Stratégie coopérative pour la mise en œuvre d'une flotte de robots mobiles dans un milieu ouvert et encombré

Guillaume Lozenguez

université de Caen
Basse-Normandie

U.F.R. : Sciences

ÉCOLE DOCTORALE : SIMEM

THÈSE

présentée par

Guillaume Lozenguez

et soutenue

le 11 Décembre 2012

en vue de l'obtention du

DOCTORAT de l'UNIVERSITÉ de CAEN

spécialité : Informatique et applications

(Arrêté du 7 août 2006)

Stratégie coopérative pour la mise en œuvre d'une flotte de robots mobiles dans un milieu ouvert et encombré

MEMBRES du JURY

Dominique Duhaut	Professeur	Univ. de Bretagne Sud	(Rapporteur)
Raja Chatila	Directeur de recherche	CNRS, Paris	(Rapporteur)
Abderrafiâa Koukam	Professeur	Univ. de Tech. Belfort-Montbéliard	
Nadine Lefort-Piat	Professeure	Ecole ENSMM de Besançon	
Aurélie Beynier	Maître de Conférences	Univ. Pierre et Marie CURIE	(Co-encadrant)
Lounis Adouane	Maître de Conférences	Univ. Blaise Pascal	(Co-encadrant)
Philippe Martinet	Professeur	Ecole Centrale de Nantes	(Directeur)
Abdel-Illah Mouaddib	Professeur	Univ. Caen Basse-Normandie	(Directeur)

Mis en page avec la classe thloria.

Remerciments

Je remercie, en premier lieu, l'équipe d'encadrements constituée autour de ma thèse : Lounis Adouane, Aurélie Beynier, Philippe Martinet et Abdel-Illah Mouaddib. Le co-encadrement n'est pas un exercice évident. Malgré tout, ils m'ont accordé une bonne autonomie tout en m'apportant des soutiens complémentaires.

Je souhaite remercier également les autres membres de mon jury pour avoir accepté de prendre de leur temps pour mon travail. J'espère que ça aura été un investissement profitable. Je remercie particulièrement mes rapporteurs Dominique Duhaut et Raja Chatila pour la qualité de leur rapport malgré les contraintes de temps.

Je remercie les populations des laboratoires du GREYC à Caen et de l'institut pascal à Clermont-Ferrant, pour l'accueil qui est réservé aux nouveaux arrivants avec une pensé amicale pour les membres de l'équipe MAD à Caen et l'équipe ROSASE à Clermont-Ferrant. Les échanges lors de réunions formelles et informelles furent très enrichissants. Je remercie particulièrement aussi les doctorants et les jeunes docteurs (toujours des deux laboratoires) pour les excellents moments passés ensemble.

Enfin, je remercie ma famille pour m'avoir porté tout au long de mes études. Je les remercie notamment pour les nombreux documents dont ce manuscrit qui sont passés par leur relecture. Je remercie particulièrement ma femme, Abir, pour m'avoir supporté, encouragé et parfois managé surtout ces derniers mois. Sans elle, je serais en train de courir, en catastrophe, après une réinscription...

Table des matières

Table des figures

Liste des tableaux

Introduction

Le 6 août de cette année (2012), le robot Curiosity [1] réalisait avec succés son atterrissage sur Mars. Les équipes de scientifiques, ingénieurs et techniciens montraient qu'il était désormais possible de positionner, à quelques mètres près et avec délicatesse, un objet de 699 kg sur une cible à 248 millions de km. Cette mission nécessitait la coopération de plusieurs modules semi-autonomes : Le module spatial transportant le robot depuis la Terre jusqu'à Mars, le module d'entrée dans l'atmosphère martienne, un drone de transport pour les derniers kilomètres, le robot Curiosity et les satellites d'observation et de communication en orbite autour de Mars.

Le déroulement de cette mission a été réalisé en semi-autonomie notamment à cause des longues minutes (environ 8) qui séparent l'envoi et la réception de messages entre Mars et la Terre. Chaque module robotisé était alors autonome pour réaliser les tâches qui lui étaient imparties (freiner, se positionner, détacher, déployer les parachutes, treuiller, guider ou encore relayer les communications). Bien que plusieurs possibilités aient été considérées, la décision finale des actions à accomplir revient aux robots eux-mêmes, suivant les mécanismes et les règles qui leur ont été dictés.

Les unités d'Intelligence Artificielle embarquées doivent permettre aux robots de faire face à toutes les situations possibles dans un cadre de missions définies. Pour l'atterrissage sur Mars, le cadre de mission est extrêmement contraint : les trajectoires à suivre et les conditions de déclenchement des événements ont été définies avec précision. D'autres robots ont à évoluer dans des environnements plus ouverts. Citons, par exemple, la mission consistant à aller chercher un livre dans une bibliothèque à l'aide d'une flotte de robots hétérogènes (des robots mobiles transporteurs, des robots manipulateurs et des drones) [2].

Dans ce scénario, la position du livre dans la bibliothèque n'est pas forcément connue et la position de la bibliothèque dans l'appartement non plus. La configuration de l'appartement lui-même et le positionnement des meubles et des autres objets ne sont pas, non plus, déterminés. Il est alors difficile de prédéfinir les successions de mouvements que doivent effectuer chacun des robots pour réaliser leur mission. Les mécanismes animant chaque robot, consistent à analyser une situation, à communiquer au besoin avec ses coéquipiers et à prendre des décisions en fonction de l'état de ses connaissances. L'espace des possibilités sur l'état des connaissances individuelles est potentiellement, excessivement large.

Approches développées

Les travaux réalisés dans le cadre de cette thèse visent à la mise en œuvre d'une flotte de robots mobiles évoluant dans un environnement ouvert et encombré. L'objectif consiste à proposer des stratégies permettant au groupe de robots d'être autonome dans la réalisation de sa mission.

1. Mission "Mars Science Laboratory" : http://www.nasa.gov/mission_pages/msl/index.html
2. Projet européen "Swarmanoid" : http://www.swarmanoid.org

Ces travaux s'inscrivent dans le cadre du projet "R-Discover" [3] soutenue par l'ANR [4]. Le projet vise la couverture visuelle d'un espace par un drone et une flotte de robots mobiles terrestres. La mission est décrite par des tâches à effectuer, localisées sur des positions différentes. De plus, la réalisation d'une tâche peut en amener de nouvelles (comme dans un cadre d'exploration). Enfin, la solution attendue doit permettre à chacun d'évoluer en étant affecté au minimum par une perte possible de communication avec les autres robots.

Notre approche consiste à permettre aux robots de raisonner sur des objets abstraits plutôt que sur le problème dans sa globalité (e.g. jusqu'au plus bas niveau de réalisation des actions). Ces objets abstraits sont définis par des déplacements entre des positions clefs distantes, des zones de l'environnement, par les tâches à réaliser et par des groupes de tâches. Ainsi, nos mécanismes décisionnels manipulent ces objets sans appréhender l'ensemble des mouvements continus (directions et vitesses) appliqués simultanément par les robots. En fait, notre approche s'apparente un peu à ce que ferait un groupe d'humains.

Prenons la problématique d'un mercredi chargé. Considérons une petite famille avec des parents en repos. La problématique des parents consiste à réaliser un certain nombre de tâches localisées dans des endroits différents de la ville : accompagner les enfants dans leurs activités respectives, faire des courses, s'occuper un peu de la maison, faire à manger, régler des formalités, aller à la poste, à la banque, faire une visite à la maison de retraite, etc. En anticipant une charge importante, les parents vont chercher, normalement, à coordonner leurs actions en début de journée.

Les parents vont se distribuer les tâches en même temps que chacun d'eux planifie le déroulement de sa journée. Il n'est alors pas envisageable de visualiser toutes les actions dans leurs détails (e.g. le nombre de tours de clef pour fermer la maison, le nombre de marches pour descendre dans la bouche de métro, etc.). Dans nos réflexions, nous considérons, de façon naturelle, un niveau de granularité adapté. Au besoin, nous avons la possibilité mentale de descendre et monter sur plusieurs niveaux d'abstraction (la journée entière, des parcours en voiture, la localisation de magasins dans des rues piétonnes, etc.). Cette faculté permet de produire, dans la limite de nos capacités, une réponse à la fois globale et localement cohérente (intégrant des détails sur des points critiques).

Dans un cadre robotique, cette faculté naturelle d'abstraction, n'est pas évidente à mettre en pratique. Chaque force ou chaque vitesse appliquée à un instant donné requiert son lot de modélisations et de calculs. Dans le cadre d'une approche par planification, il n'est pas possible de prédéterminer toutes les forces et vitesses optimales à appliquer à chaque instant, pour toutes les situations potentielles, dès lors que le cadre de la mission induit une importante variété de situations. Si l'on souhaite tout de même mettre en place des mécanismes de planification, il est alors nécessaire de différencier la définition des actions dans leurs détails (forces, vitesses à chaque instant) et les objets manipulés par les mécanismes décisionnels.

Les contributions présentées dans cette thèse reposent sur 3 thèmes : la définition d'une architecture de contrôle permettant de raisonner sur des actions haut niveau ; la possibilité de regrouper ces actions et les tâches qu'elles permettent d'atteindre en zones ; un protocole qui donne la capacité aux robots de coordonner la réalisation d'une mission multi-tâches. Cette thèse se compose de 5 chapitres : 2 chapitres pour présenter le cadre général de travail et la problématique ciblée, suivies d'un chapitre de contributions pour chacun des 3 thèmes cités ci-dessus.

3. R-Discover : http://www.mis.u-picardie.fr/R-Discover

4. ANR : Agence Nationale de la Recherche avec pour mission "d'augmenter la dynamique du système français de recherche et d'innovation en lui donnant davantage de souplesse."
http://www.agence-nationale-recherche.fr/missions-et-organisation/missions/

Organisation des chapitres

Le chapitre 1 "Vers des robots autonomes" présente, de façon générale, le cadre applicatif défini par le contrôle de robots mobiles. Nous nous interrogeons sur les problématiques et les réponses scientifiques et techniques permettant de rendre des robots artificiellement autonomes dans leurs déplacements. Nous y abordons les notions de contrôle, de perception et d'incertitude liées aux différentes missions envisageables. Ce chapitre est dédié principalement au positionnement de nos travaux dans un cadre multi-robots mobiles.

Le chapitre 2 "Planifier pour explorer" pose la problématique de la planification des déplacements dans le contexte difficile qu'est l'exploration. Par définition, dans ce cadre de mission, tout ou partie de l'environnement est inconnu. Ce type de mission est caractérisé par une dynamique forte et incertaine sur l'état des connaissances des robots et par un ensemble potentiellement important de tâches unitaires d'exploration à réaliser. Nous proposons alors de nous appuyer sur le formalisme des processus décisionnels de Markov orientés par plusieurs objectifs pour modéliser et planifier les déplacements des robots vers leurs tâches d'exploration. Ce formalisme a la particularité d'intègrer une notion d'incertitude de façon à optimiser une réponse en tenant compte des différentes évolutions possibles du systéme.

Le chapitre 3 "Une architecture de contrôle/commande hiérarchique" définit une architecture basée sur des actions haut niveau, c'est-à-dire des actions permettant à chaque robot de se déplacer entre deux positions distantes tout en tenant compte des obstacles détectés. Ces actions haut niveau permettent notamment de libérer la planification de contraintes pas à pas, ou en d'autres termes, de la libérer de la production de réponses (forces et vitesses) applicables à chaque instant. L'architecture proposée se décompose en 4 modules : Perception, Représentation, Délibération et Contrôle. Cette décomposition permet de mieux appréhender les difficultés liées à la mise en mouvement de robots mobiles.

Le chapitre 4 "Décomposer des politiques individuelles multi-objectifs" propose une abstraction automatisée pour répondre à la problématique du traitement d'un nombre important d'objectifs (de tâches à réaliser). Sur la base des processus décisionnels de Markov décomposés, les contributions reposent sur un partitionnement de l'ensemble des actions de déplacement en zones. Cette décomposition est rendue possible, en cours de mission, par l'utilisation d'un algorithme glouton. Chaque robot peut alors planifier selon deux niveaux d'abstraction. Un premier plan de niveau supérieur ordonne les déplacements entre les zones. Il permet d'orienter des plans calculés de façon locale à chaque zone. Les plans locaux visent alors la réalisation des tâches contenues dans les zones tout en tenant compte des futures zones à atteindre.

Enfin, dans le chapitre 5 "Coordination multi-agents par négociation", nous nous intéressons à l'allocation des tâches à réaliser entre les robots. Sur la base de processus décisionnels de Markov individuels, chaque robot est en mesure d'exprimer des préférences sur la réalisation de telle ou telle tâche. Ces préférences peuvent être utilisées, dans le cadre d'un protocole de ventes aux enchères, pour générer l'allocation des tâches. Cependant, la préférence sur la réalisation d'une tâche dépend de la localisation des autres tâches qui doivent être réalisées par un même robot. Notre contribution repose alors sur une succession de ventes aux enchères où, à chaque itération, toutes les tâches sont remises simultanément en vente.

Chacun des chapitres 3, 4 et 5 est composé d'une section "état de l'art" et d'une section "expérimentation". Ces sections permettent de positionner et de valider nos contributions au fur et à mesure que celles-ci seront présentées. En effet, bien que le fil conducteur soit la mise en œuvre d'une flotte de robots mobiles autonomes, les contributions apportées dans chacun de ces chapitres, reposent sur des problématiques scientifiques différentes.

Enfin, en conclusion, nous prendrons un peu de recul sur le travail réalisé vis à vis d'une problématique similaire à un multi-voyageurs de commerce dynamique, évoluant dans un environnement incertain. Les approches proposées ici permettent à une flotte de robots de planifier les déplacements minimaux par un processus distribué, de façon à ce que toutes les tâches d'une mission soient accomplies. Ce processus distribué peut ensuite être relancé au besoin, en cours de mission, au fur et à mesure que l'état des connaissances évolue.

Chapitre 1

Vers des robots autonomes

L'application d'une organisation scientifique du travail [Taylor 11] dans les industries a été propice au développement de robots dans les chaînes de production. Sur une chaîne de production, chaque agent (humain ou robot) a une tâche spécifique à exécuter dans un environnement complètement adapté. Le robot peut exécuter certaines opérations répétitives plus longtemps et de façon plus rapide et plus précise que l'humain. Il est aussi capable de déployer plus de force et de réaliser des opérations qui ne seraient pas à la portée d'un humain.

Ce contexte industriel a permis la mise en œuvre de robots conçus pour la réalisation de tâches bien précises dans des environnements particuliers. Malgré les avancées en robotique, il reste nombre d'opérations difficiles à réaliser pour un robot, notamment des opérations nécessitant des gestes non seulement précis mais adaptés à l'objet manipulé. Dans les faits, par exemple, les robots sont incapables de reproduire le comportement de vendangeurs humains. La vigne doit être adaptée à la machine pour que le fonctionnement automatique de la machine cueille effectivement les fruits. La machine n'a pas la dextérité d'un humain pour chercher et sectionner les grappes d'une vigne traditionnelle (plus naturelle) où les mouvements à effectuer ne sont pas complètement déterminés.

Les travaux réalisés dans le cadre de cette thèse visent à augmenter l'autonomie décisionnelle de robots mobiles évoluant dans des environnements qui n'ont pas été spécifiquement adaptés pour eux. Le Larousse en ligne [5] définit l'autonomie d'une entité comme : "capacité de quelqu'un à être autonome, à ne pas être dépendant d'autrui ; caractère de quelque chose qui fonctionne ou évolue indépendamment d'autre chose". L'autonomie d'un robot correspond alors à sa capacité à agir, sans l'aide d'un opérateur, dans un environnement qui n'a pas été adapté pour lui. Cette autonomie passe par une interaction forte entre l'environnement et le robot de façon à ce que le robot puisse réagir correctement aux événements.

En premier lieu, le robot doit être capable de percevoir son environnement de façon à pouvoir en extraire le maximum d'informations utiles. En deuxième lieu, le robot doit être capable d'organiser dans sa mémoire, les informations sur son environnement. Enfin, en troisième lieu, le robot doit choisir la meilleure action à réaliser. Ces actions peuvent amener le robot à améliorer sa connaissance de l'environnement (communication, réglage de capteurs, etc) et/ou à modifier l'environnement (déplacement d'objets, usinage, etc).

La robotique mobile présente un cadre de travail intéressant pour pousser plus loin la réflexion sur l'autonomie des robots. L'objectif alors, est de permettre aux robots de se déplacer dans un environnement encombré (c'est-à-dire en présence d'obstacles) pour réaliser des tâches distantes les unes des autres. La robotique mobile autonome est motivée par plusieurs cadres applicatifs :

5. Dictionnaires Larousse : http://www.larousse.fr/dictionnaires/francais/

exploration planétaire, robotique domestique, véhicule autonome urbain, industriel, agricole et militaire.

Dans le cadre de cette thèse, on souhaite que l'environnement n'ait pas à être adapté aux robots (e.g. installation de rails), que les robots puissent travailler dans différents environnements et on souhaite aussi minimiser les interventions d'opérateurs humains. Il est attendu en retour, un comportement sécurisé et permettant la réalisation de la mission donnée. L'objectif dans ce chapitre, est de définir la problématique de la prise de décision en robotique mobile de façon à positionner les contributions réalisées dans cette thèse et à dégager des critères d'évaluations.

1.1 Modélisation et Architectures

"Un modéle en science est une image stylisée et abstraite d'une portion de réalité. L'activité scientifique consiste principalement à faire des modèles des phénomènes et des objets qu'elle étudie." (J. Feber) [Ferber 95]. L'activité robotique vise quant à elle à mettre en œuvre des modèles pour permettre à des machines d'interagir dans le monde réel. Il existe alors, toujours une confrontation entre les modèles et la réalité. Cette confrontation conduit à des approximations sur la précision du contrôle voire à observer des phénomènes qui sortent du cadre du modèle (e.g. glissement). De plus, l'étude globale de phénomènes et d'objets complexes comme pour le contrôle de robot mobile, conduit à une décomposition de la problématique en parties qui seront étudiées séparément.

1.1.1 Modélisation pour commander un robot mobile

Les robots sont capables d'agir physiquement dans leur environnement réel. Leurs capacités d'actions leur sont données par des effecteurs mécaniques animés par une énergie électrique, pneumatique ou encore thermique (moteur à explosion). Les effecteurs d'un robot mobile lui donne la capacité de se déplacer dans son environnement.

Pour déplacer un robot mobile il faut établir un modèle du robot lui-même. Le livre "Introduction to Autonomous Mobile Robots" [Siegwart 05] présente un état de l'art complet de la problématique. Un modèle cinématique du robot va permettre d'exprimer les mouvements qu'il est capable de réaliser. Il va en découler un certain nombre de variables décrivant des éléments composants le robot (position, orientation, vitesse) ainsi qu'un certain nombre de relations entre ces variables. Ces relations modélisent les liaisons des éléments entre eux et avec l'environnement. La motorisation du système robotisé permet de commander certaines de ces variables [Campion 96].

Plusieurs types de robots

Il existe une grande variété de robots mobiles terrestres : des robots à roues (Figures 1.1, 1.2), des robots à chenille (Figure 1.3), des robots sur jambes (figures 1.4, 1.5) et d'autre solutions plus atypiques (Figure 1.6). Un robot est donc en partie défini par le nombre de roues/jambes qu'il possède et par les mouvements libres ou motorisés appliqués à chacune de ses roues/jambes.

La première condition à l'autonomie d'un robot mobile est sa capacité à contrôler ses déplacements. Il est donc nécessaire de proposer un premier modèle permettant de représenter le robot et les commandes applicables pour passer d'une configuration à une autre. Les commandes forment ensuite les outils permettant au contrôle de déplacer le robot dans un environnement en présence ou non d'obstacles.

FIGURE 1.1 – Pioneer : robot quadricycles développé par Adept MobileRobots

FIGURE 1.2 – Tartan Racing's Boss : voiture robotisée vainqueur du "Darpa Grand Challenge 2007" (course de véhicules autonomes en milieu urbain).

FIGURE 1.3 – iRobot-710-Warrior : robot chenillard téléopéré, équipé d'un bras et d'une pince

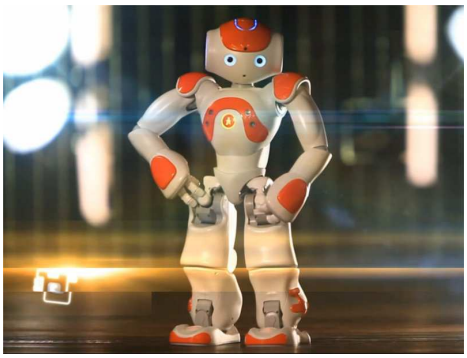


FIGURE 1.4 – Nao : robot humanoïde d'Aldebaran

Un premier modèle

Une première modélisation consiste à considérer le robot comme un point en mouvement dans un repère cartésien global à l'environnement R_E. La configuration du robot à un instant t peut être définie par sa position, sa vitesse et son accélération (Figure 1.7) si le système est commandable en position, vitesse ou en force.

La commande est définie par les variables modifiables (et leurs domaines de variation) qui vont permettre au robot d'agir dans son environnement. Une commande en position signifie qu'il est possible de déplacer instantanément le robot d'une position à une autre au choix. Ce type de commande autorise, en théorie, à appliquer une vitesse infinie. Une commande en vitesse autorise à appliquer instantanément une vitesse au robot (avec des accélérations théoriques infinies). C'est le type de commande le plus répandue pour les robots mobiles évoluant à des vitesses modérées. Une commande en force permet de jouer sur les accélérations effectuées et s'accompagne, généralement, d'une modélisation plus fine intégrant l'évolution des forces présentes au cours du temps (modèle dynamique).

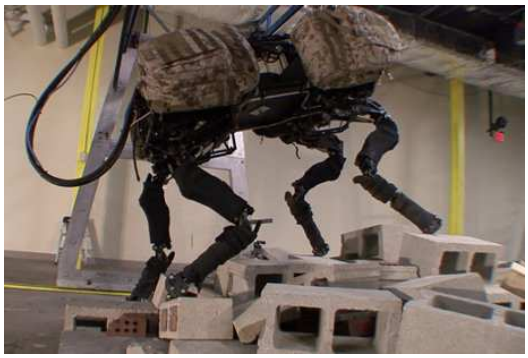


FIGURE 1.5 – BigDog : robot quadrupéde tout terrain développé par Boston Dynamics

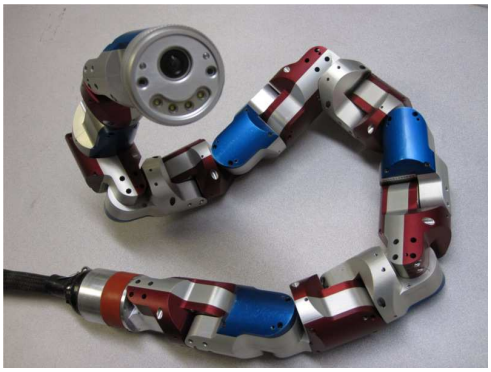


FIGURE 1.6 – Uncle Sam : robot biomorphique hyper-redondant ressemblant à un serpent développé par Biorobotics Laboratory, Carnegie Mellon University

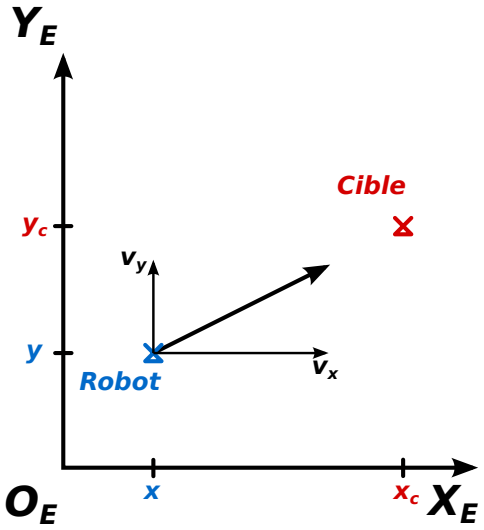


FIGURE 1.7 – Représentation minimale pour un robot holonome

Robots holonomes

Les commandes à appliquer sont définies pour conduire le robot d'une configuration initiale à une configuration cible. En considérant des robots mobiles commandables en vitesse, la définition de la configuration du robot peut se limiter à sa position. Le contrôle pour atteindre la configuration cible, définit les variables de commande, donc les vitesses à appliquer au robot à chaque instant. En considérant un robot holonome, c'est-à-dire : capable de tourner et de se déplacer indifféremment et instantanément dans les 2 directions du plan ; Le robot et sa cible peuvent être modélisés par deux points dans un repère R_E orthonormé fixe dans l'environnement (Figure 1.7).

$$robot_t = \begin{bmatrix} x_t \\ y_t \end{bmatrix}, \qquad cible = \begin{bmatrix} x_c \\ y_c \end{bmatrix}, \qquad commande = \begin{bmatrix} v_x(t) \\ v_y(t) \end{bmatrix}$$

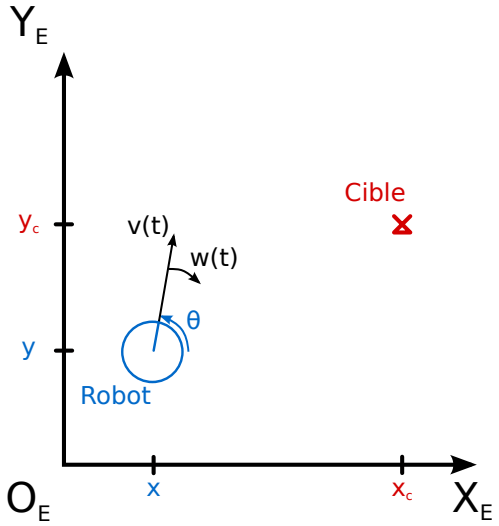


FIGURE 1.8 – Représentation minimale pour un robot non-holonome

En considérant un robot holonome commandé en vitesse, une loi de commande théorique consiste alors à ordonner la vitesse maximale en direction de la cible jusqu'à ce que celle-ci soit atteinte.

Robots non-holonomes

La plupart des robots mobiles ne sont pas capables d'avancer indifféremment dans toutes les directions du plan. Les robots dits non-holonomes, sont des robots ne pouvant avancer que dans la direction donnée par leur orientation [Canudas de Wit 92]. Nos voitures, par exemple, peuvent être considérées comme non-holonomes. La commande en vitesse d'un robot non-holonome est donnée par la vitesse $v(t)$ dans l'axe d'orientation du robot et une vitesse de rotation $w(t)$ (Figure 1.8). Il devient alors nécessaire d'ajouter l'orientation θ du robot dans son modèle.

$$robot_t = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix}, \qquad cible = \begin{bmatrix} x_c \\ y_c \end{bmatrix}, \qquad commande = \begin{bmatrix} v(t) \\ w(t) \end{bmatrix}$$

Une loi de commande simple consiste à appliquer une vitesse angulaire positive ou négative de façon à orienter le robot vers sa cible et appliquer ensuite une vitesse linéaire jusqu'à atteindre la cible [Brooks 86, Canudas de Wit 92]. Les vitesses (linéaire et angulaire) sont plus ou moins contraintes. Ainsi, certains robots ont la possibilité de tourner sur place et d'autres pas. Le contrôle du robot va dessiner une trajectoire lisse qui doit lui permettre d'atteindre sa cible le plus rapidement possible.

La commande d'un robot est définie directement par sa conception et il existe différents types méca-électroniques de robots mobiles terrestres [Siegwart 05]. Tout un pan de la robotique cherche à contrôler des robots toujours plus complexes, avec plus ou moins de précision et/ou dans des situations incommodantes. En effet, il n'est pas évident de maintenir en équilibre des robots sur jambes en mode marche/course et ce, sur tout type de sols [Huang 01, Saranli 01].

La cinématique d'un véhicule va impacter sa trajectoire la rendant difficile à pré-calculer. Les modèles vont être enrichis de façon à exprimer des relations plus fines entre la position réelle du robot, son orientation, la vitesse commandée et les accélérations possibles (e.g., la problématique de commander un véhicule à haute vitesse sur sols glissant [Lenain 11]).

Dans la mesure où l'on possède un système commandable permettant d'asservir le robot vers certaines configurations, l'étape suivante consiste à définir un contrôle pour qu'un robot mobile atteigne sa cible dans un environnement encombré. Les robots vont devoir alors naviguer de façon à atteindre leur position tout en évitant les obstacles sur leur route.

1.1.2 Architecture de contrôle

Le contrôle en boucle ouverte d'un robot mobile définit la commande à appliquer à un instant donné pour un état du système (Figure 1.9). Le contrôle en boucle ouverte permet d'adapter la commande appliquée en fonction de retours perceptif. Nous avons vu que la commande peut être définie à partir d'une configuration cible (position, orientation et vitesse) à atteindre depuis la configuration courante du robot. L'état du système robotique est donné par la perception que le robot a de son environnement.

FIGURE 1.9 – Schéma de contrôle/commande

Le contrôle donc, définit la commande en fonction de l'état courant du système robotisé. Le contrôle va être différent en fonction des tâches de déplacement à réaliser : atteindre une position ; contourner un obstacle ; suivre un leader ou un chemin ; etc. Ainsi, plusieurs contrôleurs élémentaires vont pouvoir être définis pour chaque tâche différente. Si certains contrôleurs permettent, seuls, des déplacements autonomes du robot dans un environnement encombré, d'autres auront besoin d'être combinés pour générer un comportement adéquat à toute situation.

Approches réactives / délibératives

A un instant t, la perception dépend de l'état courant de l'environnement et de la configuration du robot dans cet environnement. La perception courante peut être enrichie d'une mémoire des perceptions et des actions passées. Cette mémoire peut être plus ou moins organisée, nous reviendrons plus tard sur la représentation de l'environnement d'un robot mobile. Un contrôleur réactif vise à limiter au minimum les données et les processus utiles à la définition des commandes à un instant t.

Un contrôleur réactif se base sur l'hypothèse qu'il n'est pas nécessaire de précalculer la trajectoire totale du robot pour lui permettre d'atteindre une configuration. Pour une commande en vitesse, la définition de règles simples de contrôle conditionnées par la perception courante permettrait donc de calculer les vitesses courantes à appliquer sans se soucier de toutes les configurations intermédiaires jusqu'à la cible.

Une autre démarche consiste à considérer l'ensemble des connaissances que l'on a à un instant t de façon à calculer un contrôle le plus optimal possible. Un contrôleur délibératif va chercher à examiner un problème pour construire une décision complexe. Pour des robots mobiles, le

problème de déplacement va impliquer d'examiner la configuration de l'espace de façon à calculer un chemin voire une trajectoire.

De cette façon, dans le cadre du déplacement d'un robot, le contrôle du robot ne va pas s'effectuer directement sur la cible finale à atteindre mais sur une succession de cibles intermédiaires. Cette succession définit un chemin jusqu'à la cible. Ces cibles intermédiaires, ou points de passages, permettent de déplacer le robot en tenant compte de la configuration des obstacles. L'objectif consiste alors à définir la succession de cibles en fonction de la connaissance courante.

Contrôleur relatif / global

Il est possible d'exprimer les règles de contrôle dans un repère global R_E (fixe dans l'environnement) ou dans un repère relatif au robot R_R (fixe par rapport au robot). Dans le cadre d'une perception locale avec des capteurs embarqués sur le robot, il est naturel d'exprimer directement les éléments de l'environnement dans le repère relatif R_R. Ainsi, le modèle n'exprime plus le déplacement du robot mais le déplacement des éléments de son environnement (dont la configuration cible) par rapport à lui-même considéré comme fixe.

Cette structuration est plus cohérente avec les données de perceptions exprimées par rapport au robot notamment dans le cas où la configuration cible est directement définie dans le flot des capteurs embarqués. Rouler au milieu d'une voie nécessite simplement de détecter le milieu de la voie relativement à la configuration courante du robot. Par contre, le repère relatif R_R est inapproprié pour exprimer la connaissance sur environnement. Ainsi, un contrôle basé sur une connaissance mémorisée s'effectuera préférentiellement dans un repère global.

Le contrôle relatif peut se faire relativement à un véhicule maître à suivre [Parker 93], à une marque (comme une bande blanche) à longer ou encore, à un rail virtuel [Courbon 09]. Le rail virtuel est défini comme une succession de repères perceptifs (e.g. images). Le contrôle global s'appuie quant à lui généralement sur une carte de l'environnement [Elfe 87, Thrun 98], le robot est localisé dans sa carte, à chaque pas de temps, en fonction de sa perception avant de définir les commandes à appliquer.

Le choix de la mise en place d'un contrôleur relatif ou global dépend du volume de données qui nécessitera un changement de repère. Un contrôle relatif est souvent employé dans une approche réactive mettant en œuvre peu d'informations mémorisées. Un contrôle global sera défini s'il dépend d'une connaissance géométrique de l'environnement mémorisée de façon globale.

Architecture multi-contrôleurs

Les architectures multi-contrôleurs partent du postulat que l'on possède des briques de contrôle élémentaires (Figure 1.10). La difficulté réside alors dans la fusion des commandes issues de diffèrents contrôleurs élémentaires. La fusion consiste à définir les commandes finales à appliquer au robot en fonction des commandes données par chacun des contrôleurs.

Les premières solutions proposées reposent sur l'activation d'un contrôleur au détriment des autres. Dans l'architecture par subsomption [Brooks 86], une hiérarchie est définie entre les contrôleurs. Ainsi, la priorité est donnée au contrôleur de plus haut niveau en ignorant les commandes définies par les autres. Si le contrôleur de plus haut niveau ne propose pas de commandes alors on regarde le contrôleur de niveau inférieur et ainsi de suite jusqu'à ce qu'un des contrôleurs définisse la commande à appliquer.

Il faut alors se poser la question suivante : à quel moment le contrôleur d'évitement d'obstacles doit libérer la main ? De ces conditions dépend l'efficacité du système robotique. Ainsi, par

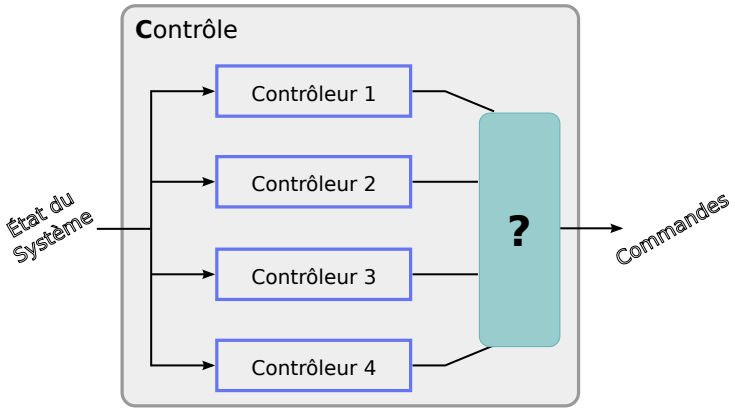


FIGURE 1.10 – Architecture multi-contrôleurs

exemple, une commutation entre évitement et attraction opérée dès que le champ est libre couplé à un évitement systématique par la droite ne garantit pas que l'objectif sera atteint (Figure 1.12).

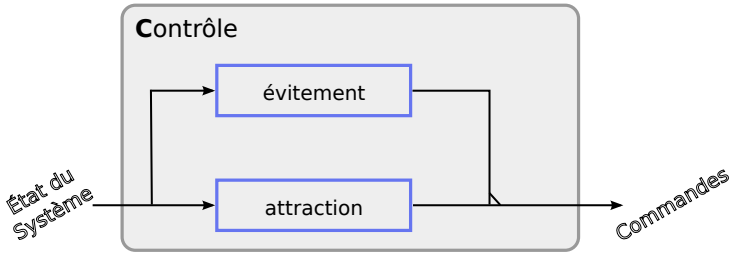


FIGURE 1.11 – Architecture subsomption pour le déplacement d'un robot mobile dans un environnement encombré

De nombreuses approches se basent sur des architectures multi-contrôleurs de part la modularité possible due à l'application variée de contrôleurs élémentaires. De plus, la commutation entre les contrôleurs (comme l'architecture subsomption) est simple à mettre en œuvre. Par contre, la commutation instantanée entre deux contrôleurs peut générer des chocs sur la commande et nuire à la qualité du déplacement, surtout si ces commutations opèrent avec une grande fréquence. Si certaines approches cherchent à définir une commutation plus souple entre deux contrôleurs [Adouane 09], une véritable fusion des commandes portées par chacun des contrôleurs est difficilement généralisable et dépend fortement des contrôleurs élémentaires mis en œuvre notamment si les contrôleurs élémentaires commandent des parties distinctes du robot (robot mobile équipé de bras).

1.1.3 Mission multi-robots

Toute une branche qui étudie les comportements coordonnés dans les systèmes multi-robots et plus largement dans les systèmes multi-agents, cherche à caractériser le comportement d'un groupe partageant des objectifs communs à partir de comportements individuels pré-définis simples et/ou experts. Des études bio-inspirées partent du constat que, la définition de comportements individuels simples permet de faire émerger une forme d'intelligence collective. Ainsi, C. W. Reynolds [Reynolds 87] a mis en évidence que l'application de trois règles simples permet

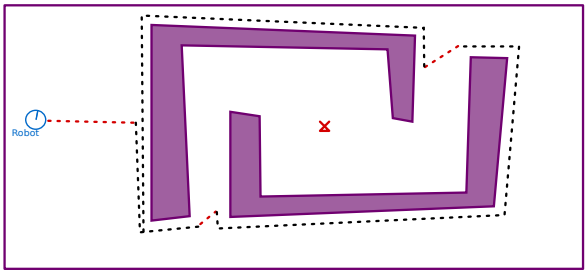


FIGURE 1.12 – Exemple d'exécution d'une "mauvaise" architecture multi-contrôleurs

à un groupe de se déplacer en essaim (d'une manière coordonnée). Les règles consistent en garder une distance de sécurité avec les autres agents, orienter son cap dans la direction moyenne donnée par les autres agents et adapter sa vitesse sur la vitesse moyenne des autres agents.

Les algorithmes basés sur une colonie de fourmis pour trouver le chemin le plus court entre deux points (une source de nourriture et la fourmilière) ont fourni d'excellents résultats dans des environnements même de type labyrinthe. L'interaction entre les fourmis est définie par un dépôt continu de phéromones le long de leur passage. Les phéromones sont des marqueurs dans l'environnement qui se volatilise avec le temps. Le comportement des fourmis s'appuient alors sur un ratio entre exploration aléatoire et suivi de phéromones. Plus la piste est fraîche et fréquentée plus la fourmi délaissera l'exploration pour suivre la piste.

Pour des robots mobiles, le challenge consiste à définir les contrôleurs individuels qui permettent à un groupe d'atteindre un ou plusieurs objectifs. Sur la base de la somme des actions de déplacement exécutées simultanément, la difficulté consiste à définir un système sûr et stable, c'est-à-dire que, quels que soient les événements, le groupe de robots tend vers la réalisation de ses objectifs.

Architecture de contrôle distribuée

Comme initié par les études sur les colonies de fourmis, la robotique mobile compte plusieurs exemples de mise en place de comportements individuels simples permettant la réalisation de missions multi-agents. Les comportements individuels simples sont généralement définis par des architectures multi-contrôleurs embarquées sur chaque robot, on parle alors d'architectures multi-contrôleurs distribuées [Benzerrouk 10].

Les architectures multi-contrôleurs permettent de définir simplement le contrôle pour déplacer un robot dans un environnement encombré. L'élaboration d'architectures distribuées consiste alors à adapter les règles appliquées individuellement pour permettre au groupe de converger vers leurs objectifs. Cependant, un contrôle totalement distribué sur la base des perceptions individuelles ne garantit pas d'éviter des situations de blocage induisant un échec de la mission multi-robots [Duhaut 07]. Dans de nombreuses missions, il est nécessaire de partager un certain nombre d'informations pour trouver un compromis entre contrôle distribué et centralisé.

Un contrôle distribué est plus "naturel" qu'un contrôle centralisé. La perception et la commande sont définies de façon individuelle en rapport aux capteurs et effecteurs embarqués par chaque robot. Une architecture centralisée nécessite alors des communication efficaces. Les perceptions individuelles sont fusionnées à chaque pas de temps sur une unité centrale qui calcule et envoie en retour le contrôle à appliquer à chaque robot. Un contrôle centralisé permet part contre, d'optimiser les actions individuelles sur la base de l'ensemble des données connues.

De nombreuses approches s'appuient sur des architectures distribuées avec fusion des connaissances. Chaque robot est maître de son comportement mais, dans la mesure du possible, les robots construisent ensemble une connaissance commune sur l'environnement. Par exemple, dans le cadre de l'exploration d'une zone pour la cartographier, des résultats intéressants ont été présentés sur la base de comportements individuels simples couplant attraction aux frontières les plus proches et éloignement des autres agents [Simmons 00, Burgard 05, Matignon 12]. La carte, embarquée par chaque robot, est construite par fusion des connaissances de chaque robot à chaque pas de temps.

Une solution distribuée avec fusion des connaissances permet de relâcher la contrainte sur la communication ; la connaissance est fusionnée uniquement quand cela est possible [Burgard 05]. Par contre, la bonne efficacité des comportements individuels n'est pas garantie sur les intervalles de temps sans communication. La mise en place de contrôles distribués cohérents nécessite des interactions entre les robots pour partager un minimum d'informations.

Observabilité et Interaction

La définition des comportements de chaque robot pour la réalisation d'une mission donnée dépend de la connaissance de chacun à chaque instant t par rapport au système multi-robots global. La connaissance d'un robot dépend de sa perception locale et sa capacité à échanger des informations avec son groupe. Pour un agent, le choix du contrôle à réaliser ne peut être basé que sur son état interne, pourtant la réalisation de la mission dépend de choix basés sur l'état global découlant des états internes de tous les robots. L'état interne d'un robot peut être alors défini sur la base d'une observabilité partielle ou totale.

L'observabilité d'un robot définit la différence entre sa connaissance et la somme des connaissances des autres. Avec une observabilité totale, à chaque instant la perception du robot lui permet de déduire l'état global incluant les états de chacun des autres robots. Avec une observabilité partielle, les données perçues ne permettent pas au robot, de reconstruire l'état global du système. Une observabilité partielle inclut alors une partie plus ou moins importante d'informations partagées entre plusieurs robots du groupe.

Une observabilité partielle peut être suffisante pour la bonne réalisation d'une mission. Par exemple, des comportements individuels pour un déplacement en essaim pour une nuée d'agents peuvent être définis uniquement avec une connaissance relative des positions, caps et vitesses des quelques agents alentours [Reynolds 87]. La définition des comportements individuels sur la base d'une observabilité partielle, s'appuie alors sur les observations partagées entre les agents d'un groupe.

Plusieurs agents d'un groupe peuvent percevoir les mêmes éléments de l'environnement. Ainsi, il est possible de définir des interactions entre plusieurs robots. Par exemple, des robots s'observant les uns les autres peuvent renforcer leur connaissance sur leur localisation [Karam 06] ou encore il est possible pour un robot d'adapter son comportement après observation des actions réalisées par un autre agent (robot ou humain) [Karami 10].

D'autre part, les actions sur l'environnement permettent de le modifier. Certaines actions peuvent alors être réalisées dans l'intention d'altérer la perception des autres. Ce principe, sous-jacent à la communication, consiste à utiliser l'environnement pour transmettre de l'information lorsque les observations partielles individuelles ne sont pas suffisantes pour coordonner les agents.

Communication

Les communications sont un type particulier d'interaction basées sur l'échange de messages significatifs. Pour que deux robots "se comprennent" il faut mettre en place un protocole partagé permettant d'interpréter les messages et permettant de définir quand envoyer quelle information. En fonction du biais utilisé dans l'environnement pour l'échange de messages et des protocoles mis en place, la communication va se faire selon des architectures différentes. Il est possible de classifier les architectures de communication selon 3 types *unicast*, *broadcast* ou *blackboard*.

Dans une communication *unicast*, le message, encapsulant l'information transmise, est envoyé à un unique destinataire ciblé (e.g. courriers postaux). Les réseaux d'ordinateurs sont construits sur cette architecture. Un message est associé à une adresse désignant l'ordinateur cible dans le réseau. Le message est porté par un signal électrique (câble) ou électromagnétique (Wi-Fi). Le Wi-Fi est largement utilisé en robotique mobile. La communication *multicast* consiste à adresser un message à plusieurs destinataires individuellement.

Avec une communication en *broadcast*, le message n'est pas adressé. Tous les agents percevant le signal porteur du message peuvent avoir accès aux informations transmises. Ce type de communication est plus naturel. La communication orale humaine peut être qualifiée de communication en *broadcast*. Une parole est alors intelligible à toute personne alentour partageant le même langage. Dans un réseau informatique, le signal est naturellement accessible à toute antenne, ce sont les protocoles de bas niveau qui assurent l'*unicast*.

La communication standard s'appuie sur des phénomènes physiques ponctuels, des signaux. Le principe de la communication en *blackboard* ou "tableaux noirs", consiste à utiliser des éléments de l'environnement permettant de stocker les messages avec une certaine durée d'existence. C'est le procédé utilisé via les phéromones dans les algorithmes de colonies de fourmis. Le message a alors une existence propre, il peut être lu voire modifié et enrichi, par plusieurs agents pendant sa durée d'existence.

Un *blackboard*, s'il existe, peut permettre de partager un état global du système par fusion des états perceptifs de chacun des agents. Cet outil a un intérêt certain dans des missions d'exploration multi-robots avec cartographie [Simmons 00, Burgard 05, Rooker 07]. La carte construite doit être partagée au mieux pour que chaque agent connaisse l'état d'avancement de l'exploration à chaque instant ainsi que la position des autres robots. Un *blackboard* virtuel peut être défini sur la base de communication sur signaux ponctuels. Un service de *blackboard* peut être centralisé sur un agent particulier qui va fusionner les messages de chacun et redistribuer l'état global du système. Le *blackboard* peut aussi être porté par tous les agents, chacun faisant le travail de fusion de données.

En pratique, et notamment en robotique, les possibilités de communication sont soumises à des contraintes. La communication peut être qualifiée de *connectée*, si deux robots peuvent communiquer. Les messages sont alors échangés efficacement. Elle peut aussi être qualifiée *avec perte* ou *bruitée*. En effet, il se peut qu'un message n'arrive jamais à son destinataire et, s'il arrive, il se peut que le message reçu diffère du message envoyé.

En général, plus la technologie est proche du phénomène physique permettant la communication, moins elle fournit de garantie sur cette communication. Les protocoles réseaux, en informatique, superposent plusieurs couches de traitement permettant, artificiellement, de fournir les garanties attendues au prix du tout ou rien. La déconnexion interviendra avec anticipation, alors que des bribes de messages seront encore reçues. De plus, ces couches induisent une certaine lourdeur lors de la transmission de messages, on parle alors du coût de communication. Dans la mesure où la transmission d'un message consomme des ressources temporelles sur une

bande passante, chaque agent doit contrôler son volume de communication pour permettre une coordination efficace.

1.2 Déplacement dans un environnement encombré

La problématique centrale à la robotique mobile terrestre consiste à permettre aux robots de se déplacer d'une position à une autre dans un environnement encombré. Comme nous l'avons vu, cette problématique sous-entend dans un premier temps, de modéliser le robot de façon à le mettre en mouvement. Cependant, l'environnement peut être partitionné, à partir de la configuration du robot, en espace libre et en espace obstacles.

Dans un second temps, l'objectif consiste à définir les routes que les robots vont suivre de façon à atteindre leurs cibles tout en évitant les obstacles présents. La définition de chaque route peut se faire de façon plus ou moins réactive ou délibérative en fonction des approches et des ressources mises en œuvre.

Par contre, la fluidité du système dépend des déplacements simultanés de tous les robots. Une optimisation de la coordination des robots appelle plutôt une résolution centralisée. Pourtant, par nature, le contrôle de chaque robot est plutôt distribué. Chaque robot possède sa propre perception, ses propres ressources de calcul pour définir ses commandes. Il existe alors plusieurs approches plus ou moins centralisés/distribués pour coordonner les déplacements des robots.

