparation. Aussi, dans la suite de ce chapitre nous introduisons différentes perspectives en vue de futurs développements de notre approche.

9.2 LIMITES ET PERSPECTIVES DU PROTOCOLE DE DÉCISION

Dans cette section, nous mettons en lumière certaines limitations de l'implémentation actuelle du protocole de décision de HaucTioN et proposons des pistes d'améliorations.

9.2.1 *Vers une meilleure gestion des contraintes issues de la décision*

Nous utilisons un schéma de décision décentralisé basé sur des enchères. Une des principales difficultés de la prise en compte des dépendances complexes dans un tel schéma est de pouvoir déterminer les intentions des enchérisseurs sur le plan global d'équipe. Nous avons proposé une première méthode reposant sur la connaissance des liens causaux entre tâches.

Limites des liens causaux

Dans HaucTioN, chaque enchérisseur planifie pour chaque mise : les tâches multirobots qu'il doit effectuer, ses actions locales, les tâches multirobots permettant de compléter la mission. Avec ce dernier point, l'agent intègre les tâches qui sont accomplies par ses coéquipiers et suppose celles qui le seront. La solution obtenue par l'agent est un plan global théorique qui inclut certaines de ses actions locales. Cette solution permet à l'agent de déterminer la valeur de sa mise et de transmettre ses intentions sur les tâches multirobots, c.-à-d. son plan.

Dans le plan solution, cependant, toutes les variables sont fixées, y compris les *timepoints* de début et de fin des tâches. L'instanciation de ces éléments est nécessaire au solveur pour atteindre une solution. Si l'enchérisseur transmettait directement ce plan solution avec sa mise, tous les *timepoints* seraient fixés avec des contraintes de synchronicité. Comme présenté dans la Sec. 6.5.4, il est donc nécessaire à l'agent de relâcher les contraintes superflues de ce plan afin de ne pas surcontraindre le problème de MRTA.

Pour relâcher ces contraintes, nous nous appuyons sur les liens causaux entre les tâches. Avec cette méthode, les contraintes de synchronicité sont supprimées et remplacées par des contraintes de précédence lorsqu'un lien causal existe. Cette méthode comporte cependant des limites.

Parfois, certaines contraintes de synchronicité peuvent être réellement nécessaires, par exemple, pour établir des rendez-vous entre agents. Dans notre cadre, ces rendez-vous peuvent être associés à des objectifs de relais de communication ou de recharge auprès d'un agent USV. Les liens causaux ne permettent pas d'identifier explicitement ces contraintes, qui sont donc relâchées.

De plus, les chaines de causalité sont construites sur les conditions et effets des tâches, certains liens implicites entre tâches ne peuvent pas être capturés par ces chaines. Par exemple, un agent possède une certaine autonomie qui peut décroitre en fonction de ces actions. L'encodage explicite de cette autonomie dans les conditions et effets des tâches peut surcontraindre inutilement la mise, mais l'absence d'une

telle information peut mener à des problèmes à l'exécution, p. ex. parce que l'agent doit absolument se recharger entre deux tâches.

Enfin, cette autonomie peut également décroitre avec le temps qui passe, même si l'agent n'effectue aucune action. Dans un tel cas, un lien existe entre l'autonomie et les *timepoints* instanciés de la solution. Tenir compte de ces éléments est un défi important du relâchement des contraintes de la mise.

Restreindre l'ajout de contraintes aux participants d'une enchère

Dans notre protocole de décision, un enchérisseur peut contraindre une tâche qu'il ne possède pas. La seule condition à cela est qu'une chaine de causalité existe entre cette tâche et la tâche qu'il estime (ou une des tâches qu'il a déjà acquises). Cette absence de limitations peut entrainer un blocage dans la mission. Nous représentons un tel cas dans Ex. 9.1.

Exemple 9.1 : Ajout de contraintes sur les tâches d'un robot absent

Soient $AUVe1 \bullet (\, l_x \,)$, $AUVe2 \bullet (\, l_y \,)$ et $AUVe3 \bullet \varnothing$ avec la contrainte : $(l_x \rightarrow l_y)$, c.-à-d. $\mathsf{start}_y > \mathsf{end}_x$.
Après une enchère en ligne, à laquelle seuls $AUVe1$ et $AUVe3$ participent, soient $AUVe3 \bullet (\, l_z \,)$ et une nouvelle contrainte issue du processus de décision : $(l_z \rightarrow l_y)$.
Durant la suite de la mission, $AUVe1$ diffuse deux premiers *achievements* pour signaler qu'il a commencé puis terminé l_x aux instants t_1 et t_2. À la réception de ce message, $AUVe2$ commence l_y, car il n'a pas connaissance de $(l_z \rightarrow l_y)$. En commençant cette tâche, il diffuse un *achievement* signalant le début de l_y à l'instant t_3.
À la réception de ce dernier *achievement* par $AUVe3$ (ou $AUVe1$), il devient impossible de résoudre le problème de supervision, car les contraintes $(\mathsf{end}_x = t_2)$, $(\mathsf{start}_y = t_3)$, et $(\mathsf{start}_z >= t_c)$ sont intégrées au problème de supervision et la contrainte $\mathsf{start}_y > \mathsf{end}_z$ ne peut plus être respectée.

Pour éviter de tels cas, nous pensons qu'il est nécessaire d'imposer certaines restrictions dans le processus d'estimation des mises des enchérisseurs. Ces restrictions visent à empêcher l'intégration dans le WDP de mises dont les contraintes supplémentaires non pas été validées par la totalité des agents concernés par ces contraintes.

Dans un environnement où les communication sont limitées, il est difficile de mettre en place un processus de consensus visant pour un agent à obtenir l'approbation des coéquipiers dont sa mise dépend. Aussi, nous pensons que ces restrictions doivent être encodées dans le problème d'estimation de façon à empêcher un enchérisseur de faire dépendre son plan des tâches d'un agent ne participant pas à l'enchère.

Ensuite, le processus de clôture du tour d'enchère pourrait comporter une phase d'approbation générale des enchérisseurs. Cette phase serait construite sur le modèle déjà en place des accusés de réception, mais en imposant l'obtention d'un accusé de réception non pas uniquement de l'agent ayant gagné une mise mais également des coéquipiers dépendants de la mise.

9.2.2 *Complexité du problème de détermination des gagnants*

Une des principales limitations de notre approche réside dans la complexité du problème de détermination des gagnants. Comme vu dans le Chap. 8, le temps de résolution du WDP augmente exponentiellement avec le nombre de mises à considérer. Bien qu'il soit possible de considérer un nombre important de mises en un temps raisonnable, p. ex. 80 mises en environ 3 minutes avec notre implémentation d'évaluation, il

est impératif de tenir compte de cette limite afin de respecter les spécificités de la prise de décision en ligne.

Enfin, certaines méthodes pourraient être en mesure d'améliorer le temps de résolution du WDP. Par exemple, en utilisant des heuristiques pour guider la recherche, il existe des contributions dans ce sens dans l'état de l'art [Ber+17]. Ou, de manière plus spécifique au protocole de décision, en réduisant artificiellement le nombre de mises à considérer.

Limitations du nombre de mises formulées

Grâce à $\mathcal{H}_\delta$, la structure hiérarchique de l'article en vente, les robots peuvent formuler des mises sur des ensembles de tâches. Dans HaucTioN, les robots sont dans l'obligation d'estimer et d'envoyer une mise pour chaque label en vente dans $\mathcal{H}_\delta$.

Dans notre évaluation, ce processus exhaustif d'estimation ne semble pas poser de réels problèmes, car le temps de résolution d'un problème d'estimation reste très inférieur à celui de la détermination des gagnants. La réelle limitation de cette approche exhaustive est dans l'obligation pour le commissaire-priseur d'intégrer l'ensemble des mises réceptionnées. Sous hypothèse de communications parfaites, cela signifie que le commissaire-priseur doit tenir compte de toutes les mises possibles, c.-à-d. le nombre de labels en vente multiplié par le nombre d'enchérisseurs. Cette intégration exhaustive des mises au WDP peut augmenter de manière importante son temps de résolution.

Cependant, il est envisageable de réduire artificiellement le nombre de mises en limitant les enchérisseurs à ne transmettre qu'un sous-ensemble de leurs mises. Ce sous-ensemble peut être construit de différentes manières. Par exemple, l'enchérisseur peut s'appuyer sur un modèle de référence représentant un agent générique pour estimer une mise de référence. En s'aidant de cette mise de référence, l'enchérisseur peut ne sélectionner que ses meilleures mises, p. ex. la valeur de la mise de référence peut être utilisée comme prix d'appel.

Il est également possible de s'appuyer sur la structure hiérarchique pour réduire le nombre de mises. Par exemple, en limitant le nombre de mises quand plusieurs alternatives sont possibles ou en favorisant les mises sur les tâches abstraites.

Enfin, le commissaire-priseur peut également filtrer les mises à intégrer au WDP. Par exemple, grâce à l'ensemble des mises reçues, il peut définir un premier coût de revente pour les labels en vente. Il peut ensuite ne conserver que les mises dont la valeur n'est pas trop éloignée de ce coût de revente.

Il est important de noter qu'en filtrant artificiellement les mises, l'ensemble des solutions du WDP est réduit. Par conséquent, il est possible de perdre la solution optimale, ou d'autres solutions de bonne qualité. Il est donc nécessaire de traiter de telles méthodes avec prudence en procédant, par exemple, à une étude empirique de leurs paramètres.

9.2.3 *D'une gestion implicite à une gestion explicite des incertitudes*

Dans notre approche, nous priorisons la capacité de réparation à celle de l'amélioration dynamique d'un modèle de connaissance. Ce choix est guidé par le besoin vital pour le système d'adaptation du plan via la détection de divergences la réparation.

Nos travaux portant avant tout sur la coopération entre robots, nous n'avons pas considéré explicitement les problématiques d'incertitudes sur la localisation et la perception, mais uniquement leurs conséquences lors de l'exécution.

Ces problématiques d'incertitudes représentent cependant un défi important du déploiement d'un système robotisé en environnement réel. Elles apparaissent généralement à l'échelle de l'agent qui doit, par exemple, estimer sa position. Dans un milieu sous-marin, il n'est pas possible de se reposer sur un GPS, aussi le robot doit utiliser des capteurs comme des IMU ("Inertial Measure Unit") ou DVL ("Doppler Velocity Log"). Cependant, avec ces méthodes, l'erreur entre la position estimée du robot et la réalité ne fait qu'augmenter. Une dérive apparait, imposant aux drones sous-marins d'utiliser des dispositifs de localisation, tels que des balises acoustiques, ou de remonter régulièrement à la surface faire des points GPS.

Dans notre approche, les conséquences de ces incertitudes sont détectées par le protocole de supervision des agents et réparées grâce au schéma de décision. Néanmoins, anticiper ces conséquences en intégrant explicitement ces incertitudes au processus de décision peut participer à l'amélioration de la résilience du système.