1.2.1 Approches bio-inspirées

Les comportements bio-inspirés trouvent leur source dans l'observation du vivant et notamment du monde des insectes. Malgré des capacités perceptives et cognitives qui semblent limitées, les insectes sont capable de réaliser des tâches complexes comme atteindre une position particulière et de faire, par exemple, des aller-retours entre la fourmilière et une source de nourriture.

En robotique mobile, ces observations ont conduit à la recherche de règles de contrôle simples appliquées de façon locale et distribuée qui permettent d'atteindre un objectif défini. C'est une approche cohérente avec des architectures multi-contrôleurs basées sur des contrôleurs élémentaires de type attraction / évitement.

Dans le cadre du déplacement, nous avons vu (Figure 1.10) que la commutation entre contrôleurs élémentaires, ne permet pas de garantir systématiquement que le déplacement terminera sur toute position cible atteignable. L'objectif consiste alors à définir des contrôleurs et des règles de commutation qui soient à la foi simples, efficaces et qui permettent de garantir que tous les robots atteindront leurs cibles respectives s'il existe un chemin.

Algorithme *Bug*

L'algorithme Bug [Lumelsky 90] est sûrement un des algorithmes les plus simples pour permettre à un unique robot mobile d'atteindre sa position. Il se présente comme suit :

Cet algorithme permet de garantir le déplacement du robot jusqu'à une cible atteignable. En effet, chaque contournement d'obstacles ramène le robot sur la route à une position plus avancée et cela, jusqu'à ce qu'il n'y ait plus d'obstacles sur sa route. Chez les fourmis la route est définie par les phéromones déposées collectivement. Dans le cadre d'un robot mobile qui connaît sa position dans l'environnement la route peut être initialisée comme la connexion en ligne droite entre la position initiale du robot et la position cible (figure 1.13).

La force de ce type d'algorithme réside dans le fait que la configuration des obstacles n'a pas besoin d'être connue initialement. En plus du positionnement du robot, seule la perception locale

Algorithme 1 : Bug

Données : une *route* jusqu'à la *cible* ***et*** sens pour longer un obstacle
Résultat : le robot sur la *cible*

```
répéter
    actualiser position;
    actualiser obstacle;
    si position sur route et pas d'obstacle alors
        re-initialiser route depuis position;
        avancer selon route;
    sinon
        longer obstacle;
jusqu'à position = cible ;
```

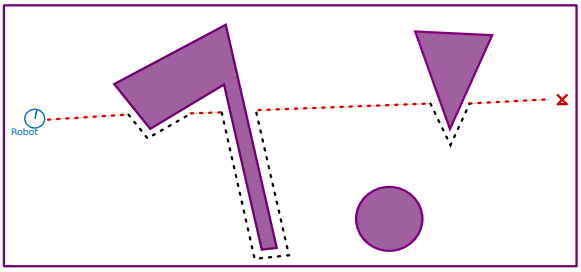


FIGURE 1.13 – Route suivie par l'éxecution de l'algorithme Bug

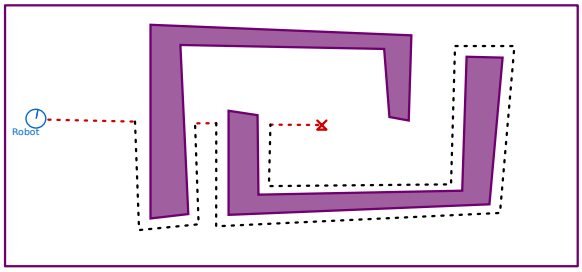


FIGURE 1.14 – Algorithme Bug dans un labyrinthe simple

de l'obstacle immédiat est requise. Le rayon de perception doit être défini de façon cohérente en tenant compte de la distance de sécurité à garder entre les obstacles et le robot.

En revanche une critique spontanée peut être formulée quant à l'optimalité du déplacement réalisé. En effet, un simple regard sur le chemin du robot permet de pointer plusieurs améliorations possibles pour raccourcir son déplacement. Il est alors possible de modifier les contrôleurs mis en œuvre et les règles de commutation entre contrôleurs. Les architectures multi-contrôleurs réactifs cherchent, en employant des règles de calcul simples, à concilier stabilité (garantie que la cible sera atteinte) et qualité. La qualité du déplacement dépend alors de l'application visée. On peut chercher à minimiser : les déplacements, le temps d'exécution, et/ou les à-coups sur la commande (changement rapide de direction/vitesse).

Multi-contrôleurs distribués

Il est alors possible de distribuer une architecture multi-contrôleurs sur chaque robot évoluant dans des environnements multi-robots [Dudek 96, Fierro 01]. Dans la mesure où les autres robots sont détectés comme des obstacles mobiles (position et vitesse), il est possible d'extrapoler un contrôleur d'évitement pour éviter des obstacles mobiles. Par contre, l'étude de la stabilité du système est plus compliquée. Il faut garantir que la somme des déplacements individuels ne conduira pas le système dans un état de blocage où les robots vont s'empêcher, les uns les autres, d'atteindre leurs positions cibles.

Il est aussi possible de proposer des contrôleurs spécifiques permettant des déplacements dans un environnement multi-robots. Ces nouveaux contrôleurs peuvent aussi s'appuyer sur la

définition de rôles différents entre les robots. Ainsi, il est possible d'intégrer des règles simples et locales de suivi de leader, de priorité à gauche ou à droite ou de déviation par rapport à un groupe, etc.

Les architectures multi-contrôleurs élémentaires distribuées ont été particulièrement étudiées dans le cadre de missions de navigation en convois [Benzerrouk 11]. Une flotte de robots doit alors atteindre une position tout en maintenant une certaine géométrie dans les positions respectives des robots les uns par rapport aux autres. Il faut alors commuter entre trois contrôleurs élémentaires : attraction, évitement et maintien en formation. Le maintien en formation peut être défini avec [Gustavi 08] ou sans [Benzerrouk 10] une hiérarchie entre les robots.

1.2.2 Champs de potentiels

Dans le cadre du contrôle des déplacements d'un robot mobile par une architecture multi-contrôleurs, le problème de fusion traite essentiellement de la fusion entre attraction et évitement de façon à définir une commande appropriée. Les méthodes basées sur les champs de potentiels [Khatib 86, Arkin 87] permettent de définir un chemin lisse jusqu'à une cible dans un environnement encombré. Ces méthodes considèrent le robot comme un point soumis à un champ de forces artificiel construit en connaissant les positions respectives de la cible et des obstacles par rapport à la position du robot.

La cible agit comme une force attractive alors que les obstacles déploient des forces répulsives. Ces forces virtuelles sont appliquées sur le robot et celui-ci sera poussé par la résultante de ces forces. De façon à produire un champ de force cohérent, la force d'attraction doit être présente dans tout l'espace alors que les forces de répulsion vont décroître en s'éloignant de l'obstacle (Figure 1.15, 1.16). De cette façon, les forces de répulsion vont masquer complètement la force d'attraction si le robot est trop proche d'un obstacle alors que les forces de répulsion des obstacles lointains n'influenceront pas la trajectoire du robot.

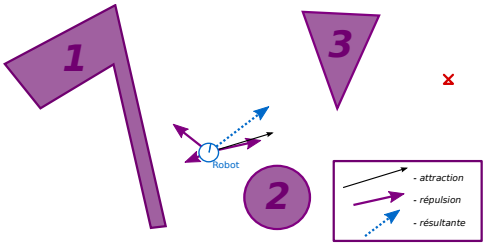


FIGURE 1.15 – Application des forces virtuelles sur le robot mobile

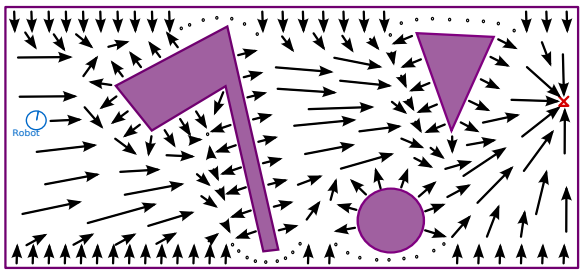


FIGURE 1.16 – Champ de force simple pour l'ensemble de l'environnement

Propagation itérative

Pour que le robot puisse trouver son chemin dans un environnement de type labyrinthe le champ de potentiels peut être construit de façon itérative [Barraquand 92]. Le champ de potentiels est calculé pour chacun des points de l'espace en fonction des forces environnantes proches. La force d'attraction sera propagée de proche en proche. Le champ de potentiels peut être décomposé avec des points d'attractions intermédiaires influençant des zones différentes (Figures 1.17 - 1.21). De cette façon la direction à prendre par le robot n'est pas définie directement par la cible mais bien par le passage le plus proche conduisant à la cible.

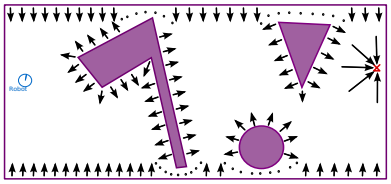


FIGURE 1.17 – Champ de force initial

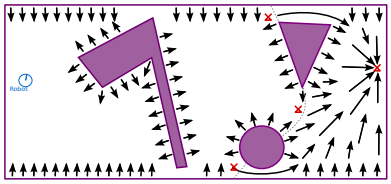


FIGURE 1.18 – Champ de force après propagation dans la première zone

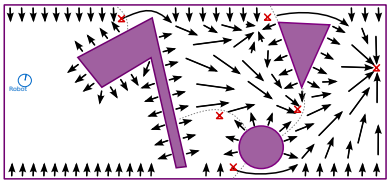


FIGURE 1.19 – Champ de force après propagation dans la seconde zone

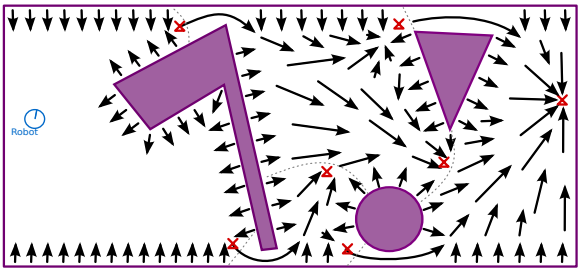


FIGURE 1.20 – Champ de force après propagation dans la troisième zone

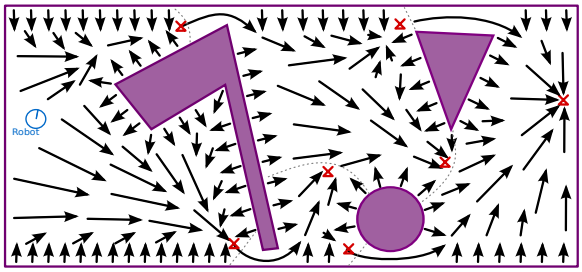


FIGURE 1.21 – Champ de force total après propagation.

Forces et limites des champs de potentiels

Un contrôle basé sur les champs de potentiels permet de déplacer un unique robot vers une cible. Contrôler un robot comme une particule dans un champ de forces virtuel permet d'apporter une certaine souplesse à la commande. Le robot va suivre une trajectoire fluide. Pour garantir que le robot atteigne bien sa cible, il est nécessaire de pré-calculer un champ de force itératif intégrant la connaissance de tous les obstacles. Cette méthode permet de conduire le robot sur un chemin court et sûr (à une distance raisonnable des obstacles).

Dans le cadre d'une mission multi-robots, il faudra calculer un champ de force par robot. Chaque robot doit alors intégrer, dans son champ de force, des forces de répulsion par rapport aux positions des autres robots. Cette solution est mal adaptée pour planifier un déplacement dans un environnement dynamique où les forces de répulsion évoluent dans le temps. En effet, du fait des déplacements des robots, les champs de force individuels devront être actualisés à chaque instant.

1.2.3 Planification de chemin

Nous avons vu qu'il était possible de diriger le robot d'une configuration initiale vers une configuration cible. L'objectif d'une planification de déplacement consiste à définir la succession des configurations intermédiaires qui permettront à un robot de rejoindre une position cible distante dans un environnement encombré.

La planification peut s'effectuer de façon réactive, en cherchant, à chaque instant, uniquement la prochaine configuration cible dans l'espace donnée par la perception courante ou de façon délibérative en établissant l'ensemble des configurations intermédiaires. Une planification délibérative dans un cadre multi-robots doit tenir compte des déplacements des autres robots de façon à ce que les successions de configurations intermédiaires de tous les robots ne génèrent pas de collisions.

Connexions, Chemin et Trajectoire

La planification d'un déplacement peut se faire selon plusieurs niveaux de granularité. Il est possible de définir une connexion, un chemin et une trajectoire de façon incrémentale (Figure 1.22). Une connexion correspond à l'existence d'un chemin reliant deux points. L'information contenue dans la connexion ne permet pas, directement, de contrôler le robot. Le robot doit s'appuyer sur sa perception embarquée pour naviguer entre deux positions connectées. Un chemin quant à lui, prend en considération la configuration de l'environnement et des obstacles. La succession de points de passage définie par un chemin permet directement de naviguer en évitant les obstacles. Le chemin est orienté et les points de passage doivent être atteignables par le robot. Une trajectoire est une définition beaucoup plus précise du déplacement du robot, les points de passage vont contenir une information temporelle plus riche (orientation, vitesse du robot, accélération, etc) permettant d'en déduire directement la commande à appliquer au robot [Laumond 89, Morette 11].

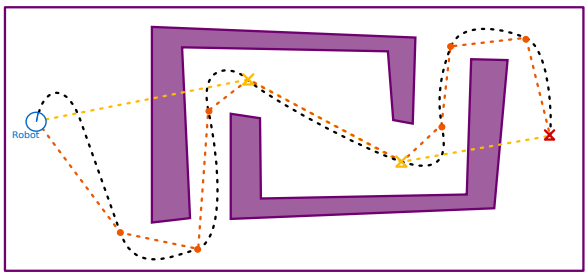


FIGURE 1.22 – Exemple de connexion (jaune), chemin (orange) et trajectoire (noir) reliant deux positions dans un environnement encombré

Dans la mesure où les points de passages du chemin sont deux à deux directement atteignables, il n'est pas nécessaire de pré-calculer la trajectoire à suivre. En effet, il est difficile de pré-calculer avec précision, une trajectoire. Pour la plupart des robots, la trajectoire dépend de leur modèle cinématique, de leur modèle dynamique et des possibilités de commandes qui en découlent.

La plupart des approches de planification de déplacement cherche à définir le chemin sous forme de succession de points de passage, à suivre jusqu'à la cible. Dans un deuxième temps, il reste possible de calculer une trajectoire à partir d'un chemin. La planification consiste à construire un ensemble des chemins possibles puis à en sélectionner un. L'ensemble des chemins forme un graphe des déplacements. Les nœuds correspondent aux points de passage et un arc est ajouté chaque fois qu'il est possible, pour le robot, de passer d'un nœud à un autre.

Le chemin optimal dans le graphe sera alors le chemin qui permet d'atteindre une cible en optimisant un critère donné. Ce critère peut alors être, au besoin, le chemin avec la distance la plus courte, ou le chemin minimisant les changements de direction (la commande en rotation) ou encore le chemin le plus sûr, le plus rapide... De nombreux algorithmes basés sur la théorie des graphes permettent de trouver le chemin optimal dans un graphe pondéré [Hart 68, Cormen 01, West 01].

La construction du graphe de déplacement définit l'ensemble des chemins possibles. Il peut être construit de différentes manières. Cela consiste en une discrétisation de l'espace en un ensemble fini de points de passage particuliers puis en la vérification de l'existence ou non, d'une connexion entre chacun des points de passage.

Construction du graphe de déplacements

La première solution pour la construction d'un graphe de déplacement, consiste à fabriquer manuellement le graphe. Un opérateur expert sur la base de sa propre connaissance de l'environnement cible va établir lui-même les chemins que peut parcourir le robot. Cette solution est exploitée dans le cadre de l'apprentissage supervisé, où l'opérateur conduit, une première fois, le robot en lui montrant ainsi les déplacements autorisés.

Les solutions automatiques les plus simples dérivent d'une discrétisation régulière de l'espace (Figure 1.23). Le graphe est défini en plaçant les points de passage à une distance régulière les uns des autres de façon à former une grille. Ensuite les nœuds sont connectés les uns avec les autres si deux positions sont suffisamment proches et si le passage entre elles est suffisamment éloigné de tout obstacle (Figure 1.24).

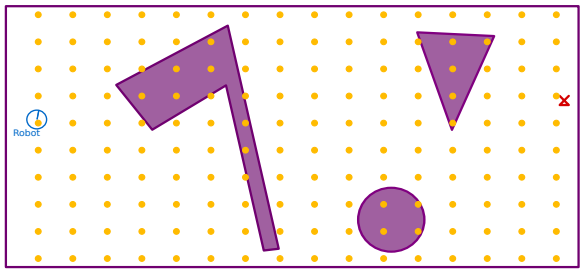


FIGURE 1.23 – Définition régulière des points de passage

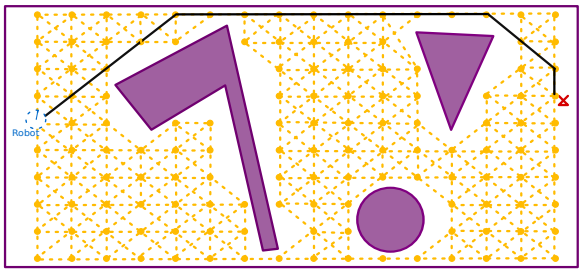


FIGURE 1.24 – Construction du graphe de déplacement

Un graphe généré par une discrétisation régulière va contenir beaucoup de points de passage, ce qui va alourdir les recherches de chemins optimaux. Il est toujours possible, dans un second temps, de supprimer les points de passage non significatifs mais autant placer directement ces points en fonction de la configuration des obstacles.

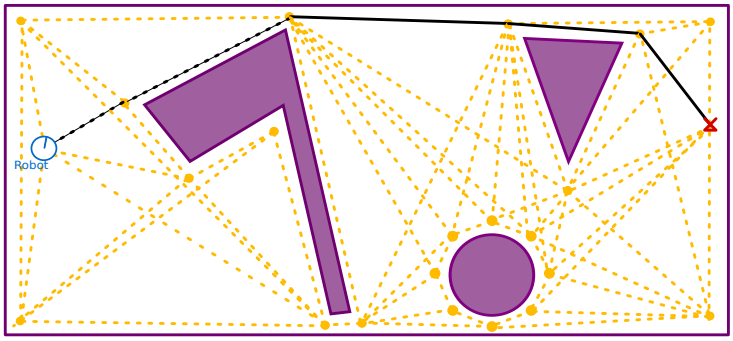


FIGURE 1.25 – Chemin dans le graphe de visibilité

Un graphe de visibilité (Figure 1.25) peut être construit en plaçant les points de passage à chaque extrémité de tous les obstacles et en considérant une distance minimale de sécurité. Les connexions sont ensuite créées de façon à ce qu'elles n'intersectent pas d'obstacles. La construction d'un graphe de visibilité pour le contrôle de robot mobile peut s'effectuer en englobant les obstacles par des polygones, en grossissant ces polygones pour respecter les distances de sécurité et enfin, en connectant entre eux les sommets visibles deux à deux.

Il est aussi possible de contrôler le robot en cherchant à le tenir le plus éloigné possible des obstacles. Le graphe de Voronoï (Figure 1.26) est construit en considérant l'ensemble des points

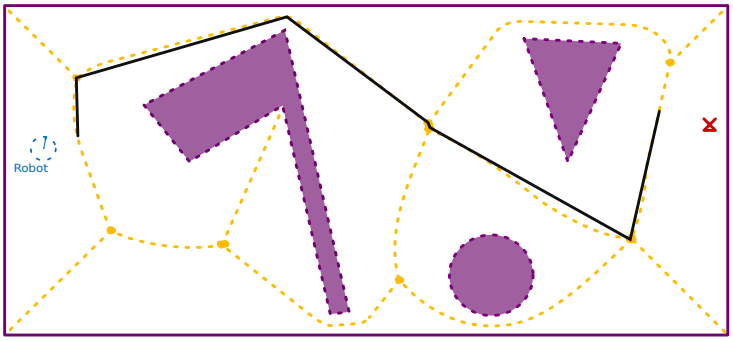


FIGURE 1.26 – Chemin dans le graphe de Voronoï [Choset 95a, Choset 95b]

à équidistance des deux obstacles les plus proches. Les nœuds du graphe sont alors composés de l'ensemble des positions de l'espace qui sont à équidistance d'au moins trois obstacles voisins [Choset 95a, Choset 95b].

Enfin, il existe des solutions basées sur une génération aléatoire de chemins [Kavraki 96]. Les algorithmes RRT (Rapidly-exploring Random Trees) [LaValle 00, Guitton 09] vont générer un certain nombre de chemins sous forme d'un arbre à partir d'une position initiale jusqu'à ce qu'une branche atteigne la cible. Les chemins sont générés en fonction des mouvements possibles du robot.

Les graphes de déplacement sont construits sur la base d'une connaissance de la configuration des obstacles. Leur utilisation est valide dans la mesure où le robot a connaissance, à chaque instant, de sa position dans le graphe. Ces hypothèses théoriques ne sont pas toujours vérifiables en pratique lors de l'utilisation de robots mobiles. La première limitation est due au contrôle erroné des mouvements du robot basé sur des modèles imparfaits pour la commande. La seconde limitation découle des capacités perceptives liées aux capteurs physiques mis en place : la technologie ne va permettre de construire qu'une connaissance partielle et imprécise de l'environnement.

1.3 Incertitudes en robotique

Les robots sont capables de se déplacer dans un environnement dans la mesure où ils connaissent les positions de leurs cibles respectives relativement à leur position courante et l'ensemble des obstacles entre eux et les cibles. Des conditions parfaites sont définies lorsqu'aucune différence entre la commande et les effets réels n'existe, une connaissance précise de sa position est possible à chaque instant par rapport à la cible et par rapport aux obstacles et lorsqu'une communication efficace est possible à tout instant. Du fait d'évoluer dans un environnement réel, ces conditions ne sont souvent pas respectées ce qui conduit à un certain nombre d'incertitudes lors du contrôle d'un robot mobile.

Ces incertitudes sont dues : à l'imprécision des mesures des capteurs ; au manque de connaissance sur l'environnement et au non contrôle d'événements extérieurs aux robots. Tout cela dépend des systèmes mis en place et de l'environnement dans lequel doit évoluer le robot mobile.

1.3.1 Environnements réels

Un des grands challenges actuels, comme l'exploration planétaire, consiste à permettre aux robots d'évoluer dans un environnement sans qu'il n'y ait besoin d'adapter l'environnement pour les robots, c'est-à-dire que les robots se suffisent à eux-mêmes pour réaliser la mission. Dans ces conditions, garantir la réalisation d'une tâche demandée dans les conditions de sécurité et d'efficacité voulues nécessite de considérer l'environnement tel qu'il est et tel qu'il peut être perçu par des capteurs embarqués.

Types d'environnement

En fonction de l'application, le robot va agir dans des environnements cibles différents. Ainsi, un aspirateur autonome évolue dans un appartement, une tondeuse sur une pelouse et un taxi robotisé se déplacera sur le réseau routier. Plus l'application s'éloigne d'un environnement structuré et statique plus elle induit des difficultés pour la mise en œuvre des robots mobiles.

L'environnement peut être plus ou moins structuré : un environnement adapté est enrichi d'éléments spécifiques facilitant la localisation du robot et l'identification des éléments de l'environnement ou des chemins à suivre [Wurman 08]. Un environnement intérieur profite d'une structuration conçue initialement pour l'homme. Ce type d'environnement est caractérisé par un sol quasiment plat. Les murs droits facilitent la détection de l'espace navigable et les autres obstacles seront facilement détectés avec un simple télémètre plan positionné en parallèle au sol [Elfe 87].

Il existe ensuite plusieurs types d'environnements extérieurs. Les environnements urbains (à l'èchelle piéton ou automobile) restent encore très structurés mais avec des configurations possibles beaucoup plus variées que ce que l'on peut trouver dans des environnements intérieurs. Le sol ne sera pas forcément plat et homogène. Les délimitations de l'espace navigable du robot ne sont pas forcément évidentes. Vient ensuite les environnements agricoles qui restent façonnés par l'homme mais beaucoup moins pratiques pour des systèmes robotisés (sol cabossé, glissant). Enfin, les environnements naturels offrent le challenge le plus difficile. Rendre un robot tout terrain autonome sous entend (en plus d'arriver à le contrôler) de définir ce que peut être son espace navigable et de le détecter.

L'environnement d'un robot mobile est aussi caractérisé par sa dynamique. La configuration des éléments qui composent l'environnement va pouvoir être modifiée par des événements dus au système robotisé ou à des agents extérieurs. Dans le cadre du déplacement d'un mobile, la configuration des obstacles va pouvoir changer dans le temps. L'environnement peut être qualifié avec plusieurs degrés de dynamique (faiblement à fortement dynamique) en fonction de la fréquence et de l'importance (vitesse, durée) des modifications de l'environnement émanant d'événements extérieurs aux robots.

La dynamique de l'environnement est la plupart du temps due à la présence d'autres agents humains, animaux, ou robotisés. Ces autres agents vont développer différents comportements animés par des objectifs particuliers. Ainsi, l'environnement pourra être aussi qualifié de coopératif, s'il y a correspondance entre les objectifs du système robotisé et les objectifs des autres agents, neutre si les objectifs ne sont ni communs ni antinomiques ou encore hostile si des agents vont chercher à empêcher le robot d'atteindre ses objectifs.

Déviation du déplacement

A partir de la commande du robot mobile depuis sa position initiale, il est possible de déduire sa position à un instant t. Cette déduction se fait en intégrant la somme des déplacements

commandés. En raison des erreurs de déplacements émanant d'une commande imprécise et d'un environnement imparfait, la position déduite ne pourra être qu'approximative. Dans le cas d'un véhicule, les vitesses appliquées par la commande, les vitesses observées et les vitesses réelles seront toutes les trois différentes. Le robot dévie immanquablement de la trajectoire attendue.

L'odométrie consiste en la mesure du déplacement effectué par le robot entre deux instants t et $t+1$, elle sera plus ou moins précise en fonction des méthodes utilisées. La mesure odométrique est par nature erronée. Dans le cas d'un robot non-holonome commandé en vitesse, la position du robot à l'instant $t+1$ sera comprise dans un espace construit relativement à l'erreur possible sur la vitesse linéaire v_t et sur la vitesse angulaire w_t (Figure 1.27). De la même façon, l'orientation réelle du robot à l'instant $t+1$ ne pourra être donnée que dans une fourchette relative à l'erreur possible sur la vitesse de rotation. A l'instant $t+2$, l'intervalle des poses (positions et orientations) possibles dépend des erreurs sur les vitesses v_{t+1} et w_{t+1} à partir de l'ensemble des possibilités calculées à $t+1$ sur la base des erreurs sur v_t et w_t. Ainsi, l'erreur de mesure de localisation sur la seule base de l'odométrie se propage au fur et à mesure du déplacement du robot.

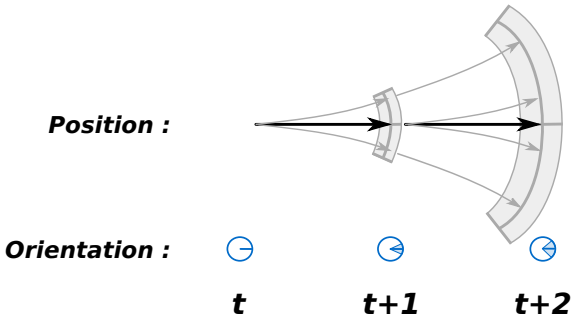


FIGURE 1.27 – Exemple de déviation possible lors d'un déplacement linéaire

L'intervalle des poses possibles après un déplacement donné n'est pas homogène. C'est-à-dire que la probabilité de dévier d'une trajectoire vers une autre est de plus en plus faible au fur et à mesure que l'on s'écarte de la trajectoire supposée. Dans le livre "Autonomous Mobile Robot" [Siegwart 05], est présenté un modèle de l'espace des poses possibles basé sur une ellipse. L'ellipse est définie par deux rayons et une orientation, elle va grossir et tourner au fur et à mesure que le robot se déplace de façon à maintenir une forme simple et cohérente de la croyance du robot.

Dans le livre "Probabilistic Robotic" [Thrun 05], une approche permet de maintenir cette croyance sous la forme d'un espace discret des poses possibles. Avec cette méthode, on ne cherche plus à maintenir une forme géométrique de l'espace de croyance mais à considérer une population finie et dénombrable de poses (positions et orientations) générées récursivement.

Il est possible aussi de maintenir une croyance sur la position du robot dans l'environnement sur la base d'un graphe des déplacements connus. Les positions et les orientations possibles ne seront pas générés à partir de la position initiale. Seule les poses données par le graphe sont considérées possibles. La technique consiste à maintenir une population de probabilités distribuées sur chacun des nœuds du graphe. Cette technique est largement utilisée dans le cadre d'un graphe basé sur une discrétisation régulière de l'espace [Simmons 95].

Maintenir une connaissance cohérente de la position du robot basée sur l'odométrie impose donc de caractériser la déviation possible. Le robot doit alors évaluer les bornes sur les erreurs des vitesses linéaires et angulaires observées. Ces erreurs ne sont pas constantes. Elles dépendent de l'état du robot, des vitesses appliquées et du type de sol sur lequel le robot évolue. Maintenir une telle localisation ne peut suffire que pour des déplacements sur une distance limitée. Il faudra

ensuite se référer à une perception autre pour permettre au robot de se localiser dans un repère absolu et de corriger son odométrie.

1.3.2 Capacité perceptive

L'objectif de la perception pour un robot mobile autonome est donc de pouvoir maintenir une connaissance sur sa position ainsi qu'une connaissance sur les éléments composant l'environnement. Cette connaissance est construite par le biais des capteurs qui vont mesurer des grandeurs physiques avec une certaine précision. Les capteurs les plus courants travaillent sur la lumière, le son, les ondes radio, la gravité ou les forces de contact. Les mesures dépendent alors de la sensibilité et de la position du capteur dans l'espace.

Sur la base des mesures données par les capteurs, le roboticien met en place des modèles permettant donc de caractériser les éléments à portée des capteurs. La perception du robot est capable de donner une description des éléments alentour dans la limite de ces capacités capteurs. Dans le cadre d'un déplacement autonome on s'intéresse en premier lieu à définir l'espace navigable délimité par les obstacles.

Perception par télémétrie

Les capteurs de type télémètre permettent de mesurer des distances avec des surfaces solides. Les télémètres les plus répandus sont de type laser ou ultrason. La mesure donne une distance entre la source et un point d'impact dans une direction donnée. De cette façon il est possible de constituer un nuage de points défini dans un repère attaché au capteur. Cette détection des obstacles est limitée par les obstructions. En effet, un obstacle détecté va cacher les obstacles positionnés derrière lui. La mesure télémétrique est aussi limitée dans l'espace, les points d'impacts ne sont donnés que dans les directions admises par le capteur et dans la limite d'une certaine distance.

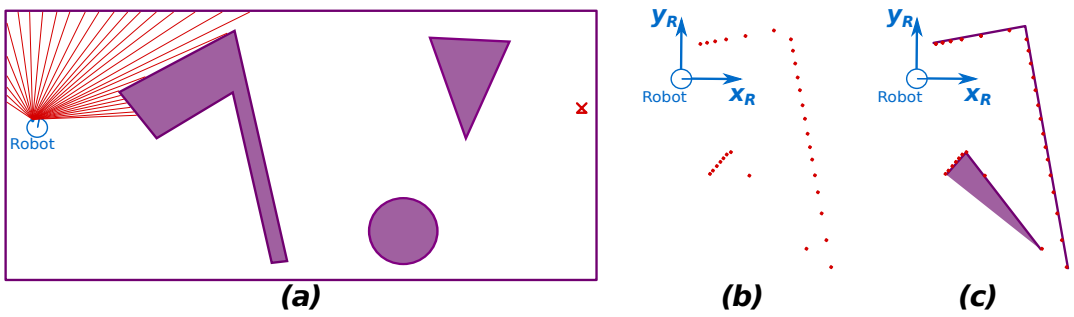


FIGURE 1.28 – (a) exemple de mesures perçues par un capteur télémétrique par rapport (b) à sa position dans l'environnement et (c) la forme géométrique des obstacles sensée être reconstituée à partir de la donnée.

D'autre part la caractérisation des obstacles comprend des erreurs de mesure et des erreurs de modèle. Les erreurs de mesure vont varier avec les conditions d'utilisation, les lasers sont connus pour être plus précis en intérieur (dû à la luminosité) et à courte distance. Les erreurs de modèle sont définies par la différence entre la nature du capteur et le modèle qui permet d'en extraire des connaissances. Les lasers, par exemple, sont sensibles au type de surface et à l'inclinaison entre la surface et le rayon si bien que, le point d'impact peut se perdre et la mesure sera incohérente avec la réalité.

Perception par vision

Les caméras forment le deuxième type de capteurs le plus utilisé en robotique mobile. L'information contenue dans une image est riche mais non exploitable directement. Un seul capteur caméra va pouvoir couvrir toute une zone sans discontinuité. Par contre l'image ne donne pas directement une mesure métrique des délimitations des obstacles et de leurs positions.

L'image retourne des mesures de colorimétrie selon deux dimensions. Reconstruire la troisième dimension peut être fait de façon adhoc. Ainsi, si l'on connaît les paramètres de la caméra et sa position par rapport à un sol plan il est possible d'évaluer la distance d'un objet posé sur le sol [Haralick 92]. Pour un robot mobile dans un environnement structuré il est possible de détecter le sol de par sa couleur et d'en déduire l'espace navigable. En considérant donc cet espace navigable comme plan, il est possible d'en déduire la configuration des obstacles dans le repère relatif au robot [Merveilleux 10].

La troisième dimension peut être reconstruite de façon plus robuste, en utilisant un système de caméra stéréo à champ recouvrant. A partir des différences entre deux images relatives à deux sources différentes mais orientées vers une même cible, il est possible d'évaluer la profondeur de certains pixels par triangulation. Les algorithmes se basent sur des zones remarquables de l'image qui pourront être mises en concordance dans les deux images. Cette technique peut être appliquée à un système à une seule caméra en déplacement où la 3ème dimension est reconstruite à partir de la succession d'images.

La perception basée vision a les mêmes limites que les télémètres standards. Elle est limitée par obstruction, dans une zone donnée et avec une précision qui décroît avec la distance entre la source et les surfaces observées. La détection de surfaces et de zones particulières est sensible aux ombres, et aux reflets.

Il est possible d'extraire de l'image plus d'informations sur les éléments partageant l'espace du robot mobile. La reconnaissance d'objets dans l'image permet de classifier les éléments qui sont aux alentours. Sur cette base, le suivi d'objet dans l'image permet de différencier les objets en mouvement des autres dans un environnement dynamique. Le robot va pouvoir se localiser avec les éléments statiques, observer les obstacles en mouvement et évaluer leurs vitesses (linéaire et de rotation).

Localisation absolue

La perception embarquée d'un environnement, bien que limitée dans l'espace, peut être riche. L'objectif alors est d'utiliser cette perception pour corriger la localisation basée sur l'odométrie. L'odométrie permet de développer un ensemble des poses (positions et orientations) possibles et la perception caractérise l'environnement par rapport à la pose réelle du robot. Dans la mesure où l'on a des connaissances sur ce qui peut être observé en fonction de la position du robot, il est possible pour le robot de se re-localiser.

A partir de la détection de l'espace navigable sous la forme d'un nuage de points d'impacts délimitant ce qui est obstacle de ce qui ne l'est pas, il est donc possible de corriger l'odométrie par différence entre les distances aux obstacles observés et les distances théoriques qui auraient dû être observées. Cette méthode nécessite une connaissance précise de l'environnement permettant la reconstruction des points d'impacts théoriques. Le filtre de Kalman [Kalman 60] permet de corriger la localisation à un instant donné sur la base des observations faites l'instant d'avant. Cette méthode permet une localisation continue pendant le déplacement du robot [Elfe 87, Hébert 96].

Cette correction permet d'affiner l'odométrie mais reste soumise à une erreur d'imprécision qui va aussi se propager au fur et à mesure du déplacement. D'autre part cette méthode est mal

adaptée aux environnements dynamiques où l'évolution de la configuration de l'espace navigable n'est pas uniquement due au mouvement du robot.

Les solutions de reconnaissance de lieux permettent de pallier à la déviation d'une localisation continue du robot (quelques exemples : [Thrun 98, Nehmzow 00, Lisien 03, Korrapati 12]). Ponctuellement, le robot est capable de faire correspondre sa perception courante avec la connaissance qu'il a de l'environnement. La perception, quelle que soit son type (télèmétrie, vision) fournit un tuple de mesures, à un instant donné. Il est alors possible de faire correspondre ce tuple avec l'ensemble des tuples mémorisés de façon à trouver les positions dans l'environnement qui fournirait une perception similaire.

De nouveau, une localisation robuste sous-entend l'élaboration d'une base de données conséquente et particulière aux capteurs embarqués par le robot. Pour essayer de pallier à ce travail fastidieux, il est possible de définir une re-localisation sémantique. C'est-à-dire que, à partir de l'ensemble des mesures perçues, le robot va chercher à caractériser son environnement immédiat. Par exemple, dans un environnement intérieur, le robot va chercher à définir s'il est dans une pièce, dans un couloir ou encore sur une intersection [Kuipers 00]. La mémoire référence non plus directement des ensembles de mesures perceptives mais elle référence des caractéristiques locales. Une localisation sémantique ne permet pas directement une re-localisation métrique ; par contre, la connaissance sémantique pourra être plus facilement partagée entre plusieurs agents hétérogènes. La mémoire peut alors se rapprocher d'une ontologie spatiale qui va faciliter le dialogue entre l'opérateur humain et le robot [Belouaer 11].

1.3.3 Représenter l'environnement, les cartes

Se déplacer et percevoir son environnement n'est pas chose aisée pour des robots mobiles. Les approches utilisées influencent l'organisation de la mémoire spatiale des robots, donc leurs cartes. Nous avons vu qu'un contrôle réactif se passe de carte et qu'un contrôle délibératif s'accommode de graphes de types différents. La carte va donc être surtout liée aux approches perceptives utilisées pour la localisation absolue des robots dans leur environnement. Elle va servir à organiser la mémorisation des mesures effectuées et/ou des caractéristiques déduites de façon à suivre virtuellement le déplacement réel du robot mobile.

Il existe deux familles de cartes : les cartes sous forme de représentation euclidienne et les cartes sous forme de représentation causale. La première famille vise à la mise en place d'une carte métrique qui organise une connaissance géométrique des éléments composant l'espace du robot. La seconde famille organise les successions de perceptions en fonction des déplacements réalisés sous forme de carte topologique. La carte topologique est alors un graphe où les nœuds représentent des perceptions particulières et les arcs décrivent les déplacements qui vont permettre de passer d'une perception particulière à une autre.

Carte métrique

La plupart des cartes métriques utilisées en robotique mobile définissent l'espace navigable total d'un environnement dans une représentation unifiée euclidienne de l'espace. Cet espace est défini à chaque instant par les télémètres (vision, laser, etc) qui équipent le robot, comme un ensemble des points d'impacts modélisant la limite des obstacles locaux. L'objectif d'une carte métrique est alors de fusionner dans un repère absolu, l'ensemble de ces points d'impacts relatifs au robot [Elfe 87, Thrun 98]. La carte permet en retour de prédire les points d'impacts théoriques locaux à partir d'une pose (position/orientation) supposée du robot.

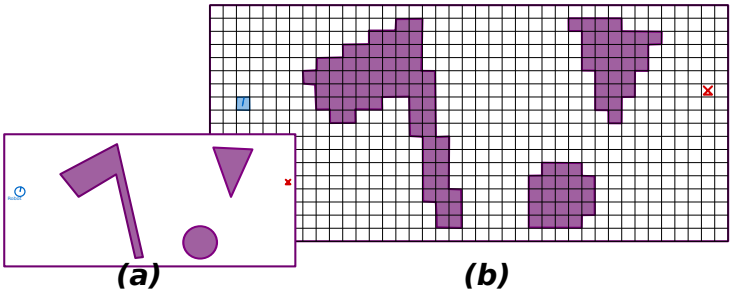
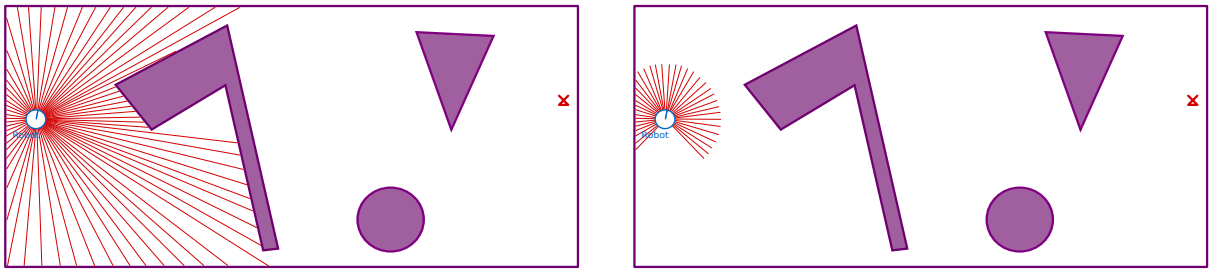


FIGURE 1.29 – Illustration d'une grille d'occupation (b) à partir d'un environnement modélisé en (a)

De nombreux travaux utilisent une grille d'occupation (Figure 1.29) comme carte métrique de l'environnement [Elfe 87, Thrun 98, Matignon 12]. Chaque case de la grille correspond à un espace élémentaire coloré comme obstacle, navigable ou inconnu. La cartographie en grille d'occupation consiste alors à partitionner l'ensemble des cellules de la carte en obstacle ou navigable. Les grilles d'occupation ont montré des résultats convaincants pour une localisation et une cartographie simultanées avec une perception consistante dans un environnement statique.



(a) (b)

FIGURE 1.30 – (a) une perception consistante permettant une localisation absolue précise du robot mobile. (b) une perception ne permettant qu'une correction partielle de la localisation odométrique.

Une perception va être considérée consistante (Figure 1.30(a)) si quelle que soit la position du robot dans l'environnement, il existe des particularités dans la configuration de l'espace navigable permettant de re-positionner avec précision le robot dans la carte entre deux instants. Inversement s'il existe des zones dans l'environnement où la perception est monotone (mesures identiques à une erreur près) alors la correction de l'odométrie restera limitée dans ces zones (Figure 1.30(b)). Il est plus facile pour le robot (comme pour un humain) d'évaluer la longueur d'un couloir homogène s'il perçoit simultanément les deux extrémités plutôt que s'il doit parcourir ce couloir à partir d'une extrémité jusqu'à l'autre.

Les grilles d'occupation permettent une localisation précise au prix d'une carte riche en données et donc assez lourde. Ce modèle de carte est souvent utilisé conjointement à un contrôle pas à pas ou basé sur les champs de potentiels. Un nœud du graphe de déplacement est attaché à chaque cellule de la grille. La commande est pré-définie pour chaque cellule et actualisée chaque fois que la carte est mise à jour.

De part sa définition plane, la taille mémoire, et les erreurs de localisation, les grilles d'occupation, utilisées seules, induisent des déformations des obstacles sur la carte construite [Thrun 98]. Ces déformations limitent l'utilisation des grilles d'occupation si l'environnement est grand, en 3

dimensions (e.g., avec des pentes, des ponts, etc) et ouvert. Ouvert, c'est-à-dire que la perception de l'espace navigable autour du robot est homogène dans certaines zones de l'environnement.

Carte topologique

Une carte topologique (Figure 1.31) organise la connaissance de l'environnement sous la forme d'un graphe des perceptions locales [Kuipers 91]. Chaque nœud de la carte topologique va correspondre à une mémorisation d'éléments de la perception pour un instant donné dans la navigation du robot. Les nœuds vont être connectés les uns aux autres par des arcs dans la mesure où il existe une connexion directe entre deux nœuds, c'est-à-dire que le robot peut se déplacer d'un nœud à l'autre sans passer par un nœud tiers. Les arcs incluent la définition du contrôle à appliquer pour passer d'un nœud topologique à l'autre. Le contrôle est alors défini relativement au nœud initial (exemple de contrôle : longer un mur, prendre à droite à l'intersection, avancer de 3 mètres en direction nord-est, etc).

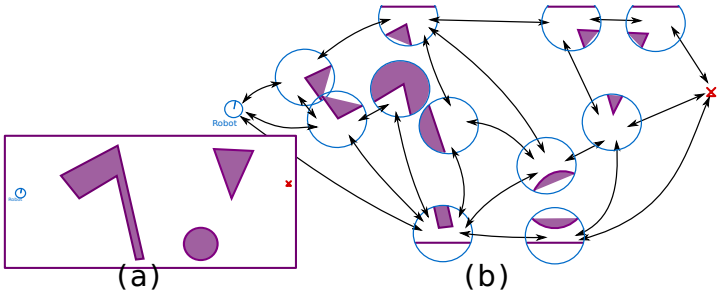


FIGURE 1.31 – Illustration d'une carte topologique (b) à partir d'un environnement modélisé en (a)

L'utilisation d'une carte topologique pure induit une localisation relative. Le robot peut uniquement évaluer quel nœud a été croisé et sur quel arc il est en train de se déplacer. Il existe deux familles principales de cartes topologiques : les topologies référencées perceptions [Nehmzow 00, Choset 01, Courbon 09] et les topologies sémantiques [Kuipers 91, Mataric 92].

Les topologies référencées perception construisent leur graphe directement sur la base des informations données par les capteurs. Cette topologie peut alors être construite sur le Voronoï [Choset 01]. Avec un contrôle basé perception, la commande du robot mobile est définie de façon à minimiser l'erreur entre une perception courante à l'instant t et une perception de référence. La carte topologique définit alors la succession de perceptions de référence qui permettent au robot de se déplacer d'un point à un autre dans l'environnement [Courbon 09].

Les cartes sémantiques se basent sur une perception caractérisée [Kuipers 91, Mataric 92]. Les nœuds sont définis à partir d'une sémantique ponctuelle (e.g., une intersection, un coin, etc) et les arcs définissent des directions à prendre ou des tâches à accomplir (e.g., longer un couloir, etc). Ces cartes sont utilisées dans des environnements structurés comme des bureaux inter-connectés par des couloirs.

Explorer pour construire une carte

En considérant les frontières entre les zones connues et les zones inconnues de la carte, l'objectif de l'exploration traditionnelle consiste à étendre les frontières jusqu'à ce qu'il n'y ait plus de zones inconnues. Dans une grille d'occupation la frontière est matérialisée à l'intersection des

cellules navigables et inconnues. Dans une carte topologique, la frontière est définie par l'ensemble des demi-arcs correspondant à des passages détectés mais non encore connectés à un nœud connu.

La construction d'une grille d'occupation se fait en continu avec le déplacement du robot. A chaque instant, le robot actualise au mieux son positionnement puis enrichit et corrige sa carte [Elfe 87, Thrun 98]. La localisation continue induit une erreur qui se propage avec le déplacement. Cela induit une déformation dans la carte construite lors d'une localisation et d'une cartographie simultanée. L'utilisation de carte topologique non localisée permet de s'affranchir du problème de déviation de l'odométrie en s'appuyant sur la reconnaissance de lieux particuliers. Le challenge consiste alors à créer les nœuds et à fermer les boucles. C'est-à-dire à déterminer si le robot passe deux fois au même endroit [Thrun 98, Nehmzow 00].

Les solutions métriques et topologiques présentent toutes deux des caractéristiques complémentaires ce qui a conduit à la mise en place de cartes hybrides. Une carte topologique enrichie de données métriques sous forme d'une localisation des nœuds dans l'espace euclidien permet d'aider à la reconnaissance de lieux et à délimiter les zones inexplorées [Engelson 92]. Ces techniques sous-entendent de maintenir une localisation cohérente des nœuds lors de la fermeture de boucle. A l'inverse, avec une navigation du robot basée sur une grille d'occupation, la superposition d'une topologie à la grille permet de redresser la carte métrique lors d'une fermeture de boucle [Thrun 98]. Il est aussi possible d'utiliser une carte topologique pour organiser plusieurs grilles d'occupation représentant des zones différentes [Simhon 98].

Dans un cadre de coordination multi-robots, les robots explorateurs autonomes doivent alors choisir à chaque instant une frontière (cellule ou arc) à explorer dans leur carte. Les choix effectués pour un robot doivent prendre en considération les choix effectués par les autres robots. Dans une approche cherchant à optimiser le temps nécessaire à l'exploration totale d'un environnement, le choix de la frontière à repousser à un instant t doit considérer l'ensemble des frontières laissées intactes et qui nécessiteront un retour en arrière. D'un autre côté, s'il existe des restrictions sur la communication, disperser les robots peut conduire à détériorer la transmission de messages. De plus, si l'on cherche à planifier l'exploration, les conséquences liées à la suppression d'une frontière sont soumises à incertitude avec une probabilité de fermer une boucle en arrivant dans une zone connue ou d'ouvrir de nouvelles frontières si de nouveaux nœuds intersections apparaissent.

Au vue de la complexité de modéliser un problème d'optimisation d'exploration multi-robots et étant donné la taille de l'espace de recherche qui découlerait d'une telle modélisation, la plupart des stratégies d'exploration se basent sur un choix de frontière heuristique. Les robots parcourent l'espace selon un chemin pré-défini (e.g. en spirale) ou ils choisissent chacun la frontière la plus proche ou la position qui maximise la taille de la frontière repoussée tout en respectant un minimum de distance avec les autres robots.

Il existe aussi des stratégies d'exploration basées sur des techniques de planification [Matignon 12]. Dans la mesure où les robots possèdent une connaissance même partielle de l'environnement, il est possible d'utiliser des planificateurs pour ordonner l'importance des frontières à un instant t pour chaque robot. Ces méthodes se basent davantage sur des re-planifications régulières chaque fois qu'un robot actualise sa carte plutôt que sur une planification à long terme induisant une énumération exhaustive des possibilités qui, notamment dans le cadre de l'exploration, conduit à des espaces de recherche exponentiels.

1.4 Positionnement

Les robots mobiles sont capables, aujourd'hui, de réaliser des tâches "unitaires" plus ou moins complexes : naviguer, explorer, nettoyer, chercher, manipuler etc. Ce savoir faire s'appuie sur des approches variées qui possèdent leur points forts et leur points faibles. L'autonomie des robots vis à vis de missions plus complexes nécessite alors la combinaison de plusieurs de ces capacités unitaires. Dans le cadre de missions d'exploration multi-robots, un groupe de robots doit construire une carte leur permettant de se localiser et de naviguer dans un environnement ouvert et encombré. La problématique globale s'appuie alors sur les capacités de se déplacer, se localiser, communiquer, fusionner des connaissances, décider, se coordonner, etc...

Cette thèse vise la mise en œuvre de stratégies pour la réalisation de mission multi-robots d'exploration. Si elle est positionnée sur des problématiques de décision, chaque choix que doit réaliser chaque robot dépend de ses capacités et, par conséquent, des approches permettant la réalisation de tâches élémentaires.

1.4.1 Récapitulatif des approches existantes

La problématique de définir des comportements pour une flotte de robots mobiles explorateur est une problématique large. Il existe des approches différentes sur plusieurs aspects de la commande jusqu'à la planification. Ces approches se basent sur des hypothèses plus ou moins restreignantes et apportent des garanties différentes. Dans un premier temps, je proposerai dans cette section, des critères pour classifier les différentes approches en se basant sur les sections précédentes. Ensuite, les approches fondamentales du domaine seront brièvement discutées. Enfin, je présenterai les travaux récents qui permettront de positionner mes travaux de thèse.

Système de robots mobiles

Les approches existantes se différencient vis à vis des techniques de perception et de contrôle utilisées pour déplacer les robots, du type d'environnement ciblé et du modèle permettant la représentation de la connaissance. Je propose donc de classifier les approches selon plusieurs mot-clefs définis sur la base de l'état de l'art présenté dans les sections précédentes :

Perception *globale* ou *relative* : Au pire, le robot n'a accès qu'à des données *relative*s produites par ses propres capteurs : une description de ses propres mouvements et de l'environnement alentour. Certaines approches imposent que le robot ait une connaissance globale (par exemple avec un GPS), il existe alors une borne constante sur l'erreur de localisation.

Contrôle *multi-Contrôleurs*, par *Champ de Potentiels*, par suivi de *chemin* ou de *trajectoire* : il existe plusieurs solutions pour définir la commande à appliquer au robot mobile. Les méthodes *multi-Contrôleurs* et *Champ de Potentiels* permettent, dans une certaine mesure, à plusieurs robots de se déplacer dans un environnement encombré. Un contrôle sur *chemin* ou sur *trajectoire* pré-calculé permet de planifier l'ensemble des déplacements des robots en tenant compte de la configuration des obstacles.

Environnement *adapté*, *intérieur*, *urbain*, *rural* ou *naturel* : Nous avons vu que plus l'environnement est structuré plus il y est possible de fournir des garanties sur la réalisation des tâches par un ou plusieurs robots. D'un autre côté, l'autonomie des robots mobiles se mesure aussi par sa capacité à se déplacer dans des environnements non préparés spécifiquement pour eux.

Dynamisme *statique* ou de dynamique *faible*, *moyenne* ou *forte* : L'environnement se caractérise aussi par l'apparition d'événements non contrôlés par le système. Ainsi, dans un

environnement *statique* seuls les robots se déplacent et modifient l'environnement, avec une dynamique *faible*, *moyenne* ou *forte*, il existe d'autres agents qui vont agir dans l'environnement, qui vont rendre de plus en plus difficile l'utilisation d'une connaissance stable. La problématique de se localiser notamment n'est pas la même si le robot évolue au milieu d'une foule ou dans un hall vide.

Carte *métrique*, métrique en *grille* d'occupation, *topologique*, *double* (grille + topologie) ou *Voronoï* : La représentation de la connaissance est au cœur de la différenciation entre les approches. La plupart des approches utilisent une carte *métrique* sous forme d'une *grille* d'occupation. Une carte *topologique* organise sous forme d'un graphe une information basée sur une sémantique ou basée sur une collection d'amers perceptifs. Enfin, le *Voronoï* permet de construire un graphe sur une métrique basée sur la distance aux obstacles.

Localisation *connue*, *continue* ou *ponctuelle* Sur la base de la représentation choisie et de la présence ou non d'une perception globale (type GPS), la localisation du robot sera supposée *connue*, *continue* ou *ponctuelle*. Ainsi, dans le cadre d'une localisation *continue* le robot, à chaque instant, actualise sa connaissance sur son positionnement par odométrie ou par référencement dans une carte. Cette connaissance peut être enrichie par la reconnaissance de lieux particuliers permettant une re-localisation *ponctuelle*.

Équipe de robots *unique*, *groupe*, *flotte* ou *essaim* : caractérise des approches validées avec : un seul robot (*unique*) ; quelques robots (de 2 à 5) (*groupe*) ; autour d'une dizaine de robots (*flotte*) ou de plusieurs dizaines à plusieurs centaines de robots (*essaim*).

Planification et décision

La mise en œuvre de robots est étudiée dans différents cadres applicatifs. Nous distinguerons 4 cadres applicatifs : la *navigation* où les robots doivent atteindre des positions particulières ; l'*exploration* où les robots cherchent à enrichir leur connaissance ; la *cartographie* est une forme d'exploration où les robots construisent une carte de leur environnement et les missions *interactives* basée sur des jeux entre plusieurs agents (coopération homme-robot, poursuite, déplacement d'éléments de l'environnement). Pour chaque cadre applicatif, les approches se différencient de nouveaux selon plusieurs critères :

Communication *sans*, *efficace*, *connectée* ou *bruitée* : Dans le cadre de mission multi-robots, les approches se passent de communication (*sans*), la considèrent tout le temps *efficace* ou soumise à plus ou moins de contrainte. Considérer une communication *connectée* dans un rayon limité autour du robot suppose qu'aucun message ne sera perdu dans la mesure ou deux robots sont suffisamment proches. Avec une communication *bruitée*, l'approche tient compte du fait que les messages peuvent être que partiellement transmis ou avec un bruit plus ou moins fort.

Transmission *unicast*, *broadcast* ou *blackboard* : Sur la base des contraintes de communication les approches proposent une coordination des robots basée sur une transmission *unicast* ou *broadcast* des messages. Certaines approches vont plus loin en construisant une mémoire partagée sous forme d'un *blackboard* virtuel centralisé sur un robot ou décentralisé sur tous les robots.

Planification *pré-définie*, *heuristique*, *déterministe*, *stochastique* ou *aléatoire* : La planification est de difficulté variable en fonction des types de problèmes abordés. Dans le cadre de la navigation il va être possible de déterminer une solution optimale via une résolution *déterministe* ou *stochastique*. Une planification *stochastique* tient compte de l'aspect incertain

	Perception Contrôle	Environn. Dynamisme	Carte Localisation	Mission Équipe	Communication Transmission	Planification Résolution
[Barraquand 92]	relative	intérieur statique	topologique ponctuelle	navigation unique		stochastique
[Simmons 95]	relative	intérieur statique	topologique ponctuelle	navigation unique		stochastique
[Kavraki 96]	globale potentiel	intérieur statique	métrique continue	navigation unique		
[Thrun 98]	relative	intérieur statique	double continue	cartographie unique		
[Simmons 00]	relative chemin	intérieur statique	grille continue	cartographie flotte	efficaces blackboard	heuristique
[Choset 01]	relative voronoï	intérieur statique	topologique voronoï	cartographie unique		heuristique
[Guo 02]		statique	métrique connue	navigation flotte	efficaces broadcast	déterministe décentralisée
[Zlot 02]	relative chemin	intérieur statique	grille	cartographie flotte	rayon limité broadcast	heuristique vente aux enchères
[Gerkey 02]	relative	intérieur faible		interaction flotte	rayon limité broadcast	heuristique vente aux enchères
[Chades 02]	relative chemin	intérieur statique	grille continue	interaction flotte	rayon limité unicast	stochastique décentralisée

TABLE 1.1 – Classification de travaux de planification pour robots mobiles

de certaines données du problème. Dans le cadre de missions multi-objectifs comme l'exploration il est plus compliqué de définir un plan optimal. Les méthodes *déterministe*s où *stochastique*s effectuent alors des re-planifications régulières pendant le déroulement de la mission. D'autre solutions se basent sur un plan *pré-défini*, sur des recherches (*aléatoire*s) ou calculées de façon *heuristique*.

Résolution *centralisée*, *décentralisée* et plus particulièrement, décentralisée par *ventes aux enchères* : Enfin la planification multi-robots peut se faire de façon *centralisée*, un agent unique calcule et distribue les plans à tous les robots ou de façon *décentralisée*, plusieurs robots se partagent la tâche de planifier. Dans le cadre d'une résolution *décentralisée* de nombreuses solutions adoptent des protocoles de *ventes aux enchères* où les tâches sont alloués un par un, au robot proposant la meilleur offre.

Approches fondatrices en planification des déplacements de robots

Les années 1990 ont vu apparaître les approches qui définissent les fondamentaux de la mise en œuvre de robots mobiles autonomes [CAO 97]. Ainsi, [Elfe 87, Kuipers 91] ont permis à un robot de cartographier un environnement intérieur de façon autonome.

Dans l'objectif d'étendre l'autonomie des robots mobiles, une deuxième vague de travaux va chercher à repousser les limites des approches fondamentales en combinant des solutions entre elles ou en appliquant des méthodes issues de la planification (déterministe ou stochastique) à la robotique mobile (Tableau 1.4.1.0).

Ainsi, il est possible pour un robot de cartographier et naviguer dans un environnement intérieur inconnu en combinant les avantages des approches métriques et topologique avec, par exemple, deux cartes [Thrun 98] ou encore, un graphe de Voronoï [Choset 01]. Plusieurs travaux se concentrent alors sur la navigation du robot de façon à planifier le meilleur chemin (distance, sécurité...) pour atteindre un objectif en respectant les capacités sensori-motrices du robot [Simmons 95, Kavraki 96, Guo 02].

	Perception Contrôle	Environnement Dynamisme	Carte Localisation	Mission Équipe	Communication Transmission	Planification Résolution
[Berhault 03]	global chemin	intérieur statique	grille connue	exploration flotte	rayon limité broadcast	déterministe vente aux enchères
[Burgard 05]	relative chemin	intérieur statique	grille continue	exploration flotte	rayon limité blackboard	heuristique décentralisée
[Tovey 05]	global chemin	intérieur statique	grille connue	navigation équipe	efficaces blackboard	déterministe vente aux enchères
[Rooker 07]	relative chemin	intérieur statique	grille connue	exploration flotte	rayon limité blackboard	heuristique décentralisée
[Franchi 07]		intérieur statique	topologique connue	exploration flotte	rayon limité broadcasts	aléatoire décentralisée
[Vincent 08]	relative	intérieur statique	Double	cartographie essaim	rayon limité unicast	heuristique décentralisée
[Capitan 12]	relative chemin	intérieur statique	grille continue	navigation équipe	rayon limité broadcasts	stochastique décentralisée
[Matignon 12]	relative chemin	intérieur statique	grille continue	cartographie équipe	efficaces broadcasts	stochastique décentralisée

TABLE 1.2 – Classification de travaux récents pour la mise en œuvre de missions multi-robots et multi-objectifs

Ces avancées plurielles ont permis de considérer des problématiques plus complexes avec la mise en œuvre de flottes de robots pour la réalisation de missions impliquant une réalisation de tâches plurielles comme dans le cadre de l'exploration avec ou sans cartographie. Au delà des contraintes ajoutées (communication, fusion de données), les problèmes de planification soulevés ne sont pas solvables par des méthodes de planification optimales. Les premières solutions proposées s'appuient naturellement sur une recherche heuristique distribuée où chaque robot calcule son propre plan [Simmons 00, Guo 02, Gerkey 02, Zlot 02]. La coordination se fait par une allocation des objectifs entre les robots principalement via des protocoles de vente aux enchères. Cette solution n'est alors pas applicable à des problématiques basées sur des interactions nécessitant une certaine synergie dans les actions des robots (e.g. [Chades 02]).

Travaux multi-robots et multi-objectifs récents

Le sujet de cette thèse porte sur les architectures et les stratégies mises en œuvre permettant à une flotte de robots autonomes de réaliser une mission à objectifs multiples. C'est-à-dire que la mission peut être décomposée à chaque instant en plusieurs sous-objectifs réalisables de façon indépendante. L'accent est porté sur l'autonomie des robots vis à vis de l'environnement. L'objectif étant de mettre en place des robots capables de remplir leur mission sur la seule base de leur perception embarquée, sans l'aide d'un opérateur et malgré une connaissance initiale de leur environnement incomplète voire nulle. De nombreux travaux en robotique mobile et en informatique décisionnelle ont apporté des éléments de réponse sur cette problématique (Tableau 1.4.1.0).

On constate que la plupart des approches se base sur une grille d'occupation. Cela s'explique par la maturité de cette technique pour la cartographie et par la cohérence entre une discrétisation régulière et les méthodes de planification traditionnelles. De plus, les cartes métriques (comme les grilles d'occupation) permettent une évaluation simple des actions de déplacement basées sur les distances.

Les grilles d'occupation à la base de ces méthodes impliquent des difficultés pour passer à l'échelle. Les solutions proposées ne permettent pas de considérer un environnement large, plus

ou moins naturel, ouvert et non plan. Un des freins réside dans la taille de la carte créée. D'autre part, l'utilisation de la grille d'occupation telle quelle, génère un espace d'états trop important pour une planification à long terme des actions des robots.

La planification se base alors :

- sur un espace d'états réduit avec des grilles de seulement plusieurs centaines de cellules [Chades 02, Berhault 03],
- sur des calculs constants pour actualiser l'action à réaliser à chaque instant [Vincent 08, Matignon 12],
- des hypothèses fortes sur la perception du robot : le robot connaît sa position à chaque instant [Vincent 08],
- sur des heuristiques sur les comportements individuels (e.g. les déplacements des robots doivent permettre de maintenir une communication efficace [Vincent 08]).

En considérant les nœuds comme des états clefs, les cartes topologiques permettent de modéliser un problème de planification avec un espace d'états raisonnable. Ils est notamment possible de générer les nœuds en ligne, pour planifier un déplacement parmi plusieurs possibilités générées aléatoirement [Franchi 07]. Mais de nouveau, une planification à long terme nécessite un maillage complet de l'environnement appuyé par des mécanismes de navigation et de localisation autonomes.

1.4.2 Contributions

Je pars de l'hypothèse qu'une planification à long terme permet d'améliorer la réalisation d'une mission multi-robots. La mise en place de telle stratégie s'oppose aux approches basées sur une actualisation de l'action à réaliser à chaque instant. D'un autre côté, les rafraîchissements fréquents de la connaissance sont difficilement prévisibles et induisent un espace de possibilités intraitables lors des phases de planification.

La solution proposée ici s'articule autour de trois contributions. Dans un premier temps je propose la mise en place d'une architecture basée sur une carte topologique couplée à des capacités réactives du robot. Cette carte permet de diminuer l'espace d'états considérés lors des phases de planification. Malgré cela, la taille des problèmes reste difficile à traiter. La seconde contribution cherche à calculer une solution approchée basée sur une résolution hiérarchique avec un niveau global et un niveau local. Enfin, dans le cadre de la mise en œuvre de mission multi-robots, une troisième contribution est proposée pour permettre à plusieurs robots de coordonner leurs plans lors des phases de planification.

Hiérarchisation du contrôle d'un robot mobile

Plusieurs contributions ont été apportées en robotique mobile sur la base de cartes topologiques (en navigation, en localisation ou en cartographie). D'un autre côté, il y a peu de travaux en planification qui cherchent à combiner les avantages de ce type de représentation. Une localisation absolue ponctuelle par reconnaissance de lieux couplés à une navigation sur le Voronoï devrait permettre la mise en place d'une carte à la fois légère mais permettant de bonnes garanties pour maintenir la localisation du robot dans son environnement.

La réalisation d'un tel projet implique d'une part, de faire converger plusieurs compétences et d'autre part, de redéfinir le modèle permettant la planification des actions de déplacement. L'idée maîtresse consiste à mettre en place une planification basée sur des capacités fonctionnelles. Un niveau fonctionnel est défini pour permettre aux robots de se déplacer de façon réactive d'un point à un autre dans un environnement encombré. Un niveau délibératif peut ainsi s'abstraire

d'une planification des actions à réaliser à chaque instant pour travailler sur des actions de haut niveau. Il est alors possible d'identifier les états clefs correspondants à des intersections de façon à décider uniquement des actions à réaliser jusqu'à l'état clef suivant.

La première contribution de cette thèse vise à mettre en place une architecture permettant à plusieurs modules de travailler de concert. L'architecture proposée ici s'appuie sur deux niveaux, un niveau réactif et un niveau cognitif. Ainsi, des modules réactifs vont permettre la localisation et la navigation du robot alors que des modules délibératifs vont avoir pour charge la planification et la supervision du niveau réactif.

Des modules réactifs mis en œuvre vont découler des propriétés particulières avec des réussites soumises à incertitude. L'organe maître de l'architecture réside alors dans la carte topologique permettant d'exprimer les capacités de déplacement et de localisation et leurs incertitudes liées. Sur la base de cette connaissance il va alors être possible d'appliquer des techniques de planification traditionnelles. Cette planification ne va plus construire un chemin à suivre tenant compte des obstacles mais plutôt une connexion abstraite définie par des points de passages distants.

L'architecture proposée cherchera alors :

- à garantir le maintien de la localisation d'un robot réel dans une carte erronée lors d'une mission de navigation multi-objectifs (type surveillance),
- à vérifier que la taille de la carte et les fréquences d'actualisation permettent une planification à moyen terme.
- à comparer lors d'une réalisation d'une mission d'exploration le déroulement via une planification à moyen terme avec un déroulement via des stratégies heuristiques simples.

Planification à long terme

Au vu des incertitudes liées à la réalisation d'un déplacement, incertitudes modélisées dans la carte topologique, la méthode de planification proposée s'appuie sur les Processus Décisionnels de Markov (MDP). Malgré l'utilisation d'une carte topologique, l'espace d'états lors des phases de planification reste trop important. Nous verrons qu'il est difficile de réaliser une planification à long terme impliquant la réalisation de plusieurs dizaines d'objectifs indépendants.

La seconde contribution de cette thèse consiste en la proposition d'algorithmes de résolution approchée de MDP basés sur une hiérarchisation sur deux niveaux. Une résolution à un niveau global permet au robot d'évaluer une priorité entre les régions contenant les tâches à réaliser. Ensuite, une résolution locale permet de planifier effectivement les actions de déplacement de façon à réaliser les tâches contenues dans une zone.

L'état de l'art sur les MDP décomposés est riche. Les solutions reposent sur un partitionnement de l'espace d'états couplé à une résolution, région par région, jusqu'à convergence vers une solution globalement optimale. Le problème du partitionnement d'un ensemble en n régions reste difficile et il existe de nombreuses heuristiques concurrentes. Je proposerai un algorithme permettant de partitionner rapidement la carte topologique en vue de la construction d'un MDP décomposé.

Une bonne décomposition d'un MDP permet d'accélérer sa résolution optimale. Je proposerai un nouvel algorithme de résolution permettant d'accélérer encore la convergence, mais vers une solution sous-optimale. L'algorithme proposé se décompose alors en deux phases. Dans une première phase, il construit une politique globale approchée permettant d'ordonnancer l'exploration de chacune des régions. Dans une seconde phase, l'algorithme calcule la politique locale à appliquer à la région en cours. L'heuristique consiste donc à construire, au fur et à mesure de l'évolution du robot, une politique localement optimale qui soit globalement acceptable.

Les expérimentations permettront d'analyser l'approche comparativement à une résolution optimale. L'objectif de l'algorithme présenté est aussi de permettre à des robots d'actualiser leurs politiques en cours de mission. Étant impossible de considérer initialement toutes les modifications qui peuvent avoir lieu dans la carte, un second critère d'évaluation est basé sur la vitesse de convergence vers la solution retenue. En effet l'objectif est de permettre au robot d'actualiser ponctuellement sa politique avec l'évolution de sa connaissance.

Coordination des politiques individuelles

L'objectif suivant est d'appliquer les solutions proposées dans le cadre d'une coordination multi-robots. L'intérêt d'une planification à long terme reste à démontrer lorsque la connaissance est constamment remise en cause. Par contre, en considérant des communications limitées dans un rayon autour de chaque robot, la coordination des robots ne peut se faire que ponctuellement lorsque ceux-ci sont suffisamment proches entre eux pour communiquer.

Contrairement aux approches visant à maintenir la connectivité de communication entre les robots, l'idée est de permettre à chaque robot d'évoluer de manière autonome entre plusieurs phases de communications. Une planification à long terme, basée sur les connaissances de chacun doit alors permettre un meilleur partage des tâches à réaliser en autonomie.

Dans le cadre de phases ponctuelles de coordination, l'objectif est alors d'allouer l'ensemble des tâches connues à réaliser. Je propose dans cette thèse une méthode distribuée permettant une vente aux enchères des tâches avec une évaluation des gains basée sur une planification mono-agent.

L'évaluation des gains dans le cadre d'une planification à long terme dépend de l'ensemble de toutes les tâches allouées à l'agent. Je propose un protocole basé sur une succession de plusieurs ventes aux enchères simultanées. Ainsi, toutes les tâches vont être remises en ventes automatiquement, plusieurs fois jusqu'à stabilisation de l'allocation des tâches. Il est alors prouvé que le processus converge effectivement vers une solution stable.

Le protocole est expérimenté de façon à montrer que la convergence vers une solution sous-optimale est rapide permettant aux robots de se coordonner ponctuellement en cours de mission. De plus, la qualité des solutions construites est mesurée comparativement à l'allocation optimale (calculée de façon centralisée) de façon à évaluer l'efficacité statistiquement de l'approche proposée.

1.5 Conclusion

Nous avons débuté ce chapitre sur le Taylorisme. La découpe scientifique des chaînes de production a permis des gains de productivité et à faciliter l'émergence de la robotique. Cette vision s'appuie sur une planification complète où chaque agent se voit attribuer un rôle défini selon une succession précise d'actions. Il devient alors rapidement impossible de planifier les actions à réaliser à chaque instant pour un groupe de robots et pour tous les cas de figures imaginables dès que le problème sort d'un cadre restreint à quelques états.

A travers la problématique de concevoir une flotte de robots mobiles autonomes, nous avons vu qu'il est difficile d'appliquer une planification optimale des déplacements simultanés possibles de tous les robots pour des missions d'exploration. La réalisation des actions et la perception de chaque robot sont soumises à incertitude. Une connaissance robuste et commune à plusieurs agents est difficile à mettre en œuvre. Cela engendre un espace intraitable des configurations possibles du système à contrôler.

Mon hypothèse de travail repose sur l'idée que, augmenter les capacités d'un système complexe ne peut se faire sans l'augmentation de l'autonomie des éléments composant le système : de l'autonomie pour chaque robot du groupe à travers une architecture distribuée et de l'autonomie pour des capacités fonctionnelles vis à vis d'une planification plus abstraite. Nous même, en prévoyant un déplacement, nous ne planifions pas, pas à pas, notre trajectoire jusqu'à notre cible. Dans une situation normale, nous faisons confiance, sans réfléchir, à notre capacité de marcher. Nous planifions uniquement les passages clefs qui différencient une route plutôt qu'une autre.

A travers mes contributions, pour l'architecture robotique mise en place, je cherche à définir les limites de l'autonomie d'un niveau réactif vis à vis d'un niveau cognitif. Les contributions apportées à la planification individuelle reposent sur une résolution hiérarchique où une solution globale s'abstrait de solutions locales exactes. Ce procédé permet de traiter sous-optimalement et en cours de mission, un problème sans forte restriction sur sa taille. La solution construite doit alors être affinée localement pour être effective. Enfin, pour la coordination multi-agents, l'approche proposée s'appuie sur des procédés de délibérations distribuées où chaque agent est autonome dans la création de son plan particulier et coopératif.

Chapitre 2

Planifier pour explorer

Le Petit Larousse illustré de 1905[6] définit le verbe "*Explorer*" par : "visiter, aller à la découverte (explorer les mers) et au sens figuratif, étudier, scruter, sonder (explorer les sciences)". La définition peut conduire à des activités et des problématiques différentes. Il est possible de différencier ces deux sens comme : l'exploration au sens premier, désignant un déplacement sur une frontière définie entre le connu et l'inconnu, cette exploration permet une acquisition de données nouvelles par l'étude d'éléments inconnus ; et l'exploration au sens figuré, consistant à fouiller une base de connaissances à la recherche de nouveaux savoirs.

En informatique, l'exploration fait davantage écho à une exploration au sens figuré. L'exploration se traduit par des activités de fouille de données et de recherche dans de grands graphes (moteur de recherche Internet). Les données existent déjà, il s'agit de les parcourir efficacement pour en faire ressortir une connaissance ou un élément particulier. En robotique mobile, l'exploration vise plus communément un sens premier avec l'activité de créer une carte.

L'acquisition de données est alors localisée sur des zones inconnues à la limite de la carte connue. L'activité d'exploration en robotique mobile s'accompagne généralement de la création de cartes modélisant les routes praticables (e.g. délimitant l'espace libre et les obstacles). Mais l'exploration peut définir des missions où l'on possède déjà une carte. Ce type d'exploration cible une recherche d'informations particulières (pour actualiser, pour enrichir la carte, pour secourir, pour surveiller). On peut donc distinguer les missions d'exploration en général et les missions de cartographie. A chaque instant d'une mission d'exploration, l'environnement peut être défini par des zones à visiter. L'objectif pour un robot ou pour une flotte de robots consiste donc à couvrir, le plus efficacement possible, les zones inconnues jusqu'à ce qu'il n'y en ait plus.

Dans le cadre de missions multi-robots, les choix individuels doivent prendre en considération la présence de coéquipiers de façon à répartir efficacement les tâches. Les choix individuels reposent donc sur deux problématiques principales : fusionner les connaissances et coordonner les actions. Par nature, chaque robot embarque des capteurs et construit sa connaissance propre. Ainsi, la représentation de l'environnement connu est différente d'un robot à un autre. Le challenge consiste alors à fusionner les connaissances et à coordonner les robots, quand la communication le permet. La fusion permet d'unifier la carte et de localiser les robots. Ensuite, la coordination consiste à allouer à chacun des robots leurs tâches d'exploration [Simmons 00, Zlot 02, Burgard 05, Matignon 12].

Ce chapitre présente la problématique liée à la planification des actions dans le cadre de missions d'exploration. Les modèles et les approches liées à la planification sont brièvement présentés de façon à se concentrer plus particulièrement sur les processus décisionnels de Markov

6. Petit Larousse illustré de 1905 mis en ligne par l'université de Cergy : http ://dictionnaire1905.u-cergy.fr/

(MDP) dans un cadre d'exploration mono-robot puis multi-robots. En effet, nous verrons que les MDPs forment une famille de formalismes expressifs pour modéliser des problèmes de prise de décisions complexes.

2.1 Modélisation pour la planification

La notion d'agent est largement utilisée en Intelligence Artificielle pour modéliser les interactions entre plusieurs éléments d'un même environnement. L'agent est défini comme une entité autonome capable de percevoir et d'agir dans un environnement [Wooldridge 95, Ferber 95]. L'agent est animé par des buts à atteindre. Le robot peut alors être défini comme un agent particulier où la perception et l'action sont définies par ses interfaces avec le monde réel.

Tout un pan de l'Intelligence Artificielle cherche à définir les mécanismes propres à chaque agent, liant les perceptions aux actions. L'ensemble de ces mécanismes définissent le comportement d'un agent. Le comportement de l'agent doit lui permettre d'atteindre ses objectifs. C'est-à-dire que la succession d'actions, guidées par sa perception, doit conduire l'agent vers un état où il est validé que ses objectifs sont atteints ou sont inatteignables.

Le comportement des agents d'un groupe peut être défini sur des approches plus ou moins réactives/délibératives et plus moins centralisées/décentralisées en fonction des ressources mises en œuvre et de l'architecture utilisée entre les agents pour prendre des décisions. Une approche réactive cherche à définir des comportements adéquats tout en minimisant les calculs et la mémoire nécessaires à l'exécution de ces comportements. Dans une approche délibérative, les techniques de planification cherchent à définir le choix des actions à réaliser sur la base d'une évaluation des conséquences de ces choix.

Les conséquences de ces choix sont définies par les actions à venir qui pourraient être exécutées. La planification vise à contrôler les successions possibles d'actions, jusqu'à la réalisation des objectifs de la mission. La planification consiste alors à définir une modélisation permettant d'exprimer l'évolution du système en fonction des actions pouvant être réalisées puis à explorer virtuellement le modèle pour définir un plan d'actions. Enfin, le modèle pour la planification peut (ou non) intégrer une connaissance probabiliste sur l'évolution de l'environnement ; on parlera alors de planification stochastique.

2.1.1 Planification déterministe

Les méthodes classiques de planification consistent à discrétiser l'espace d'états du système de façon à pouvoir itérativement les parcourir en fonction des actions testées [Bellman 57, Fikes 71, Boutilier 96, Buffet 08]. Le principe fondamental consiste en une représentation des actions sous forme d'une fonction de transition $t : S \times A \rightarrow S$. La connaissance sur l'évolution du système s'exprime telle qu'une action a réalisée dans un état du système s conduit le système dans un état s'. Dans le cadre où le système est décrit par un nombre fini d'état, on parlera d'un automate à ensemble fini d'états.

La perception et l'action d'un agent peuvent être caractérisées par deux ensembles finis de variables. À chaque instant, l'agent peut accéder aux variables de perception et définir ses variables d'actions. Pour le robot mobile, les variables de perception et d'action sont données par les mesures capteurs et les possibilités de commande. L'état courant d'un agent peut être défini sur la base des perceptions courantes et d'une mémoire.

La mémoire peut être définie simplement par l'historique des couples (perception, action) jusqu'à l'instant présent. Le nombre d'états d'un agent construit uniquement sur une mémorisation des perceptions et des actions passées est potentiellement infini. Définir un modèle de

l'environnement (une carte pour un robot mobile), permet d'organiser la mémoire de façon à exprimer une connaissance. L'état d'un agent est alors défini par l'union de son état perceptif et de l'état de sa connaissance.

Planifier le comportement d'un agent consiste à définir une fonction $plan : S \rightarrow A$ qui associe une action à réaliser pour chaque état du système de façon à ce que, à partir de l'état initial s_0, l'exécution du plan termine dans un état s où les objectifs sont atteints ($s \in S_{objectifs} \subset S$) [Bellman 57, Fikes 71]. La fonction de validation d'un plan à partir d'un état initial s_0 peut être définie de façon récursive (avec S' l'ensemble des états visités initialement vide $S' = \emptyset$) :

$$valide(s,\ plan,\ S') = \begin{cases} vrai & \text{si} \quad s \in S_{objectifs} \\ faux & \text{si} \quad s \in S' \\ valide(\ t(plan(s)),\ plan,\ S' \cup \{s\}\ ) & \text{sinon} \end{cases}$$

$$\text{et : } valide(s_0,\ plan) = valide(s_0,\ plan,\ \emptyset)$$

Ainsi, à chaque itération l'ensemble S' des états visités est incrémenté d'un état. Dans la pire configuration, le plan impose de visiter l'ensemble des états S, la vérification d'un plan se fait donc en $O(|S|)$ opérations.

Exemple de problématique

Considérons un robot nettoyeur qui travaille dans un petit appartement comprenant une chambre, un séjour et une cuisine (Figure 2.1(a)). Son état est constitué de 3 variables booléennes (Ch, Se, Cu) respectivement vrai si la chambre, le séjour ou la cuisine est propre. A cela s'ajoute une quatrième variable *ressources* représentant l'état de charge du robot selon 4 niveaux. Le robot peut : nettoyer la cuisine et la chambre pour une unité de ressource, nettoyer le séjour pour deux unités de ressources ou se recharger. Enfin, le robot ne peut plus se recharger s'il a consommé toutes ses ressources. L'état du système comprend donc une variable à 4 valeurs et 3 variables booléennes soit un total de 32 états ($4 * 2^3$). Seulement 23 états de l'automate (Figure 2.1) sont atteignables par l'ensemble des combinaisons d'actions possibles depuis l'état initial (s_0 : aucune pièce propre, robot en charge maximale).

En considérant comme état but l'état s_7 où toutes les pièces sont nettoyées (Ch, Se et Cu sont vrais) et le robot est complètement chargé, le planificateur a pour objectif de déterminer un plan sous la forme d'un chemin dans l'automate qui se termine dans l'état s_7. Ce chemin décrit un comportement partiel défini uniquement pour chacun des états traversés (par exemple Figure 2.2 : (s_0, nettoyer Ch), (s_9, se charger), (s_1, nettoyer Cu), (s_{13}, se charger), (s_5, nettoyer Se) et (s_{23}, se charger)).

Définition factorisée des transitions

Les automates à ensemble fini d'états peuvent rapidement être très lourds, du fait du nombre d'états qui définissent le système. De plus, une définition brute de la fonction de transition $t : S \times A \rightarrow S$ impose une énumération de l'ensemble des couples (état, action). Le langage lié au planificateur STRIPS [Fikes 71] permet une définition factorisée des actions. En définissant un état comme un ensemble de formules atomiques valides (par exemple l'état s_{18} : *propre*(Chambre, faux), *propre*(Séjour, vrai), *propre*(Cuisine, faux) et *ressource*(robot, 1)), une action peut être définie par 3 conjonctions de formules atomiques : précondition, suppression et ajout. Ainsi, l'action est exécutable si l'état initial vérifie les formules de précondition, et l'exécution de l'action conduit dans un état construit relativement à l'état initial après suppression

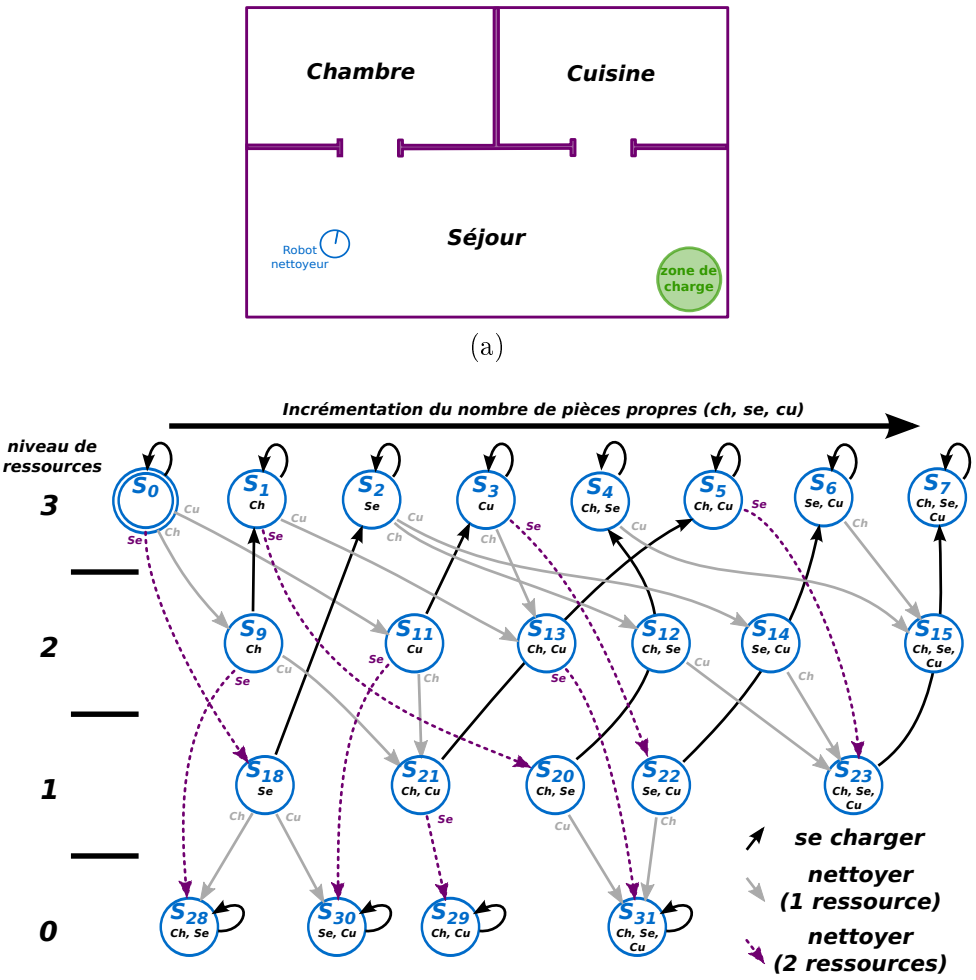


FIGURE 2.1 – Appartement à nettoyer par le robot mobile (a) et automate à ensemble fini d'états du robot nettoyeur (b)

et ajout des formules décrites par l'action. Voici un exemple de description de l'action "netoyer" quand le niveau de ressources du robot est de 3 :

$$\begin{array}{rrl} & \textit{precondition} : & \textit{ressource}(\text{robot}, 3),\ \textit{propre}(\text{Séjour}, \textit{faux}) \\ \textit{nettoyerSe3}(\text{robot}) : & \textit{suppression} : & \textit{ressource}(\text{robot}, 3),\ \textit{propre}(\text{Séjour}, \textit{faux}) \\ & \textit{ajout} : & \textit{ressource}(\text{robot}, 1),\ \textit{propre}(\text{Séjour}, \textit{vrai}) \end{array}$$

Les buts sont exprimés de la même façon par un ensemble de formules. Dans notre exemple le but regroupe les formules : *propre*(Chambre, vrai), *propre*(Séjour, vrai), *propre*(Cuisine, vrai) et *ressource*(robot, 3). La recherche d'un plan consiste alors à visiter la succession des états découlant de l'exécution des actions disponibles [Fikes 71]. Cette recherche peut être très coûteuse en calculs. La possibilité d'évaluer une distance entre chaque état et l'état but permet d'accélérer la résolution à la manière de l'algorithme A^* [Hart 68]. L'exploration des états construits par l'application des actions est alors orientée de façon à minimiser cette distance à chaque étape. D'autres approches [Lasdon 70, Dean 95] proposent une résolution basée sur une structuration de plusieurs plans partiels appliqués successivement pour réaliser chacun un sous-groupe de buts.

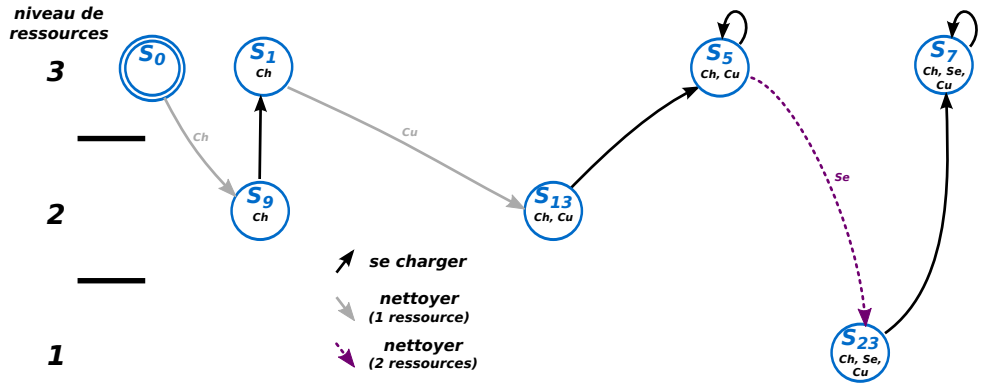


FIGURE 2.2 – Exemple de plan valide sous-optimal (vis à vis de terme du nombre d'actions réalisées) pour nettoyer l'appartement

Limitation des transitions déterministes

Une modélisation par automate à ensemble fini d'états ou en utilisant le langage STRIPS reste peu expressive par rapport à des cas concrets de planification :

- Le modèle suppose que les actions s'exécutent de façon séquentielle, une par une. Il n'est donc pas possible d'exécuter plusieurs actions simultanément sans construire des super-actions comme le produit de plusieurs actions unitaires exécutées simultanément. Cette limitation apparaît notamment lors de la modélisation de problèmes de planification mettant en œuvre plusieurs agents. Dans notre exemple, le nettoyage des pièces pourrait se faire simultanément avec deux robots. Chaque action est alors définie par l'union des actions individuelles (e.g., robot1 nettoie Ch, robot2 se charge, etc.) en tenant compte du fait que chaque action individuelle n'a pas forcément la même durée d'exécution.
- Le modèle ne propose pas de traiter directement des problèmes d'optimisation. Une recherche ciblant le plan optimal (minimal ou maximal) nécessite une relation d'ordre entre les plans valides. Dans le cadre du robot nettoyeur, il est possible d'imaginer une métrique basée sur le nombre de charges effectuées lors de la réalisation du plan. Un plan optimal est alors un plan qui minimisera le nombre de charges.
- Le modèle ne permet pas d'exprimer clairement les interactions entre l'agent et les éléments de son environnement. Il est cependant possible de conditionner les transitions par l'observation de certains événements. Le comportement de l'agent est alors défini selon un triplet (état, événement, action).

L'utilisation de la notion d'événement permet l'application de planificateurs déterministes dans des environnements où certains éléments ne dépendent pas de l'agent contrôlé. La modélisation de l'incertain permet de passer dans une nouvelle classe de planificateurs où l'exécution d'une action peut conduire le système dans différents états.

2.1.2 Planification stochastique

De façon similaire à la planification déterministe, les approches de planification sous incertitude basent la représentation des actions sous forme de transitions t qui associent des triplets (s, a, s') (état initial, action réalisée, état atteint) [Bellman 57]. Dans le cas de transitions déterministes l'état atteint s' pour un couple (s, a) est unique. En considérant une évolution incertaine du système, la réalisation d'une action a dans un état s peut mener le système dans un état

parmi plusieurs possibles. Il existera alors plusieurs états s' tels que les triplets (s, a, s') soient réalisables.

Dans la mesure d'une connaissance fine sur l'évolution d'un système il est possible d'associer une distribution de probabilités sur l'ensemble des états atteignables. Une probabilité est alors définie sur un intervalle $[0, 1]$ telle que 0 correspond au cas où la transition (s, a, s') n'est jamais réalisée, et 1 au cas où la transition est certaine.

Notion de politique

La résolution d'un problème de planification sous incertitude consiste à définir les actions à réaliser pour chacun des états de façon à maximiser l'espérance que les buts soient atteints. Dans le cadre de planification sous incertitude, il y a un glissement sémantique d'une problématique de résolution de problèmes vers une problématique de prise de décision [Bellman 57, Puterman 94, Buffet 08]. Dans le cadre où le choix des actions à réaliser ne fournit qu'une garantie statistique de réussite, on parlera de politique d'action π plutôt que de plan. Dans la mesure où les états $s \in S$ du système sont construits par l'union des états perceptifs et des états de la connaissance, la fonction politique π est définie en associant une action à réaliser à chaque état du système :

$$\pi : S \rightarrow A$$

La planification sous incertitude et orientée par des objectifs est, par nature, un problème d'optimisation. La solution (politique optimale) est la politique qui maximise les chances d'atteindre un des états objectifs $s \in S_{objectifs} \subset S$. Les algorithmes classiques de résolution se basent sur des techniques similaires à la planification déterministe avec optimisation. Ils parcourent, itérativement, les états du système en fonction des actions choisies de façon à évaluer une politique [Puterman 94]. L'optimalité d'une politique s'évalue comparativement aux autres politiques possibles.

Dans la mesure où le l'évaluation d'une politique ne dépend plus uniquement d'états objectifs mais prend en compte d'autres éléments (e.g. minimisation des actions de charge), la politique optimale ne pourra pas être définie uniquement sur une maximisation des probabilité d'atteindre un des états objectifs. Le formalisme des Processus Décisionnels de Markov (MDP) fait référence en terme de modélisation de problèmes de prise de décision dans l'incertain. Il associe à un automate à transition probabiliste une fonction de coût/récompense. Nous verrons plus tard comment les MDPs et ses extensions proposent des modèles adaptés à la planification de missions d'exploration en robotique.

Retour sur l'exemple

Dans le cadre d'un robot nettoyeur où la consommation de ses ressources est incertaine (par exemple, conditionnée par l'état réel de propreté des pièces), les actions de "nettoyer" vont pouvoir conduire à différents états. Par exemple, l'action de nettoyer la chambre peut être soumise aux probabilités : 0.2 pas de diminution du niveau de charge, 0.7 une simple diminution et 0.1 une double diminution du niveau de charge. Ainsi, à partir de l'état initial s_0, l'exécution de l'action "nettoyer la chambre" peut conduire le robot dans les états s_1 : *propre*(Chambre, vrai), *ressource*(3) avec une probabilité de 0.1 (la chambre était déjà propre), s_9 : *propre*(Chambre, vrai), *ressource*(2) avec une probabilité de 0.7 (situation normale), ou s_{16} : *propre*(Chambre, vrai), *ressource*(1) avec une probabilité de 0.2 (la chambre était très sale). Les transitions à partir du couple (s_0, nettoyer la chambre), vers tous les autres états, sont alors nulles (Figure 2.3).

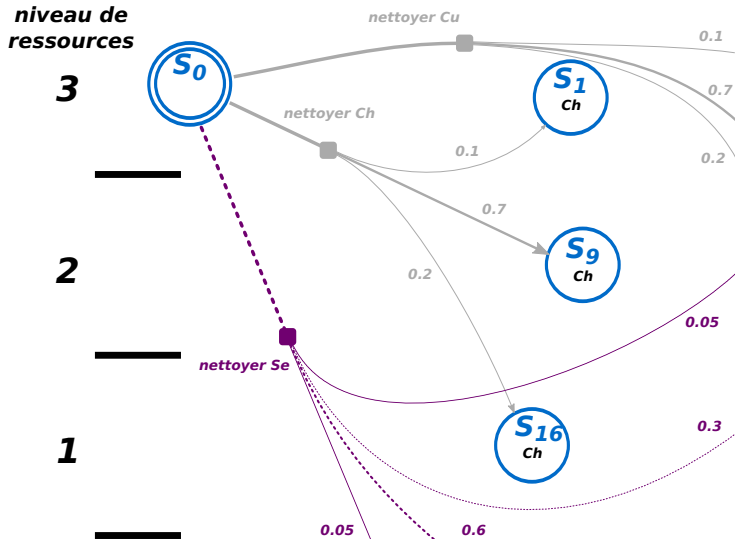


FIGURE 2.3 – Zoom sur l'état initial s_0 et l'action "nettoyer chambre" avec des transitions probabilistes.