Pour ce faire, il serait par exemple possible d'exploiter le schéma d'enchère avec des mises incertaines. Un agent pourrait indiquer des valeurs minimale et maximale pour sa mise qui pourront ensuite être exploitées lors de la résolution du WDP.

9.2.4 *Refonte du formalisme des enchères*

Le coeur du protocole de décision est basé sur un schéma d'enchère. Dans le but d'évaluer au mieux les performances de ce protocole, il serait intéressant de le comparer à d'autres schémas d'allocations par enchères.

Pour cela, il est possible de s'appuyer sur une taxonomie des schémas d'enchères afin d'identifier les différences fondamentales entre les approches comparées. Bien que l'originalité de notre approche soit d'inclure des structures hiérarchiques au sein d'enchères, il est par exemple possible de définir notre schéma comme un schéma d'enchères parallèles (Parallel Single-Item auctions (PSI)) où les items en vente sont des ensembles de tâches selon les schémas définis dans [Dia+06].

Ensuite, l'exploitation d'un formalisme commun peut améliorer la pertinence d'une étude comparative. Cependant, les méthodes d'allocation par enchères sont aussi nombreuses que variées et, bien que leurs bases soient communes, il n'existe pas à notre connaissance de référence sur la manière de les formaliser afin d'en faciliter la comparaison.

De récents travaux ont pour objectif de surmonter ce problème en proposant une formalisation des enchères [MP20b]. Bien que la formalisation proposée soit encore incomplète, nous pensons qu'il important de suivre son évolution et de faire l'exercice de cette formalisation.

9.3 LIMITES ET PERSPECTIVES DU PROTOCOLE DE SUPERVISION ET D'EXÉCUTION

Pour pouvoir évoluer dans une mission aussi incertaine que la chasse aux mines, le système doit être tolérant aux fautes.

La supervision d'un plan peut se faire en utilisant l'état du plan perçu à un instant courant comme référence. Dans notre approche, cette perception de l'état du plan repose uniquement sur les *achievements* échangés entre les robots. Cependant, de nombreux problèmes en lien avec l'utilisation de ces *achievements* peuvent apparaitre.

9.3.1 *Rigidité de la planification du problème de supervision*

Dans l'architecture, le protocole de supervision utilisé oblige les agents à replanifier intégralement la mission en intégrant les informations connues sur l'état du plan, pour ce faire un agent :

— Utilise les *achievements* reçus pour fixer dans le "passé" les timepoints associés ;

— Établit des contraintes avec le temps courant pour les éléments pour lesquels aucun *achievement* n'est connu.

En procédant ainsi le problème de supervision doit trouver une solution satisfaisante pour :

1. les tâches en cours d'exécution ;
2. les tâches à accomplir ;
3. les tâches déjà accomplies.

Ce dernier point, que nous appelons la **sensibilité au passé**, peut être problématique en raison des divergences de connaissances possibles entre agents. Dans certains cas, le problème de supervision peut ne pas avoir de solution alors même qu'il est théoriquement possible au robot d'exécuter la fin de son plan. Dans Ex. 9.2 nous illustrons un tel cas.

Exemple 9.2 : Sensibilité au passé du problème de supervision

Soient la situation de Ex. 9.1 et $AUVe4 \bullet (\, l_w \,)$. Aucune contrainte ne lie l_w à l_x, l_y, ou l_z. De plus, $AUVe4$ a connaissance de la contrainte $(l_z \rightarrow l_y)$ issue d'une enchère en ligne.
À la réception des mêmes *achievements* que décrit dans Ex. 9.1, $AUVe4$ va également se retrouver bloqué dans son processus de supervision alors que sa tâche ne dépend pas déroulement de l_x, l_y, ou l_z.

Dans une optique de tolérance aux fautes et de réussite "autant que possible" de la mission, un tel comportement est problématique. Pour pallier ce défaut, une première amélioration peut passer par le relâchement des contraintes ajoutées dans le problème de supervision dont l'agent ne dépend pas. Par exemple, l'agent peut tirer profit des chaines de causalité sur les tâches multirobots.

Ces chaines de causalité sont connues à chaque résolution d'un problème de planification, et donc à minima lors d'une prise de décision grâce à la résolution du WDP. Le commissaire-priseur pourrait ainsi transmettre les chaines de causalité qu'il a déterminées en résolvant le problème de MRTA.

Ensuite, pour superviser son plan, l'agent peut prendre en considération ces chaines de causalité pour ajouter ou non les contraintes issues des *achievements* et du temps courant. Si aucune chaine de causalité ne relie les tâches qu'il doit accomplir à une autre tâche, alors il n'est pas nécessaire de contraindre cette autre tâche.

Cette proposition peut également aller de pair avec le comportement strict actuel afin de proposer à l'agent plusieurs modes de supervision. Par exemple, dans la Fig. 9.1 nous présentons un protocole permettant d'exploiter deux modes : le mode strict actuel et le mode avec relâchement des contraintes.

Par défaut, l'agent peut vérifier son plan en intégrant toutes les informations dont il a connaissance, s'il trouve une solution "*Valid plan*" alors le plan multirobot tel que connu par l'agent se déroule bien. S'il ne trouve pas de solution, alors l'agent peut procéder à une seconde supervision en relâchant les contraintes sur les tâches dont il ne dépend pas. S'il trouve une solution à ce second problème, alors l'agent peut continuer à exécuter son plan tout en sachant qu'une incohérence existe dans le plan de ses coéquipiers. La connaissance d'une telle information peut, par exemple, servir à déclencher une réparation de groupe

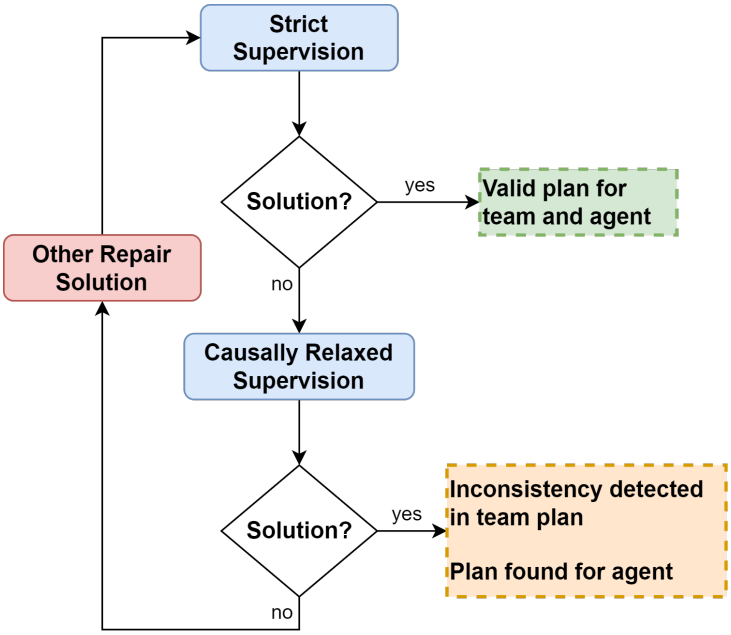


FIGURE 9.1 – Protocole de supervision alternant supervision stricte et supervision relaxée.

où tous les coéquipiers se concertent. S'il ne trouve pas de solution, alors le plan de l'agent est mis en défaut et il doit faire appel au protocole de décision pour essayer de le réparer.

9.3.2 *Exploitation des plans locaux*

Dans notre approche, l'exécution des tâches multirobots repose sur l'exécution d'un ensemble d'actions locales déterminé à la résolution du dernier problème de supervision. Les tâches multirobots sont supervisées grâce à des *achievements* générés à partir de ces actions locales. Les prédicats locaux et multirobots étant séparés, nous raisonnons sur l'état des actions locales pour inférer des informations sur les tâches multirobots, p. ex. est-ce que la première action locale a débuté et est-ce que toutes les actions locales sont terminées.

À chaque résolution d'un problème de supervision, l'ensemble des actions locales associé à une tâche multirobot est susceptible d'être modifié. Par exemple, si une tâche multirobot consiste à exécuter une série de 5 déplacements et que cette séquence locale est interrompue pour accomplir une autre tâche, un ou plusieurs autres déplacements peuvent être nécessaires pour reprendre la première tâche.

Pour éviter de recommencer toutes les actions locales lorsqu'un tel cas arrive, nous identifions ces actions locales et ne répétons pas les actions qui ont déjà été achevées dans le cadre d'une tâche multirobot en particulier. En procédant ainsi, nous évitons des répétitions inutiles, mais nous limitons les possibilités de réparations.

Cependant, il peut parfois être nécessaire à un agent de recommencer une action locale. Par exemple, dans une optique de maitrise des incertitudes, il peut être imposé que des actions de couvertures ne doivent pas être interrompues.

De plus, avec cette approche les tâches locales dont l'exécution a débuté, mais ne s'est pas encore achevée seront entièrement recommencées.

Pour prendre en compte ces problèmes, nous pensons qu'il est possible d'exploiter des *achievements* locaux ainsi que des modélisations dynamiques des réseaux de tâches locaux. Grâce aux *achievements*

locaux, l'état d'une action locale peut être encodé. Il est ensuite possible de considérer deux décompositions locales lors de la supervision d'une tâche multirobot, avec une première décomposition où les actions déjà exécutées sont encodées grâce à leurs *achievements*, et une seconde décomposition générique où ces mêmes actions sont recommencées, car sans *achievements*.

Ainsi, lors de la résolution d'un problème de supervision, un agent peut planifier quels éléments de son plan local doivent être repris, abandonnés, ou recommencés.

9.4 VERS UNE MEILLEURE TOLÉRANCE AUX FAUTES

Lorsqu'un système multirobot est déployé en conditions réelles, il doit faire face à des aléas pouvant provoquer des erreurs dans les différents processus mis en oeuvre. Dans cette section, nous présentons plusieurs perspectives pour améliorer la tolérance aux fautes du système.

9.4.1 *Horizon de supervision*

Dans notre approche, la supervision qu'un agent fait de son plan repose sur la résolution d'un problème de planification. Ce processus de supervision est fait en prenant en compte l'état connu du plan via la mécanique des *achievements*.

Malheureusement, dans la version actuelle de l'architecture, cette information est exploitée de façon binaire : si un plan existe, alors l'exécution du robot peut continuer, si aucun plan n'est trouvé, alors il est nécessaire de réparer. En procédant ainsi, il est impossible à un agent d'anticiper des erreurs. L'agent ne peut détecter des erreurs que lorsque leurs conséquences se font ressentir en l'empêchant de trouver une solution à son problème de supervision.

Pour un système multirobot évoluant en environnement incertain, la résilience du système face aux erreurs est primordiale. Anticiper suffisamment une erreur pour, soit la prévenir, soit préparer en amont la réaction du système, peut être un moyen d'améliorer cette résilience.

Dans HaucTioN, une manière simple d'améliorer l'horizon d'anticipation d'un agent serait d'exploiter les contraintes avec le temps courant. Pour rappel, ces contraintes sont établies pour les éléments dont aucun *achievement* n'est connu.