Un plan valide sera défini si les actions permettent, avec certitude, d'atteindre un des états où les objectifs sont atteints, ici s_7 : *propre*(Chambre, vrai), *propre*(Cuisine, vrai), *propre*(Séjour, vrai), *ressource*(3). La nature incertaine du problème induit une validité probabiliste sur la politique. L'objectif de la planification consiste alors à définir les actions optimisant les chances de valider la mission (ici, d'arriver à nettoyer les 3 pièces).

Factorisation de la fonction de transition

La fonction de transition doit être définie pour tous les triplets possibles (état initial, action réalisée, état atteint). Comme dans le cadre déterministe, il est possible de factoriser la fonction de transition. Les réseaux bayésiens permettent de représenter l'influence des variables de l'état les unes par rapport aux autres lors de la réalisation d'une action. Ce modèle permet une définition factorisée des transitions [Naïm 07]. Par exemple, dans le cadre du nettoyage de la chambre, si l'on possède 2 variables "état de la pièce" et "fréquence de nettoyage" dans l'état du système, il est possible de les relier au nombre de ressources qui seront probablement consommées lors du nettoyage (Figure 2.4).

L'expression de l'incertitude dans les transitions permet au modèle d'exprimer une grande variété de problèmes liés à l'optimisation du comportement d'un agent. La modélisation par transitions probabilistes, par rapport aux transitions déterministes, permet de lever la limite liée à l'évolution incertaine d'un système. Par contre, le modèle suppose toujours que les actions s'exécutent de façon unique et séquentielle, ce qui n'est pas pertinent dans un cadre multi-agents.

2.1.3 Planification multi-agents

La notion d'agent définit une entité capable de percevoir et d'agir dans son environnement. Les systèmes multi-agents (SMAs) modélisent des systémes mettant en œuvre plusieurs entités évoluant dans un même environnement [Wooldridge 95, Ferber 95]. Les SMAs permettent ainsi de modéliser des problèmes de contrôle distribué multi-robots. Chaque agent est autonome pour choisir ses actions à exécuter et dans une approche par planification, l'exécution d'une politique

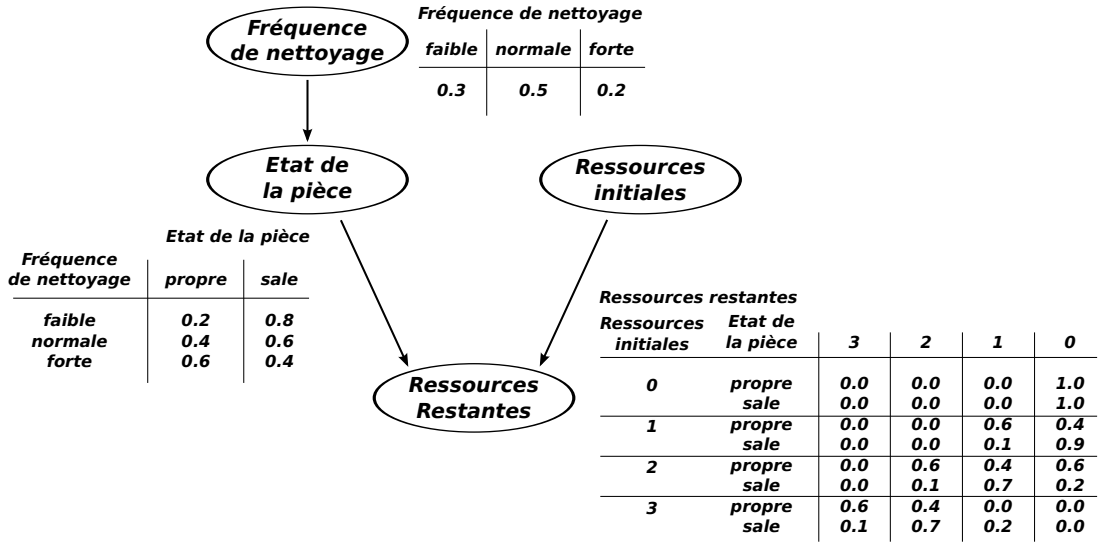


FIGURE 2.4 – Réseau bayesien attaché à la fonction "nettoyer la chambre"

se fait de façon distribuée. Le choix des actions à réaliser par chaque agent se fait sur la base de ses perceptions propres (incluant de possibles communications).

Dans le cadre de la planification multi-agents, on ne cherche plus à organiser les actions réalisées par un agent mais les actions réalisées en parallèle par plusieurs agents. Les principales modélisations pour la planification (mono-agent) s'appuient sur une discrétisation de l'espace d'états du système couplé à une exécution séquentielle des actions. Cette approche est inappropriée à la planification d'actions réalisées de façon simultanée comme dans un cadre multi-agents. La politique globale doit alors être composée de politiques individuelles qui, exécutées séparément par chaque agent, permettront au système d'atteindre les objectifs fixés.

Il existe 2 principales approches pour traiter un problème de planification multi-agents. La première approche, avec modélisation centralisée, considère un unique modèle intégrant des états et des actions construits sur les capacités jointes de tous les agents contrôlés. La seconde approche, avec modélisation distribuée, cherche à coordonner plusieurs modèles interdépendants, où un modèle est construit pour chaque agent.

Modélisation centralisée

Avec une approche par modélisation centralisée, la planification des actions à réaliser conjointement par un groupe d'agents implique de traiter l'ensemble des actions individuelles comme une action jointe unique dans le modèle du système. L'ensemble des états S est défini par l'union des états individuels (S_{Ag_i}). L'ensemble des actions A est défini comme le produit cartésien des actions simultanées de chacun des agents (A_{Ag_i}). Ce modèle suppose alors que les actions individuelles se réalisent en même temps et avec une même durée.

$$S = S_{Ag_1} \times \ldots \times S_{Ag_n}$$
$$A = A_{Ag_1} \times \ldots \times A_{Ag_n}$$

Si nous reprenons l'exemple du nettoyage d'appartement par une flotte de deux robots, chaque action du système est décrite par les actions réalisées simultanément par chacun des deux robots. Le système évolue selon les actions individuelles combinant nettoyage, attente et chargement. En

supposant que le nettoyage du séjour prend deux fois plus de temps que le nettoyage de la chambre, de la cuisine ou que la charge, il est alors nécessaire de diviser cette action en deux actions unitaires. Les actions jointes peuvent s'énumérer comme suit :

1. action attendre-nettoyer(Ch) : le robot 1 attend et le robot 2 nettoie la chambre.
2. action attendre-nettoyer(Cu) : le robot 1 attend et le robot 2 nettoie la cuisine.
3. action attendre-nettoyer1(Se) : le robot 1 attend et le robot 2 débute le nettoyage du séjour
4. action attendre-nettoyer2(Se) : le robot 1 attend et le robot 2 finit le nettoyage du séjour.
5. action attendre-charger : le robot 1 attend et le robot 2 se charge.
6. action nettoyer(Ch)-attendre : le robot 1 nettoie la chambre et le robot 2 attend.
7. action nettoyer(Ch)-nettoyer(Ch) : les deux robots nettoient la chambre.
8. action nettoyer(Ch)-nettoyer(Cu) : le robot 1 nettoie la chambre et le robot 2 la cuisine.
9. ...

La définition de la fonction de transition devra alors intégrer le fait que si un robot débute le nettoyage du séjour alors, à l'état suivant, il n'a pas d'autre choix d'action que de finir le séjour. D'autre part, si les politiques sont exécutées de maniére décentralisée, la politique construite sera distribuée sur chacun des agents. Les politiques individuelles doivent alors respecter les capacités perceptives et communicatives de chaque agent.

Contrainte sur une politique décentralisée

Au delà de l'explosion combinatoire due à la modélisation d'un tel système, il est important, dans la phase de planification, de garder une cohérence entre les actions individuelles réalisées et les parties de l'état global du système connues par chacun des agents. En d'autres termes, la politique globale doit être définie comme l'union de politiques individuelles $(\pi_{Ag_1},\ \ldots,\ \pi_{Ag_n})$.

$$\begin{array}{rcl} \pi : S & \rightarrow & A \\ s_{joint} = (s_{Ag_1},\ \ldots,\ s_{Ag_n}) & \rightarrow & \pi(s_{joint}) = (\pi_{Ag_1}(s_{Ag_1}),\ \ldots,\ \pi_{Ag_n}(s_{Ag_n})) \end{array}$$

La politique globale doit respecter une contrainte d'unicité des actions individuelles pour un même état individuel. La politique ne peut pas définir deux actions individuelles différentes pour le même état d'un agent.

$$\forall Ag_i, \quad \forall s_{joint},\ s'_{joint} \in S \quad \text{si } s_{Ag_i} = s'_{Ag_i}, \quad \text{alors } \pi_{Ag_i}(s_{joint}) = \pi_{Ag_i}(s_{joint})$$

Une modélisation simple d'un probléme multi-agents peut se baser sur une connaissance indépendante des capacités perceptives de chaque agent. Il y a souvent redondance entre les informations connues par chaque agent et l'état du système peut être construit par fusion des états individuels. Dans notre exemple, cela correspond à des états construits sur une connaissance de la propreté de chacune des pièces. Après l'exécusion de l'action jointe "nettoyer(Ch)-nettoyer(Cu)", le système atteint un état où la chambre et la cuisine sont nettoyées sans garder de trace de qui à fait quoi.

Cela ne porte pas à conséquence dans un environnement avec des communications efficaces ; si l'état du système est connu par chacun des agents $(\forall Ag_i, \quad S = S_{Ag_i})$. Dans le cas d'une

possibilité de communication efficace et peu coûteuse, il est possible de mettre en place un blackboard permanent. Dans le cas contraire, après l'exécution d'une action jointe, chaque agent possède une connaissance partielle de l'état du système. Par exemple, après l'action "nettoyer(Ch)-nettoyer(Cu)", le robot 1 connaîtra l'état de la chambre mais ne saura pas forcément si la cuisine aussi a pu être nettoyée.

Nous verrons, avec les modèles de la familles des Processus Décisionnels de Markov (MDP), qu'une solution pour une modélisation générique de problémes multi-agents coopératifs consiste à introduire une notion d'observabilité. Le système est toujours décrit par un ensemble d'états mais la politique est définie sur les observations possibles de l'état du système faites par chacun des agents.

Modélisation distribuée

Une autre approche pour exprimer une problématique multi-agents et calculer la politique jointe, consiste à s'appuyer sur une modélisation distribuée avec un modèle individuel construit pour chaque agent. Chaque modèle individuel (état, action, transition) est construit sur l'espace d'états restreints à ceux de l'agent en accord avec ses capacités perceptives. Cette approche est particulièrement appropriée pour une planification distribuée où chaque agent cherche à construire sa propre politique coordonnée [Bernstein 00, Chades 02, Nair 03].

Par contre, les modèles individuels sont inter-dépendants. En effet, la dynamique de l'environnement perçue par un agent dépend des actions réalisées par les autres agents. Formellement, la fonction de transition d'un agent Ag_i peut dépendre de la politique de tous les autres agents. Dans notre exemple, si le robot 1 cherche à nettoyer une pièce qui a déjà été nettoyée par le robot 2, il y aura une forte probabilité pour que le robot 1 ne consomme aucune ressource pour cette action.

Une modélisation distribuée nécessite d'exprimer une connaissance sur les interactions entre les agents. La plupart des approches s'appuient sur des éléments clefs définissant les règles d'interaction. Ces règles permettent de paramétrer chaque modèle individuel. Les éléments clefs prennent généralement la forme de ressources/tâches à allouer, de règles d'usage à définir, de rôle à distribuer et/ou de mise en place de rendez-vous [Ren 08]. Nous parlerons alors d'un ensemble de variables d'interaction $V_1 \dots V_k$ associées à des domaines de variation $D_1 \dots D_k$.

Par exemple, dans le cadre d'une coordination basée sur une simple allocation de tâches (les pièces à nettoyer), les variables d'interaction représentent les tâches à allouer. Les domaines de variation associés sont définis par l'ensemble des robots. L'allocation de tâches permet de définir individuellement les états objectifs à atteindre. En supposant que chaque robot est indépendant dans ses déplacements, il pourra être autonome pour calculer sa politique une fois les objectifs alloués.

La définition heuristique des variables d'interaction ne permet pas de garantir l'optimalité des solutions calculées. Dans notre exemple, l'indépendance dans les déplacements de chaque robot est une hypothèse restrictive forte. Dans les faits, les robots peuvent se gêner les uns les autres. D'un autre côté, avec un nombre de variables d'interaction restreint, une modélisation distribuée permet de réduire considérablement la taille d'un problème de coordination. Une modélisation globale implique $(|A_{Ag_i}|^n)^{|S_{Ag_i}|^n}$ politiques jointes possibles avec $|A_{Ag_i}|^n$ actions et $|S_{Ag_i}|^n$ états joints. Une modélisation distribuée implique $n * |A_{Ag_i}|^{|S_{Ag_i}|} * |D_j|^k$ politiques jointes possibles soit $|A_{Ag_i}|^{|S_{Ag_i}|}$ politiques individuelles par agent (n) et par configuration des variables d'interaction ($|D_j|^k$).

2.2 Exploration par un robot unique

L'exploration consiste à permettre à un robot de se déplacer dans son environnement de façon à acquérir une connaissance sur des zones à visiter. Pour valider la bonne couverture des zones à visiter, le robot doit maintenir une connaissance sur sa localisation lors de ses déplacements. A chaque instant, le robot a à choisir une action de déplacement pour lui permettre d'étendre sa carte connue. Quelle que soit la représentation de son environnement, grille d'occupation ou carte topologique (cf. Section 1.3.3), ce choix est matérialisé par une position à atteindre parmi l'ensemble des positions frontières, à la limite des zones connues.

La problématique de planifier une exploration peut être considérée comme double avec une phase de planification hiérarchisant les zones à visiter et une phase de planification du chemin permettant de naviguer entre les zones à visiter. La fin d'une mission d'exploration est définie lorsqu'il n'existe plus de zones atteignables à visiter.

Le principe des algorithmes de planification repose généralement sur l'évaluation d'un nombre important de politiques pour converger vers la meilleure. Cependant, une énumération des politiques en considérant tous les chemins possibles reliant toutes les zones à visiter et en considérant toutes les possibilités dans la construction de la carte, va s'avérer fastidieuse et computationellement infaisable.

2.2.1 Des variables vers les états

Dans le cadre de la planification pour un robot mobile, les politiques sont définies comme une succession d'actions de déplacement permettant au robot, à partir d'un état initial, d'atteindre un état où tous les objectifs de sa mission sont réalisés. La succession d'actions va permettre de prévoir la succession d'états qui seront possiblement traversés. Un état du robot est alors composé de l'ensemble des variables qui définissent sa connaissance à un instant t.

L'espace de recherche est basé sur les possibilités de succession entre les états. L'espace de recherche est donc exponentiel par rapport à l'espace des états de l'agent. Le robot est une entité réelle et l'espace des états d'un robot est limité, à défaut, par la taille de sa mémoire. En pratique, en considérant que le robot embarque un ordinateur conventionnel, la mémoire disponible permet d'énumérer un volume important d'états. Ce volume est borné, en considérant des états construits sur un ensemble de variables booléennes, par :

$$1 \text{ octet } = 2^8 \text{ états potentiels} \qquad 1 \text{ gigaoctet } = 2^{8*10^9} \text{ états potentiels}$$

Mémorisation complète des successions de perceptions

Les états du robot construits sur une simple mémorisation des perceptions depuis le début de la mission et jusqu'à saturation de la mémoire induit un nombre exponentiel d'états par rapport à la taille de la mémoire. En considérant l'ensemble des transitions possibles entre les états, il en découle un espace de recherche doublement exponentiel par rapport à la taille de la mémoire. De ce fait, la planification basée sur une simple mémorisation des perceptions est impossible à mettre en œuvre pour des applications réelles. Il est indispensable de proposer une planification basée sur un espace d'états du robot contenu pour réduire l'espace de recherche. Ainsi, l'utilisation de modèles compacts pour représenter la connaissance du robot va permettre de limiter l'espace de recherche.

La connaissance d'un robot navigant dans un environnement, est généralement composée de sa carte, de sa position et de la position des objectifs dans sa carte. Les objectifs à un instant

donné, en exploration, sont donc modélisés par l'ensemble des positions présentes et futures à explorer. Les positions futures dépendent de la carte qui est construite au fur et à mesure des déplacements. De plus, l'évolution de la carte est incertaine, il en résulte un espace de recherche conséquent pour définir tous les états possibles.

États d'un robot découlant d'une modélisation

Concrètement l'état s d'un robot à l'instant t est défini par sa carte courante M_s (Map) et sa position p_s. La position du robot appartient à l'ensemble des positions P_s définies par la carte M_s. Une carte M_s permet de lier un ensemble de positions P_s entre elles via un ensemble de connexions C_s (les déplacements possibles).

$$s = (M_s, p_s) \quad | \quad M_s = \{P_s, C_s\}, \quad p_s \in P_s$$

L'espace des états correspond à l'ensemble de tous les états atteignables à partir de l'état courant. La taille de l'espace des états dépend alors de la taille et du nombre de cartes potentielles qui peuvent être construites. La carte et la position du robot sont définies de façon différente en fonction de la représentation de l'environnement choisi.

Rappelons que l'on trouve 2 principaux types de cartes : les grilles d'occupation et les cartes topologiques. Dans le premier cas, la position du robot est liée à des coordonnées cartésiennes. Dans le second cas, la position du robot est définie relativement aux connexions existantes avec les autres positions. Une position topologique est alors définie par l'ensemble des positions permettant de l'atteindre et l'ensemble des positions atteignables. Il est bien entendu possible de lier des coordonnées métriques à une position topologique.

Évaluation de l'espace d'états

L'espace d'états du robot dépend du potentiel de sa connaissance. De façon à borner le nombre de cartes potentielles, on définit l'ensemble P comme l'ensemble maximal de toutes les positions potentielles modélisables dans la carte du robot. C'est-à-dire que, quelque soit un état s à un instant donné, P_s (défini par la carte courante M_s) est un sous-ensemble de P. L'ensemble $P - P_s0$ représente alors l'ensemble des positions inconnues à l'état initial $s0$. On définit alors : $\mathbb{P}$ les ensembles potentiels de positions ($\mathbb{P} = \{P_s\} \mid P_s \subset P$) et $\mathbb{C}_s$ l'ensemble des configurations des connexions possibles pour les positions dans P_s. En considérant P fini et dénombrable, l'ensemble des cartes potentielles $\mathbb{M}$ est défini comme le produit de $\mathbb{P}$ et de $\mathbb{C}_s$.

$$|\mathbb{M}| = \sum_{P_s \in \mathbb{P}} |\mathbb{C}_s| = \sum_{P_s \in \mathbb{P}} 2^{\left(\frac{|P_s|^2}{2}\right)} \text{ avec } |\mathbb{P}| = 2^{|P - P_s0|} \text{ possibilités pour } P_s$$

$$2^{|P-P_s0|} * 2^{\left(\frac{|P_s0|^2}{2}\right)} < |\mathbb{M}| < 2^{|P-P_s0|} * 2^{\left(\frac{|P|^2}{2}\right)}$$

$2^{|P-P_s0|}$ définit le nombre d'ensembles de positions potentielles et $2^{\left(\frac{|P_s|^2}{2}\right)}$ le nombre de configurations potentielles pour chaque ensemble P_s. Une configuration dépend de l'existence ou non de connexions entre chaque paire de positions de la carte. Avec $|P_s|$ positions possibles pour le robot à un instant t, l'ensemble des états possibles S du robot est défini comme le produit cartésien des cartes potentielles et des positions possibles du robot pour chaque carte :

$$|S| = \sum_{P_s=P_s0}^{P} |\mathbb{C}_s| * |P_s| = \sum_{P_s=P_s0}^{P} 2^{\left(\frac{|P_s|^2}{2}\right)} * |P_s|$$

$$2^{|P-P_s0|} * 2^{\left(\frac{|P_s0|^2}{2}\right)} * |P_s0| < |S| < 2^{|P-P_s0|} * 2^{\left(\frac{|P|^2}{2}\right)} * |P|$$

Il est possible de réduire l'espace des états en considérant une borne sur la connectivité de chaque position. Avec une discrétisation régulière basée sur une grille d'occupation, chaque position ne pourra connecter qu'un nombre maximal de voisins c_{max} réduisant ainsi le nombre de configurations possibles $\mathbb{C}_s = 2^{P_s * c_{max}}$. Cependant, la taille de l'espace des états liés au nombre de cartes potentielles reste exponentiel en le nombre maximal de positions qui peuvent être découvertes ($|P|$).

2.2.2 Construction de l'espace de recherche

L'objectif de la planification consiste à définir la succession d'actions à réaliser. Dans le cadre de l'exploration, la perception va modifier la carte et va induire un état s du robot parmi plusieurs possibles. Dans la mesure où il existe une distribution de probabilités sur les états atteignables, il est cohérent de modéliser le problème de planification avec une fonction de transition t stochastique de façon à construire une politique π d'actions : $\pi : S \rightarrow A$.

L'espace de recherche des solutions se compose alors de l'ensemble des politiques applicables. Les politiques solutions, permettant d'atteindre un état où il n'existe plus de position inconnue atteignable, sont à chercher parmi $|A|^{|S|}$ politiques possibles. De plus, nous avons vu que le nombre d'états $|S|$ est lui-même exponentiel en le nombre $|P|$ de positions potentielles. Il est donc excessivement difficile de parcourir l'ensemble des politiques possibles pour sélectionner celle qui permet l'exploration la plus efficace.

Espace de recherche dans un exemple concret

En considérant une grille d'occupation définie sur 10×16 cellules le robot peut être dans 160 positions différentes ($10 * 16$). On suppose un robot capable d'exécuter 3 actions : tourner de 45° à gauche, tourner de 45° à droite et avancer d'une cellule. De plus, la perception du robot lui permet d'établir la connectivité des positions à une distance de une cellule (Figure 2.5). Avec 8 orientation possibles, il existe 1280 ($160 * 8$) poses possibles (positions $\times$ orientations) et potentiellement 640 ($160 * 8/2$) passages bidirectionnels.

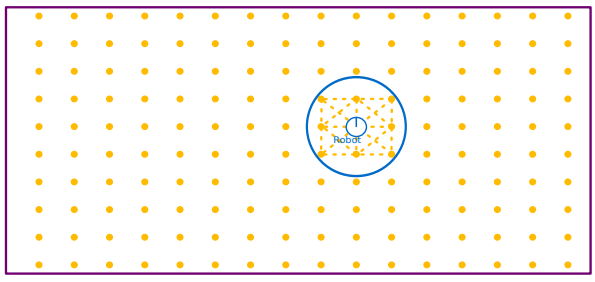


FIGURE 2.5 – Définition régulière de point de passages

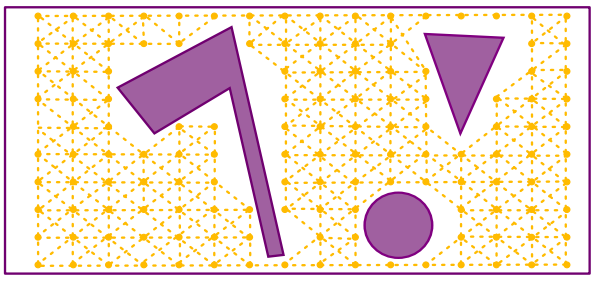


FIGURE 2.6 – Construction du graphe de déplacement

En considérant simplement que chaque passage inconnue existe ou non, il est alors possible de construire 2^{640-20} cartes (en supposant 9 cases et 20 passages initialement connues Figure 2.5)

parmi lesquelles se trouve la carte correspondant à l'environnement (Figure 2.6). Dans un tel environnement, les états possibles pour le robot appartiennent à $|S| = \mathbb{M} \times P$ soit $2^{620} * 1280$. Enfin, avec une politique à chercher parmi $|A|^{|S|}$ possibilités, l'espace de recherche s'évalue à $3^{2^{620}*1280}$ solutions potentielles.

Il est possible de se limiter aux états atteignables à partir de la position courante. Les états atteignables peuvent être construits en déroulant les séquences d'actions et de possibilités de perception à partir de l'état initial (Figure 2.7). Cependant, même en se limitant aux états atteignables, on reste dans des espaces de recherche trop importants (avec une borne minimale à $|A|^{2^{|P|-|P_s0|}}$ et dans l'exemple $3^{2^{620}*1280}$) pour pouvoir tester toutes les solutions. Même une génération de politique heuristique linéaire (en $O(n)$) est difficilement imaginable avec $2^{|P|-|P_s0|}$ états potentiels (dans l'exemple $n = 2^{620} * 1280$).

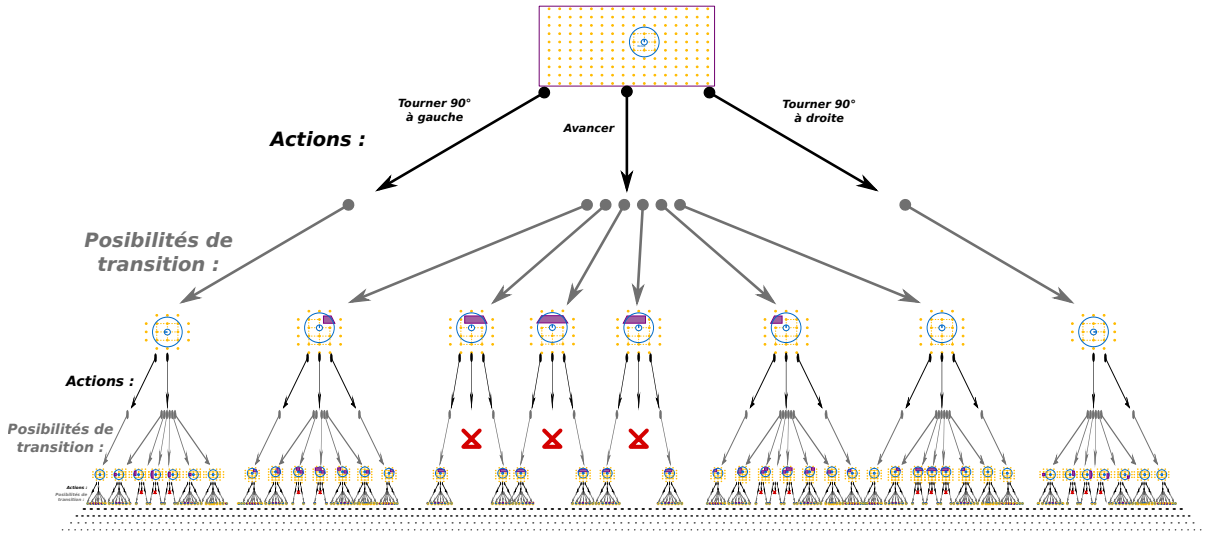


FIGURE 2.7 – Arbre de recherche déroulant l'ensemble des états atteignables à partir d'un état initial

Vers des re-planifications régulières

Il est raisonnablement impossible de planifier les actions à réaliser, de la configuration initiale du robot, jusqu'à l'exploration totale d'une zone et ce, quelles que soient les évolutions possibles de la carte. En raison de la taille de l'espace de recherche qui découle d'une modélisation complète du problème d'exploration, la plupart des stratégies d'exploration se basent sur une résolution à horizon limité sur l'évolution de la carte (incluant un nombre limité de modifications possibles dans la carte).

La politique doit donc être orientée par les états où les connaissances des robots couvrent tout l'environnement. Une résolution à horizon limité consiste à viser des objectifs intermédiaires où, la connaissance a été enrichie sans couvrir encore tout l'environnement. La politique est ensuite actualisée au fur et à mesure que la connaissance s'accroît pour viser de nouveaux objectifs intermédiaires vers des connaissances toujours plus riches. Pour calculer ces politiques à horizon limité sur l'évolution de la carte, des frontières sont définies entre les zones connues et inconnues pour représenter les objectifs intermédiaires. La difficulté de la mise en œuvre d'une stratégie d'exploration réside dans les mécanismes permettant au robot de hiérarchiser, en cours de mission, l'importance de chacune des portions frontières courantes (Figure 2.8).

Les solutions répertoriées dans la littérature proposent une planification sans présupposer des évolutions possibles de la carte [Thrun 98, Simmons 00, Matignon 12]. En d'autres termes, la politique construite est rendue obsolète après toute modification de la carte. Cette solution permet de considérer des états du robot construits uniquement à partir de la position du robot dans la carte courante. Par contre, une politique nouvelle doit alors être calculée entre deux mises à jour de la carte.

A chaque actualisation de politique, le robot a le choix de viser un état objectif (modélisant une portion de frontière à explorer) parmi plusieurs états objectifs. En fonction de la portion de frontière qui sera explorée, le robot terminera dans une configuration plus ou moins avantageuse pour continuer son exploration lorsque la carte sera mise à jour. Cependant, une planification sans présupposer des évolutions de la carte ne permet pas de hiérarchiser l'intérêt immédiat des portions de frontières les unes par rapport aux autres (Figure 2.8).

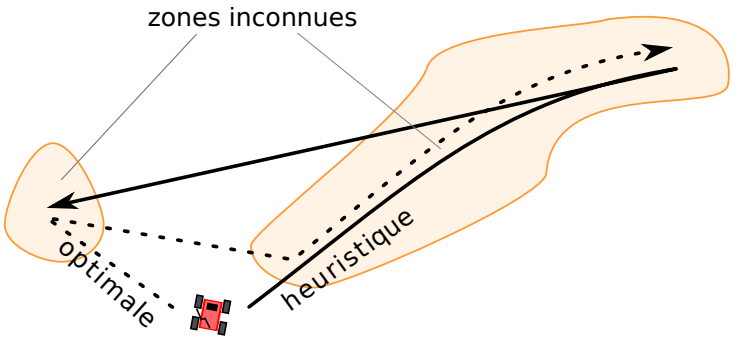


FIGURE 2.8 – Exemple où l'heuristique visant la frontière la plus proche ne permet pas une stratégie d'exploration optimale.

L'évaluation et le choix de quelle frontière il faudrait explorer en priorité peut se faire de façon heuristique. Généralement, l'heuristique consiste à choisir la frontière la plus proche [Elfe 87, Thrun 98, Choset 01]. Cependant, l'heuristique peut être définie arbitrairement : selon une route pré-définie (en spirale par exemple), de façon à maximiser la taille de la frontière repoussée, pour privilégier les fermetures de boucle, etc.

Une seconde approche pour évaluer l'intérêt immédiat des frontières, consiste à introduire une évaluation des positions frontières dans le cadre d'une planification des mouvements du robot. Le mécanisme consiste à évaluer les gains d'explorer chacune des portions de frontière (ou cellule frontière) comparativement aux autres et à propager ces gains dans la carte [Burgard 05, Matignon 12]. Ainsi, le choix final de la frontière à explorer maximise le ratio : "coût du déplacement" *versus* "gain associé à la portion de frontière".

La Figure 2.8 met en évidence que la taille et la configuration de zones distinctes à explorer peut être déterminant par rapport à un choix privilégiant uniquement la proximité des frontières. Une évaluation fine de l'intérêt de viser une frontière plutôt qu'une autre passe alors par une planification de chemin englobant plusieurs frontières à explorer [Berhault 03]. Ce constat appelle alors une planification multi-objectifs pour pouvoir éviter au robot des retours coûteux sur des zones laissées en arrière pour plus tard.

2.2.3 Les Processus Décisionnels de Markov (MDP)

La problématique de planification pour l'exploration relève d'un problème d'optimisation de chemin permettant de visiter plusieurs zones frontières dans un cadre incertain. Les Processus

Décisionnels de Markov (MDP) sont considérés comme un outil mathématique de référence pour modéliser des problèmes d'optimisation de décision sous incertitude. Le modèle est utilisé dans de nombreux travaux pour la planification de chemin en robotique mobile dont voici un échantillon : [Burgard 05, Teichteil-Königsbuch 06, Foka 07, Matignon 12]. Il permet de modéliser l'incertitude par des probabilités de transition et d'exprimer les performances par une fonction de coût/récompense liée à la réalisation des actions.

Du fait de la popularité des MDPs, il existe une bibliographie riche proposant des algorithmes permettant le calcul efficace de politiques ou permettant d'étendre le modèle pour exprimer des problèmes plus complexes. Parmi les extensions, nous verrons qu'il est notamment possible d'exprimer une notion d'observabilité et des problématiques de planification multi-agents.

Le formalisme

Un MDP est défini par un tuple $\langle S, A, t, r\rangle$ [Bellman 57, Puterman 94] représentant respectivement l'ensemble des états, l'ensemble des actions, la fonction de transition et la fonction de récompense. Ce tuple définit le système et ses possibilités d'évolution. En robotique mobile, les états sont généralement construits sur la base de la carte du robot à un instant donné et des actions unitaires permettant au robot de passer d'une position à une autre dans sa carte.

La fonction de transition t est définie par $t : S \times A \times S \to [0, 1]$ qui donne la probabilité $t(s, a, s')$ d'atteindre s' depuis s en exécutant l'action a. Une valeur de $t(s, a, s')$ égal à 1 ou 0 indique respectivement que la transition de s vers s' en réalisant l'action a est certaine (déterministe) ou impossible. Pour être cohérent avec l'évolution probable du monde la somme des transitions depuis s.

$$\sum_{s' \in S} t(s, a, s') \ = 1$$

r est la fonction de coût/récompense permettant d'exprimer les préférences sur les actions à réaliser en fonction de l'état courant. La fonction de coût/récompense r est définie par $r : S \times A \to \mathbb{R}$, où $r(s,\ a)$ retourne la récompense obtenue en exécutant l'action a depuis s si $r(s,\ a)$ est positif ou le coût d'exécuter a en s si $r(s,\ a,)$ est négatif.

Politique optimale

Trouver une solution optimale à un MDP consiste à chercher la politique optimale $\pi^* : S \to A$ qui maximise les gains espérés sur les récompenses. Bellman [Bellman 57] a proposé une fonction permettant une évaluation récursive des états par rapport à π la politique employée $V^\pi : S \to \mathbb{R}$. La valeur d'un état $V^\pi(s)$ est alors définie comme une somme pondérée de sa récompense immédiate après réalisation de l'action $\pi(s)$ et des valeurs des états atteignables proportionnellement aux probabilités de les atteindre :

$$V^\pi(s) = r(s, \pi(s)) + \gamma \sum_{s' \in S} t(s, \pi(s), s') V^\pi(s')$$

Le paramètre $\gamma \in [0, 1]$ pondère l'importance entre les récompenses immédiates et futures. La politique optimale π^* est la politique qui maximise la fonction de valeur $V^\pi : S \to \mathbb{R}$ donnée par l'équation de Bellman [Bellman 57].

$$\forall s \in S, \quad \forall \pi \in A^S, \quad V^{\pi^*}(s) \geq V^\pi(s)$$

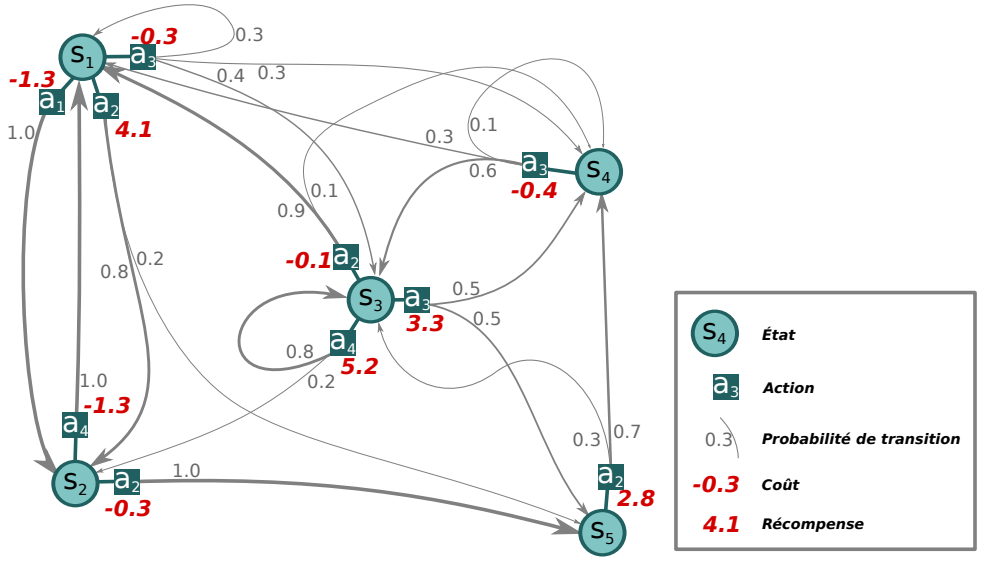


FIGURE 2.9 – Exemple d'un Processus Décisionnel de Markov avec 5 états et un total de 4 actions disponibles (e.g. $t(S_3,\ a_2,\ S_1) = 0.9$ et $r(S_2,\ a_4) = -1.3$).

$$\forall s \in S, \quad \pi^*(s) = \underset{a \in A}{argmax} \left(r(s,a) + \gamma \sum_{s' \in S} t(s,a,s') V^{\pi^*}(s') \right)$$

$$\text{avec} \quad \underset{a \in A}{argmax}\ (\ f(a)\ ) = a^* \quad \Leftrightarrow \quad \forall a \in A, \quad f(a^*) \geq f(a)$$

Résolution d'un MDP

A partir de l'équation de Bellman, l'algorithme "*Value iteration*" [Bellman 57, Bellman 70] repose sur une technique de programmation dynamique pour calculer une politique optimale. Le principe consiste en des ré-évaluations approximatives successives de chaque état (Algorithme 2).

Algorithme 2 : Value iteration

Données : un MDP $\langle S, A, t, r \rangle$, $\gamma \in [0,\ 1]$ et $\epsilon \ll \min_{s \in S,\ a \in A} (\ |r(s,\ a)|\ )$

Résultat : π^* (selon ϵ et γ) et V^{π^*}

1 initialiser V : $\forall s \in S, \quad V[s] \leftarrow 0$;

2 **répéter**

3 initialiser V' : $V' \leftarrow V$;

4 Actualiser π^* : $\forall s \in S, \quad \pi^*[s] \leftarrow \underset{a \in A}{argmax} \left(r(s,a) + \gamma \sum_{s' \in S} t(s,a,s') V'[s'] \right)$;

5 Actualiser V : $\forall s \in S, \quad V[s] \leftarrow r(s, \pi^*[s]) + \gamma \sum_{s' \in S} t(s, \pi^*[s], s') V'[s']$;

6 **jusqu'à** $\forall s \in S, \quad |V'(s) - V(s)| < \epsilon$;

Nous avons vu que γ représente le facteur d'atténuation des gains futurs. Ce facteur permet notamment de traiter des problèmes à horizon infini. Dans un cadre robotique, cela peut correspondre à des missions où les objectifs sont perpétuellement remis en cause. Par exemple,

pour une mission de surveillance d'entrepôt, le robot doit alors maintenir une connaissance d'un environnement potentiellement dynamique. Dans le cas d'un problème à horizon fini γ peut être paramétré à 1. Sur la base d'une atténuation des coûts/récompenses à l'infini, le paramètre ϵ conditionne l'arrêt de l'algorithme lorsque l'actualisation des valeurs de gains est considérée comme négligeable.

L'algorithme "*Policy iteration*" [Howard 60], variante de "*Value iteration*", propose d'actualiser, à ϵ près, les valeurs de l'équation de Bellman $V^{\pi}(s)$, entre deux itérations sur la politique construite π. Chaque itération dans "*Policy iteration*" demande alors plus de ressources calculatoires qu'une itération dans "*Value iteration*" [Puterman 94]. Cependant, "*Policy iteration*" a statistiquement besoin de moins d'itérations pour converger vers la politique optimale. La résolution d'un MDP est P-Complet et les deux algorithmes ont des résultats similaires [Puterman 05]. "*Value iteration*" et "*Policy iteration*" permettent de traiter des problèmes de l'ordre du million d'états [Sutton 98].

2.2.4 Exploration : un MDP orienté par des objectifs (GO-MDP)

Dans le cadre d'une planification de l'exploration d'une zone par des robots mobiles, l'utilisation des Processus Décisionnels de Markov (MDP) à horizon limité sur les évolutions des connaissances impose de matérialiser les frontières à atteindre comme des objectifs intermédiaires. Un coût est associé à chacune des actions de déplacement relativement à l'énergie ou le temps consommé. Des récompenses sont également associées à chaque action permettant de visiter une zone inconnue [Burgard 05, Matignon 12].

La tâche d'exploration à un instant donné, en cours de mission, est donc composée de plusieurs zones à visiter (objectifs). Une évaluation homogène de ces objectifs conduit le robot sur la zone minimisant les coûts des déplacements. Sans faire des hypothèses sur les zones inconnues, il est possible d'étendre ce modèle utilisé en robotique mobile, de façon à choisir la frontière à repousser en considérant les frontières laissées derrière le robot. Il est alors nécessaire d'ajouter, dans la définition des états, une mémoire des objectifs atteint ou non, de façon à construire une politique englobant l'ensemble des objectifs à réaliser pour une carte à un instant donné.

Définition des espaces d'états et d'actions

L'ensemble des états S est donc construit sur la base des positions et orientations atteignables dans la carte P incluant l'ensemble des positions frontières (Goal) $G \subset P$. Les positions frontières sont définis par l'ensemble des positions connues d'où il est possible, pour le robot, de percevoir une portion des zones inconnues. Un MDP orienté par des objectifs (GO-MDP) est construit en enrichissant chaque état s d'une mémoire des objectifs qui ont été atteints G_s. L'ensemble G_s des objectifs atteints est un sous-ensemble de tous les objectifs G. La mission s'arrête lorsque l'ensemble des objectifs est atteint : $G_s = G$.

$$S = \{s \quad | \quad s = (p_s,\ G_s), \quad p_s \in P, \quad G_s \subset G\}$$

Le système est donc défini sur $|P| * 2^{|G|}$ états (Figure 2.10) en considérant toutes les successions possibles dans l'ordre de la réalisation des objectifs ($2^{|G|}$ possibilités pour G_s). L'ensemble des actions est défini selon les capacités du robot. Si l'on reprend l'exemple du robot explorant une grille de 10×16 avec 4 orientations et 3 actions possibles, à l'état initial, ses objectifs sont localisés à la limite de sa perception (Figure 2.11). A l'état initial, avec une carte de 9 positions dont 4 positions frontières, le système est défini sur 576 états ($9 * 4 * 2^4$).

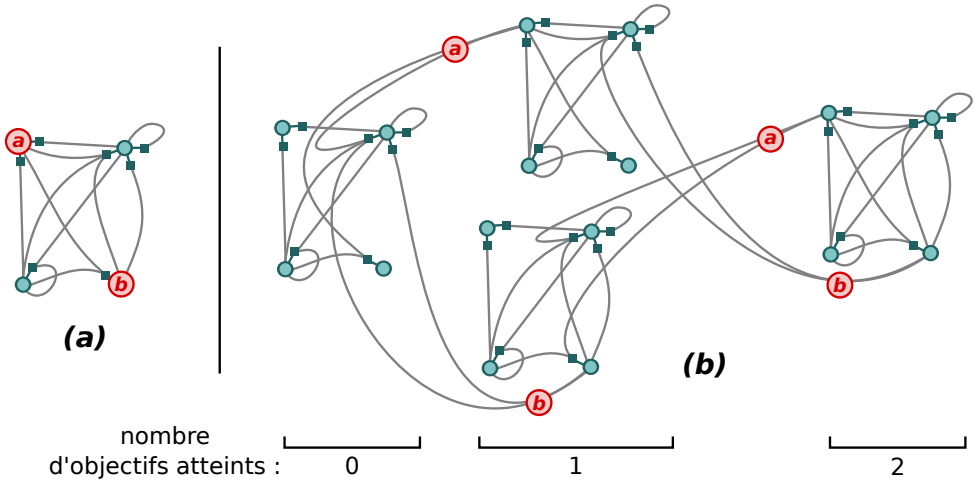


FIGURE 2.10 – Exemple de GO-MDP (b) construit sur la base du modèle illustré en (a) incluant un ensemble de 2 objectifs $\{a,\ b\}$.

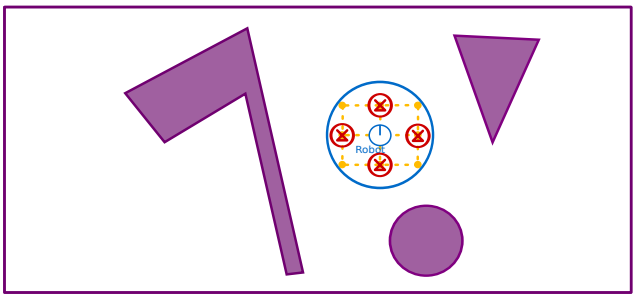


FIGURE 2.11 – État initial du robot explorateur dans son environnement avec ici 4 positions cibles frontières.

Au cours de l'exécution de sa mission d'exploration, le robot va actualiser son GO-MDP au fur et à mesure que des frontières seront ajoutées (Figure 2.12) modifiant à chaque fois la politique en conséquence, cependant la politique du GO-MDP orientera le robot sur une frontière en considérant toutes les autres frontières laissées pour plus tard. La politique construite sera alors valable et optimale jusqu'à la fin de l'exécution de la mission si la visite des frontières connues n'ouvre pas de nouvelles positions frontières (Figure 2.13).

Définition des fonctions de transition et de récompense

L'exécution d'une action de déplacement connectant deux positions connues $a = (p_s, p'_s)$ dans la carte est caractérisée par une probabilité de réussite et un coût. La transition peut alors être définie suivant les déviations possibles du robot lors de son contrôle [Burgard 05, Matignon 12].

Ainsi, la fonction de transition attachée à un déplacement peut exprimer une probabilité que l'agent n'ait pas bougé, ou que l'agent ait atteint une autre position voisine. De plus, dans un cadre d'exploration avec une carte en construction, il est possible aussi d'exprimer des fermetures de boucles potentielles. C'est-à-dire qu'il est possible de connecter des frontières proches entre elles avec une probabilité d'échec proportionnelle à la distance.

Sur la base d'une connaissance exprimée sur les déplacements sous forme d'une fonction de transition $t_D : P \times A \times P \to [0,\ 1]$ et une fonction de récompense $r_D : P \times A \to \mathbb{R}^-$ exprimant les

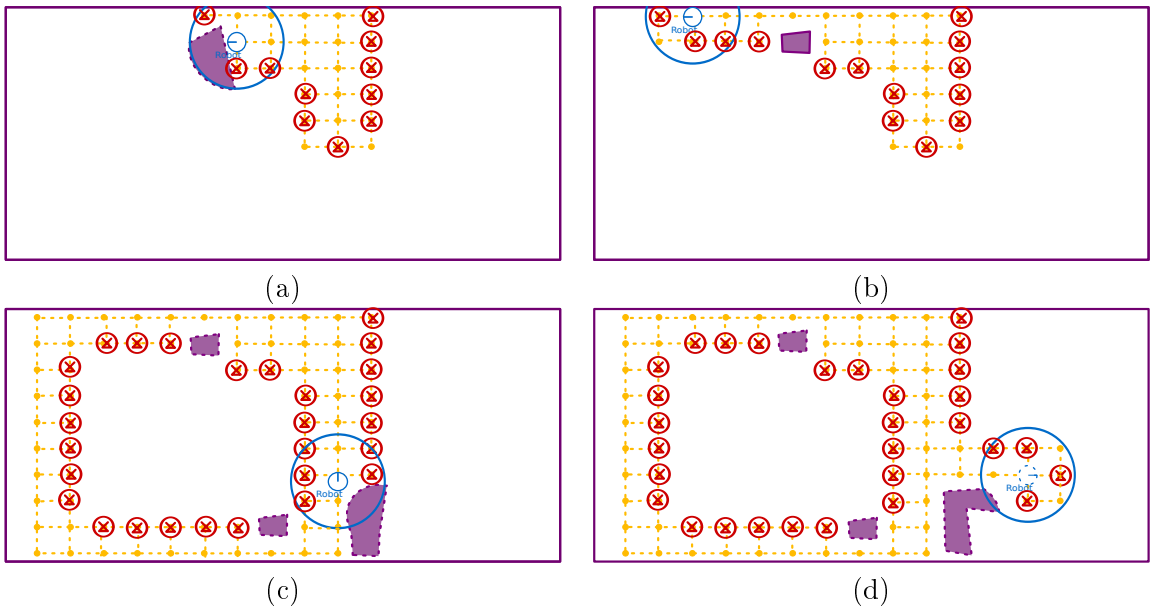


FIGURE 2.12 – Exemples de plusieurs états du robot pendant l'exécution de sa mission d'exploration.

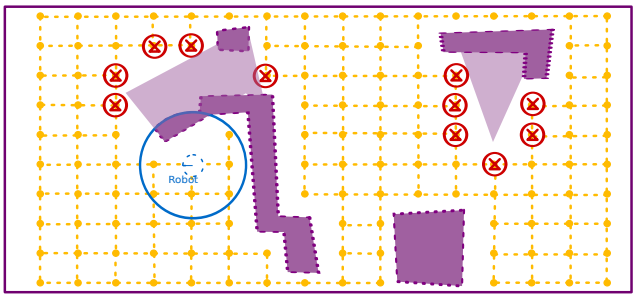


FIGURE 2.13 – Exemple de situation avec encore 11 positions frontières où la politique calculée sera optimale jusqu'à la fin de son exécution au vu des limites d'obstacles (en clair) qu'il reste à découvrir. Dans cette situation le système totalise 1 089 536 états ($532 * 2^{11}$).

coûts des déplacements, il est possible de construire les fonctions de transition t et de récompense r définies sur l'espace d'états du GO-MDP.

La fonction de transition t retourne la probabilité d'atteindre l'état incluant la position suivante et l'ensemble des objectifs actualisés en accord avec les transitions données par les déplacements. L'ensemble des objectifs atteints est alors augmenté si la position atteinte est un objectif.

$$t(s = (p_s,\ G_s),\ a,\ s = (p_{s'},\ G_{s'})) = \begin{cases} t_D(p_s,\ a,\ p_{s'}) & \text{si} \quad G_{s'} = G_s \cup (G \cup S') \\ 0 & \text{sinon} \end{cases}$$

D'une façon similaire, la fonction de récompense est construite relativement aux coûts de déplacement et à un gain perçu pour visiter une nouvelle frontière. Le gain est défini pour chacun des objectifs. Pour l'exploration, le gain peut être défini de façon homogène pour toutes les positions frontières. Pour garantir que toutes les frontières soient visitées, la valeur de gain doit être supérieure au coût cumulé pour traverser l'environnement de part et d'autre.

Le gain perçu est alors proportionnel aux chances qu'une frontière soit effectivement visitée après réalisation de l'action de déplacement.

$$r(s = ((p_s,\ G_s),\ a)) = r_D(p_s,\ a) + gain \sum_{p \in G,\ p \notin G_s} t_D(p_s,\ a,\ p)$$

Particularités de la politique

A chaque pas de temps, la politique optimale construite dicte au robot le chemin le plus court à suivre, lui permettant de visiter l'ensemble des frontières. Par exemple dans la figure 2.12(a), la politique du robot lui dicte de tourner à droite pour visiter la portion de frontière en haut à gauche, pour mieux revenir sur les 10 autres positions à visiter. La possibilité de revenir est remise en cause au fur et à mesure que le robot avance.

Cette stratégie conduit le robot à ceinturer un obstacle jusqu'à fermer une boucle. L'obstacle ceinturé dans l'exemple (Figure 2.13) est alors la bordure extérieure. La mise en œuvre de cette stratégie nécessite de nombreux calculs de politiques en-ligne. Toutefois, un GO-MDP est construit sur $|P| * 2^{|G|}$ soit 540672 états dans le cadre de l'exemple Figure 2.12(a) ($33 * 2^{14}$).

En considérant la borne de quelques millions d'états [Papadimitriou 87, Littman 95, Sutton 98] comme borne maximale pour la résolution optimale d'un MDP, l'utilisation d'un processus décisionnel de Markov orienté par plusieurs objectifs est limité à 16 objectifs (en fonction de la taille de la carte). Cependant, cette solution est parmi les solutions applicables les plus complètes en terme d'expressivité du modèle pour un problème d'exploration en robotique.

2.3 Extensions des MDPs au cadre multi-agents

Le modèle des Processus Décisionnels de Markov a été enrichi pour exprimer des problématiques de planification multi-agents. Le modèle des Processus Décisionnels de Markov Décentralisés (Dec-MDP) propose une modélisation centralisée du problème décisionnel permettant de construire des politiques individuelles exécutables de façon décentralisée [Bernstein 00]. Les politiques individuelles doivent alors coordonner les actions de chaque agent en considérant les capacités perceptives de chacun.

Il est possible de classifier les problèmes de coordination en deux groupes. Les missions coopératives et les missions collaboratives. La coopération est définie par l'action d'opérer conjointement avec plusieurs agents et la collaboration par l'action de travailler à plusieurs personnes à un ouvrage commun. Formellement, dans le premier cas, les objectifs de chaque agent peuvent être disjoints, la réalisation des objectifs ne nécessite pas d'exécution d'actions jointes. Dans le cadre de la coopération, les objectifs communs peuvent être sub-divisés et distribués sur l'ensemble du groupe. Dans le cadre de la collaboration, la réalisation des objectifs est soumise à une coordination plus fine des actions de chaque agent. Les effets attendus sur l'environnement dépendent de la synergie d'actions unitaires réalisées en même temps.

La force du formalisme des Dec-MDPs est de pouvoir exprimer indifféremment des problèmes de coopération ou de collaboration. Le formalisme repose sur deux notions fondamentales : la notion d'observabilité et la notion d'actions jointes. La notion d'observabilité va permettre d'exprimer la différence entre la perception d'un agent et l'état du système. Les actions jointes expriment quant à elles, la possibilité d'exécuter simultanément une action par chacun des agents. Nous verrons que, si le formalisme des Dec-MDP est complet, il n'existe pas d'approche, à l'heure actuelle, pour résoudre de manière optimale des problématiques de taille standard (1000 états,

3 agents et 100 observations). Par contre, il existe plusieurs approches dérivées utilisées dans un cadre multi-robots.

2.3.1 Notion d'observabilité

La notion d'observabilité intervient dans un cadre où il est possible de définir l'évolution du système de façon indépendante à la perception de l'agent. C'est-à-dire que l'agent a des connaissances sur l'évolution du système en fonction des actions qu'il accomplit mais il n'a qu'une perception limitée de cette évolution pendant l'exécution de ses actions. Formellement, cela se traduit par une connaissance probabiliste des observations possibles pour chaque transition effectuée (état initial, action, état atteint).

Si l'on prend la célèbre série télévisé "Dr House", toute l'intrigue tourne autour de l'administration du bon traitement au patient. Le patient évolue en passant par plusieurs états en fonction de sa maladie et des traitements donnés. La recherche en médecine fournit une connaissance sur l'évolution des maladies et de leurs symptômes en fonction des traitements administrés. L'équipe de médecins n'a qu'une observation des symptômes et doit choisir les traitements à administrer. Chercher à optimiser la prise en charge d'un patient, consiste à planifier la succession de traitements à donner en fonction des observations faites de façon à optimiser le rétablissement du patient.

Dans le cadre d'un robot qui se déplace d'une position à une autre dans un environnement ouvert connu, l'état du robot est donné par sa position absolue alors que les capacités du robot peuvent limiter cette connaissance à une localisation relative [Foka 07]. La politique calculée doit minimiser les chemins probables parcourus en respectant les probabilités de perdre le robot. Dans un cadre d'exploration multi-robots avec une planification à l'horizon d'un objectif (la prochaine frontière), l'état courant du système global peut être défini sur la base de la fusion des cartes individuelles [Burgard 05, Matignon 12].

Modélisation

Le formalisme des Processus Décisionnels de Markov Partiellement Observable (POMDP) permet d'exprimer la problématique d'un robot évoluant dans un environnement multi-agents. Ce formalisme permet de tenir compte du fait que la connaissance du robot dépasse ses seules capacités perceptives [Foka 07]. Une modélisation distribuée [Beynier 06, Nair 05], avec un POMDP par agent, ne permet pas d'exprimer l'ensemble des actions jointes. Le POMDP de chaque agent modélise l'évolution de l'environnement multi-agents en fonction de son comportement individuel sur la base d'hypothèses sur le comportement des autres agents. Il est possible d'adapter successivement les modèles individuels au fur et à mesure que les politiques des agents se précisent. Cette approche consiste alors à itérer sur le calcul des politiques individuelles jusqu'à converger vers une politique jointe sous-optimale [Chades 02, Nair 03].

Concrètement, le formalisme des POMDPs, est défini sur un tuple $\langle S,\ A,\ t,\ r,\ \Omega,\ o\rangle$ [Cassandra 94]. Un POMDP est basé sur un MDP standard $\langle S,\ A,\ t,\ r\rangle$ respectivement les ensembles d'états et d'actions et les fonctions de transitions et de récompenses. Le tuple $\langle S,\ A,\ t,\ r\rangle$ est augmenté d'un ensemble Ω des observations et d'une fonction o d'observations.

L'ensemble des observations Ω définit le nombre fini d'observations que peut réaliser un agent. Dans le cadre de notre robot mobile naviguant sur une grille avec une perception d'un rayon égal à une cellule, l'ensemble d'observations Ω inclut toutes les configurations locales possibles (Figure 2.14). Avec 8 cellules autour du robot, chacune définie comme étant libre ou encombrée d'un obstacle, on a 256 observations possibles (2^8) dont certaines sont peu probables.

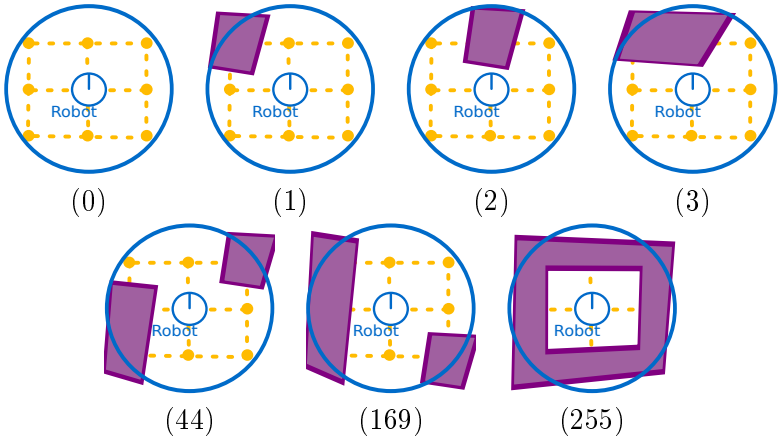


FIGURE 2.14 – Quelques exemples d'observations possibles pour notre robot mobile.

La fonction d'observations $o : S \times A \times S' \times \Omega \rightarrow [0, 1]$ associe une probabilité de percevoir une observation $\omega \in \Omega$ en fonction des transitions $(s,\ a,\ s')$ effectuées par le système. Ainsi, $o(s,\ a,\ s',\ \omega)$ est la probabilité de percevoir ω si l'agent était en s et qu'il a atteint s' en exécutant l'action a.

En robotique mobile, une fonction d'observations $o : S \times A \times S' \times \Omega$ définie sur tout $[0, 1]$ permet d'exprimer un bruit entre la perception de l'environnement et l'environnement réel [Foka 07]. Ce bruit peut résulter des erreurs de mesures ou d'artefacts (e.g. porte en verre avec une perception par vision).

Résolution

Résoudre un POMDP consiste à construire une politique optimale, comme pour un MDP, qui permet de maximiser les gains espérés. Par contre, la politique ne peut plus associer une action à chacun des états du système puisque l'agent n'est pas capable de percevoir, avec certitude, l'état courant. La politique doit alors être construite sur l'historique des observations perçues avec un historique borné à k observations :

$$\pi : \underbrace{\Omega \times \ldots \times \Omega}_{0,\ 1,\ \ldots,\ k} \rightarrow A$$

Lors de l'exécution de la politique, en considérant que l'agent connaît l'état initial du système, la succession de ses actions est guidée par l'accumulation de ses observations. k définit alors le nombre maximal d'actions/observations à exécuter jusqu'à réalisation de la mission. La complexité de la résolution optimale d'un POMDP passe de N-complet (pour un simple MDP) à NP-Complet [Papadimitriou 87] à horizon fini. A horizon infini, un POMDP est un problème indécidable [Madani 99].

Il existe des heuristiques permettant de traiter un POMDP à horizon fini ou infini [Cassandra 98, Aberdeen 03]. Le principe repose alors sur la définition d'un espace discret d'états de croyances. Un état de croyances $b_t : S \rightarrow [0,\ 1]$ est défini par une distribution de probabilités sur l'espace d'états [Astrom 65, Cassandra 94]. Un état de croyances $b_t(s)$ sur l'état s, à un instant t, définit la probabilité que le système soit en s. L'état de croyance peut être mis à jour au fur et à mesure des perceptions que l'agent a de son environnement. Ainsi, la politique sera définie sur l'ensemble des états de croyance possibles B. B est alors construit par rapport aux domaines de variation D_s des probabilités d'être dans chacun des états ($s \in S$).

$$\pi : B \to A \qquad \text{avec} \quad B = \prod_{s \in S} D_s, \quad |B| \simeq \underset{s \in S}{moyenne}(|D_s|)^{|S|}$$

Utilisation en exploration

Un POMDP représente un modèle complet pour exprimer des problématiques en robotique mobile. Les probabilités de transitions représentent les déviations possibles. Les observations et la fonction d'observations permettent de représenter les possibilités de perception et le bruit associé. Par contre, la résolution d'un POMDP étant trés difficile, l'utilisation de ce modèle pour l'exploration en robotique est limitée.

En considérant qu'il est impossible d'utiliser un MDP pour planifier une exploration à long terme (plusieurs (toutes les) modifications possibles de la carte), il est encore plus difficile de s'appuyer sur un POMDP. Cela signifie que la planification optimale d'exploration à long terme et en considérant les capacités perceptives réelles d'un robot n'est pas un problème actuellement solvable pour des tailles d'environnement même limitées à 10×16 positions potentielles (soit, pour rappel, $2^{620} * 1280$ états possible vis à vis des cartes potentielles) et 256 observations.

De façon similaire à l'utilisation d'un MDP sans considérer les évolutions possibles de la carte, il est possible d'imaginer recalculer séquentiellement un POMDP au fur et mesure que la carte se construit. Cependant, il est nécessaire avec cette approche, de considérer, pendant la construction de la carte, les possibles déviations du robot modélisées par son état de croyance. En effet, il se pose la difficulté de déterminer la position des éléments perçus dans la carte à partir d'une position hypothétique du robot [Sim 05].

Cette solution permettrait une planification à horizon limité sur les évolutions des connaissances (jusqu'à la réalisation d'un objectif) en tenant compte des capacités perceptives du robot. Par contre, une planification multi-objectifs en cours de mission (via un GO-MDP), déjà limitée sans observation partielle, ne peut pas être mise en place avec un POMDP. Le nombre d'états de croyances B d'un GO-POMDP construit alors sur $|P|*2^{|G|}$ états ne permettrait pas sa résolution, en cours de mission.

$$|B| \simeq \underset{s \in S}{moyenne}(|D_s|)^{|P|*2^{|G|}}$$

2.3.2 Processus décisionnels de Markov décentralisés (Dec-MDP)

La planification permet de definir une succession d'actions pour contrôler un système en optimisant les gains espérés. Dans le cadre d'un système multi-agents, chaque action a doit décrire l'ensemble des actions individuelles réalisées dans un même temps. On parle alors d'actions jointes. Le formalisme des Processus décisionnels de Markov Multi-agents (MMDP) est basé sur un MDP classique avec un ensemble A d'actions jointes.

Cependant le formalisme des MMDPs impose que tous les agents aient accès à l'état courant du système. Cette restriction ne permet pas de modéliser un problème impliquant une exécution décentralisée de la politique sauf quant l'état est globalement totalement observable (e.g. avec une communication gratuite parfaite). En effet, chaque agent doit pouvoir choisir son action à accomplir au regard de ses propres capacités perceptives. Le formalisme des processus décisionnels de Markov partiellement observables et décentralisés (Dec-POMDP : Decentralized Partially Observable Markov Decision Process) [Bernstein 00] fusionne les capacités expressives des MMDPs et des POMDPs. Ce formalisme permet de différentier les observations individuelles pour calculer des politiques jointes.

Modélisation multi-agents

Un Dec-POMDP est défini de façon similaire à un POMDP avec tuple $\langle S,\ A,\ t,\ r,\ \Omega,\ o\rangle$ respectivement les ensembles d'états S et d'actions A, les fonctions de transition $t : S \times A \times S \rightarrow [0,\ 1]$ et de récompense $r : S \times A \rightarrow \mathbb{R}$, l'ensemble des observations Ω et la fonction d'observations $o : S \times A \times S \times \Omega \rightarrow [0,\ 1]$.

La particularité d'un Dec-POMDP par rapport à un POMDP classique, est que les ensembles d'actions et d'observations sont définis comme l'ensemble des actions et des observations jointes de chacun des agents Ag_i. Ainsi, une action $a = \{a_{Ag_0},\ \ldots,\ a_{Ag_n}\} \in A$ définit les actions réalisées conjointement par l'ensemble des agents $Ag_0,\ \ldots,\ Ag_n$ et une observation $\omega = \{\omega_{Ag_0},\ \ldots,\ \omega_{Ag_n}\} \in \Omega$ définit les observations individuelles perçues simultanément par les n agents.

$$A = A_{Ag_0} \times \ldots \times A_{Ag_n}, \quad \Omega = \Omega_{Ag_0} \times \ldots \times \Omega_{Ag_n}$$

Résoudre un Dec-POMDP consiste alors à construire l'ensemble Π^* des politiques individuelles (une politique pour chaque agent) permettant au système multi-agents d'optimiser ses gains. Les politiques individuelles sont chacune construites sur l'état de croyances de chaque agent $b_{t-Ag_i} \in B_{Ag_i}$.

$$\Pi = \{\pi_{Ag_i} \mid \quad i \in [0, n], \quad \pi_{Ag_i} : B_{Ag_i} \rightarrow A_{Ag_i}\}$$

Classe de problèmes et approches de résolution

Un Dec-POMDP permet d'exprimer un problème multi-agents, où chaque agent n'a qu'une vision partielle du problème. Il a été démontré qu'une résolution optimale d'un Dec-POMDP est excessivement difficile et donc limitée à des problèmes incluant un petit nombre (< 10) d'agents, d'états, d'actions et d'observations possibles [Bernstein 00].

Il est possible de profiter de structures particulières pour chercher à réduire la complexité liée au calcul de la politique optimale [Goldman 04]. Ces structures se formalisent généralement par des hypothèses sur les problèmes traités. Ainsi, la classe de problème la plus citée dans les Dec-POMDPs, est la classe formée par les Dec-MDP. L'hypothèse définissant les Dec-MDPs est que l'état du système est globalement totalement observable. C'est-à-dire qu'une fusion des observations de tous les agents permettrait de déterminer l'état global.

D'autres hypothèses sont utilisées : en considérant une indépendance sur les transitions, les observations et les récompenses acquises par le groupe [Guestrin 02, Goldman 04] ; l'existence de contraintes temporelles [Beynier 06] ; une fonction de récompenses définie sur plusieurs critères [Boussard 07, Abdel-Illah Mouaddib 07] ; une capacité de communication gratuite définie dans un rayon autour des agents [Spaan 08] ou encore, des interactions définies uniquement via une allocation de tâches [Hanna 02, Varakantham 09]. Ces solutions ne s'appliquent qu'à des problèmes particuliers qui, malgré cela, restent difficiles à résoudre.

Une autre approche consiste à proposer des algorithmes permettant de converger vers des solutions sous-optimales. Ces approches se basent sur une décomposition du Dec-POMDP ou du Dec-MDP selon deux axes, une décomposition par agent ou une décomposition par régions.

Une décomposition par agent consiste à itérer sur des calculs indépendants des politiques de chaque agent jusqu'à converger sur un ensemble de politiques sous-optimales [Chades 02, Nair 03]. On peut parler d'algorithme de co-évolution itératif. Une autre approche consiste à identifier des sous-problèmes impliquant des interactions fortes entre les agents au sein du problème global (e.g.

les carrefours dans la circulation urbaine). Ainsi, le Dec-POMDP peut être décomposé en sous-ensembles d'états définissant des sous-problèmes résolues séparément [Boussard 07, Dibangoye 09, Canu 11].

Dans un cadre d'exploration multi-robots

Les états d'un Dec-MDP pour des robots explorateurs peuvent être construits sur la base de la fusion virtuelle des cartes individuelles. C'est-à-dire qu'un état du système peut être défini par la somme des connaissances des robots et de leurs positions dans l'environnement. De tels états correspondraient à l'utilisation d'une connaissance partagée (blackboard) construite avec une communication efficace. Avec des contraintes sur la communication, les observations de chaque agent définissent alors la différence entre cette carte globale virtuelle et les cartes individuelles.

La politique optimale construite sur un tel Dec-MDP permettrait de définir les déplacements optimaux permettant de valider que toutes les zones inconnues ont éte visitées quelles que soient les évolutions de la carte globale virtuelle. La politique jointe optimale inclurait certainement un certain nombre de rendez-vous potentiels permettant aux robots de fusionner leurs connaissances et de re-centrer leurs états de croyances. Malheureusement, pour les mêmes raisons que celles pour lesquelles il est impossible de modéliser un problème mono-robot d'exploration en considérant toutes les cartes potentielles avec un MDP (vis à vis de la taille de l'espace d'états potentiels), il est impossible de résoudre un Dec-MDP pour planifier une mission d'exploration multi-robots jusqu'à l'exploration totale des zones à visiter.

2.3.3 Modélisation et résolution sous-optimale

Force est de constater que, si les formalismes issus des MDPs permettent d'exprimer une problématique d'exploration en robotique, la taille des systèmes générés ne permet pas leur utilisation pour calculer une politique couvrant la totalité des situations potentielles pour un robot (avec un MDP) et, *a fortiori*, pour plusieurs robots (avec un Dec-MDP). Il est alors possible de se rabattre sur une approche ne présupposant pas d'évolution à la carte, c'est-à-dire une planification des actions basée sur la connaissance courante fixe. Une telle approche induit de recalculer régulièrement, en cours de mission, la politique jointe au fur et à mesure que la connaissance est actualisée. Cependant un recalcul régulier de la politique jointe implique une communication efficace et continue.

La planification pourrait donc s'appuyer sur une modélisation d'une problématique multi-agents, multi-objectifs (un objectif par frontière à visiter) et individuellement totalement observable. La politique ainsi construite permettrait de hiérarchiser les frontières et de coordonner les actions des robots. Cependant, l'hypothèse qu'il est possible de communiquer en tout point de l'environnement est fortement restrictive.

Choix successif des frontières à explorer

Dans l'hypothèse d'une communication efficace dans tout l'environnement et d'une bonne connaissance sur la localisation des robots, chaque robot est en mesure, à chaque instant, de localiser les positions des frontières connues et les positions des autres robots du groupe. Il est alors possible d'appliquer une approche par re-planification régulière en cours de mission, à l'horizon d'un objectif [Matignon 12]. A chaque instant, après fusion des connaissances individuelles sur un *blackboard* partagé, le modèle basé sur des MDPs permet de planifier un chemin pour chaque robot jusqu'à la prochaine frontière à visiter.

Sur la base de cette première solution, il est possible de chercher à planifier l'ensemble des chemins des robots leurs permettant de visiter toutes les frontières actuelles. Cette solution permet alors des choix d'exploration tenant compte des frontières laissées pour plus tard. Le problème peut se modéliser par un processus décisionnel de Markov multi-agents orienté par des objectifs (GO-MMDP).

Un GO-MMDP est défini par le tuple $\langle S,\ A,\ t,\ r\rangle$ pour une carte donnée $M = \langle P, C\rangle$ définissant les ensembles de positions et les connexions connues et un ensemble de positions frontières G. Un état s du système est alors défini par les positions $p_{Ag_0},\ \ldots,\ p_{Ag_n}$ des n robots dans la carte (Figure 2.15). L'ensemble des actions A regroupe les actions de déplacements joints.

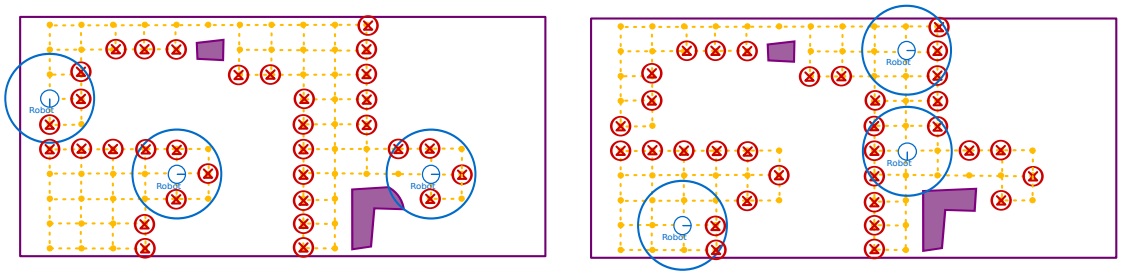


FIGURE 2.15 – Exemple de 2 états multi-robots définis pour une même carte

Les fonctions de transition et de récompenses sont construites de façon similaire à un GO-MDP. Elles sont construites sur la base d'une connaissance des transitions et récompenses individuelles définies sur les connexions entre les positions et un gain pour visiter une position frontière. Notons que, les transitions et/ou les récompenses dans un cadre multi-agents seront affectées pour éviter les collisions entre robots et éviter d'explorer les mêmes frontières. Ainsi, le gain lié à l'exploration d'une frontière ne peut être perçu qu'une fois et pour un seul agent et atteindre un état où deux robots sont sur une même position sera fortement pénalisé dans la fonction de récompense (accident et dégradation du matériel).

Résoudre un GO-MMDP permet de construire la politique optimale $\pi^* : S \rightarrow A_{Ag_0} \times \ldots \times A_{Ag_n}$ qui définit les n plus courts chemins permettant de relier l'ensemble des positions à visiter G et ce sans collision entre les robots. Par contre, la résolution du GO-MMDP s'effectue sur la base de $|P|^n * 2^{|G|}$ états. Si le GO-MDP mono-agent est limité à 16 positions frontières, il faudra considérablement diminuer le nombre de frontières considérées ($|G|$) dans un GO-MMDP résolu en ligne, pour chaque nouveau robot dans le groupe (n).

Il y a une question qu'il est possible de se poser : dans la mesure où l'exécution des politiques est décentralisée, la construction de ces politiques peut-elle se faire aussi de façon décentralisée ? L'idée repose sur le constat que la planification multi-agents est un problème difficile à résoudre et que chaque agent d'un groupe possède des ressources calculatoires. Il devrait alors être possible de mutualiser ces ressources distribuées.

Planification Décentralisée

Les formalismes issus des MDPs (POMDP, MMDP, Dec-MDP et Dec-POMDP) expriment une problématique dans un modèle centralisé qui est donc résolu généralement de façon centralisée. Décomposer un MDP permet de distribuer sa résolution mais la distribution pose des problématiques particulières. Comment décomposer un MDP ? Quels mécanismes mettre en place pour que les résolutions distribuées convergent vers une solution globalement optimale ? Enfin,

une distribution bien organisée ne permet qu'une accélération de la résolution. La résolution optimale de POMDPs, de Dec-MDPs et de Dec-POMDPs reste limitée à de petits problèmes.

Par contre, de nombreuses heuristiques se basent sur une résolution distribuée notamment en planification multi-agents, où les ressources sont distribuées et la solution doit être elle-même une politique décentralisée. Chaque agent va chercher à construire sa propre politique coordonnée avec les autres agents. Ainsi, chaque robot va construire son MDP ou son POMDP défini uniquement sur ses actions propres.

Dans un cadre d'exploration avec localisation efficace, la fusion des connaissances permet (si elle est possible) à chaque agent de posséder les données complètes du problème. Une coordination définie sur cette hypothèse peut alors consister à distribuer des zones à visiter entre les agents après une évaluation relative à la position de tous les agents [Burgard 05, Matignon 12]. Ainsi, l'intérêt d'une frontière pour un robot dépend, entre autres, de la distance de cette frontière avec les autres robots. Le robot planifie son chemin vers la frontière la plus intéressante. Cette solution distribue complètement les calculs (intérêt des frontières et calcul de chemins) sur chaque robot avec une connaissance unique partagée (blackboard). Cette approche s'apparente aux techniques basées sur l'utilisation de récompenses sociales et individuelles [Boussard 08, Beynier 10] où la planification individuelle cherche un compromis entre des récompenses acquises personnellement et des récompenses distribuées sur l'ensemble du groupe.

Avec une re-planification régulière impliquant une re-coordination, la communication doit être efficace tout au long du déroulement de la mission. Cette communication sert à fusionner les connaissances permettant une définition commune de l'état du système. L'utilisation d'un *blackboard* n'empêche pas les robots de perdre, ponctuellement la possibilité de communiquer [Burgard 05]. Dans cette situation, chaque agent continue de planifier sur la base des connaissances issues de la dernière fusion et des avancées individuelles. Le *blackboard* est alors actualisé dès que la communication est de nouveau possible. Cette solution ne permet que de courtes interruptions de communication. Il est aussi possible de mettre en place une exploration "en formation" de façon à garantir l'efficacité de la communication à tout instant [Vincent 08].

Coordination par négociation

Plutôt que de fusionner toutes les connaissances des agents (une tâche qui peut être fastidieuse et compliquée) il est possible de se restreindre aux éléments clefs définissant la coordination. Cette solution passe par la définition de variables d'interaction partagées entre les agents. La communication est alors utilisée pour identifier les variables d'interaction puis définir une option qui fasse consensus dans un processus de planification distribuée.

Les variables d'interaction communes à l'ensemble du groupe, prennent la forme de ressources/tâches à allouer, de règles d'usage à définir (e.g. priorité à droite) et/ou de mise en place de rendez-vous. Nous parlerons alors d'un ensemble de variables d'interaction $V_1 \dots V_k$ associées aux domaines $D_1 \dots D_k$. Les valeurs affectées aux variables d'interaction $V_1 \dots V_k$ associées aux domaines $D_1 \dots D_k$ doivent être communes à l'ensemble du groupe.

Sur la base d'une option o définie comme un vecteur de valeurs associé à l'ensemble des k variables d'interaction, chaque agent va pouvoir planifier sa politique. En considérant plusieurs options possibles, chaque agent va être en mesure d'exprimer des préférences.

$$o = \{x_1, \ \dots, \ x_k\} \in D_1 \times \dots \times D_k$$

Dans une mission d'exploration l'ensemble des frontières à visiter représente les tâches à allouer entre les robots. Une variable d'interaction est définie pour chaque frontière avec pour domaine, l'ensemble des robots candidats :

$$o = \{x_1, \ldots, x_k\} \in D^k, \quad D = \{Ag_1, \ldots, Ag_n\}, \quad k = |G|$$

L'ensemble des agents d'un groupe doit alors tomber d'accord sur une option qui définit un consensus.

La recherche de consensus est une problématique à part entière et notamment en robotique mobile [Ren 08]. Les algorithmes de recherche de consensus définissent un protocole de communication distribué sur l'ensemble des agents leur permettant d'atteindre une solution (si elle existe). Le protocole définit pour chaque agent : quoi, quand et à qui communiquer. Il doit respecter les contraintes données par la topologie des communications définissant ainsi qui peut communiquer avec qui de façon à converger le plus rapidement possible sur un consensus.

La vitesse de convergence dépend du nombre d'options candidates qui seront échangées entre les agents pour être individuellement testées. Il y a $|D_1 \times \ldots \times D_k|$ options possibles. Pour des variables définies sur l'ensemble des k positions frontières il y a n^k possibilités avec n agents. En raison des coûts de communication et de la complexité pour chaque agent de vérifier la validité d'une option, la vitesse de convergence des algorithmes de recherche de consensus est évaluée sur le nombre d'options échangées.

Le consensus peut s'exprimer sous forme de contraintes ou de préférences. Dans le premier cas, une option o est individuellement valide ou invalide. Les solutions consensus sont définies par l'ensemble des options valides pour tous les agents. Dans le second cas, chaque agent exprime sa préférence entre les options possibles. La recherche de consensus consiste alors à trouver une option qui optimise l'ensemble des préférences de chacun.

Dans le cadre de préférences, il est possible de travailler avec des protocoles basés sur des votes qui font se succéder des étapes d'évaluation individuelles d'options et des étapes de sélection d'options candidates.

Dans le cadre de l'allocation de ressources ou de tâches, comme pour l'exploration, les procédures de votes peuvent être simplifiées par des procédures de ventes aux enchères puisque chaque agent représente une option pour chaque variable [Berhault 03]. Les protocoles de ventes aux enchères permettent une allocation rapide des ressources et des tâches entre un ensemble d'agents enchérisseurs. Ces protocoles fonctionnent tel que chaque agent propose une valeur pour chaque variable et la variable est allouée à l'agent avec la plus forte proposition.

2.4 Conclusion

La planification d'une mission d'exploration multi-robots est une tâche complexe mettant en œuvre des modèles qui, s'ils sont complets d'un point de vue de la formalisation, ne permettent pas d'être résolus de façon optimale, sur des problèmes de taille raisonnable (plusieurs dizaines de positions et quelques agents). Ce constat s'applique même pour une planification impliquant un unique robot. Cela s'explique par l'explosion combinatoire des cartes potentielles pouvant être construites.

Les approches délibératives mises en place se basent sur une planification à court terme des actions de chacun des agents. Cette planification est remise en cause à chaque actualisation de la carte construite. Dans un cadre multi-agents, ces solutions se heurtent au problème de coupure de communications empêchant toute fusion de connaissance et coordination d'actions. Cela impose des stratégies d'exploration limitant les coupures de communications entre les agents. Une planification à plus long terme à travers une modélisation multi-objectifs peut permettre plus d'autonomie entre les robots explorateurs d'un groupe en distribuant l'ensemble des zones à visiter.

Dans le cadre des approches proposées dans cette thèse, nous sommes partis de problèmes d'exploration où les robots peuvent communiquer s'ils sont suffisamment proches les uns des autres et possèdent une carte initiale avec des zones inexplorées. Cette hypothèse peut correspondre à plusieurs cas de figure :

- en cours d'exploration, plusieurs robots se croisent, ils ont déjà fait une partie du travail. Après fusion de leurs cartes, ils cherchent à se redistribuer les tâches restantes.
- l'environnement a été modifié, (e.g. après une catastrophe), les cartes connues sont erronées et doivent être actualisées.
- en cas de collaboration sol/air pour des missions de reconnaissance, une connaissance acquise par imagerie aérienne doit être affinée par des robots au sol.

Dans l'ensemble de ces scénarios, l'état initial intègre une connaissance non nulle des tâches d'exploration à exécuter dispersées dans l'environnement. L'objectif consiste à proposer des stratégies permettant au groupe de robots d'être autonome dans le calcul des politiques décentralisées et dans la réalisation de leurs tâches. La solution attendue doit permettre à chacun d'évoluer en étant affecté au minimum par une perte de communication.

Les zones à visiter doivent alors être allouées entre les robots à chaque fois qu'il est nécessaire et que les robots sont en capacité de communiquer. L'utilisation d'une planification multi-objectifs (GO-MDP et GO-MMDP) peut permettre cette allocation totale en respectant les synergies entre les tâches d'exploration. Cette planification multi-objectifs est limitée à quelques objectifs (16 dans un cadre mono-robot).

Dans un premier temps (Chapitre 3), il est proposé une architecture cohérente avec les capacités structurelles des robots mobiles autonomes. Cette architecture se base sur une représentation topologique sémantique de l'environnement de façon à limiter la taille de la carte et par conséquent le nombre de positions frontières considérées à chaque instant. L'objectif est double : alléger les charges de calcul pour la planification en diminuant les données utiles et garantir que les hypothèses mises en œuvre respectent les capacités des robots dans un environnement réel (ouvert et encombré). Cette contribution va permettre de poser une nouvelle base pour repousser l'horizon des politiques construites pour l'exploration.

Sur la base de l'architecture proposée, une mission d'exploration induit plusieurs re-planifications ponctuelles. Ces re-planifications s'effectuent en ligne de façon à construire une politique englobant l'ensemble des objectifs à réaliser. Dans un second temps (Chapitre 4), nous chercherons à nous affranchir de la limite posée sur le nombre d'objectifs en proposant une planification approchée sur plusieurs niveaux. Cette contribution consiste à proposer une application des théories de décomposition de problèmes pour la robotique mobile.

Enfin, distribuer cette solution pour une résolution multi-robots nécessite la mise en place de protocoles permettant d'allouer rapidement l'ensemble de tous les objectifs. Sur la base d'une évaluation individuelle des préférences corrélées sur les objectifs, la solution distribuée proposée (Chapitre 5) vise un consensus via une succession de ventes aux enchères simultanées. Cette troisième contribution apporte une nouvelle approche pour exprimer les synergies entre des objets à allouer dans un protocole rapide de ventes aux enchères (permettant d'allouer des objets dans un laps de temps court).

Chapitre 3

Une architecture de contrôle/commande hiérarchique

La planification permet de déterminer les actions à effectuer en fonction des objectifs à atteindre. L'activité de planifier s'appuie alors sur une modélisation du système à contrôler de façon à exprimer une connaissance sur les aboutissements possibles des actions. Dans le cadre où le système est un ensemble de robots mobiles, le modèle ou la combinaison de modèles permet d'anticiper les évolutions possibles des poses (positions et orientations) des robots en fonction des commandes appliquées. Ces évolutions dépendent de la configuration qui précède le choix de chaque action réalisée.

La planification du contrôle de robots mobiles est confrontée à la résolution de problèmes complexes. En effet, l'espace des configurations et des actions possibles est important. De plus, les robots n'ont qu'une perception limitée et leur connaissance sur leur environnement et sur eux-mêmes peut évoluer. La complexité résultant d'une modélisation multi-robots complète ne permet pas de pré-programmer toutes les commandes potentielles à appliquer dans des environnements ouverts. Une telle démarche implique une énumération de toutes les successions de situations possibles jusqu'à atteindre tous les objectifs.

Les solutions retenues consistent alors à s'appuyer sur des processus décisionnels embarqués basés sur les capacités perceptives, plutôt que sur des comportements pré-calculés. C'est-à-dire que les réponses à une situation sont calculées par chaque robot, à chaque instant, en fonction du nouvel état de sa connaissance. Les processus décisionnels sont alors soumis à des contraintes temporelles fortes avec des commandes qui doivent être définies en quelques micro-secondes. Il est alors difficile d'appliquer des approches plaçant chaque décision dans une optimisation de la résolution de la mission à long terme.

Les architectures hiérarchiques en robotique permettent de diviser la problématique de contrôle sur plusieurs niveaux. Chaque niveau supervise le niveau inférieur selon des directives données par le niveau supérieur. Plus on monte dans l'architecture, plus les algorithmes travaillent sur des données abstraites avec des temps de réponses moins contraints.

Sur cette approche, l'architecture développée dans ce chapitre se divise en un niveau réactif et un niveau cognitif pour permettre à une flotte de robots mobiles de réaliser des missions complexes (plusieurs objectifs) dans des environnements ouverts et encombrés. L'objectif est d'exploiter au mieux les ressources des robots pour calculer des réponses réactives aux événements perçus le long des chemins suivis, tout en permettant à des processus délibératifs de planifier les déplacements à effectuer. Ces travaux ont débouché sur la définition d'une architecture PRDC permettant de séparer les problématiques selon 4 parties : Perception, Représentation, Décision et Contrôle.

Ce chapitre s'organise avec une première section présentant un état de l'art sur les architectures de contrôle hiérarchiques. La section suivante introduit l'architecture "2 axes" et les interactions existantes entre les différentes parties. Chaque partie est ensuite détaillée dans la section 3 pour des missions de navigation et d'exploration. La section 4 cherche à valider expérimentalement l'utilisation de l'architecture PRDC sur une flotte de robots coopératifs. Un accent particulier est porté sur la capacité du niveau réactif à alléger le calcul des politiques de contrôle des robots effectuées par le niveau cognitif. Enfin, le chapitre est finalisé par une conclusion et une discussion sur les orientations futures possibles, pour le contrôle hiérarchique de robots mobiles.

L'interaction proposée entre les deux niveaux (réactif/cognitif), basée sur une carte topologique particulière, a été publié dans la conférence international IAS (Intelligent Autonomous Systems) [Lozenguez 12b]. Un papier sur l'architecture "2 axes" dans son ensemble est actuellement soumis à l'"International Journal of Advanced Robotic Systems" pour un numéro sur le sujet : "Latest Trends in Mobile Robotic Research".

3.1 Fondement théorique, les architectures de contrôle

L'autonomie des robots dépend de leurs capacités à produire des réponses cohérentes et sures, en ligne, aux situations rencontrées et, en même temps, à planifier les actions à effectuer pour réaliser la mission. Les approches hiérarchiques cherchent à diviser les problématiques liées au contrôle de robots sur plusieurs niveaux caractérisés notamment par des contraintes temporelles différentes.

Chaque niveau définit des fonctionnalités unitaires qui peuvent être combinées, au niveau supérieur, pour réaliser les objectifs ciblés. Ces objectifs, s'ils sont paramètrables, permettent de définir de nouvelles fonctionnalités, avec un niveau d'abstraction supérieur. La politique de contrôle global d'un robot consiste alors à superviser la réalisation de tâches fonctionnelles (se déplacer jusqu'à, attraper un objet, ouvrir un tiroir, etc.).

3.1.1 Plusieurs niveaux de contrôle

Un contrôle optimal de robots consiste à définir les successions possibles, des commandes à appliquer aux actionneurs, qui optimisent la réalisation de la mission. Une modélisation complète du contrôle doit alors permettre de prévoir et d'évaluer les conséquences liées au choix des commandes appliquées. L'évaluation de la pertinence de chaque choix s'effectue alors sur une projection jusqu'à la réalisation de la mission. Cette projection doit tenir compte de tous les événements extérieurs possibles et des imperfections sensori-motrices.

De telles approches ne permettent de définir qu'une autonomie pour de petites missions, avec peu de robots et des incertitudes faibles. Une approche consistant à calculer les commandes optimales à appliquer n'est pas envisageable pour permettre à une flotte de robots d'explorer un environnement encombré et ouvert.

D'un autre côté, comme nous l'avons vu pour le contrôle des déplacements (Section 1.1.2), des approches réactives basées sur des stratégies heuristiques fournissent de bonnes garanties sur l'exécution de tâches fonctionnelles. Ces approches s'inspirent notamment du vivant (comme les fourmis), ou des comportements simples individuels ou de groupe permettent d'observer l'émergence d'une intelligence sans planification préalable [Reynolds 87, Lumelsky 90]. Les approches réactives s'opposent alors à une approche délibérative basée sur un modèle fin qui exprime le problème à résoudre. Elle consiste à définir un comportement heuristique, puis à démontrer un certain nombre de caractéristiques (e.g. stabilité, rapidité d'exécution, sécurité, etc).

Les architectures hiérarchiques partent du constat que les approches délibératives permettent de générer automatiquement un comportement vis à vis de problèmes de taille raisonnable et que des comportements heuristiques offrent un certain nombre de garanties vis à vis de la réalisation de tâches fonctionnelles particulières. Il est donc, en théorie, possible de s'appuyer sur des capacités fonctionnelles pour planifier la réalisation d'une mission.

Distinction en 2 niveaux

Les architectures hiérarchiques divisent la problématique du contrôle des robots sur 2 niveaux principaux [Chatila 95, Alami 98, Volpe 01] : un niveau fonctionnel ; un niveau délibératif (Ficure 3.13.2). Le niveau délibératif planifie la succession de tâches à réaliser par le niveau fonctionnel. Une couche exécutive entre ces deux niveaux, veille ensuite, au bon déroulement du plan.

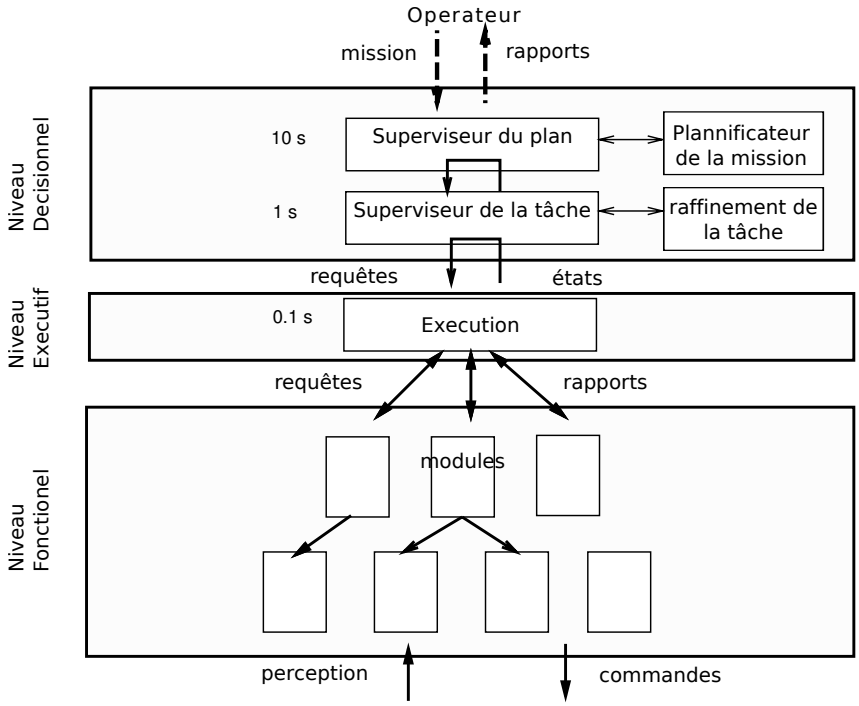


FIGURE 3.1 – Architecture Générique du LAAS [Alami 98]

Le bon fonctionnement d'une architecture hiérarchique dépend de la définition des connexions entre les niveaux. Le niveau exécutif agit comme un traducteur de politiques. A partir de requêtes du planificateur (action) données en fonction de l'état courant fusionnant les rapports du niveau fonctionnel, le niveau exécutif définit les requêtes adressées au niveau fonctionnel [Alami 98].

Contraintes temporelles différentes

Les architectures hiérarchiques se prêtent bien à une programmation multi-threads, avec des processus exécutées en parallèle. Les requêtes et les rapports définissent les données échangées entre les processus. Chaque niveau est autonome pour calculer les réponses adéquates à fournir au niveau inférieur. Enfin, l'indépendance des niveaux les uns par rapport aux autres permet de définir plusieurs niveaux de priorités sur les processus à partir de contraintes temporelles différentes.

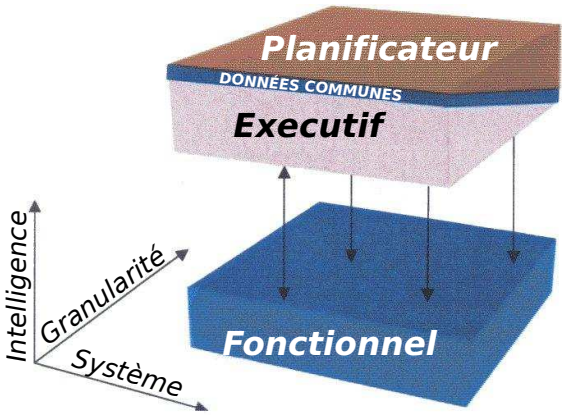


FIGURE 3.2 – Architecture CLARAty [Volpe 01]

Les temps de réponse acceptés sont différents entre les niveaux. Plus le processus est proche des capteurs et des effecteurs plus la réponse à donner doit être rapide. Cette rapidité permet d'augmenter la précision des mouvements vis à vis de stimuli perceptifs, cette précision est notamment gage de sécurité. Il est aussi évident que, plus les mouvements du robot seront rapides, plus les temps de réponse seront contraints.