Pour une tâche $[\mathsf{start}, \mathsf{end}]\mathit{mr\text{–}task}(l_x)$ commencée, mais pas encore achevée, une contrainte de ce type dans le problème de supervision est : $(\mathsf{end} >= t_c)$, avec t_c le temps courant. Si le plan de l'agent dépend de la bonne exécution de $\mathit{mr\text{–}task}(l_x)$ avant une échéance t_d donnée, il ne peut réagir que lorsque $t_c > t_d$ car : $(\mathsf{end} >= t_c)$ et $(\mathsf{end} <= t_d)$ sont incompatibles.

Cependant, en remplaçant la contrainte portant sur le temps courant par $(\mathsf{end} >= t_c + 5)$, l'agent augmente son horizon d'anticipation de 5 unités de temps. Si un plan est trouvé, l'agent peut continuer à exécuter ses tâches, si aucun plan n'est trouvé, l'agent bénéficie de 5 unités de temps pour prévenir le problème ou anticiper ses conséquences.

Il est donc possible à un agent d'anticiper de possibles erreurs sans changer la structure de notre protocole. Il est cependant important de noter que, pour être pleinement exploité, un tel mécanisme nécessite de pouvoir identifier les raisons d'un échec de planification. Ce mécanisme d'horizon d'anticipation doit donc s'appuyer idéalement sur une connaissance explicite des tâches dont l'agent dépend.

9.4.2 *Représentation des dépendances entre agents et propagation des contraintes*

Les problèmes que nous abordons comportent des dépendances complexes entre les tâches qui les composent. En répartissant ces tâches aux robots, ces dépendances sont transmises. En conséquence, les plans des robots sont dépendants entre eux. Il est donc vital pour un robot de superviser le déroulement des plans dont il dépend afin d'éviter des blocages à l'exécution.

Dans notre approche, la supervision qu'un agent fait de son plan, et de celui de ses coéquipiers, repose sur un problème de planification intégrant l'état connu des tâches multirobots. De même, des problèmes de planification sont utilisés lorsqu'il s'agit de résoudre un problème de MRTA en ligne. Ces problèmes de planification permettent à l'agent de déterminer si un plan global théorique existe. Cependant, il est difficile d'obtenir à partir de ces problèmes de planification de plus amples informations permettant, par exemple, à un agent de connaitre explicitement les tâches et coéquipiers dont il dépend.

Par exemple, si l'agent est dépendant du plan d'un coéquipier, il peut avoir intérêt à rester à portée de ce coéquipier. La connaissance explicite des dépendances le liant à ce coéquipier peut être utilisée pour guider ses décisions futures. Ainsi, il peut lui être possible d'optimiser la supervision des éléments dont il dépend.

Cette connaissance explicite peut être encodée au sein des plans des agents de différentes manières. Par exemple, dans [CLP15], les auteurs utilisent des STN pour intégrer ces dépendances. Avec ces STN, l'agent peut connaitre les tâches dont il dépend et agir en conséquence, p. ex. en propageant aux coéquipiers concernés une information sur un retard.

Dans notre cas d'usage, les tâches peuvent également comporter des alternatives, c.-à-d. plusieurs manières d'accomplir une même tâche. Au fur et à mesure des réparations, une alternative retenue peut être abandonnée au profit d'une autre. Avec notre protocole de décision, ces choix sont faits en prenant en considération les dépendances avec les plans des autres agents, c.-à-d. qu'une alternative ne peut pas être choisie si elle est incompatible avec une précédente décision. Au regard de l'importance pour un agent de connaitre explicitement ses dépendances, cela signifie qu'un agent doit considérer la tâche dont il dépend au même titre que les alternatives qui lui sont possibles.

Il peut donc être intéressant d'explorer l'utilisation de structures hiérarchiques pour encoder explicitement les plans des agents et leurs dépendances.

9.4.3 *Choix des tâches à revendre*

Durant la mission, les agents peuvent rencontrer des aléas les obligeant à abandonner ou revendre certaines de leurs tâches. Dans notre implémentation d'évaluation, le choix des tâches à abandonner ou revendre est fait grâce à une méthode d'essai/erreur reposant sur le problème de supervision. Rien ne guide cette méthode dont la sélection des essais est aléatoire. Ainsi, il est possible qu'un agent choisisse d'abandonner une tâche dont ses coéquipiers dépendent alors même qu'il serait moins dommageable à l'équipe d'abandonner une autre tâche dont aucun de ses coéquipiers ne dépend.

Cependant, la mise en place d'un encodage explicite des dépendances des agents, comme proposé dans Sec. 9.4.2, peut guider le choix de la tâche à revendre ou abandonner. Grâce à cette connaissance, l'agent peut explorer en premier les tâches n'ayant pas d'impact sur le plan de ces coéquipiers, puis les tâches ayant l'impact le plus loin, etc.

À défaut de l'encodage explicite de ces dépendances, l'agent peut à minima se reposer sur les chaines de causalités déterminées à la décision.

9.4.4 *Divergence des connaissances*

Dans les Sec. 4.3.4 et Sec. 7.3.1, nous avons introduit le problème de divergence des connaissances. Dans notre approche, ce problème se traduit par une différence entre les Multi-Robot HTN des agents.

Nous avons abordé ce problème sous l'angle du protocole de décision en soulignant que nous utilisons une méthode simple et peu efficace pour le contourner, à savoir, lors de l'ouverture d'une enchère le commissaire-priseur impose son Multi-Robot HTN comme seule vérité. Cette méthode présentant de nombreux handicaps, p. ex. la perte totale d'objectifs connus uniquement par un enchérisseur, elle n'est qu'un moyen pour nous d'établir les fondements de notre approche en priorisant d'autres défis, aussi, nous n'avons pas souhaité axer nos simulations sur l'évaluation d'une telle méthode.

De plus, les problèmes causés par ces divergences ne sont concernent pas que la phase de décision, mais également celle de supervision. Par exemple, nous avons souligné dans Sec. 7.3.1 comment il est possible qu'un agent reçoive des informations d'exécution incohérentes sur un même élément.

Association d'une valeur à une information

Pour prendre en compte ces divergences, une première amélioration serait d'associer à une information une valeur qualifiant sa fiabilité puis de mettre en place des algorithmes spécifiques exploitant cette valeur. Il existe de nombreuses manières de définir cette valeur, par exemple, si c'est la "fraicheur" d'une information qui est priorisée, alors l'information la plus récente sera conservée. Sinon, cette valeur peut représenter la distance avec son origine, c'est l'information la moins distante de sa source qui sera priorisée, p. ex. le robot transmettant une information ne fait-il que la relayer, ou en est-il à l'origine ? Ou encore, si c'est la majorité qui fait foi, l'information la plus partagée sera utilisée comme référence.

Dans le cadre dynamique de notre mission, il peut être intéressant de définir la valeur d'une information comme une combinaison de plusieurs éléments, p. ex. un scalaire représentant la fraicheur d'une information et la distance avec son origine.

La mise en place d'une telle approche et la comparaison des valeurs possibles est un travail long, d'autant plus lorsqu'il est possible de définir des combinaisons de ces valeurs.

Une autre possibilité pour faire face à ces problèmes de divergences de connaissances serait de se reposer sur des algorithmes spécialisés dans la fusion de données. Une telle approche nécessite d'analyser consciencieusement le problème à résoudre afin d'identifier les caractéristiques discriminantes d'une connaissance, p. ex. degré de confiance dans l'information, et d'étudier en profondeur l'état de l'art pour déterminer les algorithmes pouvant s'appliquer aux structures de données considérées.

9.5 VERS UNE MEILLEURE ESTIMATION DE LA VIABILITÉ DE L'ARCHITECTURE

Apporter des garanties formelles à un système de systèmes est un problème difficile en raison des nombreux paramètres et interactions de leurs processus. Ce manque de garantie est un de leurs défauts les plus courants. Aussi, une approche empirique est généralement préférée pour évaluer leur fiabilité. Dans un tel cas, le système doit idéalement être testé en conditions réelles ou au sein d'une simulation s'en approchant au maximum sur un large panel de missions.

Dans notre cas, en raison de ressources limitées l'évaluation présentée dans ce manuscrit porte sur un ensemble restreint de missions et s'appuie sur une simulation éloignée des conditions du monde réel. Nos résultats définissent donc une preuve de concept plutôt qu'une référence de performance. Aussi, une perspective directe pour notre approche serait de l'évaluer à une échelle plus importante.

Modélisation des communications

Dans HaucTioN, la prise en compte de communications incertaines est critique. Ces problèmes de communications sont dus à l'environnement sous-marin où les ondes acoustiques subissent des retards et des pertes importantes pour une bande passante limitée. La modélisation de ces ondes acoustiques est un problème complexe, pourtant, une meilleure modélisation des communications est nécessaire pour confirmer les hypothèses que nous posons.

Sans devoir simuler la propagation des ondes acoustiques, une première amélioration de l'architecture de simulation serait de simuler les communications en prenant en compte la saturation de la bande passante. Dans notre implémentation actuelle, seules les pertes de communication sont simulées grâce à un modèle de Bernoulli, il est donc impossible de déterminer l'impact de plusieurs communications simultanées.

Cependant, la mise en place d'un modèle de Gilbert-Elliot permettrait une prise en compte de la bande passante [OKS19]. Il serait ainsi possible de déterminer plus précisément la viabilité du protocole de décision de HaucTioN.

Utilisation générique de l'approche

L'approche que nous développons dans ce manuscrit a pour objectif l'accomplissement d'une mission de chasse aux mines, mais sa réelle finalité est la coopération effective de systèmes multirobots sous contraintes de communication pour surmonter des problèmes de MRTA comportant des dépendances complexes. Notre approche se veut donc générique, en mettant à jour les modèles utilisés, tels que les décompositions hiérarchiques, il est possible de l'appliquer à d'autres contextes. Aussi, une autre manière d'éprouver la fiabilité de l'architecture serait de l'exploiter sur d'autres cas d'usage.

Par exemple, dans [Mil+21b] nous avons utilisé notre architecture sur un problème de logistique comportant des contraintes de précédence et de causalité.

La mise en place d'autres cas d'usage peut également se faire dans un but de comparaison à d'autres approches visant à résoudre des problèmes de MRTA équivalent. Si la communauté scientifique de la planification peut se reposer sur des benchmarks de référence pour comparer ses approches [1], il n'existe malheureusement pas de tels benchmarks pour les problèmes de MRTA. Aussi, dans cette optique de comparaison le choix de nouveaux cas d'usage doit être fait prudemment en fonction des expérimentations proposées dans certains travaux. Ce choix peut idéalement être guidé par la taxonomie des problèmes de MRTA.

Enfin, en restant dans le contexte de la chasse aux mines, il peut être intéressant d'étendre notre approche aux scénarios *Point d'étranglement* et *Débarquement* présentés aux Sec. 2.2.2 et Sec. 2.2.3. Chacun de ces scénarios implique un nouveau lot de contraintes opérationnelles, comme la discrétion acoustique pour le scénario de débarquement. Avec ces contraintes, d'autres aspects de l'architecture peuvent être explorés.