A une échelle standard où le robot évolue avec des vitesses proches de vitesses humaines, les temps de réponse perception/contrôle sont de l'ordre de quelques dizaines de micro-seconde [Alami 98]. Ce temps comprend l'acquisition de données (image, signal laser, etc.), leur traitement pour en extraire de l'information, le calcul de la réponse et sa mise en œuvre par la commande. Le niveau exécutif doit répondre "instantanément" (moins d'une seconde) pour passer de l'exécution d'une tâche fonctionnelle à une autre et sortir d'une situation délicate au besoin. Le niveau exécutif garantit que les modules fonctionnels activés sont en adéquation avec la situation courante du robot et les objectifs à atteindre. Enfin, la politique associant les modules fonctionnels à activer en fonction des situations rencontrées est calculé au niveau délibératif. L'actualisation ponctuelle d'une politique exécutive peut prendre un peu plus de temps (de quelques secondes à quelques dizaines de secondes) dans la mesure où seule l'optimalité du comportement est mise en jeu.

Les architectures hiérarchiques permettent de positionner les travaux en robotique soit sur des aspects fonctionnels, en décrivant les capacités et des caractéristiques des modules proposés soit sur des aspects décisionnels en proposant des modèles et des algorithme réalistes. La caractérisation des modules fonctionnels doit permettre leur supervision et inversement, les hypothèses retenues pour les modèles délibératifs doivent être cohérentes avec les capacités fonctionnelles.

3.1.2 Application en robotique mobile

Les architectures hiérarchiques trouvent un intérêt certain en robotique mobile avec un niveau fonctionnel défini sur les capacités de déplacement et de localisation et un niveau délibératif centré sur l'ordonnancement de la réalisation de la mission et la planification de chemins. Ce type d'architecture semble bien adapté à la réalisation de missions d'exploration où les tâches de planification requièrent beaucoup de ressources. Il est alors nécessaire, lors de la planification, de s'abstraire des contraintes de bas niveau.

En effet, baser les problématiques de planification sur un niveau fonctionnel ouvre la voie pour des robots mobiles intelligents [Chatila 95]. Il est alors possible d'utiliser plusieurs niveaux

d'abstractions avec des modèles plutôt locaux, basés directement sur les capacités perceptives, et des modèles plus globaux mais moins précis, pour anticiper les déroulements possibles de la mission.

Densité de la carte et horizon de planification

Les solutions délibératives de planification de déplacements se basent, généralement, sur une représentation de l'environnement sous forme de grille. Ces approches, appliquées dans une architecture hiérarchique, supposent un contrôle fonctionnel permettant au robot de se déplacer cellule par cellule. La précision du contrôle repose alors sur un nombre important de cellules. De plus, au niveau délibératif, il est nécessaire d'intégrer la différence entre l'observation et la position réelle du robot ainsi que les possibilités de déviations du contrôle [Foka 07, Burgard 05].

La cohérence d'une architecture hiérarchique appliquée à des robots mobiles repose sur la densité de la carte permettant de planifier les déplacements et de superviser le niveau fonctionnel. Plus la connaissance est dense, plus la planification du contrôle sera précise mais moins la zone considérée pourra être importante. Ainsi, l'horizon de planification dépend de la densité des informations à traiter. Pour planifier sur de larges zones, il faut alors, définir des niveaux de granularité différente sur les cartes utilisées par le contrôle fonctionnel et par le planificateur [Volpe 01, Mourioux 07].

Hiérarchisation "complète"

L'utilisation d'une architecture hiérarchique en robotique mobile permet de définir un niveau fonctionnel plus évolué qu'un simple contrôle cellule par cellule. Ainsi, un contrôle des déplacements plus autonome entre deux positions distantes, peut être supervisé à l'aide de cartes avec un niveau d'abstraction plus fort et donc une densité plus faible. Pour des robots mobiles, il a été proposé de subdiviser l'architecture définie initialement sur 2 niveaux principaux en une architecture sur 5 niveaux d'abstraction [Mourioux 07] (Figure 3.3).

Au dessus de la perception et de l'application de la commande donnée par le niveau (1), le niveau fonctionnel est constitué des niveaux (2) "pilote" et (3) "navigateur". Le niveau délibératif est, lui, composé sans changement par rapport aux architectures précédentes (Figure 3.1), du niveau (4) "planification de chemin" et (5) "générateur d'objectif" (planification de la mission).

Ces types d'architectures fondent alors le module fonctionnel sur un contrôle métrique nécessitant le pré-calcul d'une trajectoire. Pré-calculer la trajectoire au niveau (3) "navigation" [Morette 09, Morette 11], permet de considérer une faible densité de points de passages au niveau (4) "planification de chemin". En effet, en considérant que l'évitement des obstacles est réalisé par le navigateur (3), la planification peut se limiter à raisonner sur des points clefs de type carrefours. Par contre, cette architecture repose sur une localisation globale du robot dans son environnement sur chacun des niveaux.

3.2 Une architecture sur 2 axes

Sur la base des architectures hiérarchiques génériques, il est intéressant d'exprimer des architectures spécifiques à des robots mobiles. En effet, les problématiques autour du déplacement autonome appellent des solutions et des modèles de la connaissance particuliers. Les architectures rencontrées jusqu'ici sont alors spécifiques aux solutions mises en places, avec une forte proportion pour les modélisations métriques impliquant une bonne localisation du robot, à tous les niveaux.

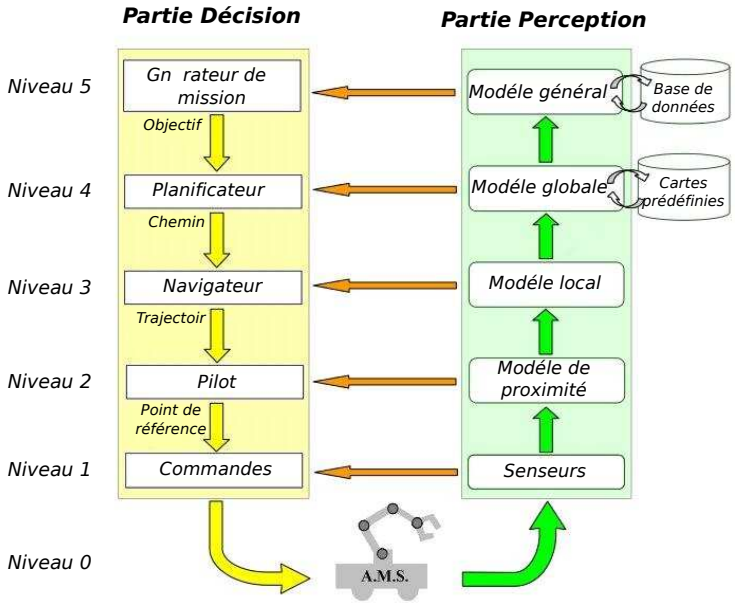


FIGURE 3.3 – Architecture hiérarchique sur 5 niveaux, pour robot mobile [Mourioux 07]

Notre proposition repose sur une architecture sur 2 niveaux, modulaire et qui sépare d'un côté, l'acquisition et le management des connaissances et d'un autre coté, son traitement pour produire les décisions adéquates. L'architecture doit pouvoir intégrer des modules de contrôle et de perception développés selon des approches variées. La hiérarchie développée se base sur un niveau fonctionnel réactif permettant le déplacement du robot et un niveau cognitif pour superviser le niveau réactif.

Au niveau fonctionnel, de nombreuses solutions heuristiques réactives permettent une réalisation efficace des tâches d'appréhension de l'environnement local et des tâches de déplacement. Si ces solutions ne garantissent pas l'optimalité du comportement, elles ont une bonne autonomie et ne consomment que peu de ressources de calcul. Il est alors possible, au niveau délibératif, de profiter des ressources restantes pour planifier sur la base d'action de déplacement définies entre des positions distantes les unes des autres.

3.2.1 Schéma général

L'approche consiste en une architecture sur 2 axes : Connaissance/Décision et Réactif/Cognitif (Figure 3.4). L'axe Connaissance/Décision sépare d'un côté, les tâches d'acquisition, de représentation et de maintien et de l'autre côté, les tâches d'analyse des connaissance suivie des prises de décision. L'axe Réactif/Cognitif permet de définir les processus mis en œuvre avec des niveaux d'abstraction différents.

Une politique haut niveau donne successivement, les directions à suivre de façon à atteindre une cible lointaine. Le contrôle réactif du déplacement est alors indépendant pour suivre au mieux les directions données tout en évitant les obstacles présents. Les données servant à la construction de la politique doivent intégrer une connaissance statistique des capacités de contrôle et de maintien d'une connaissance sur sa localisation.

L'architecture PRDC (Perception, Représentation, Décision et Contrôle) se divise en quatre parties selon un axe horizontal Connaissance/Décision et un axe vertical Réactif/Cognitif. Cette distinction permet aussi de séparer les problématiques et les compétences scientifiques d'ingé-

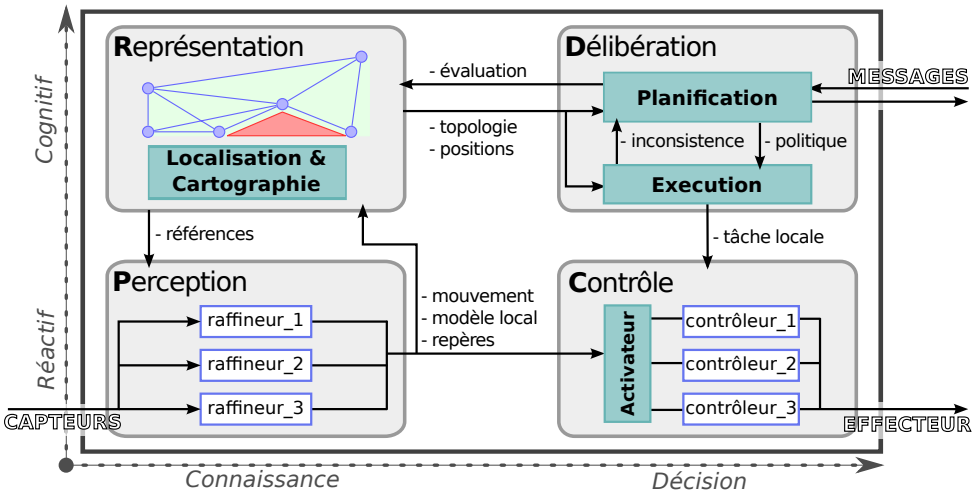


FIGURE 3.4 – Architecture PRDC "2 axes" (Connaissance/Décision × Réactif/Cognitif).

nierie liées à l'informatique embarquée des robots. Chaque partie met en œuvre des processus indépendants (mais en interaction) propres à leur sujet d'expertise :

- la partie Perception (Connaissance, Réactif) traite les informations à partir des données sensorielles.
- la partie Représentation (Connaissance, Cognitif) maintient à jour un modèle de l'environnement.
- la partie Délibération (Décision, Cognitif) supervise le contrôle du robot pour atteindre l'ensemble des objectifs ciblés.
- la partie Contrôle (Décision, Réactif) retourne aux actionneurs une réponse immédiate à une situation locale en accord avec la tâche en cours.

Axe Réactif/Cognitif

A l'image des architectures hiérarchiques standards [Alami 98, Volpe 01], l'architecture proposée s'organise sur 2 niveau verticaux principaux. L'architecture est hiérarchisée sur un niveau réactif et un niveau cognitif. Ainsi, le niveau réactif correspond à l'analyse première des perceptions et aux réponses rapides qui doivent être apportées aux stimuli perceptifs. Avec des temps de réponses très contraints, les approches heuristiques bio-inspirées basées sur l'élaboration de règles simples, permettent des actions précises et sûres. Il est cependant possible d'intégrer des approches basées sur un pré-calcul de trajectoire dans la mesure où les calculs sont rapides et réalisés sur la base des perceptions locales.

Le niveau cognitif intègre cette même dualité connaissance et prise de décision. Le Larousse [7] définit la cognition comme la faculté de connaître. Le niveau cognitif dans l'architecture proposée se divise en une partie Représentation (des connaissances), et une partie Délibération. Ce niveau vise à superviser le niveau réactif de façon à atteindre les objectifs de la mission.

7. Dictionnaires Larousse : http://www.larousse.fr/dictionnaires/francais/

Axe Connaissance/Décision

La particularité de l'architecture consiste à diviser le niveau réactif et le niveau délibératif selon un axe Connaissance/Décision à la manière de [Mourioux 07]. L'approche se base sur un contrôle des robots mobiles par des procédés heuristiques mettant en œuvre des règles simples sous-optimales mais peu coûteuses en ressources de calcul. Ce contrôle est guidé par un niveau cognitif global sans distinction entre la planification de chemin et l'ordonnancement d'objectifs.

Les processus des niveaux peuvent alors s'exécuter en parallèle. En effet, le niveau réactif peut être supervisé par une premier politique pendant que le niveau cognitif optimise cette politique ou élargit son horizon. En revanche, les mécanismes visant à actualiser des décisions à partir de l'acquisition de nouvelles connaissances s'effectuent en série et justifient alors un axe vertical.

Sur chacun des 2 niveaux, un processus horizontal construit la connaissance courante avant de définir le comportement à adopter. Ces processus sont séquentiels et ne peuvent être parallélisés. Le niveau réactif et le niveau cognitif sont divisés en une partie d'acquisition et de fusion des connaissances (parties Perception et Représentation) et une partie de traitement et de prise de décision (parties Délibération et Contrôle) (Figure 3.4).

3.2.2 Niveau réactif

Dans l'architecture PRDC (Figure 3.5), la partie Perception décrit l'environnement à partir des données capteurs et envoie cette description à la partie Représentation et à la partie Contrôle. Le contrôle a pour objectif, de calculer une réponse immédiate à la description locale courante. Cette réponse se formule en terme de commande à appliquer (vitesse, changement de direction). Enfin, la partie Contrôle oriente, dans la mesure du possible, ses réponses de façon à réaliser la tâche ordonnée.

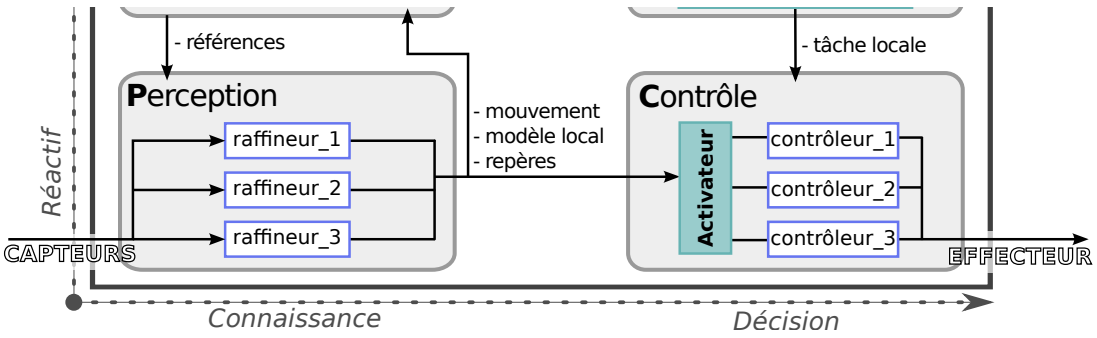


FIGURE 3.5 – Niveau réactif

En robotique mobile, la partie Perception produit un modèle local incluant principalement une description géométrique de la scène (des obstacles et autres éléments). De plus les informations envoyées au niveau cognitif doivent permettre au robot de maintenir une connaissance sur sa localisation dans son environnement.

Partie Perception

La partie Perception raffine les données produites par les capteurs de façon à produire une information utilisable pour la localisation et le contrôle du robot [Siegwart 05]. Cette partie peut être décomposée en plusieurs modules indépendants (raffineurs) pour produire des informations

différentes : localisation relative ; localisation globale ; détection d'objets particuliers, de l'espace navigable ; reconnaissance de lieux ; etc.

Dans le cadre de robots mobiles autonomes, la perception doit pouvoir, au minimum, détecter l'espace navigable local. Cette détection lui permet de contrôler ses déplacements dans un milieu encombré par exemple, via un télémètre laser [Elfe 87] ou via une caméra [Merveilleux 10]. Les capteurs odométriques, quant à eux, permettent une localisation relative qui entraine une déviation au fur et à mesure du déplacement. La caractérisation des flots sensoriels, permet de détecter des particularités dans l'environnement. Par la mise en correspondance de ces particularités, le robot est en mesure de calculer sa localisation globale [Nehmzow 00, Courbon 09, Ulrich 00]. Cette localisation globale, effectuée ponctuellement, permet de ré-initialiser l'odométrie.

La reconnaissance de particularité peut être réalisée de façon brute ou via une caractérisation sémantique. La façon brute cherche directement des correspondances entre un flot sensoriel (ou une partie de ce flot) et une collection mémorisée de flots [Nehmzow 00, Courbon 09]. La caractérisation sémantique [Kuipers 91] consiste à exprimer une connaissance sur l'environnement local, dans un environnement intérieur par exemple, l'espace local au robot peut être classé en : couloir, intersection, pièce, pas de porte, etc. Une reconnaissance sémantique s'appuie, en général sur une description topologique de l'espace local en vu d'identifier les différents passages.

Partie Contrôle

La partie Contrôle fournit des fonctionnalités permettant de réaliser des tâches. Ces fonctionnalités doivent tenir compte des contraintes du robot. Comme pour la partie Perception, la partie Contrôle peut être divisée en plusieurs modules (contrôleurs) définissant des fonctionnalités différentes. Ces fonctionnalités sont valables pour des situations perceptives locales particulières. La fonctionnalité principale attendue pour un robot mobile autonome est définie par sa capacité à atteindre une position tout en évitant les obstacles sur sa route.

Dans plusieurs approches, le robot est contrôlé en minimisant l'erreur entre une trajectoire réfèrente et la configuration courante du robot. La trajectoire (le fil des configurations du robot défini dans le temps) peut être pré-calculée en tenant compte des capacités du robot et des obstacles détectés [Laumond 89, Foka 07, Morette 11]. La trajectoire peut aussi être directement définie par référence à la perception comme une succession de flots sensoriels. [Courbon 09].

D'un autre côté, plusieurs approches contrôlent le robot sans pré-calcul de trajectoire avec des résultats tout à fait acceptables [Siegwart 05]. Les approches réactives proposent, par exemple la mise en place de régles simples permettant de basculer entre plusieurs contrôleurs élémentaires [Brooks 86, Rosenblatt 95, Adouane 09, Vilca 12] comme l'attraction à une cible et l'évitement d'obstacles. Enfin des approches intermédiaires basées sur la planification de chemins permettent de contrôler le robot en définissant uniquement une succession de points de passages.

Les contraintes concernent la partie Contrôle, dans l'architecture PRDC, la vitesse de réponse. Quelle que soit l'approche adoptée dans la définition des modules contrôleur, les algorithmes mis en place doivent retourner rapidement (quelques millisecondes à quelques dizaines de millisecondes) une réponse cohérente sur la base de la description locale fournie par la partie Perception. De plus, le contrôle devra se plier à des règles de sécurité et de confort définies en fonction de l'application visée. La supervision de la partie Contrôle définit, sous forme de tâche, les modules contrôleurs à activer ainsi que leurs paramètres.

3.2.3 Niveau cognitif

Le niveau cognitif (Figure 3.6) a pour mission de superviser le contrôle du robot en fonction des objectifs à réaliser. La supervision implique de construire et de maintenir une connaissance permettant de valider si les objectifs ont bien été atteints. Ce niveau est donc excessivement dépendant des capacités réactives du robot, que ce soit en terme de raffinement d'informations perceptives ou en terme de contrôle.

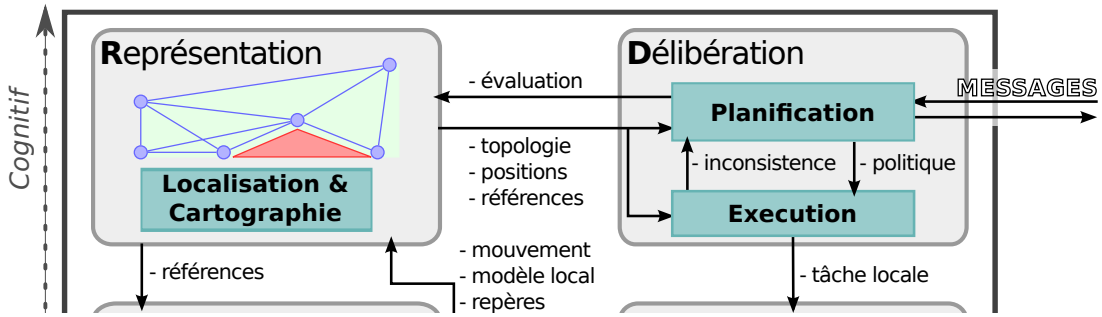


FIGURE 3.6 – Niveau cognitif

Le niveau est divisé en deux parties : représentation des connaissances et délibération. La représentation des connaissances permet de mémoriser l'état du système, elle sert notamment de base de données pour permettre aux processus délibératifs de planifier la supervision du contrôle. La partie Délibération va pouvoir modifier et enregistrer des données dans la partie Représentation.

De plus le niveau cognitif, dans la mesure où l'infrastructure matérielle le permet, va échanger des messages avec les autres agents interactifs présents de façon à enrichir, à partager sa connaissance et à actualiser ses politiques. Par nature, l'architecture de contrôle hiérarchique est embarquée sur les robots de la flotte. Chaque robot possède sa propre base de connaissances et ses propres processus. La communication permet de fusionner les cartes, d'envoyer/recevoir ces ordres et de coordonner les politiques par négociation.

Partie Représentation

Cette partie a pour objet d'organiser et de maintenir la connaissance sur l'environnement des robots de façon à permettre à chaque robot de maintenir sa localisation, de prendre des décisions et de superviser le contrôle réactif. Il existe, dans la littérature, deux types principaux de cartes : les grilles d'occupation et les cartes topologiques. Le niveau réactif de l'architecture PRDC permet des déplacements autonomes entre deux positions distantes, selon des références perceptives locales (e.g. longer un mur). Aussi, nous proposons une représentation des connaissances basée sur une carte topologique.

Unifier le flot de perceptions sur une grille d'occupation permet à chaque robot de se localiser et de cartographier son environnement simultanément [Elfe 87]. Cependant, la localisation basée sur l'odométrie induit une distorsion qui doit être redressée par l'utilisation d'une topologie [Thrun 98]. De plus, la précision de la carte dépend de la taille des cellules et de la quantité d'information contenue dans chaque cellule. Cela implique des tailles de cartes importantes.

La carte topologique est construite comme une représentation en graphe de l'environnement. Les nœuds représentent des poses (position, orientation) particulières du robot et les arcs représentent la connectivité entre les poses [Kuipers 91]. L'existence d'une connexion dans la carte

implique la possibilité pour le robot de se déplacer dans un sens d'un nœud à l'autre. La topologie peut organiser une connaissance sémantique en caractérisant par exemple, les nœuds en : intersection, coin, pas de porte, etc [Kuipers 91] ou organiser des photographies de flots perceptifs pour localiser globalement le robot [Ulrich 00, Korrapati 12] voir le contrôler [Courbon 09]. Une autre approche hybride consiste à organiser topologiquement plusieurs grilles d'occupation ensemble [Simhon 98].

Les cartes topologiques sont donc à la fois plus riches, plus souples et potentiellement plus légères (pour la décision) qu'une grille d'occupation, en concentrant uniquement les données utiles à la localisation du robot. Les nœuds sont définis en fonction des modules raffineurs mis en œuvre. D'autre part les connexions entre les nœuds correspondent aux possibilités de déplacement du robot. Ce modèle est donc aussi adapté à la possibilité d'utiliser plusieurs types de contrôle au niveau réactif. Pour que la partie Délibération soit efficace, nous proposons une carte topologique (*Road-Map*) enrichie en intégrant des informations métriques sur la localisation des nœuds.

Partie Délibération

La partie Délibération connecte le modèle de l'environnement, les objectifs du robot, les contraintes de coordination et les possibilités de supervision du contrôle. Cette partie cherche à définir, à chaque étape, la tâche locale à réaliser par rapport à une politique globale. Elle est divisée en un module de construction de décision et un module exécutif (Figure 3.4). Le module exécutif supervise le contrôle réactif du robot par rapport à une politique définie par le module de construction de décision à la manière des architectures hiérarchiques traditionnelles [Alami 98, Volpe 01].

L'indépendance donnée à la partie Contrôle pour réaliser des tâches fonctionnelles de haut niveau (attraper un objet, atteindre une position, suivre un leader, etc.) induit une certaine incertitude sur la réalisation de ces tâches. Aussi les processus décisionnels de Markov (MDPs) et ses dérivées (PO-MDP, M-MDP, Dec-MDP, etc) sont largement utilisés pour la planification en robotique mobile [Beynier 05, Burgard 05, Boussard 07, Foka 07, Matignon 12]. Généralement, ces approches sont utilisées conjointement avec des grilles d'occupation. En effet ce modèle est basé sur un ensemble fini et dénombrable d'états et d'actions unitaires. Cela permet une connexion naturelle avec un modèle MDP.

Concernant l'architecture PRDC, l'approche proposée consiste à utiliser la puissance expressive des MDPs couplée avec une représentation topologique. Ainsi, le contrôle réactif permet la définition de cartes de faible densité et par conséquent, une planification des déplacements avec un meilleur horizon (plusieurs objectifs à atteindre). Cette solution se base sur un (des) processus décisionnel(s) de Markov orientés par des objectifs et elle unifie planification de chemin et ordonnancement des objectifs.

La partie Délibération construit et exécute une politique $\pi : S \rightarrow A$. Les états sont composés (1) de la dernière position reconnue dans la carte couplée à la description locale de l'environnement (état perceptif du robot) et (2) de l'étape d'achèvement des objectifs de la mission. Les actions correspondent aux tâches fonctionnelles de contrôles disponibles pour l'état perceptif du robot. La connaissance sur la réalisation des actions est portée par la connectivité des nœuds les uns par rapports aux autres dans la *Road-Map*.

La politique construite doit alors être actualisée au fur et à mesure que la représentation s'enrichit. Plutôt que de re-calculer la politique à chaque modification il est possible d'attendre de détecter des incohérences. Le module d'exécution peut détecter ces incohérences à partir des différences entre la topologie à l'instant courant et la politique construite avec une ancienne version de la topologie.

Dans des missions multi-robots (après fusion des connaissances) la politique doit être construite de façon à respecter la coordination entre tous les robots. Cette politique peut être calculée de façon centralisée par un agent tiers (e.g. un des robots) puis communiquer aux autres ou les communications peuvent servir dans un processus distribué où chaque robot calcule sa politique propre tout en recherchant un consensus.

3.3 Développement des 4 parties

L'architecture PRDC a été mise en place et expérimentée pour des missions de navigation et d'exploration multi-robots. L'architecture s'appuie sur des modules réactifs raffineurs et contrôleurs pour définir les capacités fonctionnelles des robots mobiles Les modules réactifs sont basés sur une approche de contrôle topologique de façon à mettre en œuvre une *Road-Map* cohérente avec des capacités réelles.

Cette *Road-Map* doit permettre d'exprimer l'incertitude sur la réalisation des tâches de déplacement. De cette façon, la partie Délibération pourra chercher à optimiser au mieux la réalisation de la mission sans intégrer toutes les variables de bas-niveau. L'indépendance des parties entre elles est limitée aux interactions possibles. Les 4 parties de l'architecture PRDC sont développées avec davantage de détails, dans le cadre d'une application de robots explorateurs. Ce travail nous permet de mieux formaliser les possibilités d'interaction entre les parties (Perception, Représentation, Délibération et Contrôle).

3.3.1 Partie Perception

La perception de nos robots mobiles se compose de modules raffineurs : de localisation relative, de détection de l'espace navigable local et de modules raffineurs de caractérisation de l'environnement local. La détection de l'espace navigable sert principalement aux modules contrôleurs pour éviter les obstacles. La caractérisation de l'environnement local est utile pour la localisation globale du robot dans sa carte en faisant correspondre l'ensemble des caractérisations identiques/proches. La caractérisation sémantique de l'espace navigable permet d'activer ou non les contrôleurs adéquats.

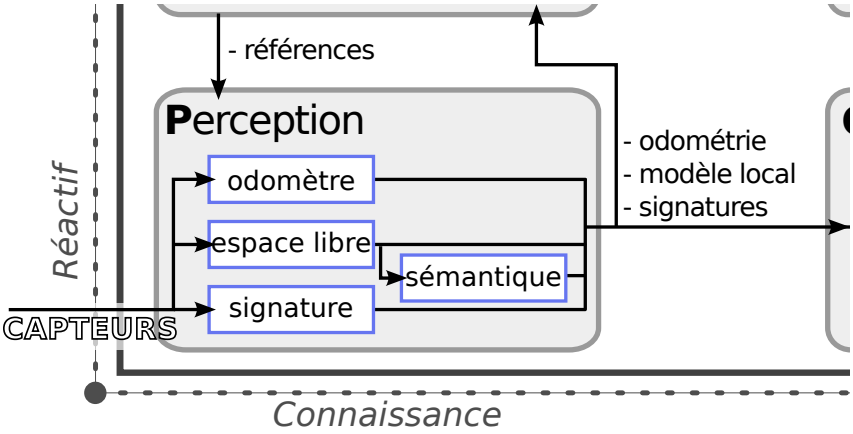


FIGURE 3.7 – Partie perception avec les 4 modules raffineurs.

4 modules raffineurs sont alors mis en place (Figure 3.7) : un raffineur de localisation par odométrie pour une première localisation relative (odomètre), un raffineur de détection d'espace

navigable (espace libre) pour décrire géométriquement l'environnement proche du robot, un raffineur de caractérisation sémantique de l'espace libre (sémantique) et un raffineur de détection de points de repère dans l'environnement (signature).

Descriptions métriques

Les informations métriques utiles au contrôle et à la localisation du robot sont données par les raffineurs "odomètre" et "espace libre". L'odométrie permet d'estimer la position et l'orientation du robot relativement à une position réfèrente. Cette information est construite avec les observations des mouvements du robot. Plus le robot se déplace, plus sa confiance sur sa localisation est faible. L'odométrie peut être ré-initialisée en actualisant le point de référence.

Le raffineur de détection de l'espace navigable permet de diviser l'espace local en zones libres et en zones "obstacles". Le raffineur mis en place décrit l'environnement sous forme d'une liste de polygones délimitant la forme des obstacles dans un rayon donné autour du robot. La définition métrique des polygones, (via une liste de points) est référencée dans le repère 2 dimensions centré sur le robot et défini dans le plan de ses déplacements possibles (Figure 3.8). Cette hypothèse est cohérente avec des capteurs lasers ou visions [Elfe 87, Merveilleux 10].

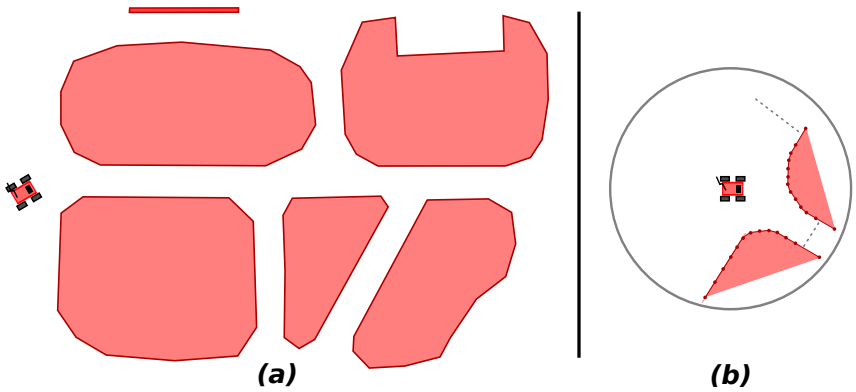


FIGURE 3.8 – (a) l'environnement du robot et (b) la représentation locale correspondante. Dans cette configuration le robot peut détecter 2 passages représentés par les lignes en pointillés gris.

Dans les faits, la détection des obstacles est efficace uniquement dans une zone limitée de l'environnement à cause de l'obstruction des éléments les uns avec les autres et à cause de la précision des capteurs lasers ou visions (nombre de rayons, nombre de pixels, ets). Dans un rayon limité autour du robot, les obstacles peuvent être dessinés avec suffisamment de précision et la perception peut être qualifiée de certaine. Au delà de ce rayon, les mesures deviennent imprécises et il devient difficile de différencier des artefacts des informations réelles dans les données capteurs.

État perceptif topologique

Sur la base de la description métrique de l'espace navigable, le raffineur "sémantique" permet de caractériser topologiquement l'espace local. La sémantique est alors définie sur la base du nombre de passages devant le robot et de leurs configurations. Les passages correspondent aux zones dans l'espace local détecté permettant d'aller au delà de la limite de perception (figure 3.8). Un passage est défini si la distance entre deux obstacles est supérieure à une borne minimale de sécurité. Au delà d'une borne maximale, plusieurs passages peuvent être définis. En considérant

uniquement l'espace à l'avant du robot, 8 états sémantiques perceptifs : libre, à gauche ou à droite d'un obstacle, fourche, couloir, intersection, cul-de-sac et ouverture (Figure 3.9).

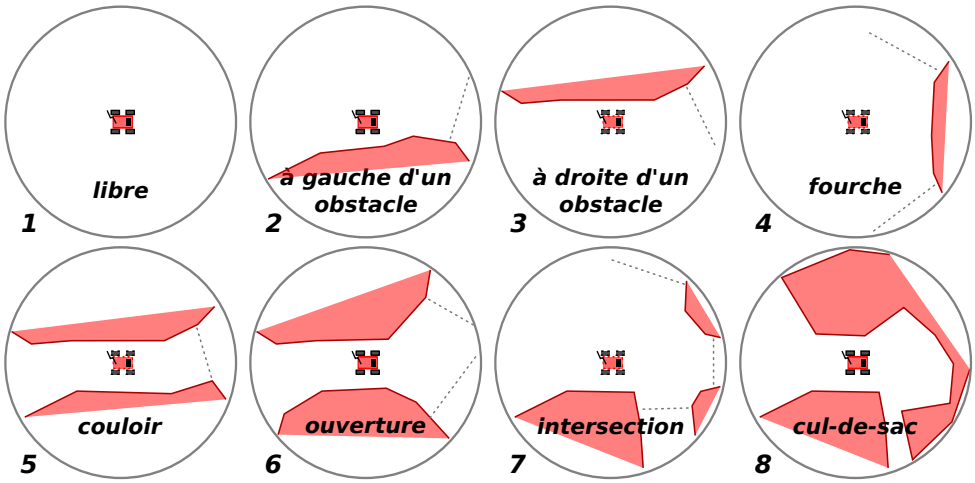


FIGURE 3.9 – Les 8 états perceptifs sémantiques possibles

Dans l'hypothèse où le rayon perceptif est plus grand que la distance maximale séparant deux obstacles, l'espace libre navigable est défini dans toutes les directions possibles. La délimitation de l'espace navigable peut alors être qualifiée de forte. De plus, le robot est quasi-systématiquement dans un état (7) intersections. Dans cette situation les passages sont définis dans les zones inobservables par obstructions. Ainsi, dans un environnement intérieur, les états du robot peuvent se limiter à : couloir, intersection ou cul de sac (pièce).

Détection de signatures

Un autre type de caractérisation de l'espace local consiste en la détection de signatures uniques dans l'environnement permettant de détecter si le robot passe plusieurs fois par un même endroit. L'efficacité de ce raffineur est définie selon l'unicité et l'universalité de la signature, de la taille de la zone de détection et de la possibilité de localiser le robot par rapport à la signature.

La signature est constituée principalement d'une clef construite sur la base du flot perceptif. L'unicité définit la probabilité qu'une même clef soit produite plusieurs fois par l'observation de zones différentes. L'unicité dépend de la technique mise en œuvre et de l'homogénéité de l'environnement. L'universalité définit la possibilité de communiquer ses clefs avec d'autres agents. C'est-à-dire, la probabilité que deux agents produisent une même clef à partir d'une même zone observée. L'universalité dépend de la sensibilité de l'approche vis à vis du capteur utilisé (et de sa configuration).

La zone de détection définit la zone dans laquelle une même clef est détectée. Par nature, le robot ne passe pas 2 fois au même endroit et avec des conditions identiques. Plus la zone de détection est importante, plus les chances de re-détecter une signature est forte. D'un autre côté, plus cette zone est grande moins la re-localisation globale du robot peut être précise. Il est cependant possible d'enrichir la signature d'une connaissance sur la position du robot $(x_r,\ y_r,\ \theta_r)$ par rapport à la signature. Ainsi, il est possible, d'évaluer les positions relatives du robot lorsque celui-ci détecte plusieurs fois une même signature.

$$landmark = (signature,\ x_r,\ y_r,\ \theta_r)$$

Concrètement avec une approche de détection de signature dans une image omnidirectionnelle [Korrapati 12] les conditions de robustesse permettent à des véhicules en zone urbaine de détecter les signatures indépendamment de l'orientation du robot. La signature inclut alors la clef unique ainsi qu'une information sur l'orientation du robot par rapport à cette clef. Le robot sera donc capable de reconnaître sa position dans une intersection quel que soit le chemin par lequel il arrive.

3.3.2 Partie Contrôle

Chaque robot est équipé de raffineurs permettant d'estimer la position du robot et l'espace libre autour de lui. La détection de l'espace libre autour du robot s'accompagne d'une caractérisation sémantique selon 8 possibilités. A partir de ces informations, il est possible de définir plusieurs types de contrôleurs : des contrôleurs métriques définis pour des tâches décrites de façon globale et des contrôleurs topologiques définis pour des tâches décrites relativement à la perception locale.

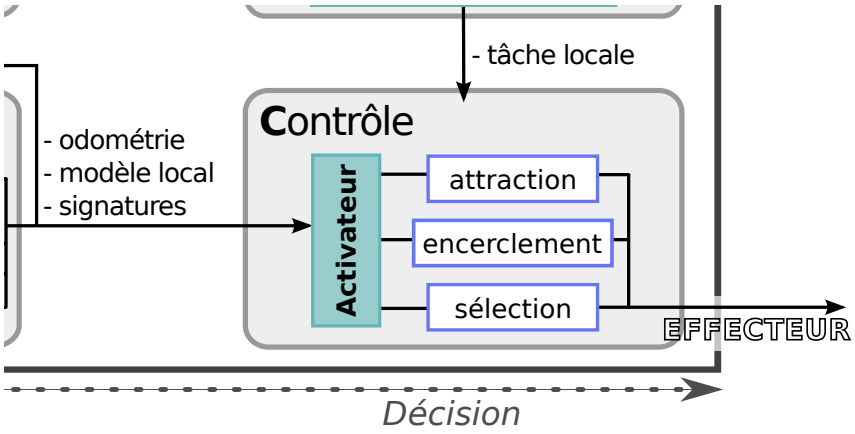


FIGURE 3.10 – Partie Contrôle avec les 3 modules contrôleurs.

Nous proposons d'intégrer à la partie contrôle, 3 contrôleurs différents (Figure 3.10) : Un contrôleur d'attraction à une cible globale (attraction) ; un contrôleur de navigation le long d'un obstacle (encerclement) et un contrôleur de navigation par sélection de passages (sélection).

Le comportement du robot est défini en fonction du contrôleur activé. Le rôle de l'activateur consiste à activer avec les paramètres cohérents, les bons contrôleurs. Les règles d'activation sont données par la tâche locale à exécuter et elles sont définies pour chacun des états sémantiques perceptifs possibles (soit un total de 8 règles à chaque instant)

Contrôle métrique

Le contrôleur d'attraction à une cible est le contrôleur basique valable quel que soit l'état sémantique perceptif. Il consiste en la définition des lois de commande permettant au robot de minimiser la distance entre sa position courante et la position cible. L'évitement des obstacles se fait sur un modèle de graphe de visibilité locale (Figure 3.11(c)).

Concrètement, s'il n'y pas d'obstacles entre la cible et le robot, le robot définit ses commandes de façon à atteindre la cible sinon le robot définit une cible locale intermédiaire. Cette cible locale est choisie dans un premier temps sur le passage le plus proche (Figure 3.11(a)). Ensuite, la cible

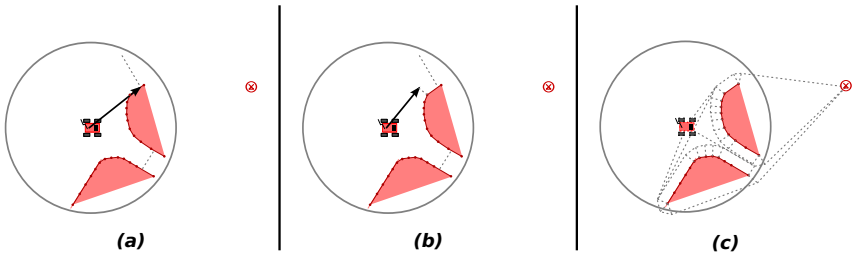


FIGURE 3.11 – Contrôle métrique avec évitement d'obstacle. (a) passage local le plus proche de la cible avec (b) respect des distances de sécurité. (c) graphe de visibilité local.

locale est raffinée de façon à passer à une distance de sécurité minimale de tous les points composant l'ensemble des polygones obstacles (Figure 3.11(b)). Le robot va donc cibler le coin de polygone le plus proche (majoré d'une distance de sécurité) lui permettant d'atteindre le passage vers sa cible finale.

Contrôle topologique

Le contrôleur de contournement d'obstacle (circling) permet de longer un obstacle par la droite ou par la gauche en fonction de son paramétrage. Ce contrôleur est valable dans les configurations où au moins un obstacle est détecté. Les règles de contrôle consistent à définir une cible locale selon le passage le plus à droite ou le plus à gauche. La sélection de la cible locale se fait à la manière d'une attraction avec évitement d'obstacle, de façon à maintenir une distance de sécurité avec les obstacles (Figure 3.12).

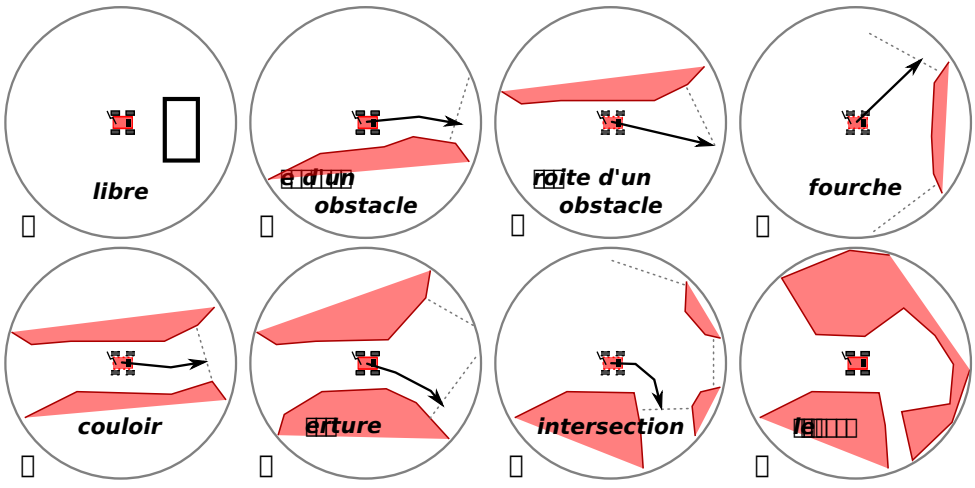


FIGURE 3.12 – Encerclement d'un obstacle par la gauche en fonction des différentes configurations possibles.

L'algorithme de contrôle est alors cohérent dans la mesure où le robot possède déjà un obstacle sur sa gauche (contournement par la droite) ou sur sa droite (contournement par la gauche). Dans les autres cas le comportement du robot va le conduire à s'éloigner de l'obstacle détecté.

Le dernier contrôleur par sélection de passages est paramétré par les numéros de passages (np_1, ..., np_i, ..., np_x). Les passages sont identifiés dans le sens anti-trigonométrique et le contrôleur conduit le robot sur la cible locale au centre du passage np_i s'il y a i passages (Figure 3.13). Le contrôleur est paramétré pour l'ensemble des tailles d'interaction possible.

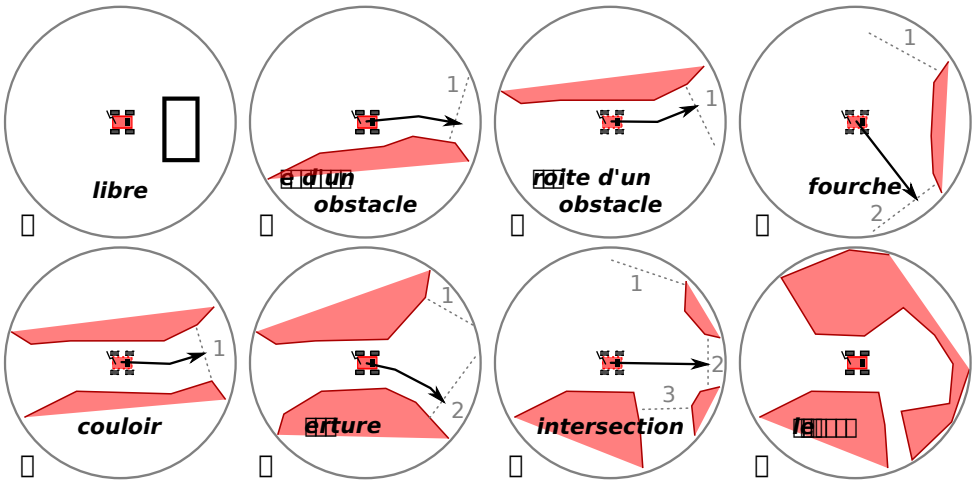


FIGURE 3.13 – Contrôle par sélection de passage (1, 2, 2), soit le passage 1 s'il y a 1 passage et le passage 2 s'il y a 2 passages ou plus.

L'utilisation du contrôleur par sélection est particulièrement adéquate dans des situations d'intersections. En règle générale, le comportement du robot est proche d'un suivi de Voronoï local dans la mesure où le robot vise continuellement le centre d'un passage.

3.3.3 Partie Représentation

La partie représentation (Figure 3.14) gère la carte de l'environnement et les mécanismes mis en œuvre pour maintenir cette carte cohérente. La carte est définie comme une *Road-Map* $< P,\ C >$ (Figure 3.14), soit un graphe stochastique, où : les nœuds P correspondent à des positions particulières et les arcs C représentent les probabilités de connectivités des positions les unes par rapport aux autres. La connectivité est construite en fonction des tâches de déplacement réactives existante (attraction, encerclement ou par sélection de passage).

Modèle pour la carte

Une position $p \in P$ est définie par sa pose dans l'environnement ($x_p,\ y_p,\ \theta_p$), un marqueur temporel (time stamp ts_p, l'état sémantique perceptif $sp_p \in PS$ (libre, couloir, intersection, etc.) et la clef de la signature détectée id_p. Un état sémantique perceptif particulier "indéfini" est ajouté ainsi qu'un id spécifique "nul" pour exprimer le manque de connaissance.

Une connexion $c \in C$ est caractérisée par une position initiale p_c, une tâche locale à exécuter $tl_p \in TL$, un coût individuel de déplacement ic_c et une fonction de déviation d_c donnant la probabilité d'atteindre les autres positions.

$$P = \left\{(x_p, y_p, \theta_p, ts_p, sp_p, id_p) \mid x_p, y_p \in \mathbb{R}^2, \quad \theta_p \in [-\pi,\ \pi] \quad ts_p \in \mathbb{R}^+ \quad sp_p \in PS, \quad id_p \in \mathbb{N}\right\}$$

$$C = \{(p_c, tl_c, ic_c, d_c) \mid p_c \in P, \quad tl_c \in TL, \quad ic_c \in \mathbb{R}, \quad d_p : W \to [0, 1]\}$$

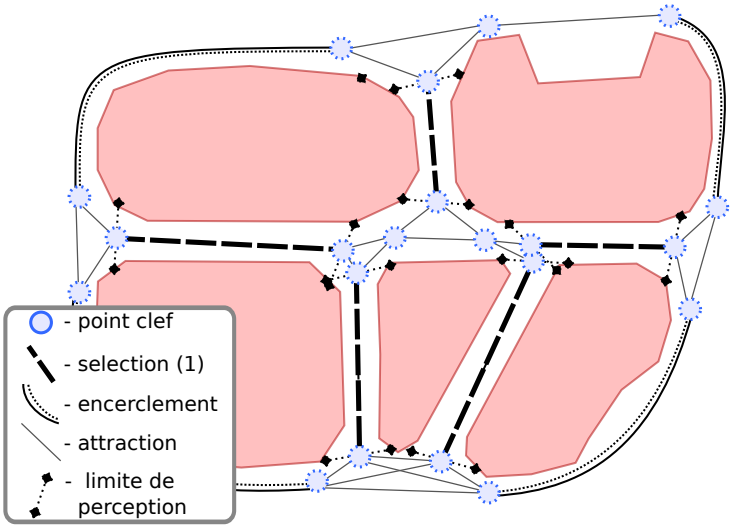


FIGURE 3.14 – Exemple de *Road-Map* construite pour permettre une navigation topologique dans l'environnement.

Le paramètre ts_p (times stamp) marque la création d'un nœud. Il permet d'évaluer une confiance sur sa localisation. Plus ts_p est petit, moins le temps qui sépare cette position de la position initiale est important et plus la confiance peut être forte. Le coût individuel mesure l'impact d'une tâche sur la consommation des ressources. En accord avec le possible manque de connaissance et les déviations durant un déplacement, la fonction normalisée de déviation d_p donne la probabilité d'atteindre une position p' en se déplaçant selon tl_c depuis la position p_c. Les notions de "coûts" et de "déviations" sont utiles pour la planification alors que les "marqueurs temporels" servent à maintenir une cohérence sur la position des nœuds.

Maintenir une connaissance métrique

La construction de cartes topologiques vise à définir ponctuellement des nœuds "positions", et à différencier les nouvelles positions des anciennes positions déjà répertoriées. La cartographie consiste soit à ouvrir de nouvelles directions à explorer, soit à fermer une boucle dans la carte. L'hypothèse d'unicité des signatures détectées permet facilement de fermer une boucle lorsque deux signatures identiques sont repérées.

Il est également possible d'utiliser la séquence de transitions entre les états sémantiques perceptifs, couplée à une connaissance métrique des positions, pour fermer les boucles et redresser une carte [Thrun 98]. Sur ce modèle, le marqueur temporel contenu par chacune des connexions permet de propager une correction en chaînage arrière lorsque qu'une boucle est fermée.

Les informations métriques permettent d'évaluer un déplacement et d'orienter le robot via le contrôleur d'attraction à une cible. Construire cette connaissance métrique sur la base d'une localisation relative implique de tenir compte de possibilités de déviation grandissantes. Une horloge robot st_r est maintenue pendant l'exploration, cette horloge cherche à maintenir une distance temporelle avec la position initiale p_0. Cette horloge est ré-initialisée chaque fois que le robot croise une position connue p avec le marqueur temporel ts_p. Cette ré-initialisation est effectuée en même temps que le robot actualise sa localisation globale avec les données de la position p.

Inversement, le marqueur temporel ts_p d'un nœud p nouvellement créé est initialisé avec l'horloge courante du robot st_r. De cette façon, la confiance sur la localisation du robot et des positions dépend de l'horloge du robot et des marqueurs temporels. Lors d'une fermeture de boucle, le robot s'applique une correction sur sa position, son orientation et son horloge. Le mécanisme de chaînage arrière permet de propager cette correction aux dernières positions créées proportionnellement à la valeur du marqueur temporel de chaque position. Par exemple, un robot construit successivement 4 positions dans la carte : A, B, C et D avant de croiser de nouveau la position A. A ce moment là, il peut actualiser sa connaissance relativement à la position précédemment enregistrée pour A (Figure 3.15).

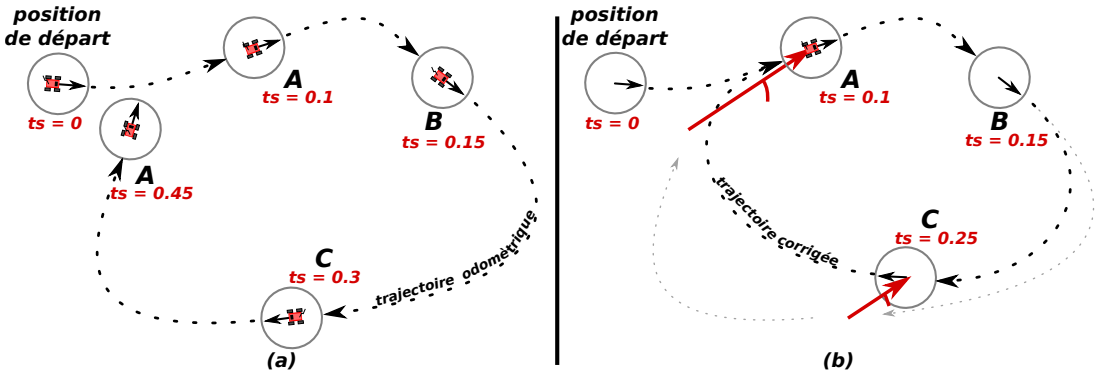


FIGURE 3.15 – Exemple de correction de localisation après fermeture de boucle (a) trace donnée par l'odométrie, (b) carte corrigée.

Au moment de la fermeture de boucle, la localisation du robot et son horloge sont corrigées selon A. La correction est alors propagée sur D, C et B en fonction de leur distance temporelle avec la correction. La correction est plus forte pour D que pour C et elle est nulle pour B qui est plus proche de A en chaînage avant que du robot en chaînage arrière. Dans les faits, cependant, cette correction récursive repose sur l'hypothèse d'une déviation homogène.

3.3.4 Partie Délibération

Une mission d'exploration multi-robots correspond à un problème de voyageur de commerce stochastique remis en cause à chaque étape. Dans l'architecture PRDC, la partie Délibération (Figure 3.16) a pour charge de maintenir une politique individuelle et de veiller à son exécution. Cette tâche est divisée en trois phases : une phase de communication, une phase de coordination et une phase d'exécution.

Lors de l'actualisation de la politique, la phase de communication permet de recenser les robots présents et de fusionner leurs connaissances. En considérant des robots homogènes dans leur conception, la fusion s'effectue sur la base d'un repère global commun (donné avec leurs positions initiales respectives) et d'une détection de signatures supposées identiques d'un robot à l'autre. La phase de coordination calcule la politique courante à exécuter sur la base de la *Road-Map* et d'une allocation des tâches d'exploration. Dans un premier temps, la coordination est faite en associant chaque tâche au robot le plus proche. Enfin la phase d'exécution supervise la réalisation de la politique en reportant des anomalies vis à vis de l'évolution de la représentation topologique. Cette dernière phase est, elle, indépendante des possibilités de communication.

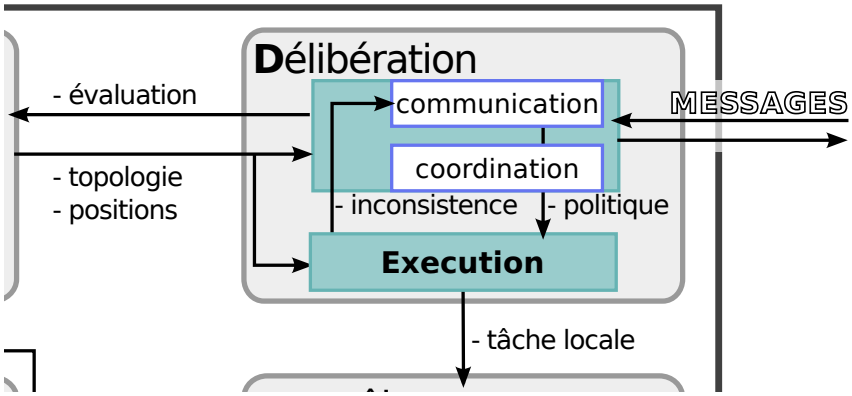


FIGURE 3.16 – Partie Délibération, mécanisme de gestion de la politique de supervision du niveau réactif

Superviser le module réactif

Les méthodes de contrôle topologiques permettent de définir des comportements par rapport à l'espace libre détecté, plutôt que via une cible métrique globale. Ces comportements sont donc moins dépendants des effets de déviation et la politique peut être définie sous forme d'une succession de règles de contrôle topologique donnée en fonction de l'état sémantique perceptif du robot.

Une politique individuelle est définie selon 4 actions : déplacement métrique, contournement par la gauche, contournement par la droite et sélection de passages. Le déplacement métrique met en œuvre uniquement le contrôle d'attraction paramétré avec une cible globale quel que soit l'état sémantique. L'action de contournement par la gauche utilise le contrôleur de contournement uniquement pour les états sémantiques "obstacle à droite" et "couloir". Dans les autres états, cette action utilise le contrôleur d'attraction. L'action de contournement par la droite est définie de façon similaire. Enfin, l'action "par sélection de passage" s'appuie sur le contrôle correspondant et en appliquant une attraction simple dans l'état sémantique perceptif "libre".

Les états du robot sont alors construits avec 3 variables : (1) une évaluation de sa position, (2) son état sémantique perceptif et (3) le chemin $p \in P$ actuellement suivi. Ainsi, au fur et à mesure que le robot se déplace dans l'environnement, la politique définit quelle tâche réactive exécuter en fonction de transition entre états sémantiques perceptifs clefs. Référencer les transitions entre états sémantiques perceptifs sur la base de la dernière signature permet de supposer que les états du système sont totalement observables par le robot. La modélisation de la problématique peut ainsi se baser sur un processus décisionnel de Markov orienté par des objectifs (GO-MDP). Le GO-MDP permet de planifier les déplacements pour visiter plusieurs positions objectifs (cf. section 2.2.4)

Le GO-MDP se présente comme un MDP traditionnel selon le tuple $\langle S,\ A,\ t,\ r\rangle$: états, actions, transition, récompense. Les états sont alors construits relativement à un ensemble d'objectifs G ici, les positions limites demandant à être visitées. Les transitions et les récompenses découlent des coûts de déplacement et des fonctions de déviations introduites dans la *Road-Map*.

Définir l'incertitude de déviation

Une fonction de déviation est définie pour chaque connexion c dans la carte et donne les transitions dans le GO-MDP. Ces fonctions permettent d'estimer l'incertitude liée à l'achèvement

d'un déplacement. Il est possible de séparer 3 sources d'incertitude : les déviations odométriques, la dynamique de l'environnement et les zones inconnues.

La déviation odométrique peut être évaluée à partir d'une représentation métrique, Plusieurs approches consistent à maintenir une ellipse délimitant l'espace de localisation possible autour de la position courante donnée par odométrie [Siegwart 05](Section 5.2.4). En simulant le déplacement du robot, il est possible d'estimer l'ellipse à partir d'une connaissance des bornes d'erreur sur les vitesses relevées. Dans un environnement statique une probabilité de déviation interviendra uniquement sur des actions de contrôle métrique. Lors de contrôle topologique, la déviation odométrique impacte uniquement la connaissance sur les positions des nœuds.

La dynamique de l'environnement dépend des possibilités d'événements indépendants des actions des robots venant modifier les possibilités de déplacements. Certains événements vont induire des modifications dans les déplacements du robot (e.g. obstacle dynamique, porte, etc...) et donc dans la connectivité des positions. La *Road-Map* telle qu'elle est définie actuellement, ne permet pas d'exprimer une connaissance temporelle sur l'apparition d'événements (e.g. ouverture/fermeture de portes). La fonction de déviation exprimera simplement une probabilité de pouvoir passer ou non.

Les zones inconnues conduisent à une incertitude forte sur la réalisation des actions permettant d'atteindre des positions frontières. Explorer un nouveau chemin dans l'environnment peut conduire le robot vers de nouveaux chemins à explorer ou vers une position connue. Atteindre une position connue permet de fermer une boucle et par conséquent de renforcer sa connaissance sur les localisations avant de cibler une nouvelle frontière. Nous avons alors proposé une approche heuristique consistant à évaluer la fonction de déviation des actions visant des positions frontières. Pour 50% des chances, le robot découvre de nouvelles directions à explorer et pour 50% des chances, le robot atteint une position connue proche. Ces chances sont modélisés via une action unique attachée à chaque nœud frontière. Les transitions données pour ces actions sont définis à 50% le robot reste dans l'état frontière et à 50% (partagé équitablement sur l'ensemble des positions proches) le robot atteint une des positions proches et ferme une boucle.

3.4 Validation : réduction de la charge délibérative

La cohérence de l'architecture PRDC (Perception, Représentation, Délibération et Contrôle) a été testée par simulation et sur des robots mobiles réels dans une zone urbaine expérimentale. Les expériences utilisent un unique robot et s'appuient sur un scénario de navigation et un scénario d'exploration. La mission de navigation consiste à visiter des points d'intérêt définis dans une *Road-Map* fonctionnelle et attribuée individuellement aux robots. La mission d'exploration consiste pour un robot, à construire la *Road-Map* de son environnement.

Le scénario de navigation permet de valider dans des conditions réelles l'efficacité de l'architecture pour maintenir la localisation du robot malgré une perception pauvre et une carte construite sur une faible densité de positions. Le scénario d'exploration, testé par simulation, permet de valider l'intérêt de la supervision d'un niveau réactif pour soulager le niveau cognitif dans son calcul de politique. Cela se traduit par la possibilité de considérer plusieurs objectifs lors des phases de planification des déplacements durant l'exploration.

3.4.1 Scénario de navigation

Dans ce scénario, le robot évolue dans un environnement urbain expérimental uniquement à l'aide d'un laser (Figure 3.17). L'espace navigable est délimité par les bords des trottoirs et la *Road-Map* du robot décrit des positions sur la chaussée et des connexions métriques et

topologiques (e.g. longer le trottoir sur x mètres). Le robot doit visiter un certain nombre de points d'intérêt (position dans la *Road-Map*) avant de retourner à sa position initiale. Le laser équipant le robot est positionné à l'avant et est orienté à 35° par rapport à l'horizontale. Le laser est, approximativement, à $50cm$ du sol et peut détecter les trottoirs ou les trous à 1 mètre devant lui (Figure 3.18). Ainsi, l'espace libre est défini par l'ensemble des points d'impacts lasers détecté entre deux distances bornes devant le robot. Inversement, les points d'impacts en dehors de la bande "distance au sol" donnée par les bornes, défnissent les obstacles positifs (e.g. un trottoir) et les obstacles négatifs (e.g. un trou).

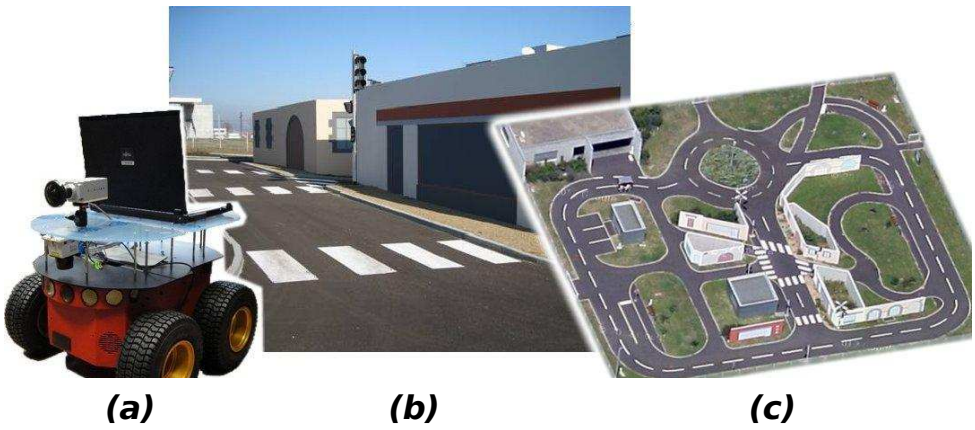


FIGURE 3.17 – (a) un robot pioneer, (b) PAVIN la zone urbain expérimentale et (c) une vue aérienne (Google).

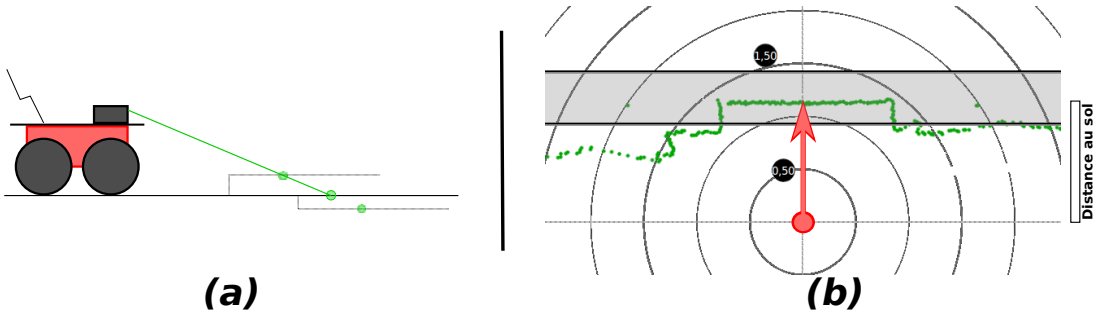


FIGURE 3.18 – (a) positionnement du laser sur le robot. (b) exemple de détection de trottoirs.

La *Road-Map* est composée de 42 positions clefs, elle couvre une zone d'une taille de 32×26 mètres incluant 6 obstacles (Figure 3.14). Le robot considère les distances de 0.8 et 4.0 mètres pour, respectivement, sa distance de sécurité et sa distance de proximité. Les démonstrations ont été réalisées dans le cadre du projet R-Discover [8].

Les positions de la *Road-Map* ont été définies par un opérateur sur la base d'une mosaïque de photos aériennes dans laquelle les bords des trottoirs sont visibles. Les positions le long des trottoirs respectent la distance de sécurité de 0.8 mètres et les possibilités de contrôle topologique sont données de façon cohérente avec la perception du robot. La communication entre le robot et

8. Démonstration : http://www.greyc.fr/node/1629

l'ordinateur de l'opérateur est réalisé par Wifi. L'opérateur construit la carte, initialise les points d'intérêt que le robot doit visiter (entre 1 et 10 points) et envoie les données de la mission au robot. Les communications sont supposées parfaites (sans perte ni erreur).

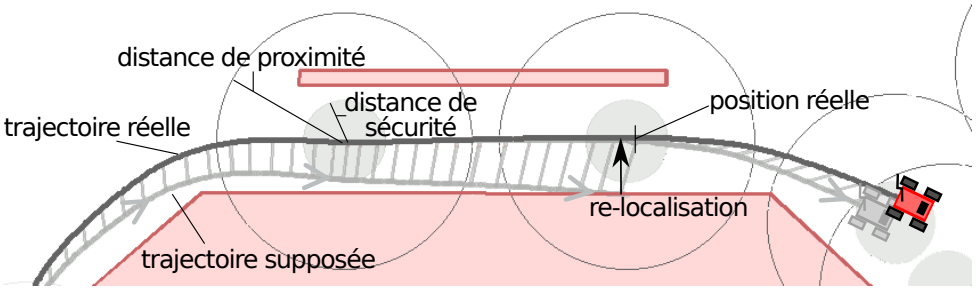


FIGURE 3.19 – Localisation globale du robot sur une position le long d'un couloir.

Localisation globale ponctuelle

Dans cette expérience la re-localisation du robot sur une position de sa *Road-Map* est réalisée sans réelle reconnaissance de signatures. Les positions sont définies sur les trajectoires théoriques du robot réalisées. Ces positions doivent être croisées lorsque le robot se déplace selon l'activation de ses contrôleurs. Ainsi, à l'instant où le robot considère par odométrie, dépasser la position du nœud ciblé, le robot actualise sa position et son orientation avec les données du nœud. Par exemple, sur la figure 3.19, le robot vient juste de corriger sa localisation à la sortie du couloir. On peut voir que cette correction a été imparfaite : on constate des erreurs de positionnement et en orientation. Aussi, il est important pour l'opérateur de construire la carte en respectant les capacités de perception et de contrôle du robot de façon à permettre au robot de se re-localiser avec le plus de précision possible.

Avec cette solution, les ré-initialisations ponctuelles de l'odométrie sont imparfaites (Figure 3.20 et 3.21). Le processus de localisation induit une erreur le long du bord de l'obstacle et cette erreur grandit tant que le robot se re-localise dans une même direction (Figure 3.20). Cette erreur peut être bornée en fonction des plus longues distances de déplacement en ligne droite dans l'environnement.

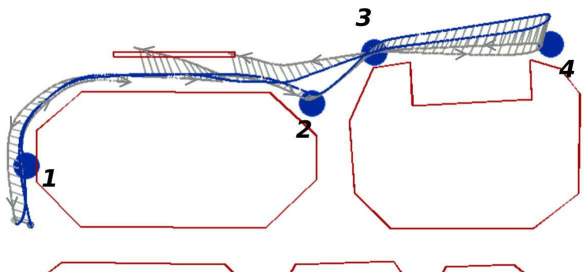


FIGURE 3.20 – Trajectoire réelle (bleu foncé) et supposée (gris) du robot (résultat par simulation sur ordinateur).

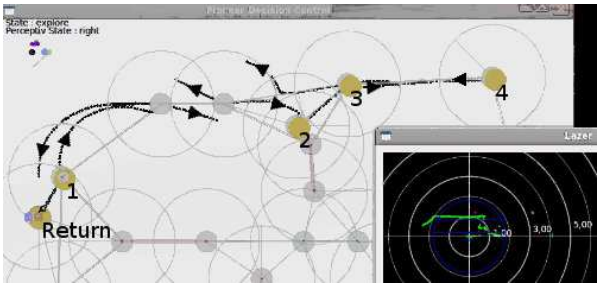


FIGURE 3.21 – Croyance sur la trajectoire du robot à la fin d'une expérimentation réelle (*Road-Map* avec 4 positions objectifs).

Observations réalisées

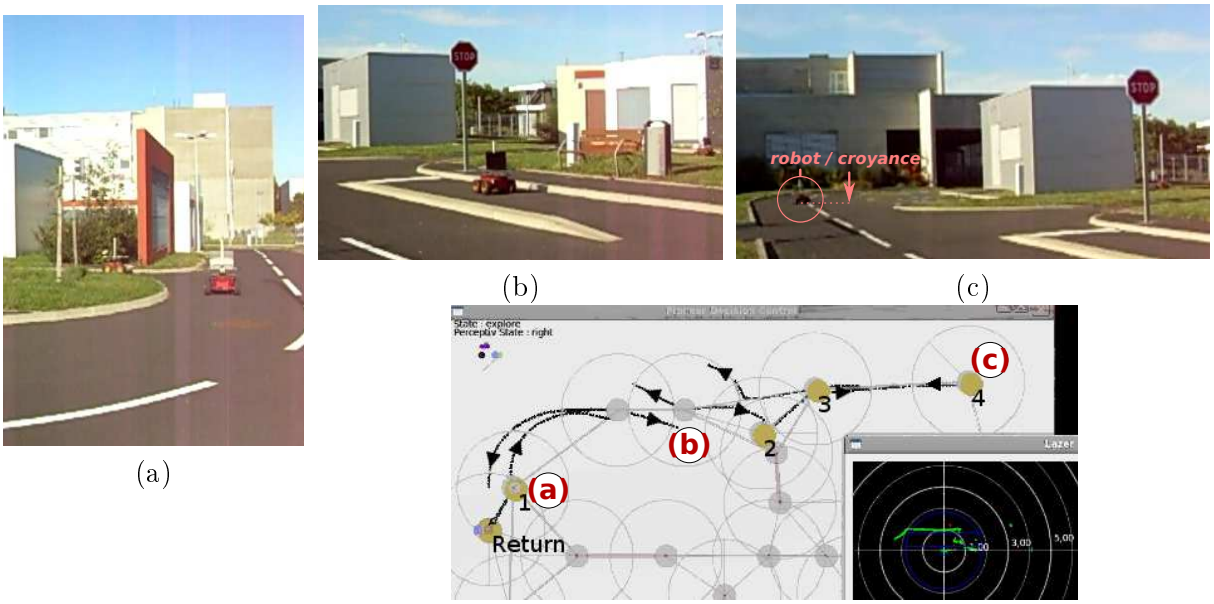


FIGURE 3.22 – Croyance sur la trajectoire du robot à l'aller : (a) le robot détecte le trottoir, (b) il se re-localise une première fois et (c) il atteint sa position objectif la plus lointaine

Les résultats expérimentaux (Figure 3.22 et 3.23) montrent que la *Road-Map* permet au robot de maintenir sa localisation dans la carte de façon à visiter l'ensemble des points et à revenir à sa position de départ. Dans l'expérience permettant d'illustrer nos conclusions (Figure 3.20), à partir de sa carte, le robot calcule une politique pour visiter les points d'intérêt 1, 2, 3 et 4 dans cet ordre. Il commence par contrôle métrique, à atteindre le point d'intérêt 1 et va commencer à détecter le trottoir (Figure 3.22(a)). Le robot longe ensuite le trottoir et il se re-localise sur le dernier nœud le long de ce trottoir (Figure 3.22(b)). Ensuite, uniquement par odométrie, le robot vise les points d'intérêt 4 à 18 mètres qu'il atteint avec une erreur de 3 mètres (Figure 3.22(d)).

Une fois les 4 points d'intérêt visités, le robot cherche à revenir à sa position de départ. A partir du point 4, alors que la localisation du robot a une erreur de 3.8 mètres, le robot vise un point après le stop (Figure 3.23(e)). Le robot arrive alors avec une erreur de 2 mètres (Figure 3.23(d)) et doit éviter le bord du trottoir pour finalement atteindre la position visée (Figure 3.23(e)). L'instant où son odométrie lui signale un dépassement de son objectif (qu'il imagine à 2 mètres à sa gauche) il actualise sa localisation globale puis longe le trottoir (Figure 3.23(f)) pour rejoindre finalement sa position initiale avec 0.5 mètres d'erreur.

Les observations en simulation (Figure 3.20) et en conditions réelles confirment que le contrôle réactif supervisé par une politique délibérative permet au robot de naviguer dans son environnement en utilisant qu'une faible densité de positions clefs. Ce constat est à nuancer, ces résultats sont obtenus dans un environnement connu et statique. Dans les faits, le processus re-localisation a une confiance aveugle dans la *Road-Map* de l'opérateur. Les paramètres de la *Road-Map* (positions, orientations, distances de sécurité et de proximité) doivent être ajustés pour garantir que le robot sera capable de trouver son chemin quel que soit son parcours dans la carte. L'étape suivante consiste alors à permettre au robot de construire et maintenir seul, sa carte.

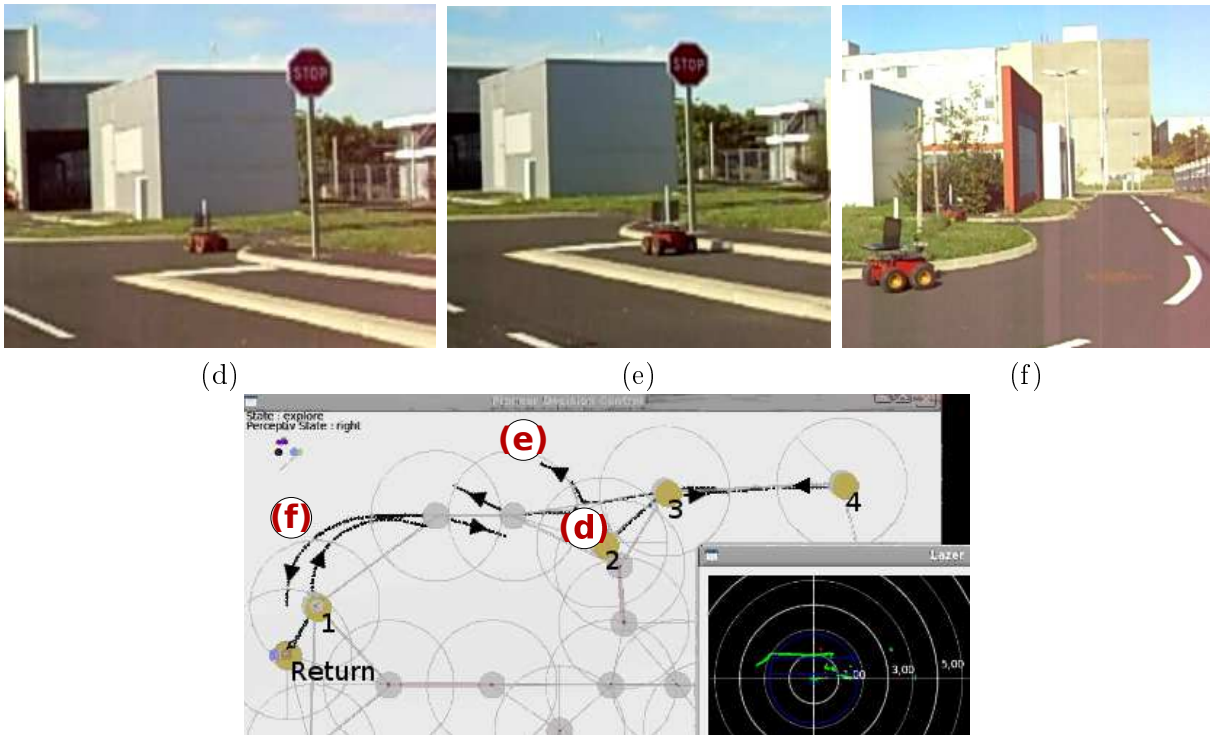


FIGURE 3.23 – Croyance sur la trajectoire du robot au retour : (d) le robot vise la position au milieu du stop mais arrive avec une déviation de 2 mètres, (e) il évite le trottoir et se re-localise le long du trottoir et (f) il continue son retour vers sa position de départ

3.4.2 Scénario d'exploration

Ce scénario permet de valider la cohérence entre les hypothèses posées au niveau réactif (perception et contrôle) et le niveau cognitif permettant la mise en œuvre d'une mission d'exploration. Ainsi, certaines modifications sur la *Road-Map* impliquent des actualisations ponctuelles de la politique dans la partie délibération. Il est montré ici que l'architecture PRDC couplée à un processus décisionnel de Markov orienté par des objectifs permet, en cours de mission, de calculer rapidement de nouvelles politiques englobant plusieurs directions à explorer.

Expérimentation par simulation

Un simulateur sur ordinateur fonctionnant sur des pas de temps constants, calcule, à chaque pas, la position du robot dans son environnement, sa perception locale des obstacles et une estimation de son déplacement. Sa perception est volontairement bruitée si bien que la localisation de l'agent virtuel dévie au fur et à mesure de ces déplacements. En effet, la position du robot est actualisée en fonction de sa vitesse linéaire puis sa vitesse angulaire données en commande et en fonction d'une erreur aléatoire proportionelle aux vitesses appliquées (et selon des bornes paramétrables). Ainsi, le simulateur produit un environnement virtuel similaire à celui pour un robot réel. La perception locale est augmentée d'une détection de signatures. Un certain nombre de zones identifiables dans un rayon donné sont positionnées dans la carte. Chaque fois que le robot entre dans une telle zone, il est capable de lire son identification.

Plusieurs paramètres caractérisent la simulation, les paramètres physiques, perceptifs et cognitifs. Les paramètres physiques décrivent :

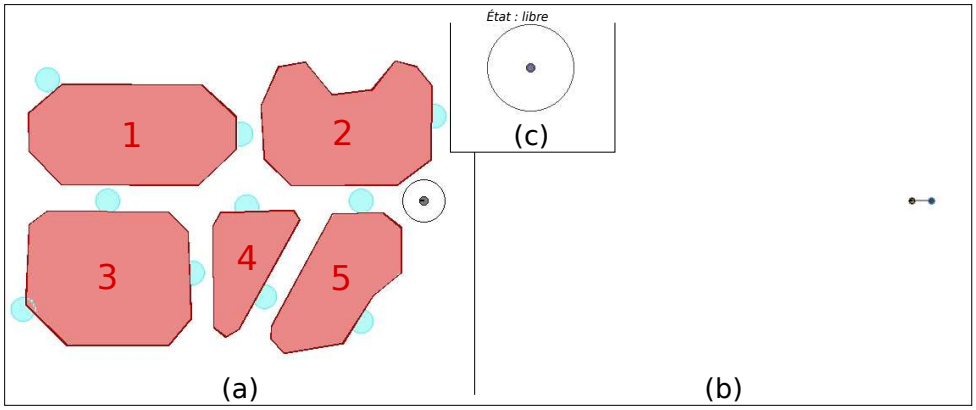


FIGURE 3.24 – Simulation d'exploration, état initial. (a) Environnement virtuel avec 5 obstacles (rouge), et 10 zones identifiables (bleu). Le robot (à l'extrémité) est modélisé avec 2 cercles (son corps et sa limite de perception). (b) État des connaissances du robot. Les informations données par la partie perception sont représentées dans le coin nord-est. La *Road-Map* est initialisée avec un point limite à explorer devant le robot. (c) État perceptif local du robot.

- la position,
- la forme et la taille des obstacles,
- les zones identifiables
- la taille du robot
- ainsi que les vitesses maximales applicables par la commande.

Les paramètres perceptifs incluent :

- les bornes sur la déviation aléatoire de l'odométrie,
- le rayon de détection des obstacles par le robot ainsi que
- un bruit sur les mesures de distance aux obstacles.

Les paramètres cognitifs définissent :

- la distances de sécurité,
- la distance de proximité et
- la distance validant une commutation entre états sémantiques.

Avec des paramètres de simulation cohérents, avec la zone urbaine expérimentale et un robot pioneer, le robot virtuel est capable d'explorer son environnement en se rapportant au mieux à la réalité (Figures 3.24-3.29). L'exploration est réalisée en ajoutant un nœud à la carte, chaque fois que l'état perceptif change de sémantique ou chaque fois que le robot entre dans une zone identifiable. Lorsque une nouvelle position est ajoutée des positions virtuelles à explorer sont ajoutées pour chaque passage local détecté. En simulation, pendant l'exploration de la zone assimilée à la zone urbaine, il a été constaté que la *Road-Map* ne possédait jamais plus de 8 positions virtuelles à explorer simultanément. Soit pas plus de 8 objectifs lors des phases de calcul de politique.

Observations réalisées

Il est difficile de conclure sur l'efficacité de la stratégie d'exploration. Dans les faits, pour être concurrentiel par rapport à d'autres méthodes existantes dans la littérature, il est nécessaire d'ajouter un mécanisme évitant d'explorer chaque passage selon les 2 directions. La difficulté

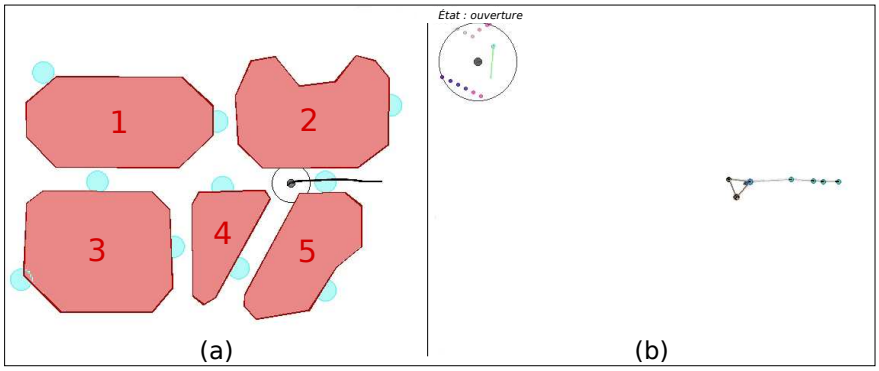


FIGURE 3.25 – Création des premiers nœuds dans la carte. Le robot est actuellement dans l'état 'ouverture' en sortant du couloir entre les obstacles 2 et 5. Le robot a alors 2 directions possibles à explorer. 5 nœuds ont été créés depuis le début de l'exploration : le nœud 1 position initiale ; le nœud 2 détection d'un obstacle sur la gauche ; le nœud 3 entrée dans le couloir ; le nœud 4 découverte d'une signature et le nœud 5 ouverture du couloir.

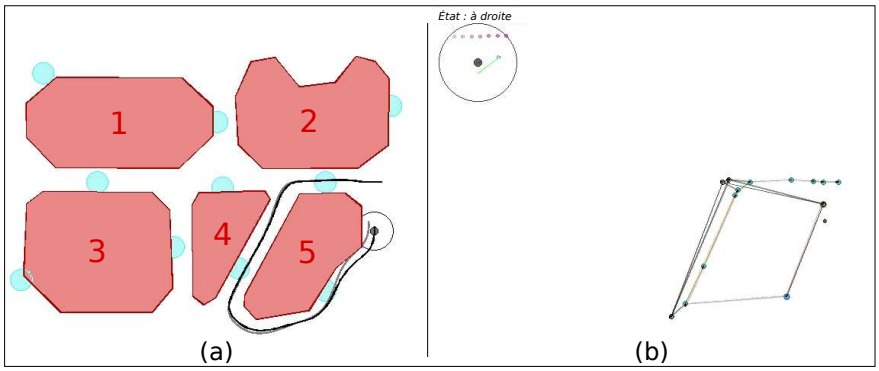


FIGURE 3.26 – Premier encerclement d'obstacle. La politique d'exploration conduit le robot à encercler l'obstacle 5. A cette étape d'exploration la *Road-Map* du robot possède 4 positions limites à explorer. La trajectoire odométrique du robot (en gris) commence à s'écarter visiblement de sa trajectoire réelle (en noire).

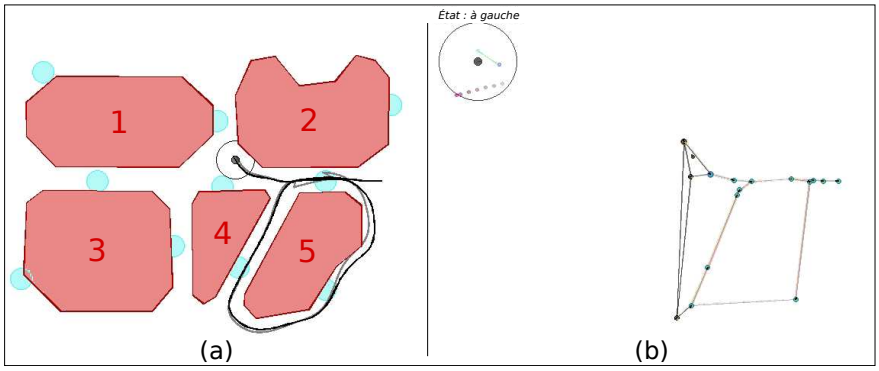


FIGURE 3.27 – Fermeture de boucle. Le robot reconnaît sa premier signature lui permettant de fermer une boucle dans la carte. La fermeture de boucle induit une première correction de la localisation odométrique. Cette correction est propagée par chaînage arrière sur les derniers nœuds créés. Le robot choisit ensuite d'explorer la position limite la plus au nord.

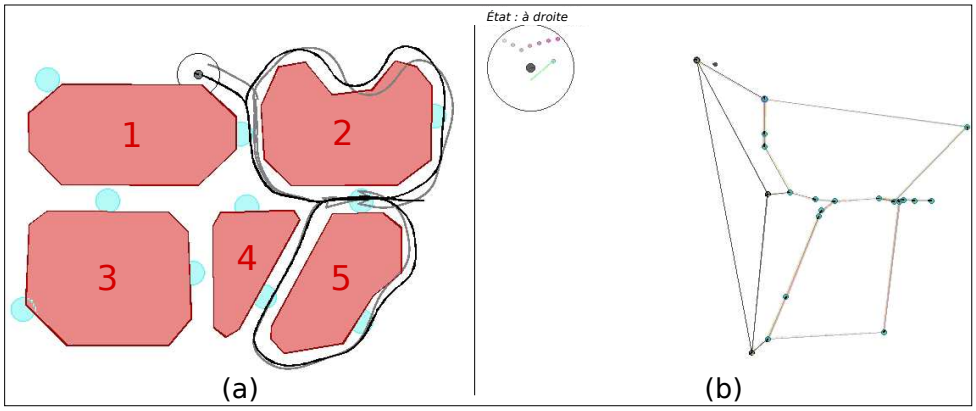


FIGURE 3.28 – Explorer les extrémités. Après encerclement de l'obstacle 2, le robot continue d'explorer le nœud le plus au nord. La politique calculée conduit le robot à explorer les positions aux extrémités. Cette stratégie permet au robot de minimiser le nombre de positions frontières derrière lui, soit, les positions qui nécessiteraient (sans évolution de la carte) un retour sur ces pas.

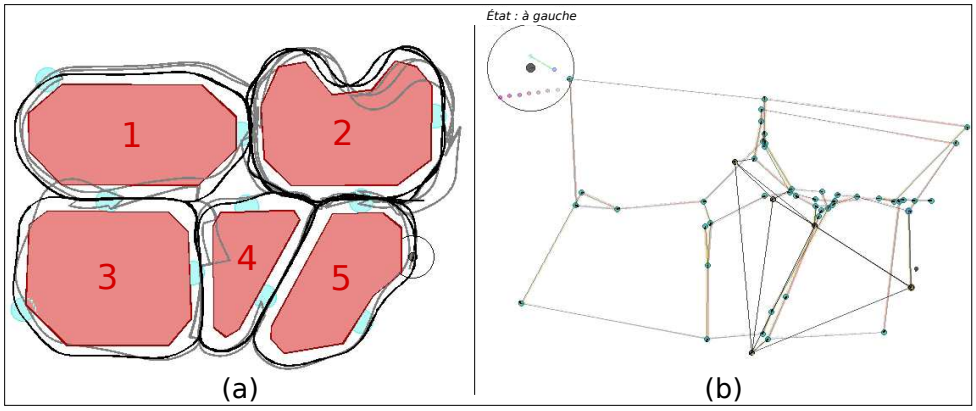


FIGURE 3.29 – Exploration dans un état avancé. Due à l'orientation des nœuds dans la *Road-Map*, le robot doit explorer les chemins topologiques dans les 2 directions possibles. A cette étape de la mission, tous les obstacles ont été encerclés mais il reste des positions frontières dans les orientations opposées. En effet, encerclé un obstacle dans un sens (par la gauche par exemple) ne permet pas actuellement, de définir les nœuds et connexions dans le sens opposé. Ici, seul l'obstacle 2 a été encerclé dans les deux directions et le robot continue son exploration.

réside alors dans la non-symétrie des trajectoires du robot et dans la non-symétrie des états perceptifs construits uniquement sur les passages devant le robot.

De plus, pour être complet, l'exploration topologique doit être complétée de tâches d'explorations métriques. Concrètement (Figures 3.24-3.29) dans l'exemple présenté, le robot ne cherche à aucun moment de se détacher des 5 obstacles centraux pour connaître les limites de l'environnement. La solution sera alors de proposer des tâches de contrôle par attraction à des cibles positionnées dans des zones inconnues.

En revanche, sur les hypothèses liées à la détection de signatures, les expérimentations permettent de valider la capacité du robot à construire sa *Road-Map* tout en maintenant sa localisation par des actions de déplacement autonomes. La *Road-Map* est alors définie sur la base d'une carte topologique qui inclut des informations métriques. L'approche par carte topologique permet ainsi de ne considérer que les positions clefs dans l'environnement.

De plus, l'exploration sur la base de l'architecture PRDC nécessite des actualisations de politiques uniquement ponctuelles et sur la base d'une *Road-Map* de faible densité. Malgré une planification englobant plusieurs objectifs (jusqu'à 8), le modèle reste contenu dans des dimensions acceptables. Ceci permet de valider la possibilité d'une planification multi-objectifs en cours de mission robotique. Enfin, il s'avère que les politiques ainsi construites à chaque ajout d'une position limite, conduisent le robot sur des tâches aux extrémités de la délimitation du connu et de l'inconnu (Section 3.28). Ainsi, sans qu'il n'y ait de récompense particulière pour la fermeture de boucle, le robot favorise régulièrement des déplacements induisant des fermetures de boucle. Ces expérimentations mettent en évidence l'idée que favoriser la fermeture de boucle (permettant de renforcer les connaissances métriques), n'est pas incompatible avec une exploration rapide de l'environnement.

3.5 Conclusion

La conception de robot mobile autonome et, *a fortiori*, d'une flotte de robot mobiles autonomes, est une tâche difficile. Une part des difficultés repose sur les mécanismes qui permettent, sur la base des perceptions, de définir la commande à appliquer. Les architectures hiérarchiques proposent de diviser ces mécanismes sur plusieurs niveaux, pour différencier notamment la réalisation de tâches fonctionnelles et la planification. L'objet de notre approche consiste à appliquer ce concept, à des robots mobiles autonomes capables de réaliser des tâches fonctionnelles de types topologiques.

Ce chapitre présente une architecture complète pour des robots explorateurs basée sur une connaissance individuelle et topologique. Cette architecture (nommée PRDC) permet de séparer les difficultés liées à la mise en œuvre du contrôle/commande des robots mobiles en décomposant l'architecture en 4 parties : Perception, Représentation, Décision et Contrôle.

Chacune des parties a été définie dans le but d'une réalisation de missions réelles et de façon à montrer les dépendances entre elles. La structure de chaque partie est modulaire pour pouvoir ajouter facilement des fonctionnalités aux robots. En effet, la définition topologique de l'environnement du robot permet une grande flexibilité quant aux modules réactifs (raffineurs et contrôleurs) mis en œuvre.

Cette architecture repose sur un niveau réactif (perception, contrôle) supervisé par un niveau cognitif (représentation, délibération). Cette solution permet la réalisation de tâches fonctionnelles sans nécessiter une consommation importante de ressources de calcul. Ainsi, le niveau cognitif profite de conditions idéales : des perceptions raffinées pour avoir des données perti-

nentes (e.g. états perceptifs sémantiques), des actions haut niveau effectives sur une à plusieurs secondes (e.g. longer un mur) et des ressources de calcul économisées.

Dans les chapitres 4 et 5, nous nous intéresserons plus particulièrement au développement du module délibératif. En effet, l'objectif de l'architecture PRDC consiste à définir un nouveau cadre de travail permettant aux approches délibératives de planifier avec un horizon plus lointain et ce, malgré une explosion des possibilités de l'évolution du système.

Orientations futures

Dans des travaux futurs concernant l'architecture PRDC, il serait intéressant d'ajouter des fonctionnalités supplémentaires (raffineurs perceptifs et contrôleurs) de façon à étendre les capacités des robots. Une collection plus riche de modules imposerait alors de mettre en place une normalisation des interactions inter-parties possibles, pour rendre le développement de chaque module dans chaque partie plus indépendant.

Parmi les fonctionnalités à ajouter, la partie Représentation doit être améliorée pour permettre un apprentissage robuste de la carte. Actuellement, les robots se contentent de localiser des transitions d'états perceptifs, le long de leur déplacement, dans un sens donné. Il devrait être possible, sur la base d'une logique sur ces transitions, d'évaluer, au moins, les transitions en sens inverse et, au mieux, des transitions futures, au-delà du champ perceptif.

A plus long terme, l'architecture PRDC peut être enrichie pour appréhender des environnements plus dynamiques. L'idée repose alors sur l'ajout de modules raffineurs spécifiques à une différenciation des éléments dynamiques (e.g. piétons). Cette différenciation doit permettre la localisation de l'agent par rapport aux éléments immuables et un contrôle adapté aux éléments mobiles avoisinants. Enfin, la complexification du niveau réactif devrait appeler une nouvelle interaction entre les parties Délibération et Perception de façon à pouvoir superviser une perception devenue active.

Chapitre 4

Décomposer des politiques individuelles multi-objectifs

L'application initiale visée par le travail de cette thèse consiste en l'exploration d'une zone par des robots mobiles autonomes. Nous nous plaçons volontairement sur une stratégie de coordination par modélisation distribuée avec des re-calculs réguliers des politiques, au fur et à mesure que la connaissance est actualisée. Ainsi, sur la base d'une allocation des tâches à réaliser, en construisant un modèle décisionnel par agent, le problème s'apparente au problème du voyageur de commerce résolu individuellement avec des ressources restreintes (temps de calcul de l'ordre de la seconde avec l'ordinateur embarqué par le robot).