1. Benchmarks de la Compétition Internationale de Planification (IPC) https://ipc.hierarchical-task.net/

ANNEXES

A FICHIERS HDDL DE SPÉCIFICATION DE MISSION DES TÂCHES MULTIROBOTS

```
(define (domain dom_4E_2I_25Z_3M_T)
(:requirements :durative :typing :hierarchy)
(:types
  zone milco mr-token - object)
(:constants
  zone-1  zone-2 zone-3 zone-4 zone-5 - zone
  zone-6 zone-7 zone-8 zone-9 zone-10 - zone
  zone-11 zone-12 zone-13 zone-14 - zone
  zone-15 zone-16 zone-17 zone-18 - zone
  zone-19 zone-20 zone-21 zone-22 - zone
  zone-23 zone-24 zone-25 - zone
  ;<<MINE>>
  ;<<MR_TOKEN>>)
(:predicates
  (mr-sync-flag ?t - mr-token))
(:task cover :parameters (?z - zone))
(:task identify-and-report :parameters (?m -
    milco))
(:task report :parameters (?m - milco))
(:task report-sync :parameters (?m - milco ?t
    - token))
; == COVER DECOMPOSITIONS ==
(:durative-method m-cover-zone-1
  :parameters ()
  :task (cover zone-1)
  :subtasks (and
    (cover zone-2)
    (cover zone-3)))
(:durative-method m-cover-zone-2-1
  :parameters ()
  :task (cover zone-2)
  :subtasks (and
    (t1 (cover zone-4))
    (t2 (cover zone-5))
    (t3 (cover zone-6)))
  :constraint (and
    (< t2.end (+ t3.end 150))))
(:durative-method m-cover-zone-2-2
  :parameters ()
  :task (cover zone-2)
  :subtasks (and
    (t1 (cover zone-7))
    (t2 (cover-primitive zone-8)))
  :constraint (and
    (< t1.end (+ t2.end 150))))
(:durative-method m-cover-zone-3-1
  :parameters ()
  :task (cover zone-3)
  :subtasks (and
    (t1 (cover-primitive zone-9))
    (t2 (cover-primitive zone-10)))
  :constraint (and
    (< t1.end t2.start)))
(:durative-method m-cover-zone-3-2
  :parameters ()
  :task (cover zone-3)
  :subtasks (and
    (t1 (cover-primitive zone-11))
    (t2 (cover-primitive zone-12))
    (t3 (cover-primitive zone-13)))
  :constraint (and
    (< t1.end (+ t2.end 150))))
(:durative-method m-cover-zone-4
  :parameters ()
  :task (cover zone-4)
  :subtasks (and
    (cover-primitive zone-14)
    (cover-primitive zone-15)))
(:durative-method m-cover-zone-5-1
  :parameters ()
  :task (cover zone-5)
  :subtasks (and
    (t1 (cover-primitive zone-16))
    (t2 (cover-primitive zone-17)))
  :constraint (and
    (< t1.end (+ t2.end 75))))
(:durative-method m-cover-zone-5-2
  :parameters ()
  :task (cover zone-5)
  :subtasks (and
    (t1 (cover-primitive zone-18))
    (t2 (cover-primitive zone-19))
    (t3 (cover-primitive zone-20)))
  :constraint (and
    (< t1.end t2.start)
    (< t2.end t3.start)))
(:durative-method m-cover-zone-6
  :parameters ()
  :task (cover zone-6)
  :subtasks (and
    (cover-primitive zone-21)
    (cover-primitive zone-22)))
(:durative-method m-cover-zone-7
  :parameters ()
  :task (cover zone-7)
  :subtasks (and
    (t1 (cover-primitive zone-23))
    (t2 (cover-primitive zone-24))
    (t3 (cover-primitive zone-25)))
(:durative-action cover-primitive
  :parameters (?z - zone)
  :precondition ()
  :effect ()
  :constraint (and
    (< end (+ start 5000))))
; == IDENTIFY & REPORT DECOMPOSITIONS ==
(:durative-method m-identify-then-report-sync
  :parameters (?m - milco ?t - token)
```

```
  :task (identify-and-report ?m)
  :subtasks (and
    (t1 (report-sync ?m ?t))
    (t2 (identify-primitive ?m)))
  :constraint (and
    (< t2.end t1.start)))
; == REPORT DECOMPOSITIONS ==
(:durative-method m-report-with-relay
  :parameters (?m - milco ?t - mr-token)
  :task (report-sync ?m ?t)
  :subtasks (and
    (t2 (relay-sync-primitive ?m ?t))
    (t1 (report-sync-primitive ?m ?t)))
  :constraint (and
    (> t1.start t2.start)
    (< t1.start (+ t2.start 10))
    (< t1.end t2.end)
    (< t2.end (+ t1.end 10))))
(:durative-method
    m-report-with-relay-unable-to-do
  :parameters (?m - milco ?t - mr-token)
  :task (report-sync ?m ?t)
  :subtasks (and
    (t1 (mr-unable-to-do))))
(:durative-action report-sync-primitive
  :parameters (?m - milco ?t - mr-token)
  :precondition (and
    (at start (mr-sync-flag ?t)))
  :effect (and
    (at end (not (mr-sync-flag ?t))))
  :constraint (and
    (< end (+ start 600))))
(:durative-action relay-sync-primitive
  :parameters (?m - milco ?t - mr-token)
  :precondition (and
    (at start (mr-sync-flag ?t))
    (at end (not (mr-sync-flag ?t))))
  :effect ())
; == IDENTIFY DECOMPOSITIONS ==
(:durative-action identify-primitive
  :parameters (?m - milco)
  :precondition ()
  :effect ()
  :constraint (and
    (< end (+ start 1000))))
; == MR-UNABLE-TO-DO DECOMPOSITIONS ==
(:action mr-unable-to-do
  :parameters ()
  :precondition ()
  :effect ()))
```

Listing A.1 – Domaine HDDL multirobot de la mission d'illustration.

```
(define (problem pb_4E_2I_25Z_3M_T)
(:domain dom_4E_2I_25Z_3M_T)
(:objects)
(:htn
  :subtasks (and
    (cover zone-1))
  :constraints ())
(:init))
```

Listing A.2 – Problème HDDL multirobot initial de la mission d'illustration.

```
(define (problem milco_alert_4E_2I_25Z_3M_T)
(:domain dom_4E_2I_25Z_3M_T)
(:objects)
(:htn
  :subtasks (and
    (identify-and-report <<MINE>>))
  :constraints ())
(:init))
```

Listing A.3 – Template du problème HDDL multirobot d'identification d'un MILCO.

B FICHIERS HDDL DE SPÉCIFICATION DE MISSION DES TÂCHES LOCALES

```
(define (domain MinehuntingLocal)
(:requirements :durative :typing :hierarchy)
(:types
  zone milco robot token mr-token - object
  auv-explorer - robot)
(:constants
  <<FL-TOKEN>> - token)
(:predicates
  (free-token ?t - token))
(:task cover-primitive :parameters (?z - zone)
    )
(:task identify-primitive :parameters (?m -
    milco))
(:task report-sync-primitive :parameters (?m -
     milco ?t - mr-token))
(:task relay-sync-primitive :parameters (?m -
    milco ?t - mr-token))
(:task mr-unable-to-do :parameters ())
(:method m-cover-primitive
  :parameters (?z - zone ?r - auv-explorer)
  :task (cover-primitive ?z)
  :ordered-subtasks (and
    (fl-cover-primitive ?r ?z)))
(:durative-action fl-cover-primitive
  :parameters (?r - auv-explorer ?z - zone)
  :precondition (and
    (at start (free-token <<FL-TOKEN>>)))
  :effect (and
    (at start (not (free-token <<FL-TOKEN>>)))
    (after (free-token <<FL-TOKEN>>)))))
(:method m-relay-sync-primitive
  :parameters (?m - milco ?t - mr-token ?r -
      auv-explorer)
  :task (relay-sync-primitive ?m ?t)
  :ordered-subtasks (and
    (fl-relay-primitive ?r ?m)))
(:durative-action fl-relay-primitive
  :parameters (?r - auv-explorer ?m - milco)
  :precondition ()
  :effect ())
; Unable
(:method m-mr-unable-to-do
  :parameters ()
  :task (mr-unable-to-do)
  :ordered-subtasks (and
    (unable-to-do)))
(:method m-identify-primitive
  :parameters (?m - milco)
  :task (identify-primitive ?m)
  :ordered-subtasks (and
    (unable-to-do)))
(:method m-report-sync-primitive
  :parameters (?m - milco ?t - mr-token ?r -
      auv-explorer)
  :task (report-sync-primitive ?m ?t)
  :ordered-subtasks (and
    (unable-to-do)))
(:action unable-to-do
  :parameters ()
  :precondition ()
  :effect ()))
```

Listing B.4 – Template de domaine HDDL local d'un *Explorer*.

```
(define (domain MinehuntingLocal)
(:requirements :typing :hierarchy :durative)
(:types
  zone milco robot token mr-token - object
  auv-identifier - robot)
(:constants
  <<FL-TOKEN>> - token)
(:predicates
  (free-token ?t - token))
(:task cover-primitive :parameters (?z - zone)
    )
(:task identify-primitive :parameters (?m -
    milco))
(:task report-sync-primitive :parameters (?m -
     milco ?t - mr-token))
(:task relay-sync-primitive :parameters (?m -
    milco ?t - mr-token))
(:task mr-unable-to-do :parameters ())
(:method m-identify-primitive
  :parameters (?m - milco ?r - auv-identifier)
  :task (identify-primitive ?m)
  :ordered-subtasks (and
    (fl-identify-primitive ?r ?m)))
(:durative-action fl-identify-primitive
  :parameters (?r - auv-identifier ?m - milco)
  :precondition (and
    (at start (free-token <<FL-TOKEN>>)))
  :effect (and
    (at start (not (free-token <<FL-TOKEN>>)))
    (after (free-token <<FL-TOKEN>>)))))
(:method m-report-sync-primitive
  :parameters (?m - milco ?t - mr-token ?r -
      auv-identifier)
  :task (report-sync-primitive ?m ?t)
  :ordered-subtasks (and
    (fl-report-primitive ?r ?m)))
(:durative-action fl-report-primitive
  :parameters (?r - auv-identifier ?m - milco)
  :precondition ()
  :effect ())
(:method m-relay-sync-primitive
  :parameters (?m - milco ?t - mr-token ?r -
      auv-identifier)
  :task (relay-sync-primitive ?m ?t)
  :ordered-subtasks (and
    (fl-relay-primitive ?r ?m)))
(:durative-action fl-relay-primitive
  :parameters (?r - auv-identifier ?m - milco)
  :precondition ()
  :effect ())
; Unable
(:method m-mr-unable-to-do
  :parameters ()
  :task (mr-unable-to-do)
```

```
  :ordered-subtasks (and
    (unable-to-do)))
(:method m-cover-primitive
  :parameters (?z - zone)
  :task (cover-primitive ?z)
  :ordered-subtasks (and
    (unable-to-do)))
(:action unable-to-do
  :parameters ()
  :precondition ()
  :effect ()))
```

Listing B.5 – Template de domaine HDDL local d'un *Identifier*.

C CHAINES DE CAUSALITÉ

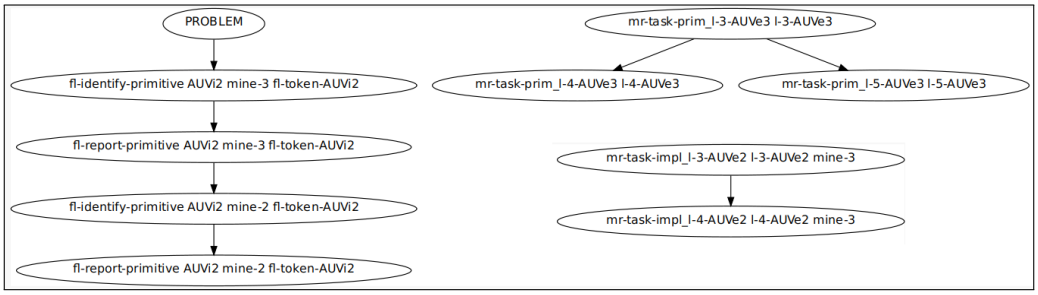


FIGURE C.2 – Exemple de chaines de causalités.

D PLAN SOLUTION D'UN PROBLÈME D'ESTIMATION

```
**** Causal links ****
Condition on (free-token fl-token-AUVe1) of action (fl-cover-primitive AUVe1 zone-23 fl-token-
    AUVe1)
  Support by effect of action PROBLEM
Condition on (free-token fl-token-AUVe1) of action (fl-cover-primitive AUVe1 zone-8 fl-token-
    AUVe1)
  Support by effect of action (fl-cover-primitive AUVe1 zone-23 fl-token-AUVe1)
Condition on (mr-sync-flag mr-token) of action (mr-task-prim_l-15-AUVe1 l-15-AUVe1)
  Support by effect of action (mr-task-prim_l-10-AUVe1 l-10-AUVe1)

**** Plan ****
template is — start : (action) [duration] —
100: (fl-cover-primitive AUVe1 zone-23 fl-token-AUVe1) [2]
100: (mr-task-impl_l-14-AUVe1 l-14-AUVe1) [2]
100: (mr-task-prim_l-9-AUVe1 l-9-AUVe1) [1]
100: (mr-task-prim_l-10-AUVe1 l-10-AUVe1) [4]
102: (fl-cover-primitive_u-loc-91 AUVe1 zone-8 fl-token-AUVe1) [1]
102: (mr-task-impl_l-5-AUVe1 l-5-AUVe1) [1]
104: (mr-task-prim_l-15-AUVe1 l-15-AUVe1) [2]
105: (mr-task-prim_l-16-AUVe1 l-16-AUVe1) [2]

**** Cost ****
228
```

Listing D.6 – Extrait de la sortie du solveur LCP pour un problème d'estimation.

E PLAN SOLUTION D'UN PROBLÈME DE DÉTERMINATION DES GAGNANTS

```
**** Causal links ****
Condition on (free-bidder AUVe1) of action (allocate_l-9-AUVe1_AUVe1 l-3-AUVe1 l-9-AUVe1 l-5-
    AUVe1 AUVe1)
  Support by effect of action PROBLEM
Condition on (free-handle-task l-9-AUVe1) of action (allocate_l-9-AUVe1_AUVe1 l-3-AUVe1 l-9-
    AUVe1 l-5-AUVe1 AUVe1)
  Support by effect of action PROBLEM
Condition on (handle-task l-5-AUVe1) of action (mr-task-prim_l-5-AUVe1 l-5-AUVe1)
  Support by effect of action (allocate_l-9-AUVe1_AUVe1 l-3-AUVe1 l-9-AUVe1 l-5-AUVe1 AUVe1)
Condition on (start-link-flag l-11-AUVe1) of action (check-bidder-plan_AUVe2_l-11-AUVe1 l-11-
    AUVe1)
  Support by effect of action (mr-task-prim_l-11-AUVe1 l-11-AUVe1)
  [...]

**** Plan ****
template is — start : (action) [duration] —
 0: (allocate_l-8-AUVe1_AUVe2 l-11-AUVe1 l-4-AUVe1 l-10-AUVe1 AUVe2) [0]
 0: (allocate_l-9-AUVe1_AUVe1 l-3-AUVe1 l-9-AUVe1 l-5-AUVe1 AUVe1) [0]
 0: (no-resell) [0]
 0: (no-resell) [0]
 0: (no-resell) [0]
0: (check-bidder-plan_AUVe1_l-5-AUVe1 l-5-AUVe1) [3]
0: (check-bidder-plan_AUVe2_l-10-AUVe1 l-10-AUVe1) [1]
0: (mr-task-prim_l-10-AUVe1 l-10-AUVe1) [1]
0: (mr-task-prim_l-5-AUVe1 l-5-AUVe1) [3]
3: (check-bidder-plan_AUVe1_l-3-AUVe1 l-3-AUVe1) [2]
3: (check-bidder-plan_AUVe2_l-4-AUVe1 l-4-AUVe1) [1]
3: (mr-task-prim_l-3-AUVe1 l-3-AUVe1) [2]
3: (mr-task-prim_l-4-AUVe1 l-4-AUVe1) [1]
5: (check-bidder-plan_AUVe1_l-9-AUVe1 l-9-AUVe1) [1]
5: (check-bidder-lan_AUVe2_l-11-AUVe1 l-11-AUVe1) [1]
5: (mr-task-prim_l-11-AUVe1 l-11-AUVe1) [1]
5: (mr-task-prim_l-9-AUVe1 l-9-AUVe1) [1]

**** Cost ****
342
```

Listing E.7 – Extrait de la sortie du solveur LCP pour un problème de détermination des gagnants.

F EXEMPLES DE STRUCTURES HIÉRARCHIQUES UTILISÉES DANS LA MISSION D'ILLUSTRATION

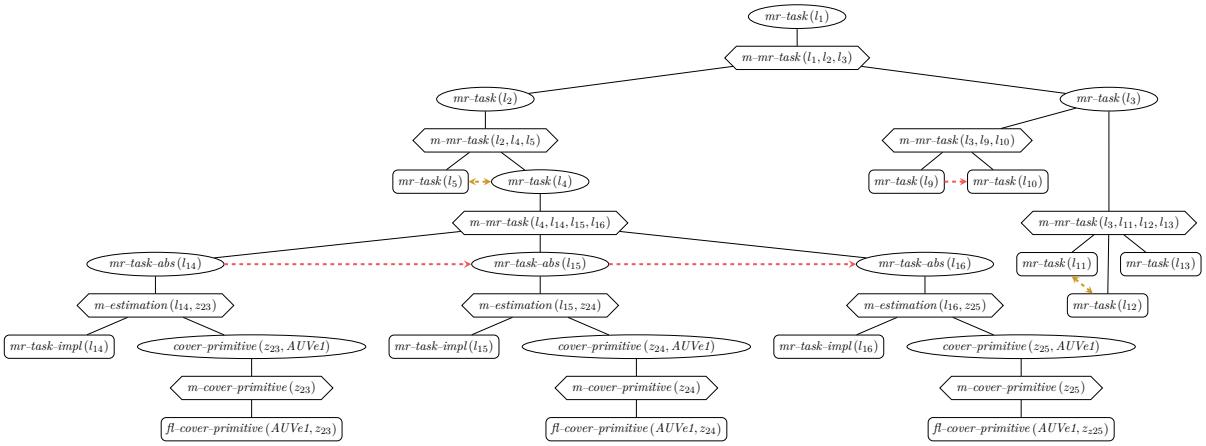


FIGURE F.3 – HTN d'un problème de supervision de *AUVe1* pour la mission d'illustration.

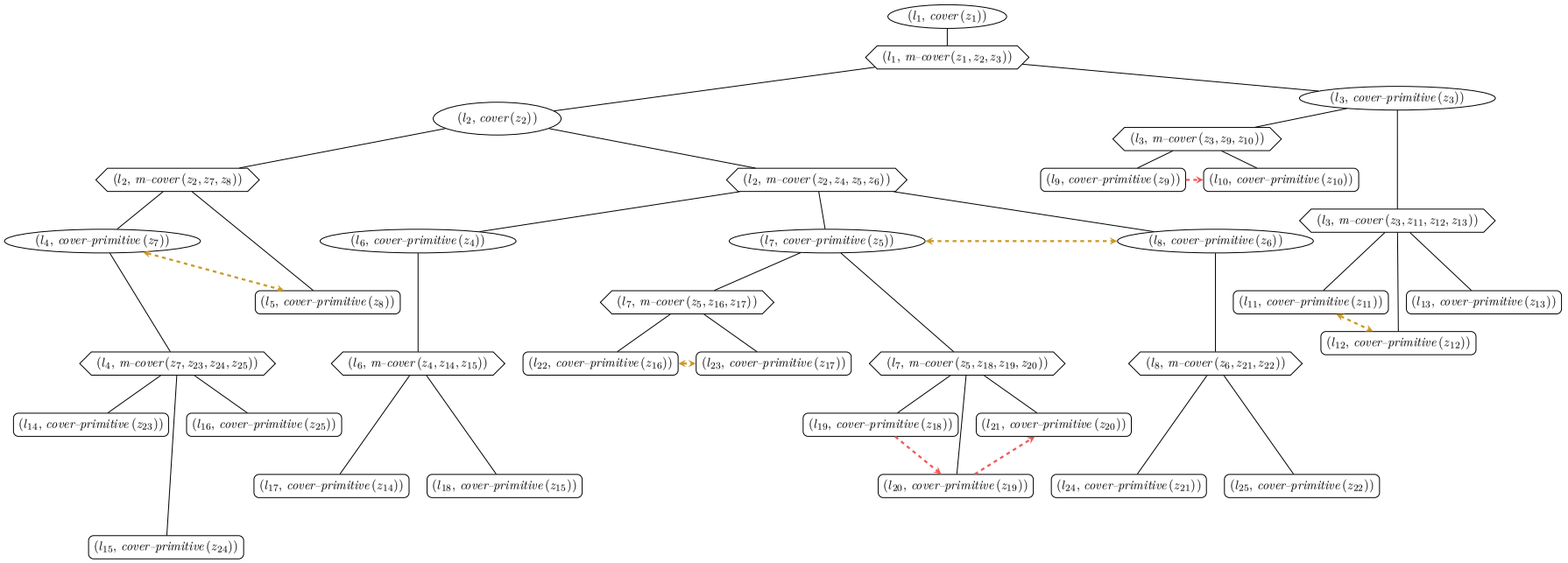


FIGURE F.4 – Multi-Robot Graph des objectifs initiaux de la mission d'illustration. Les flèches - -> et <- -> représentent respectivement les contraintes de précédence et les contraintes de synchronicité entre tâches.

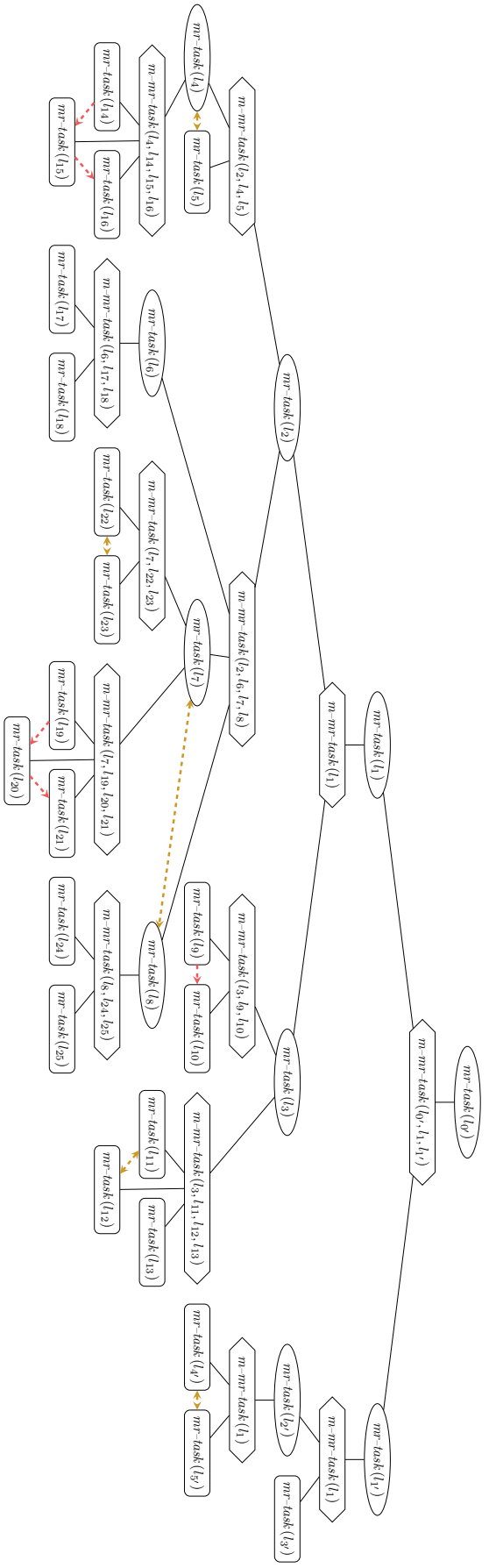


FIGURE F.5 – Multi-Robot HTN de la seconde enchère de la mission d'illustration.

BIBLIOGRAPHIE

[10] "Sudan mine blast man Stephen Allan 'was hero'". In : *BBC* (2010).

[14a] "Robots replace costly US Navy mine-clearance dolphins". In : *BBC* (2014).

[14b] "Unmanned Mine Countermeasures Update". In : *Think Defence* (2014).

[18] "The future of Royal Navy mine hunting". In : *Navy Lookout* (2018).

[19a] "Iran tanker seizure : What is the Strait of Hormuz ?" In : *BBC* (2019).

[19b] "Next-generation MCM systems for the Belgian and Netherlands Navies". In : *Mer et Marine* (2019).

[20] "Une opération de contre-minage en rade de Brest le 15 septembre". In : *Brest Métropole & ville* (2020).

[21a] "Disparition en mer d'un plongeur démineur du GPD Atlantique". In : *Mer et Marine* (2021).

[21b] "Naval Group Launches MIRICLE Next Gen Mine Warfare Project". In : *NavalNews* (2021).

[21c] "Royal Navy Takes Delivery of First Anglo-French Minehunting Drone Boat". In : *Maritime Executive* (2021).

[21d] "The First Ships of Operation Neptune". In : *The National WWII Museum - New Orleans* (2021).

[22] "Des experts préviennent que les mines de la mer Noire constituent une grave menace maritime européenne". In : *The Press Free* (2022).

[ABB17] Francesco Amigoni, Jacopo Banfi et Nicola Basilico. "Multirobot Exploration of Communication-Restricted Environments : A Survey". In : *IEEE Intelligent Systems* 32.6 (2017), p. 48-57. doi : 10.1109/MIS.2017.4531226.

[Abr+07] Jawad Abrache et al. "Combinatorial auctions". In : *Annals of Operations Research* 153.1 (2007), p. 131-164. doi : 10.1007/s10479-007-0179-z.

[Ala+20] Tauhidul Alam et al. "Stochastic Multi-Robot Patrolling with Limited Visibility". In : *Journal of Intelligent & Robotic Systems* 97 (2020). doi : 10.1007/s10846-019-01039-5.

[All20] George Allison. "Uncrewed minehunters to replace conventional vessels". In : *UK Defence Journal* (2020).

[All83] James F. Allen. "Maintaining knowledge about temporal intervals". In : *Communications of the ACM* 26 (1983), p. 832-843.

[APM05] Ian Akyildiz, Dario Pompili et Tommaso Melodia. "Underwater Acoustic Sensor Networks : Research Challenges". In : *Ad Hoc Networks* 3 (2005), p. 257-279. doi : 10.1016/j.adhoc.2005.01.004.

[BA99] S.C. Botelho et R. Alami. "M+ : a scheme for multi-robot cooperation through negotiated task allocation and achievement". In : *IEEE International Conference on Robotics and Automation (ICRA)*. Detroit, MI, USA, 1999. doi : 10.1109/ROBOT.1999.772530.

[BAH19] Pascal Bercher, Ron Alford et D. Höller. “A Survey on Hierarchical Planning - One Abstract Idea, Many Concrete Realizations”. In : *International Joint Conference on Artificial Intelligence (IJCAI)*. 2019. doi : 10.24963/ijcai.2019/875.

[Ban+18] Jacopo Banfi et al. “Strategies for coordinated multirobot exploration with recurrent connectivity constraints”. In : *Autonomous Robots* 42 (2018). doi : 10.1007/s10514-017-9652-y.

[Ban+22] Jacopo Banfi et al. “Hierarchical Planning for Heterogeneous Multi-Robot Routing Problems via Learned Subteam Performance”. In : *IEEE Robotics and Automation Letters* 7.2 (2022), p. 4464-4471. doi : 10.1109/LRA.2022.3148489.

[Bar64] Paul Baran. “On Distributed Communications”. In : *Memorandum RM* (1964).

[BBC18] Jacopo Banfi, Nicola Basilico et Stefano Carpin. “Optimal Redeployment of Multirobot Teams for Communication Maintenance”. In : *International Conference on Intelligent Robots and Systems (IROS)*. 2018. doi : 10.1109/IROS.2018.8593532.

[Bec+14] Patrick Bechon et al. “HiPOP : Hierarchical Partial-Order Planning”. In : *STAIRS*. 2014. doi : 10.3233/978-1-61499-421-3-51.

[Bec+18] Patrick Bechon et al. “Integrating Planning and Execution for a Team of Heterogeneous Robots with Time and Communication Constraints”. In : *IEEE International Conference on Robotics and Automation (ICRA)*. 2018. doi : 10.1109/ICRA.2018.8461024.

[Bel11] Assia Belbachir. “Une architecture coopérative pour la localisation de cibles marines avec des véhicules sous-marins”. Thèse de doct. 2011.

[Ber+17] Pascal Bercher et al. “An Admissible HTN Planning Heuristic”. In : *IJCAI*. 2017. doi : 10.24963/ijcai.2017/68.

[Bes+18] Graeme Best et al. “Dec-MCTS : Decentralized planning for multi-robot active perception”. In : *The International Journal of Robotics Research* (2018). doi : 10.1177/0278364918755924.

[BG06] Alexander Babanov et Maria Gini. “Deciding Task Schedules for Temporal Planning via Auctions”. In : *AAAI Conference on Artificial Intelligence (AAAI)*. 2006.

[BHK13] Mohamed Badreldin, Ahmed Hussein et Alaa Khamis. “A Comparative Study between Optimization and Market-Based Approaches to Multi-Robot Task Allocation”. In : *Advances in Artificial Intelligence* 2013 (2013), p. 1-11. doi : 10.1155/2013/256524.

[BIL12] Assia Belbachir, François Ingrand et Simon Lacroix. “A cooperative architecture for target localization using multiple AUVs”. In : *Intelligent Service Robotics* 5 (2012), p. 119-132. doi : 10.1007/s11370-012-0107-1.

[Bit+20] Arthur Bit-Monnot et al. *FAPE : a Constraint-based Planner for Generative and Hierarchical Temporal Planning*. Rapp. tech. arXiv, 2020.

[Bit18] Arthur Bit-Monnot. “A Constraint-based Encoding for Domain-Independent Temporal Planning”. In : *Lecture Notes in Computer Science* (2018), p. 30-46. doi : 10.1007/978-3-319-98334-9_3.

[Cal+90] P. Caloud et al. “Indoor automation with many mobile robots”. In : *International Workshop on Intelligent Robots and Systems*. 1990. doi : 10.1109/IROS.1990.262370.

[Cam+17] Filippo Campagnaro et al. “Multimodal Underwater Networks : Recent Advances and a Look Ahead”. In : *International Conference on Underwater Networks & Systems*. Nov. 2017, p. 1-8. doi : 10.1145/3148675.3152759.

[Cao+10] Hung Cao et al. “Complex Tasks Allocation for Multi Robot Teams under Communication Constraints”. In : 2010.

[Car13] Nicolas Carlési. “Coopération entre véhicules sous-marins autonomes : une approche organisationnelle réactive multi-agent”. Thèse de doct. 2013.

[CLI12] Hung Cao, Simon Lacroix et Félix Ingrand. “Planification d’une mission d’observation par allocation de tâches hiérarchiques pour une équipe de robots hétérogènes”. In : *Reconnaissance des Formes et Intelligence Artificielle (RFIA)*. 2012.

[CLP15] Guillaume Casanova, Charles Lesire et Cédric Pralet. “Managing Dynamic Multi-Agent Simple Temporal Network”. In : *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2015. isbn : 9781450334136.

[Cor03] Greg Cornish. “U.S. Naval Mine Warfare Strategy : Analysis of the Way Ahead”. In : *DTIC Technical Report*. 2003.

[Cus91] John H. Cushman. “WAR IN THE GULF : Sea Mines ; Allied Ships Hunt Gulf for Iraqi Mines”. In : *The New York Times* (1991).

[Dai+20] Wei Dai et al. “Multi-Robot Dynamic Task Allocation for Exploration and Destruction”. In : *Journal of Intelligent & Robotic Systems* 98 (2020). doi : 10.1007/s10846-019-01081-3.

[Dia+06] M.B. Dias et al. “Market-Based Multirobot Coordination : A Survey and Analysis”. In : *Proceedings of the IEEE* 94.7 (2006), p. 1257-1270. doi : 10.1109/JPROC.2006.876939.

[DJM02] Gregory Dudek, Michael Jenkin et Evangelos Milios. “A taxonomy of multirobot systems”. In : *Robot teams : From diversity to polymorphism* (2002), p. 3-22. doi : 10.1007/BF00240651.

[DS03] M. Bernardine Dias et Anthony Stentz. “TraderBots : A Market-Based Approach for Resource, Role, and Task Allocation in Multirobot Coordination”. In : *Technical Report, Robotics Institute, Carnegie Mellon University* (2003), p. 20. doi : 10.1109/JPROC.2006.876939.

[EHN94] Kutluhan Erol, James Hendler et Dana S Nau. “HTN planning : Complexity and expressivity”. In : *AAAI Conference on Artificial Intelligence (AAAI)*. T. 94. Seattle, WA, USA, 1994, p. 1123-1128.

[Ett03] Paul Etter. *Underwater Acoustic Modeling and Simulation*. 2003. doi : 10.1201/9781315166346.

[Eva16] Gareth Evans. “Mine-hunters : are they still needed ?” In : *Naval Technology* (2016).

[Exe09] Executive Office Littoral and Mine Warfare. *21st Century U.S. Navy Mine Warfare*. Rapp. tech. US Navy, 2009.

[Fer+17a] Gabriele Ferri et al. “A market-based task allocation framework for autonomous underwater surveillance networks”. In : *OCEANS*. Aberdeen, United Kingdom, 2017, p. 1-10. doi : 10.1109/OCEANSE.2017.8084769.

[Fer+17b] Gabriele Ferri et al. “Cooperative robotic networks for underwater surveillance : an overview”. In : *IET Radar, Sonar & Navigation* 11.12 (2017), p. 1740-1761. doi : 10.1049/iet-rsn.2017.0074.

[Fer+18] Gabriele Ferri et al. “Autonomous Underwater Surveillance Networks : A Task Allocation Framework to Manage Cooperation”. In : *OCEANS - IEEE*. Kobe, 2018, p. 1-10. doi : 10.1109/OCEANSKOBE.2018.8558813.

[Fer+19] Gabriele Ferri et al. “Cooperative Autonomy in the CMRE ASW Multistatic Robotic Network : Results From LCAS18 Trial”. In : *OCEANS 2019 - Marseille*. 2019. DOI : 10.1109/OCEANSE.2019.8867431.

[Fer+20] Gabriele Ferri et al. “Cooperative Autonomy and Data Fusion for Underwater Surveillance With Networked AUVs”. In : *IEEE International Conference on Robotics and Automation (ICRA)*. 2020. DOI : 10.1109/ICRA40945.2020.9197367.

[Fer95] Jacques Ferber. *Les Systèmes multi-agents : vers une intelligence collective*. 1995.

[FL03] M. Fox et D. Long. “PDDL2.1 : An Extension to PDDL for Expressing Temporal Planning Domains”. In : *Journal of Artificial Intelligence Research* 20 (2003), p. 61-124.

[GA15] Ilche Georgievski et Marco Aiello. “HTN planning : Overview, comparison, and beyond”. In : *Artificial Intelligence* 222 (2015), p. 124-156. DOI : 10.1016/j.artint.2015.02.002.

[Gan+05] J. Gancet et al. “Task planning and control for a multi-UAV system : architecture and algorithms”. In : *International Conference on Intelligent Robots and Systems*. 2005. DOI : 10.1109/IROS.2005.1545217.

[Gat+97] Erann Gat et al. “On Three-Layer Architectures”. In : *Artificial Intelligence and Mobile Robots*. 1997, p. 195-210.

[GB22] Roland Godet et Arthur Bit-Monnot. “Chronicles for Representing Hierarchical Planning Problems with Time”. In : *ICAPS Hierarchical Planning Workshop (HPlan)*. 2022.

[Gin17] Maria Gini. “Multi-Robot Allocation of Tasks with Temporal and Ordering Constraints”. In : *AAAI Conference on Artificial Intelligence* 31 (2017). DOI : 10.1609/aaai.v31i1.11145.

[GLB13] Thibault Gateau, Charles Lesire et Magali Barbier. “HiDDeN : Cooperative plan execution and repair for heterogeneous robots in dynamic environments”. In : *International Conference on Intelligent Robots and Systems (IROS)*. 2013. DOI : 10.1109/IROS.2013.6697047.

[GM01] Brian P. Gerkey et Maja J. Matarić. “Principled Communication for Dynamic Multi-Robot Task Allocation”. In : *Experimental Robotics VII*. Springer, 2001, p. 353-362. ISBN : 978-3-540-45118-1. DOI : 10.1007/3-540-45118-8_36.

[GM03a] Brian Gerkey et Maja Mataric. “A formal framework for the study of task allocation in multi-robot systems”. In : *International Journal of Robotic Research (IJRR)* (2003).

[GM03b] Brian P. Gerkey et Maja J. Matarić. “Multi-robot task allocation : analyzing the complexity and optimality of key architectures”. In : *IEEE International Conference on Robotics and Automation (ICRA)*. T. 3. Taipei, Taiwan, 2003, p. 3862-3868. DOI : 10.1109/ROBOT.2003.1242189.

[GM04a] Brian Gerkey et Maja Matari. “Are (explicit) multi-robot coordination and multi-agent coordination really so different”. In : *AAAI Symposium on bridging the multi-agent and multi-robotic research gap* (avr. 2004).

[GM04b] Brian P. Gerkey et Maja J. Matarić. “A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems”. In : *The International Journal of Robotics Research* 23.9 (2004), p. 939-954. DOI : 10.1177/0278364904045564.

[HBH09] Han-Lim Choi, L. Brunet et J.P. How. “Consensus-Based Decentralized Auctions for Robust Task Allocation”. In : *IEEE Transactions on Robotics* 25.4 (2009), p. 912-926. DOI : 10.1109/TRO.2009.2022423.

[Hol+11] Geoffrey A. Hollinger et al. “Communication protocols for underwater data collection using a robotic sensor network”. In : *IEEE GLOBECOM Workshops*. Houston, TX, USA, 2011, p. 1308-1313. doi : 10.1109/GLOCOMW.2011.6162397.

[Höl+20] Daniel Höller et al. “HDDL : An Extension to PDDL for Expressing Hierarchical Planning Problems”. In : *AAAI Conference on Artificial Intelligence*. 2020. doi : 10.1609/aaai.v34i06.6542.

[Höl+21] Daniel Höller et al. “The PANDA Framework for Hierarchical Planning”. In : *Künstliche Intelligenz* (2021).

[Jev+12] Aleksandar Jevtic et al. “Distributed Bees Algorithm for Task Allocation in Swarm of Robots”. In : *IEEE Systems Journal* (2012). doi : 10.1109/CASE.2011.6042503.

[Jon+06] E.G. Jones et al. “Dynamically formed heterogeneous robot teams performing tightly-coordinated tasks”. In : IEEE International Conference on Robotics and Automation (ICRA). Orlando, FL, USA, 2006. doi : 10.1109/ROBOT.2006.1641771.

[Jon+20] Simon Jones et al. “Distributed Situational Awareness in Robot Swarms”. In : *Advanced Intelligent Systems* 2 (2020). doi : 10.1002/aisy.202000110.

[Kal+05] Nidhi Kalra et al. *Market-Based Multirobot Coordination : A Comprehensive Survey and Analysis*. 7. Defense Technical Information Center, 2005, p. 1257-1270. doi : 10.1109/JPROC.2006.876939.

[KEK11] Alaa M. Khamis, Ahmed M. Elmogy et Fakhri O. Karray. “Complex Task Allocation in Mobile Surveillance Systems”. In : *International Journal on Intelligent and Robotic Systems* 64.1 (2011). doi : 10.1007/s10846-010-9536-2.

[KFS05] Nidhi Kalra, Dave Ferguson et Anthony Stentz. “Hoplites : A Market-Based Framework for Planned Tight Coordination in Multirobot Teams”. In : *IEEE International Conference on Robotics and Automation (ICRA)*. Barcelona, Spain, 2005, p. 1170-1177. doi : 10.1109/ROBOT.2005.1570274.

[KHE15] Alaa Khamis, Ahmed Hussein et Ahmed Elmogy. “Multi-robot Task Allocation : A Review of the State-of-the-Art”. In : *Cooperative Robots and Sensor Networks 2015*. T. 604. 2015, p. 31-51. doi : 10.1007/978-3-319-18299-5_2.

[KKT10] Sven Koenig, Pinar Keskinocak et Craig Tovey. “Progress on Agent Coordination with Cooperative Auctions”. In : *National Conference on Artificial Intelligence*. T. 3. Jan. 2010.

[KSD13] G. Ayorkor Korsah, Anthony Stentz et M. Bernardine Dias. “A comprehensive taxonomy for multi-robot task allocation”. In : *The International Journal of Robotics Research* 32.12 (2013), p. 1495-1512. doi : 10.1177/0278364913496484.

[Kun+18] Lars Kunze et al. “Artificial Intelligence for Long-Term Robot Autonomy : A Survey”. In : *IEEE Robotics and Automation Letters* 3.4 (2018), p. 4023-4030. doi : 10.1109/LRA.2018.2860628.

[Lag+05] Michail G. Lagoudakis et al. “Auction-Based Multi-Robot Routing”. In : *Robotics : Science and Systems*. 2005. doi : 10.15607/RSS.2005.I.045.

[LAL04] T. Lemaire, R. Alami et S. Lacroix. “A distributed tasks allocation scheme in multi-UAV context”. In : *IEEE International Conference on Robotics and Automation (ICRA)*. T. 4. 2004, p. 3622-3627. doi : 10.1109/ROBOT.2004.1308816.

[Les+16] Charles Lesire et al. “A distributed architecture for supervision of autonomous multi-robot missions”. In : *Autonomous Robots* 40 (2016). doi : 10.1007/s10514-016-9603-z.

[Liu+13] Yabo Liu et al. “Multi-Robot Coordination in Complex Environment with Task and Communication Constraints”. In : *International Journal of Advanced Robotic Systems* 10 (2013). doi : 10.5772/54379.

[LM09] Thomas Lochmatter et A. Martinoli. “Understanding the Potential Impact of Multiple Robots in Odor Source Localization”. In : *Distributed Autonomous Robotic Systems 8* (2009). doi : 10.1007/978-3-642-00644-9_21.

[Loc10] Thomas Lochmatter. “Bio-inspired and probabilistic algorithms for distributed odor source localization using mobile robots”. Thèse de doct. 2010. doi : 10.5075/epfl-thesis-4628.

[Mar18] Greta E. Marlatt. “Sea Mines and Countermeasures : A Bibliography”. In : *Dudley Knox Library - Naval Postgraduate School*. 2018.

[Mil+21a] Antoine Milot et al. “Market-based Multi-robot coordination with HTN planning”. In : *International Conference on Intelligent Robots and Systems (IROS))*. 2021. doi : 10.1109/IROS51168.2021.9636286.

[Mil+21b] Antoine Milot et al. “Solving Hierarchical Auctions with HTN Planning”. In : *4th ICAPS workshop on Hierarchical Planning (HPlan)*. 2021. url : https://hal.archives-ouvertes.fr/hal-03289854.

[Mil+22a] Antoine Milot et al. “Allocation par enchères et planification hiérarchique pour un système multirobot, application au cas de la chasse aux mines”. In : *JFSMA 2022 : 30èmes Journées Francophones sur les Systèmes Multi-Agents*. 2022. url : https://hal.archives-ouvertes.fr/hal-03694348.

[Mil+22b] Antoine Milot et al. “Autonomous and responsive multi-robot system for minehunting”. In : *OCEANS - IEEE*. 2022. doi : 10.1109/OCEANSChennai45887.2022.9775467.

[MM10] Alejandro Mosteo et Luis Montano. “A survey of multi-robot task allocation”. In : 2010. doi : 10.13140/2.1.2442.3688.

[MML08] Alejandro R. Mosteo, Luis Montano et Michail G. Lagoudakis. “Multi-robot routing under limited communication range”. In : *International Conference on Robotics and Automation (ICRA)*. Pasadena, CA, USA, 2008, p. 1531-1536. doi : 10.1109/ROBOT.2008.4543419.

[MN16] Øivind Midtgaard et Morten Nakjem. “Unmanned systems for stand-off underwater minehunting”. In : *Undersea Defence Technology (UDT 2016)*. 2016.

[MNG16] Michael Zackery McIntire, Ernesto Nunes et Maria L. Gini. “Iterated Multi-Robot Auctions for Precedence-Constrained Task Scheduling”. In : *AAMAS*. 2016. isbn : 9781450342391.

[Mos10] Alejandro R Mosteo. “Multi-robot task allocation for service robotics : from unlimited to limited communication range”. In : 2010. doi : 10.13140/RG.2.1.3601.7686.

[MP20a] Munyque Mittelmann et Laurent Perrussel. “Auction Description Language (ADL) : a General Framework for Representing Auction-based Markets”. In : *European Conference on Artificial Intelligence (ECAI*. 2020, p. 8. doi : 10.3233/FAIA200172.

[MP20b] Munyque Mittelmann et Laurent Perrussel. “Auction Description Language (ADL) : a General Framework for Representing Auction-based Markets”. In : *European Conference on Artificial Intelligence (ECAI 2020)*. 2020.

[MSØ03] Maja J Mataric, Gaurav S Sukhatme et Esben H Østergaard. “Multi-Robot Task Allocation in Uncertain Environments”. In : *Autonomous Robots* (2003), p. 9. doi : 10.1023/A:1022291921717.

[Nau+99] Dana S. Nau et al. “SHOP : Simple Hierarchical Ordered Planner”. In : *IJCAI*. 1999.

[Nay+20] Sharan Nayak et al. “Experimental Comparison of Decentralized Task Allocation Algorithms Under Imperfect Communication”. In : *IEEE Robotics and Automation Letters* 5.2 (2020), p. 572-579. doi : 10.1109/LRA.2019.2963646.

[NG15] Ernesto Nunes et Maria Gini. “Multi-robot auctions for allocation of tasks with temporal constraints”. In : *AAAI Conference on Artificial Intelligence (AAAI)*. 2015. doi : 10.1609/aaai.v29i1.9440.

[OKS17] Michael Otte, Michael Kuhlman et Donald Sofge. “Multi-robot task allocation with auctions in harsh communication environments”. In : *International Symposium on Multi-Robot and Multi-Agent Systems*. Los Angeles, CA, 2017, p. 32-39. doi : 10.1109/MRS.2017.8250928.

[OKS19] Michael Otte, Michael J. Kuhlman et Donald Sofge. “Auctions for multi-robot task allocation in communication limited environments”. In : *Autonomous Robots* (2019). doi : 10.1007/s10514-019-09828-5.

[Pel+22] D. Pellier et al. “HDDL 2.1 : Towards Defining an HTN Formalism with Time”. In : arXiv, 2022. doi : 10.48550/arXiv.2206.01822.

[QGL22a] Félix Quinton, Christophe Grand et Charles Lesire. “Communication-Preserving Bids in Market-Based Task Allocation”. In : *IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2022.

[QGL22b] Félix Quinton, Christophe Grand et Charles Lesire. “Improving the Connectivity of Multi-Hop Communication Networks through Auction-Based Multi-Robot Task Allocation”. In : *International Conference on Practical Applications of Agents and Multi-Agent Systems*. Springer. 2022.

[Rek+08] Ioannis Rekleitis et al. “Efficient Boustrophedon Multi-Robot Coverage : an algorithmic approach”. In : *Annals of Mathematics and Artificial Intelligence* 52 (2008). doi : 10.1007/s10472-009-9120-2.

[Rol+17] Juan Jesús Roldán et al. “Multi-Robot Interfaces and Operator Situational Awareness : Study of the Impact of Immersion and Prediction”. In : *Sensors* 17 (2017). doi : 10.3390/s17081720.

[San02] Tuomas Sandholm. “Algorithm for Optimal Winner Determination in Combinatorial Auctions”. In : *Artificial Intelligence* 135 (2002). doi : 10.1016/S0004-3702(01)00159-X.

[Sar07] Sanem Sariel. “An Integrated Planning, Scheduling and Execution Framework for Multi-Robot Cooperation and Coordination”. Thèse de doct. 2007.

[SB03] Ashley Stroupe et Tucker Balch. “Value-Based Observation with Robot Teams (VBORT) Using Probabilistic Techniques”. In : *Advanced Robotics*. 2003.

[SBD18a] Philipp Schillinger, Mathias Buerger et Dimos Dimarogonas. “Improving Multi-Robot Behavior Using Learning-Based Receding Horizon Task Allocation”. In : *Robotics : Science and Systems XIV* (2018). doi : 10.15607/RSS.2018.XIV.031.

[SBD18b] Philipp Schillinger, Mathias Bürger et Dimos Dimarogonas. “Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems”. In : *The International Journal of Robotics Research* 37 (2018). doi : 10.1177/0278364918774135.

[SBE08] Sanem Sariel, Tucker Balch et Nadia Erdogan. “Naval Mine Countermeasure Missions”. In : *IEEE Robotics & Automation Magazine* (2008).

[SBS06] Sanem Sariel, Tucker Balch et Jason Stack. “Distributed Multi-AUV Coordination in Naval Mine Countermeasure Missions”. In : *GVU Center Technical Reports* (2006).

[SE18] María Santos et Magnus Egerstedt. “Coverage Control for Multi-Robot Teams with Heterogeneous Sensing Capabilities Using Limited Communications”. In : *International Conference on Intelligent Robots and Systems (IROS)*. 2018. doi : 10.1109/IROS.2018.8594056.

[Sen+16] Madhubhashi Senanayake et al. “Search and tracking algorithms for swarms of robots : A survey”. In : *Robotics and Autonomous Systems* 75 (2016), p. 422-434. doi : 10.1016/j.robot.2015.08.010.

[Sha20] Tom G. Sharpe. “Hunt, the Replacement”. In : *Wavell Room* (2020).

[Sim01] Olivier Simonin. “Le modèle satisfaction-altruisme : coopération et résolution de conflits entre agents situés réactifs, application à la robotique”. Thèse de doct. 2001.

[SL10] Christopher Sotzing et David Lane. “Improving the Coordination Efficiency of Limited Communication Multi-AUV Operations using a Multi-Agent Architecture”. In : *Journale of Field Robotics* 27 (2010). doi : 10.1002/rob.20340.

[SM11] Ethan Stump et Nathan Michael. “Multi-robot persistent surveillance planning as a Vehicle Routing Problem”. In : *2011 IEEE International Conference on Automation Science and Engineering*. 2011.

[Smi80] Reid G Smith. “The Contract Net Protocol : High-Level Communication and Control in a Distributed Problem Solver”. In : *IEEE TRANSACTIONS ON COMPUTERS* 12 (1980), p. 10. doi : 10.1109/TC.1980.1675516.

[VB01] H. Van Dyke Parunak et Sven Brueckner. “Entropy and Self-Organization in Multi-Agent Systems”. In : *International Conference on Autonomous Agents*. 2001. doi : 10.1145/375735.376024.

[Ver18] Amit Verma. “Electric vehicle routing problem with time windows, recharging stations and battery swapping stations”. In : *EURO Journal on Transportation and Logistics* 7 (2018), p. 415-451. doi : 10.1007/s13676-018-0136-9.

[Vid+13] Thibault Vidal et al. “Heuristics for multi-attribute vehicle routing problems : A survey and synthesis”. In : *European Journal of Operational Research* 231 (2013), p. 1-21. doi : 10.1016/j.ejor.2013.02.053.

[VMO07] Antidio Viguria, Ivan Maza et Anibal Ollero. “SET : An algorithm for distributed multirobot task allocation with dynamic negotiation based on task subsets”. In : *IEEE International Conference on Robotics and Automation (ICRA)*. 2007. doi : 10.1109/ROBOT.2007.363988.

[VSH99] Manuela Veloso, Peter Stone et Kwun Han. “The CMUnited-97 robotic soccer team : Perception and multi-agent control”. In : *Robotics and Autonomous Systems* 29 (1999), p. 133-143. doi : 10.1016/S0921-8890(99)00048-2.

[VV03] Sven Vries et Rakesh Vohra. “Combinatorial Auctions : A Survey”. In : *INFORMS Journal on Computing* 15 (2003). doi : 10.1287/ijoc.15.3.284.16077.

[Woo18] Christopher Woody. “The US Navy faces ’a huge liability’ in countering one of Iran’s favorite and most dangerous weapons”. In : *Business Insider* (2018).

[Yor18] Veronika Yordanova. “Intelligent Adaptive Underwater Sensor Networks”. Thèse de doct. 2018.

[Zol+19] Artur Zolich et al. “Survey on Communication and Networks for Autonomous Marine Systems”. In : *Journal of Intelligent & Robotic Systems* (2019). doi : 10.1007/s10846-018-0833-5.

[ZS03] Robert Zlot et Anthony Stentz. “Market-Based Multirobot Coordination Using Task Abstraction”. In : *Field and Service Robotics*. 2003. doi : 10.1007/10991459_17.

[ZS05] Robert Zlot et Anthony Stentz. “Complex Task Allocation For Multiple Robots”. In : *IEEE International Conference on Robotics and Automation (ICRA)*. Barcelona, Spain, 2005. doi : 10.1109/ROBOT.2005.1570329.

[ZS06] Robert Zlot et Anthony Stentz. “Market-based Multirobot Coordination for Complex Tasks”. In : *The International Journal on Robotics Research* 25.1 (2006). doi : 10.1177/0278364906061160.