Le modèle individuel peut s'exprimer à l'aide d'un processus décisionnel de Markov orienté par des objectifs (GO-MDP). Les objectifs G de chaque robot à chaque instant correspondent aux positions frontières qui lui sont attribuées et qu'il doit visiter. Il y a potentiellement autant d'objectifs qu'il y a de portions frontières à visiter, ce nombre est borné par le nombre total de passages dans l'environnement.

À chaque instant, chaque robot doit avoir en mémoire l'ensemble des objectifs atteints. En effet, chaque choix de déplacement dépend des objectifs qui ont été atteints ou non, soit $2^{|G|}$ possibilités. Pour pré-définir les actions optimales à réaliser en considérant toutes les possibilités, le GO-MDP intègre, au minimum, $2^{|G|}$ états. Un GO-MDP peut donc être qualifié de MDP excessivement large, c'est-à-dire que même une simple énumération des états est fastidieuse.

Le problème de planification par résolution de MDPs larges est envisagé avec une contrainte forte sur la rapidité des calculs mis en œuvre. Une première section présente les approches existantes proposées pour accélérer la résolution de MDPs larges. Une attention particulière est portée sur les techniques de décomposition du fait de leur compatibilité avec l'utilisation de GO-MDP. Il apparaît alors qu'une résolution optimale d'un GO-MDP est difficilement envisageable.

Dans une seconde section, l'approche présentée cherche à partitionner un MDP définissant la dynamique du système (sans intégrer les objectifs) de façon à construire, ensuite, une hiérarchie de GO-MDP approchée. Cette solution permet de focaliser la résolution de problème multi-objectifs sur seulement quelques politiques locales, région par région. La troisième section propose ensuite, plusieurs séries d'expérimentations permettant de valider l'approche dans un cadre multi-robots. Les expérimentations permettent d'évaluer statistiquement la qualité et la rapidité d'une coordination par une modélisation distribuée, hiérarchique et approchée. Enfin, le chapitre se termine par une conclusion et une discussion sur des orientations futures.

Nos travaux sur la résolution hiérarchique de la planification des déplacements ont été présentés lors de la conférence internationale PAAMS (Practical Application of Agent and Multi-Agent

Sytems) [Lozenguez 11a] et font le sujet d'un article dans le journal international MAGS (Multi-Agent and Grid Systems) [Lozenguez 12a].

4.1 Fondement théorique, les processus décisionnels de Markov décomposés

Les algorithmes standards de résolution optimale de MDP sont limités à des problèmes de l'ordre de quelques millions d'états [Papadimitriou 87, Littman 95, Sutton 98]. Plusieurs travaux ont cherché à dépasser cette borne empirique en tirant partie de structures internes des problèmes. Il existe 2 principales approches : les MDPs factorisés et les MDPs décomposés.

Le principe des MDPs factorisés repose sur l'existence de liens de causalité dans l'évolution des variables des états composant un MDP. Cette approche peut être utilisée dans des problèmes où les transitions entre les états du système peuvent être définies de façon factorisée sur les variables composants le système, plutôt que sur les états eux-mêmes. Par exemple, les choix de directions à prendre, lors du contrôle d'un véhicule, dépendent du niveau des ressources, uniquement lorsque celui-ci atteint un niveau critique. Ainsi, la politique sera identique quels que soient les états ayant un niveau de ressources supérieur au niveau critique. La politique pourra être définie et calculée en tirant partie du fait que certains groupes d'états ont des actions et des valeurs similaires.

Une résolution factorisée d'un MDP permet de profiter d'une définition factorisée du problème. Comme présenté section 2.1.2, les réseaux Bayèsiens permettent d'exprimer une telle connaissance du problème. Dans le cadre des problématiques qui nous intéressent (basées sur le déplacement de robots), il n'y a pas, *a priori*, la possibilité de factoriser la modélisation. En effet, si l'on souhaite rester général, chaque position dans la carte n'est similaire à aucune autre.

L'approche par processus décisionnel de Markov décomposé, consiste à diviser un problème global en plusieurs sous-problèmes qui seront résolus séparément. Cette approche est alors plus adaptée à la planification de déplacements. Une carte peut être divisée en régions de façon à planifier une route globale composée d'une succession de chemins locaux.

4.1.1 Motivation et modélisation

L'idée maîtresse des MDPs décomposés consiste à agréger les états fortement connectés en sous MDPs de façon à décomposer le calcul des politiques [Dean 95, Boutilier 99]. La décomposition d'un MDP permet alors d'accélérer le calcul de politiques optimales en construisant une hiérarchie entre des problèmes locaux et une solution globale.

L'algorithme de référence "Value iteration" (Section 2.2.3) consiste en des ré-évaluations approximatives successives de la valeur des actions en chaque état jusqu'à stabilisation de la fonction de valeur $V^{\pi^*} : S \to \mathbb{R}$. La résolution optimale d'un MDP demande un nombre d'opérations polynômial en la taille du problème ($|S|$ et $|A|$) [Papadimitriou 87]. En théorie, avec une décomposition homogène en k parties, on devrait pouvoir passer de la résolution d'un MDP large à k résolutions polynômiales de sous-MDPs de tailles divisées par k.

Formalisme d'un Processus Décisionnel de Markov décomposé

Un Processus Décisionnel de Markov Décomposé (DMDP) est défini par un tuple $\langle S, A, t, r, P\rangle$ avec $\langle S, A, t, r\rangle$ définissant un MDP standard et P son partionnement. P est défini par l'ensemble des k régions qui divisent l'espace d'états S de façon à former une partition (Figure 4.1). C'est-à-dire que chaque état s de S appartient à une et une seule région. Dans les faits, il est nécessaire

que chaque état appartient à une unique région pour que la résolution couvre l'ensemble des états sans intersection entre les politiques locales.

$$P = \{R \mid R \subset S\}, \quad \bigcup_{R \in P} R = S \quad \text{et} \quad \forall R, R' \in P, \quad (R \cap R') = \emptyset$$

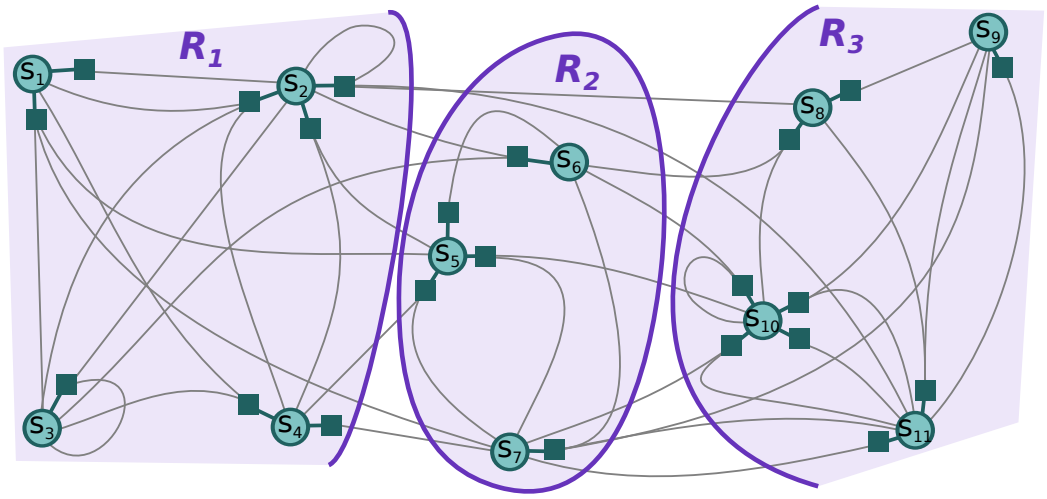


FIGURE 4.1 – Exemple de Processus Décisionnel de Markov à 11 états décomposé en 3 régions.

A partir d'un MDP décomposé en k régions, il est possible de construire un sous-MDP $\langle R, S, A, t, r \rangle$ pour chaque région R de la partition P. La résolution optimale d'un sous-MDP consiste alors à itérer uniquement sur les états s contenus par la région R (Algorithme 3).

Algorithme 3 : Local value iteration

Données : un MDP $\langle S, A, t, r \rangle$, $R \subset S$, $V : S \to \mathbb{R}$, $\gamma \in [0,\ 1]$ et $\epsilon \ll \min_{s \in S,\ a \in A} (\ |r(s,\ a)|\ )$

Résultat : π^* et V sur R (selon ϵ, γ et V sur les états en dehors de R)

répéter
  initialiser V' : $V' \leftarrow V$;
  pour $\forall s \in R$ **faire**
    Actualiser $\pi^*[s]$: $\pi^*[s] \leftarrow \underset{a \in A}{argmax} \left(r(s,a) + \gamma \sum_{s' \in S} t(s,a,s') V'[s'] \right)$;
    Actualiser V : $V[s] \leftarrow r(s, \pi^*[s]) + \gamma \sum_{s' \in S} t(s, \pi^*[s], s') V'[s']$;
jusqu'à $\forall s \in R, \quad |V'(s) - V(s)| < \epsilon$;

Dépendance entre les régions

Selon l'équation de Bellman [Bellman 57], la valeur d'un état s pour une action a dépend de la somme des valeurs des états atteignables proportionnellement à la probabilité de les atteindre. A chaque itération de l'algorithme 3, pour chaque état de la région, toutes les actions sont testées pour sélectionner l'action optimale à l'itération courante. La résolution d'un sous-MDP

dépend donc de l'ensemble des états potentiellement atteignables depuis les états de la région (Figure 4.2).

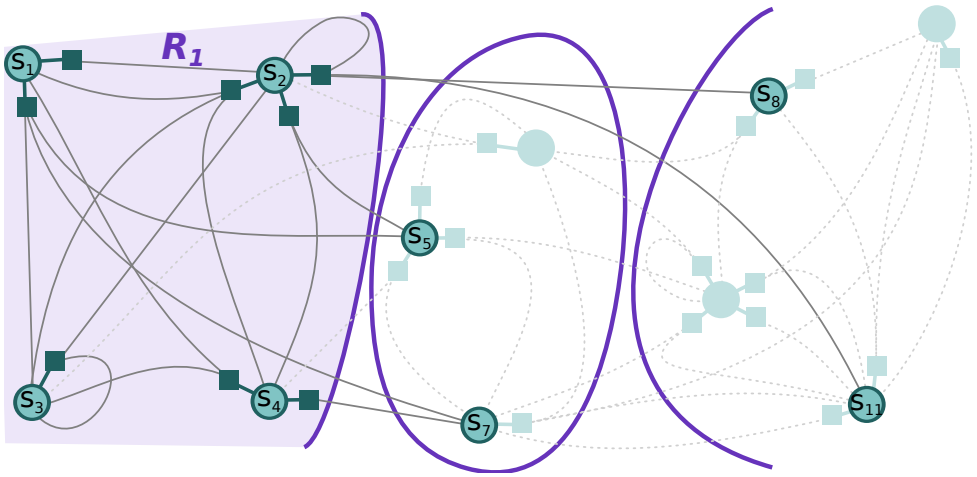


FIGURE 4.2 – Sous-MDP construit pour la région 1.

Cependant, certains états atteignables à partir de R n'appartiennent pas à la région [Dean 95]. Ces états sont définis comme les états périphériques (Perifery) à R. Inversement, les états entrées de R (Entrance) forment l'ensemble des états de R directement atteignables depuis les états des autres régions. Nous noterons ainsi $S_{Perifery}$, l'ensemble de tous les états périphériques quelle que soit la région considérée :

$$Perifery(R) = \{s \mid s \in S, \quad s \notin R, \quad \exists s' \in R, \quad \exists a \in A, \quad t(s',\ a,\ s) > 0\}$$
$$Entrance(R) = \{s \mid s \in R, \quad \exists s' \in S, \quad s' \notin R, \quad \exists a \in A, \quad t(s',\ a,\ s) > 0\}$$

$$S_{Perifery} = \bigcup_{R \in P} Perifery(R) \quad \text{et} \quad Entrance(R) = S_{Perifery} \cap R$$

Les régions voisines de R regroupent l'ensemble des régions qui incluent au moins un état périphérique à R. Une résolution locale par l'algorithme 3 nécessite alors une initialisation de la fonction de valeur sur l'ensemble des états périphériques. Ces valeurs dépendent elles-mêmes de la résolution locale des sous-MDPs voisins. Ainsi, plusieurs politiques ont besoin d'être calculées pour chaque sous-MDP de façon à coordonner les politiques locales. En effet, les politiques locales doivent être adaptées aux politiques appliquées dans les zones voisines. La décomposition d'un MDP en sous-MDPs va permettre de réduire la complexité du calcul de la politique globale optimale uniquement si les résolutions des sous-MDPs sont suffisamment indépendantes les unes des autres. Ceci suppose que le nombre de va-et-vient entre les résolutions locales ne soit pas, lui-même, polynômial en le nombre de régions construites.

4.1.2 Résolution itérative et hiérarchique

Dans les faits, construire la politique globale nécessite de calculer au moins une politique par sous-MDP. Dans le cas de dépendances entre les régions, plusieurs calculs de politiques par sous-MDP sont nécessaires à l'ajustement des politiques locales entre elles. Il existe deux approches pour coordonner les politiques locales : une approche itérative et une approche hiérarchique.

L'approche itérative consiste à re-calculer la politique d'une région au fur et à mesure que de nouvelles valeurs sont connues pour une des régions voisines. L'actualisation des valeurs périphériques se fait en même temps que les politiques locales sont affinées. L'approche hiérarchique consiste à calculer un certain nombre de politiques locales pour chaque région puis, à sélectionner les politiques locales permettant une politique globale cohérente.

Algorithme de résolution itérative

De façon similaire à "Value iteration", l'algorithme itératif "Region Iteration" parcourt l'ensemble des régions jusqu'à ce que les améliorations sur la fonction de valeur soient inférieures à un seuil donné ϵ [Dean 95] (Algorithme 4).

Algorithme 4 : Region iteration

Données : un DMDP $\langle S, A, t, r, P\rangle$, $\gamma \in [0,\ 1]$ et $\epsilon \ll \min_{s\in S,\ a\in A}(\ |r(s,\ a)|\ )$

Résultat : π^* et V associé (selon ϵ et γ)

```
initialiser V : ∀s ∈ S,   V[s] ← 0;
répéter
    initialiser V' : V' ← V;
    pour ∀R ∈ P faire
        Actualiser π* et V sur R :
        (π*, V) = Local value iteration (⟨R, S, A, t, r⟩, V', ε, γ);
jusqu'à ∀s ∈ S,   |V'(s) − V(s)| < ε ;
```

Dans le cadre de problèmes avec une certaine structure topologique sur les transitions entre états, un partitionnement cohérent permet à l'algorithme itératif de converger rapidement vers la politique globale optimale. L'algorithme n'aura alors besoin que de quelques itérations (à l'échelle du nombre total d'états $|S|$).

Résolution hiérarchique

La seconde approche consiste à résoudre un MDP abstrait construit sur l'ensemble des régions pour coordonner les politiques locales entre elles. Un MDP abstrait est un MDP classique $\langle S_A, A_A, r_A, t_A\rangle$ avec S_A et A_A respectivement, l'ensemble des macro-états et l'ensemble des macro-actions. Un macro-état est défini pour chaque région $R \in P$ d'un MDP décomposé $\langle S, A, t, r, P\rangle$. Les macro-actions correspondent aux politiques localement applicables connues. Les fonctions de transition t_A et de récompenses r_A définissent la dynamique du système abstrait en fonction des politiques locales exécutées.

La définition d'un MDP abstrait est basée sur la connaissance de politiques locales. Cette connaissance est construite par le pré-calcul d'un certain nombre de politiques locales optimales pour chaque région R. Ce pré-calcul est effectué via plusieurs vecteurs de valeurs hypothétiques $\overline{v}_0^R, \ldots, \overline{v}_{l_R}^R$ définis pour les états périphériques à la région R ($Perifery(R)$). Ainsi, un MDP abstrait est défini pour un ensemble donné de valeurs appliquées aux états périphériques.

$$S_A = P, \quad A_A = \{\pi^*_{\overline{v}_j^R} \mid R \in P, \quad j = [0, l_R]\}$$

La fonction de transition abstraite $t_A : P \times A_A \times P \to [0, 1]$ découle des politiques locales. $t_A(R,\ \pi^*_{\overline{v}_j^R},\ R')$ définit la probabilité d'atteindre un des états de la région R' en appliquant la

politique $\pi^*_{\overline{v}_j^R}$ dans la région R. La fonction de transition peut être approchée de façon récursive en considérant une équiprobabilité d'entrer dans la région R par un des états $Entrance(R)$ [Dean 95] :

$$t_A(R,\ \pi^*_{\overline{v}_j^R},\ R') = \frac{1}{|Entrance(R)|} \sum_{s \in Entrance(R)} t'_R(s,\ \pi^*_{\overline{v}_j^R},\ R')$$

$$t'_R(s,\ \pi,\ R') = \sum_{s' \in Entrance(R')} t(s,\ \pi(s),\ s') + \sum_{s' \in R,\ s' \notin Entrance(R)} t(s,\ \pi(s),\ s').t'_R(s',\ \pi,\ R')$$

L'évaluation de la fonction de récompense abstraite peut être calculée de façon équivalente à la fonction de transition [Dean 95] :

$$r_A(R,\ \pi^*_{\overline{v}_j^R}) = \frac{1}{|Entrance(R)|} \sum_{s, \in Entrance(R)} r'_R(s,\ \pi^*_{\overline{v}_j^R})$$

$$r'_R(s,\ \pi) = r(s,\ \pi(s)) + \sum_{s' \in R,\ s' \notin Entrance(R)} t(s,\ \pi(s),\ s').r'_R(s',\ \pi)$$

Ainsi, la fonction de récompense somme l'ensemble des récompenses ou pénalités perçues en exécutant une certaine politique dans une région donnée R. De plus, la récursion lors de la définition des transitions et des récompenses ne dépasse pas les états permettant d'entrer dans une des régions de la partition P (y compris la région courante R).

Ainsi, une résolution hiérarchique d'un DMDP consiste à : (1) définir un ensemble de vecteurs de valeurs possibles pour chaque région ; (2) calculer les politiques locales optimales pour chaque région et pour chaque vecteur de valeurs ; (3) construire et résoudre un MDP abstrait. L'étape (1) peut être réalisée en construisant l'ensemble des vecteurs de valeurs correspondants aux situations où chaque région R est : neutre ($\overline{v} = 0$ sur $Entrance(R)$), attractive ($\overline{v} = +constant$ sur $Entrance(R)$) ou répulsive ($\overline{v} = -constant$ sur $Entrance(R)$) [Dean 95].

Discussion

Les deux approches reposent sur des philosophies différentes avec des caractéristiques différentes en terme d'efficacité et d'optimalité. L'efficacité se mesure ici en terme de temps de calcul nécessaire à la construction de la politique globale. Elle dépend directement de la taille des régions et du nombre de politiques locales calculées par région.

L'approche itérative permet de converger vers une politique globale optimale par l'application d'améliorations successives. Par contre, cette solution peut amener la résolution à explorer un grand nombre de politiques locales par région avant de terminer. Des améliorations peuvent encore être apportées comme l'ordonnancement des régions (ordre dans lequel les politiques sont calculées pour les régions) offrant une garantie variable de l'amélioration de la convergence en fonction des structures à détecter au sein du MDP.

L'approche hiérarchique propose un bon contrôle du nombre de politiques locales calculées. En effet ce nombre est défini arbitrairement. Par contre, la résolution hiérarchique ne permet de converger que vers une solution sous-optimale. La sous-optimalité dépend du nombre et de la pertinence des politiques locales initialement calculées. Plus la population de vecteurs de valeurs utilisés sera importante et diversifiée (sur des plages de valeurs cohérentes), plus il est possible d'avoir des garanties sur la qualité de la politique globale (combinaison des politiques locales). D'autre part, la qualité de la politique calculée dépend du nombre et de l'homogénéité des entrées de chaque région. En effet, l'hypothèse d'équiprobabilité d'atteindre à partir d'une région voisine, n'importe quel état entrée n'est que rarement valide.

Quelle que soit l'approche choisie, il sera nécessaire de calculer plusieurs politiques locales par région. Même si ces approches permettent de résoudre des MDPs larges et structurés, elles ne permettent pas de considérer des MDPs excessivement larges notamment dans le cadre d'une planification en cours de mission incluant un nombre important d'objectifs ($\simeq$ 120). De plus, ces approches s'appuient sur un MDP partitionné *a priori*, avec une partition influençant pour beaucoup l'efficacité et/ou l'optimalité de la solution calculée.

4.1.3 Partitionner l'espace d'états

Les algorithmes de résolution d'un MDP décomposé se basent sur le partitionnement en régions pour accélérer la résolution. La difficulté consiste alors à partitionner l'espace d'état d'une façon cohérente avec la future résolution du DMDP. C'est-à-dire que la partition optimale sera la partition permettant la résolution la plus rapide et offrant les meilleures garanties sur l'optimalité de la solution.

Il est alors incohérent (et impossible) de calculer systématiquement la politique globale pour chaque partition possible afin de valider l'optimalité ou non d'une partition. De ce fait, les critères de partitionnement reposent sur des hypothèses *a priori* [Parr 98, Sabbadin 02] : équilibre sur la taille des régions et du nombre de régions, minimisation des transitions coupées, minimisation de l'ensemble des états périphériques ($S_{Perifery}$), etc.

Une fois le critère (ou fonction objectif) défini, le problème de la détermination de la partition optimale d'un graphe est connu pour être, lui-même, NP-difficile [Bichot 11, Garey 74]. Le partitionnement d'un MDP large, en vue de sa résolution, n'est donc intéressant que s'il existe des heuristiques rapides, permettant un partitionnement efficace.

Évaluation des partitions

La différenciation entre les partitions construites peut se faire sur 2 critères *a priori* : équilibre des régions et minimisation des coupes. L'équilibre est important pour construire des sous-MDPs et un MDP abstrait de tailles homogènes. La minimisation des coupes permet de limiter, statistiquement, la difficulté de coordonner les politiques locales.

Plus un MDP est important en taille, plus il est difficile à résoudre et donc plus les algorithmes de résolution nécessiteront de ressources. Dans le cadre d'un DMDP, une répartition équitable des états dans les régions permet de garantir statistiquement une somme de calculs des politiques locales faible. Il est possible de chercher la partition P minimisant la région de taille maximale parmi l'ensemble des partitions possibles $\mathbb{P}$:

$$\underset{P\in\mathbb{P}}{argmin}\left(\max_{R\in P}(\ |R|\ )\right)$$

Une autre approche consiste à optimiser le nombre de régions par rapport à un nombre idéal k^*. En effet, moins il y a de régions plus il sera facile de coordonner les politiques locales. Dans le cadre d'une résolution hiérarchique, la taille du MDP abstrait dépend directement du nombre de régions. Le nombre optimal k^* de régions attendu peut être défini de façon à ce que le nombre de régions soit égal au nombre moyen d'états par région. Ainsi, une partition optimale est constituée de k^* régions, chacune de taille k^*, soit, avec un nombre homogène d'états par région ($|S|/k^*$) :

$$\underset{P\in\mathbb{P}}{argmin}\left(\ ||P|-k^*|\ \right),\quad k^*=\frac{|S|}{k^*}=\sqrt{|S|}$$

Le critère peut aussi chercher à caractériser les coupes de façon à minimiser les connexions entre les régions et par conséquent entre les sous-MDPs. Ce critère repose sur l'idée que, moins il

y a de connexions entres les régions, moins il y aura de combinaisons possibles entre les politiques locales. Il existe alors plusieurs approches possibles, minimiser le nombre d'états périphériques $|S_{Perifery}|$ ou minimiser le nombre de voisins de chaque région.

$$Voisin(R) = \{R' \mid \exists s' \in R', \quad s' \in Perifery(R)\} \quad \text{et} \quad Voisins = \sum_{R \in P} (\ |Voisin(R)|\ )$$

Une autre approche consiste à minimiser la somme des probabilités de transitions coupées par la partition. Une transition coupée $t(s,a,s') > 0$ est une transition connectant 2 états dans des régions différentes. Cette approche repose sur l'idée que, à nombre égal, il est plus intéressant de couper des transitions faibles plutôt que des transitions fortes dans la mesure où la politique locale sera probablement plus faiblement impactée.

$$Transition = \sum_{R \in P} \left(\sum_{(s,\ a,\ s') \in R \times A \times Perifery(R)} t(s,a,s') \right)$$

Ainsi, une partition optimale minimise la fonction objectif définie sur l'ensemble pondéré de ces critères *a priori*. Les poids permettent, pour une résolution optimale itérative, d'avantager le score sur les coupes de transitions plutôt que le nombre total d'états périphériques. Inversement, dans une approche hiérarchique, le nombre de voisins peut conditionner le nombre de politiques locales calculées pour chaque région.

$$\underset{P \in \mathbb{P}}{argmin} \left(\alpha * |S_{Perifery}| + \beta * \max_{R \in P}(\ |R|\ ) + \gamma * ||P| - k^*| + \delta * |Voisins| + \epsilon * Transition \right)$$

La plupart des approches basent leurs fonctions objectifs de façon à générer une partition de taille fixe, la plus équilibrée possible et minimisant la coupe sur l'ensemble de transitions [Parr 98, Sabbadin 02]. Cela peut se traduire par la fonction normalisée suivante :

$$\underset{P \in \mathbb{P},\ |P| = k^*}{argmin} \left(0.5 * \frac{\max_{R \in P}(\ |R|\ )}{|S|} + 0.5 * \frac{Transition}{\sum_{(s,\ a,\ s') \in S \times A \times S} t(s,a,s')} \right)$$

Algorithme de partitionnement

Dans le cadre d'un partitionnement visant à minimiser la somme des transitions coupées, il est possible de se rapporter à un problème de partitionnement de graphes pondérés. Un graphe pondéré est un triplé $G = \langle V, E, w \rangle$ avec V l'ensemble de nœuds (vertices), E l'ensemble d'arcs (edges) et $w : E \to \mathbb{R}$ une fonction qui associe un poids (weight) à chaque arc. L'ensemble de nœuds correspond alors à l'ensemble des états du MDP et chaque arc du graphe correspond à une transition possible entre deux états. Enfin, le poids d'un arc correspond à la somme des probabilités de transitions entre les deux états.

$$V = S, \quad E = \{(s,\ s') \in S \mid \exists a \in A, \quad t(s,\ a,\ s') > 0\}$$

$$\forall e = (s,\ s') \in E, \quad w(e) = \sum_{a \in A} t(s,\ a,\ s')$$

Le problème de calculer une partition optimale d'un graphe pondéré est connu pour être NP-Difficile [Bichot 10]. La méthode multi-niveaux fait alors référence pour générer rapidement une

partition avec de bons résultats qualitatifs [Barnard 94, Karypis 98]. Cette méthode se compose de 3 étapes : contraction, partitionnement et affinage.

L'étape de contraction consiste à construire une hiérarchie de graphes $G_0, \dots G_n$ tel que chaque graphe G_i soit une simplification du graphe G_{i-1}. La simplification s'obtient en fusionnant les nœuds similaires entre eux pour créer un nœud dans le graphe de niveau supérieur [Hagen 92, Bui 93]. La construction hiérarchique s'arrête, soit lorsqu'il est impossible de trouver des nœuds similaires, soit, lorsque le graphe de plus haut niveau est suffisamment petit.

Cette première étape met en œuvre des algorithmes de fusion. Une fusion consiste à agréger plusieurs nœuds qui partagent un voisinage similaire. C'est-à-dire qu'ils sont connectés par des arcs aux mêmes nœuds (entrant et sortant) avec les mêmes poids. De façon à permettre la hiérarchisation, la similitude est définie avec un certain seuil de tolérance sur le nombre d'arcs et la différence des poids. Une autre fusion par similitude peut être définie entre des nœuds connectés par des arcs de poids importants [Karypis 98].

L'étape de partionnement consiste à découper, avec un algorithme standard, le graphe de niveau supérieur G_n dans la hiérarchie construite. Ainsi, le calcul de la partition P s'effectue sur un ensemble fortement réduit en ne considérant que les nœuds contenus par G_n. L'étape de partitionnement permet de mettre en œuvre un algorithme global basé sur la recherche de coupe optimale [Kernighan 70, Karypis 98, Martin 06, Chardaire 07] (comme l'application de bisection récursive). Les algorithmes globaux sont généralement gourmands en ressources. En effet, la détection d'une coupe optimale est elle même NP-Difficile [Bichot 10]. L'utilisation d'une approche globale est rendue possible par l'étape précédente conduisant à un graphe G_n de taille suffisamment réduite.

Enfin, la troisième et dernière étape, l'étape d'affinage, consiste à répercuter récursivement le partitionnement sur les graphes de niveau inférieur. Cette étape s'accompagne de l'utilisation d'algorithmes d'optimisation pour ajuster, entre chaque niveau, la partition vers une partition sous-optimale. Les algorithmes d'optimisation consistent, à partir d'une partition, à inter-changer les nœuds d'une région à une autre, de façon à optimiser la fonction objectif.

La méthode multi-niveaux permet de combiner des algorithmes parmi les 3 approches : approche par fusion d'états, approche globale et approche par optimisation. Elle conduit à de bons résultats pour un temps de calcul raisonnable [Bichot 10]. Malgré cela, cette méthode combinée à la résolution du DMDP ne permet pas de traiter, en cours de mission, un GO-MDP avec $2^{|G|}$ états pour G objectifs où même l'énumération des états est fastidieuse.

4.2 Vers une résolution rapide approchée d'un GO-MDP

L'objectif de notre proposition consiste à planifier rapidement les actions des agents guidés par un nombre important d'objectifs potentiels et évoluant dans un environnement incertain. La coordination des agents est réalisée par une allocation des objectifs entre eux. Pour un agent, la problématique peut être modélisée par un MDP large découlant d'un MDP orienté par les objectifs qui lui sont attribués (GO-MDP). Nous nous intéressons, dans cette section, aux mécanismes permettant à chaque agent de calculer sa politique sur la base d'une allocation définie *a priori*.

Le GO-MDP est défini à partir d'un MDP modélisant la dynamique du système à contrôler $(\langle S_d,\ A_d,\ t_d,\ r_d\rangle)$, l'ensemble des états objectifs à traverser $G \subset S_d$ et une fonction de gain $gain : G \to \mathbb{R}$ associant une récompense à chaque objectif atteint. La particularité réside dans le fait que cette récompense ne peut être perçue qu'une et une seule fois par objectif. Le GO-MDP $\langle S, A, t, r\rangle$ est alors construit tel que chaque état $s \in S$ inclut l'état courant $s_{ds} \in S_d$ dans la

dynamique du système et une mémoire des objectifs atteints $G_s \subset G$ (cf. section 2.2.4). Ainsi, le GO-MDP comptabilise $|S| = |S_d|2^{|G|}$ états et mérite sa qualification de MDP large.

Nous avons vu qu'une approche par décomposition permet d'accélérer la résolution optimale ou sous-optimale de certains MDPs larges. Cette solution repose alors sur un partitionnement de l'espace d'états en régions qui est lui-même difficile à réaliser. Cependant, cette résolution n'est pas applicable à un GO-MDP où même l'énumération de l'ensemble des états est fastidieuse. L'approche proposée consiste à s'appuyer sur la structure du GO-MDP pour proposer une résolution *ad hoc*, approchée mais rapide.

Dans une première étape un **algorithme glouton** partitionne les états du MDP modélisant la dynamique du système. Cette partition est ensuite utilisée dans le cadre d'une **résolution hiérarchique**, pour construire un GO-MDP abstrait approximatif sur l'ensemble des régions. C'est-à-dire que, les fonctions de transitions et de récompenses sont approchées sans calculer des politiques locales pour chaque région. Enfin, une **politique locale est calculée** chaque fois que le système atteint un état lui faisant changer de région.

La rapidité de l'approche repose sur 2 piliers : il n'est pas nécessaire de partitionner le GO-MDP global directement et une évaluation abstraite approximative permet d'orienter les politiques locales de façon à coordonner les politiques locales entre elles. L'évaluation abstraite approximative s'effectue au regard de 2 types de politiques basés sur l'hypothèse qu'une région peut être soit explorée entièrement (tous les objectifs locaux seront atteints) soit traversée (aucun objectif de la région n'est rempli). La coordination résultant d'une telle approche ne permet pas de garantir l'optimalité de la solution construite. Nous verrons en revanche, dans la section suivante, que cette approche permet à chaque robot de planifier, en cours de mission, des déplacements visant un nombre conséquent d'objectifs.

4.2.1 Partitionnement glouton

L'objectif visé par l'algorithme de partionnement consiste à diviser directement le MDP $\langle S_d,\ A_d,\ t_d,\ r_d\rangle$ puis, à calculer une planification hiérarchique multi-objectifs. La partition en soi n'est pas la solution finale mais un biais mis en place afin de conduire à un processus décisionnel rapide. Aussi, un algorithme glouton est proposé pour privilégier l'économie de ressources de calcul au détriment de la qualité de la partition.

Le principe de l'algorithme glouton repose sur une construction successive des régions. Une région est elle-même construite par additions successives d'états. A chaque itération, l'état optimisant un critère local est alors sélectionné pour ajout.

Le partitionnement cherche à minimiser la coupe sur l'ensemble des transitions et à équilibrer le nombre d'objectifs par région. En effet, la taille des sous-MDPs construits par la décomposition du GO-MDP est exponentielle sur le nombre d'objectifs locaux. Ainsi, un nombre équilibré d'objectifs par région permet d'équilibrer la taille des futurs GO-MDPs locaux.

Critère sur le voisinage local

Commençons par définir comment minimiser les coupes. Lors de la construction d'une région $R \in P$, l'état le plus intéressant à ajouter est l'état qui maximise la somme des transitions le connectant à un des états de la région et qui minimise la somme des transitions le connectant à un des états non encore sélectionnés dans aucune région. P, pendant le partitionnement, ne couvre pas la totalité de S_d mais uniquement, les états précédemment sélectionnés. En d'autres termes, l'état $s_d^* \in S_d - P$ à ajouter, parmi les états non encore sélectionnés, maximise le ratio entre l'ensemble des transitions intérieures à la région $t_{in}(R,\ s_d)$ et l'ensemble des transitions

extérieures aux régions du partitionnement courant $t_{out}(P,\ s_d)$.

$$s_d^* = \underset{s_d \in S_d - P}{argmax}(\ ratio(P,\ R,\ s_d)\ ), \quad ratio(P,\ R,\ s_d) = \frac{t_{in}(R,\ s_d)}{t_{out}(P,\ s_d)}$$

$$t_{in}(R,\ s_d) = \sum_{s'_d \in R} \big(t_d(s_d,\ a,\ s'_d) + t_d(s'_d,\ a,\ s_d)\big)$$

$$t_{out}(P,\ s_d) = \sum_{s'_d \in S_d - P} \big(t_d(s_d,\ a,\ s'_d) + t_d(s'_d,\ a,\ s_d)\big)$$

$$\text{avec} \quad S_d - P = \{s_d \mid s_d \in S_d, \quad \forall R' \in P, \quad s_d \notin R\}$$

Ainsi, tout état présentant un ratio supérieur à 1 connecte davantage d'états dans la région qu'il ne connecte d'états non encore sélectionnés. Le ratio s'intéresse uniquement aux transitions non encore sélectionnées pour favoriser la construction des régions le long des frontières établies par les régions précédemment construites. L'état ayant le ratio maximal optimise la différence entre les transitions qui seront sauvées et les transitions qui seront potentiellement coupées. Ce critère local n'est valable que pour l'état courant du partitionnement, ainsi, la partition finale n'offre pas de garantie d'optimalité. Ce constat est vrai notamment pour les premiers états sélectionnés d'une région. Leur sélection se fait à l'aveugle et plusieurs des états voisins comptabilisés comme extérieurs seront sélectionnés lors d'itérations suivantes.

Contrôle de la taille des régions

Une partition idéale est aussi définie par le nombre d'objectifs contenus dans chaque région ainsi que par le nombre total de régions. Cependant, cette contrainte n'est pas dure. C'est-à-dire que malgré un nombre idéal de n^* objectifs par région, la minimisation des coupes sur les transitions peut amener à des partitions hétérogènes. Par exemple, le partitionnement non-contraint d'un appartement peut se faire par rapport aux pièces même s'il y a un nombre "sensiblement" différent d'objectifs par pièce.

Dans le cadre du partitionnement glouton, cela consiste à définir la condition permettant de continuer la construction d'une région, ou de passer à la suivante. En posant comme stratégie que, "tant qu'il existe un état dont le ratio est supérieur à 1, alors continuer la région", il faut adapter le ratio pour intégrer la taille de la région comme paramètre. Dans le cadre d'une région encore trop petite, il est nécessaire de simuler artificiellement que, parmi toutes les transitions localement coupées, certaines seront sélectionnées plus tard.

En moyenne, il y a $t_{o.total}/|S_d|$ transitions sortantes par état modélisant la dynamique du système avec $t_{o.total}$ la somme des probabilités de toutes les transitions sortantes. Il est possible de considérer initialement que $t_{o.total}/|S_d|$ transitions ne seront finalement pas coupées. Cela correspond à retrancher ce volume de transitions au poids des transitions localement extérieures $t_{out}(P,\ s_d)$ pour générer une somme prédictive des probabilités de transitions extérieures $t_{o.pre}$. Ensuite, le facteur d'ajustement $t_{o.total}/|S_d|$ va être minoré au fur et à mesure que des objectifs sont ajoutés à la région et en fonction du nombre idéal d'objectifs par région n^*.

$$t_{o.pre}(P,\ R,\ s_d) = t_{out}(P,\ s_d) - \frac{n^* - |G \cap P|}{n^*}.\frac{t_{o.total}}{|S_d|}$$

$$\text{avec} \quad t_{o.total} = \sum_{s,\ s',\ a \in S^2 \times A,\ s \neq s'} t_d(s, a, s')$$

Inversement, lorsqu'une région possède plus d'objectifs que le nombre idéal, le poids des transitions localement extérieures $t_{o.pre}$ est artificiellement augmenté. Plus la région contient des objectifs, plus il sera difficile pour un état, d'intégrer la région.

$$ratio(P,\ R,\ s_d) = \frac{t_{in}(R,\ s_d)}{\max(\ t_{out}(P,\ s_d) - \frac{n^* - |G \cap R|}{n^*} . \frac{t_{o.pre}}{|S_d|},\ \epsilon\ )}$$

$$\text{avec} \quad \epsilon \ll \min(\ t_d(s,\ a,\ s')\ )$$

Le dénominateur du ratio est alors borné de façon strictement supérieure à zéro. Cela permet de contrôler la situation où le facteur d'ajustement est supérieur ou égal à la somme des transitions localement extérieures. Le ratio local dépend uniquement du voisinage de chaque état et du nombre d'objectifs dans la région courante. Ce ratio peut être calculé avec un nombre insignifiant d'opérations par rapport au nombre total d'états $|S_d|$ dans un MDP non fortement connexe ($t_{o.total} \ll |S_d|^2$).

Algorithme glouton de partionnement

L'idée générale du partitionnement glouton repose donc sur l'optimisation d'un critère local. L'algorithme glouton (algorithme 5) consiste alors à parcourir à chaque itération l'ensemble des états pour sélectionner l'état localement optimal.

Algorithme 5 : Partitionnement glouton

Données : un MDP $\langle S_d, A_d, t_d, r_d \rangle$, $s_{d0} \in S_d$, $G \subset S_d$, $n^* \in \mathbb{R}^+$
Résultat : P, $\bigcup_{R \in P} R = S_d$ et $\forall R, R' \in P,\ (R \cap R') = \emptyset$

initialiser P : $P \leftarrow \emptyset$;
tant que $\exists s_d \in S_d,\ \forall R' \in P,\ s_d \notin R'$ **faire**
  initialiser une nouvelle région : $R \leftarrow \emptyset$;
  choisir $s'_d : s'_d = \underset{s_d \in S_d}{argmin}(\ distance(s_{d0},\ s_d)\ )$;
  répéter
    ajouter s'_d à R;
    choisir $s'_d : s'_d = \underset{s_d \in S_d}{argmax}(\ ratio(P,\ R,\ s_d)\ )$;
  jusqu'à $ratio(P,\ R,\ s_d) > 1$;
  ajouter R à P;

L'algorithme 5 construit la partion sur $|S_d|$ itérations et chaque itération demande au pire $|S_d|$ calculs de ratio. Ainsi, la complexité s'évalue à $o(n^2)$. Pareillement, la distance entre un état et tous les autres états peut se calculer en $o(n^2)$. Cette distance peut être calculée par exemple, comme le nombre minimal de transitions nécessaires pour atteindre un état. Cette notion sert à choisir un état pour initialiser une nouvelle région qui soit voisine d'une région existante sans laisser d'états isolés.

Dans les faits, avec des structures de données adaptées, il est possible de réduire encore la complexité. L'idée consiste alors à ne parcourir, à chaque itération, que les états atteignables. De même, la fonction de distance peut être construite au fur et à mesure que de nouveaux états sont visités. Il en résulte une forme itérative de partitionnement acceptant un espace potentiellement

infini. A partir d'un état initial s_{d0}, l'algorithme construira un nombre de régions avec n^* objectifs attendus par région en fonction du nombre d'itérations autorisées.

Dans la section 4.3 des expérimentations sont effectuées dans un cadre robotique. Ces expériences permettent de valider la rapidité d'exécution et le bon contrôle du nombre d'objectifs par région. Cependant, le partionnement n'est pas une fin en soi, il est ensuite utilisé dans le cadre d'une planification hiérarchique approchée.

4.2.2 Approximation du MDP abstrait

La partition P du MDP $\langle S_d,\ A_d,\ t_d,\ r_d\rangle$ modélisant la dynamique d'un système permet de déduire directement un partitionnement du GO-MDP $\langle S,\ A,\ t,\ r\rangle$. Cependant, même décomposé, l'ensemble des sous-MDPs recouvre la totalité des états du GO-MDP, soit au minimum $2^{|G|}$ états. La solution retenue dans notre contribution vise une planification rapide des actions à réaliser, en considérant que même l'énumération de tous les états est impossible. Cette planification se fait alors, au détriment de l'optimalité de la solution.

La rapidité du calcul de la politique repose sur l'heuristique que : une région peut être soit traversée, soit explorée totalement. Ainsi, il est possible de découper l'ensemble des objectifs en paquets d'objectifs. Cette méthode permet, dans une résolution hiérarchique d'un GO-MDP décomposé, de réduire considérablement le nombre d'états abstraits construits pour le GO-MDP de niveau supérieur $\langle S_a,\ A_a,\ t_a,\ r_a\rangle$ (Figure 4.3) et le nombre de GO-MDPs locaux potentiels.

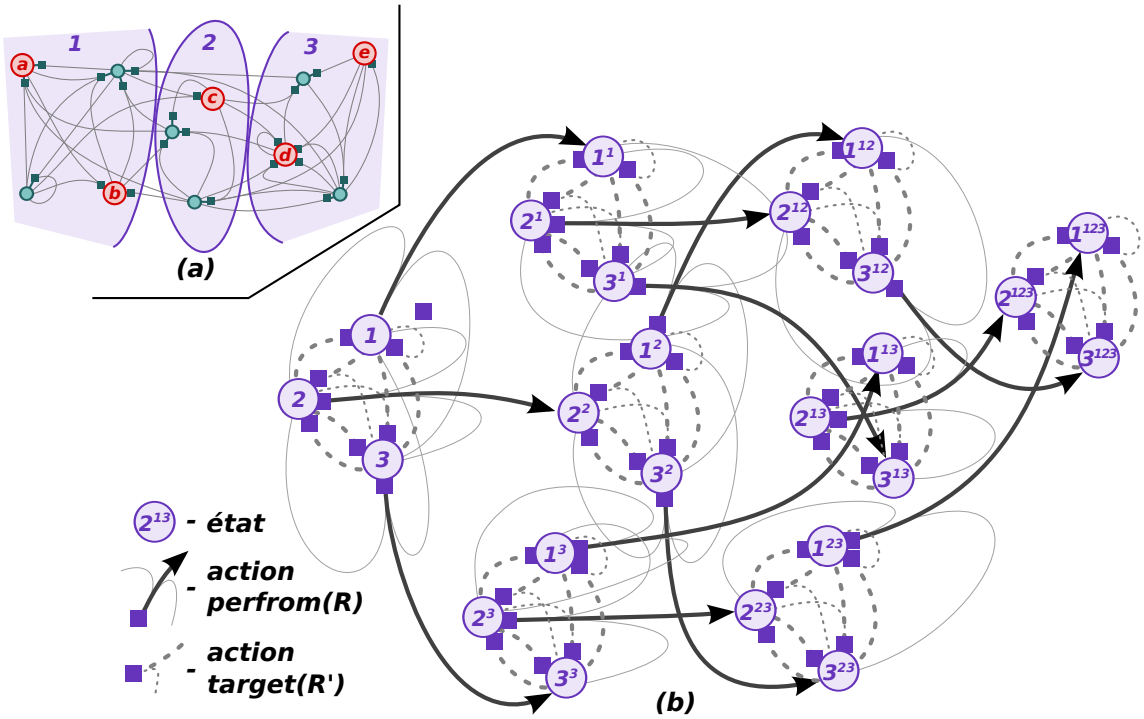


FIGURE 4.3 – Exemple de GO-MDP abstrait. Le GO-MDP abstrait (b) est construit à partir d'un MDP décomposé (a) et d'un ensemble de 5 objectifs $G = \{a,\ b,\ c,\ d,\ e\}$. Ici l'état (2^{13}) correspond à l'état $s_a = (2,\ \{1,\ 3\})$: le système est dans la région 2 et les objectifs des régions 1, 3 sont atteints.

Ensembles des états et des actions abstraits

A partir de la partition P de taille k sur le MDP $\langle S_d,\ A_d,\ t_d,\ r_d\rangle$ et un ensemble d'objectifs G, il est possible de définir un ensemble de régions objectifs $RG \subseteq P$ comme l'ensemble des régions qui possède au moins un objectif (toutes les régions avec un partitionnement glouton).

$$RG = \{R \mid \exists s \in R, \quad s \in G\}$$

Un état $s_a = (R,\ H) \in S_a$ du GO-MDP abstrait (Figure 4.3) modélise : R la région incluant l'état courant de la dynamique du système et H l'état courant d'exécution par rapport à l'ensemble des régions contenant des objectifs à atteindre RG. Dans le GO-MDP abstrait approché, depuis un état $s_a = (R,\ H) \in S_a$ les seules actions possibles sont : chercher à atteindre une région voisine ($target(R')$) ou réaliser entièrement l'ensemble des objectifs locaux ($perform(R)$).

$$S_a = \{(R,\ H) \mid R \in P, \quad H \subseteq RG\}$$
$$A_a = \{target(R') \mid R \in P, \quad R' \neq R\} \ \cup\ \{perform(R) \mid R \in RG\}$$

Ainsi, la dynamique du système dans la région 1 avec les objectifs de la région 2 atteints correspond à l'état $(1,\ \{2\})$ du GO-MDP abstrait (soit l'état (1^2) dans la figure 4.3). Les 3 actions possibles sont : réaliser les objectifs de la région 1 ; chercher à atteindre la région 2 et chercher à atteindre la région 3. Les états atteignables sont : $(1,\ \{2\})$ pas de changement ; $(2,\ \{2\})$ ou $(3, \{2\})$ l'agent a changé de région ; $(1,\ \{1,\ 2\})$ les objectifs de la région 1 ont aussi été atteints (soit les états (1^2), (2^2), (3^2), (1^{12}) dans la figure 4.3).

Évaluation approximée des transitions

La fonction de transition $t_a : S_A \times A_A \times S_A \to [0,\ 1]$ retourne chaque probabilité $t_a(s_a, a_a, s'_a)$ d'atteindre l'état abstrait s'_a à partir de s_a en exécutant l'action abstraite a_a. La fonction de récompenses abstraite $r_a : S_A \times A_A \to \mathbb{R}$ permet, quant à elle, d'évaluer l'intérêt contextuel des actions abstraites. Les transitions et les récompenses pour passer d'une région à une autre dans le MDP abstrait dépendent de l'état initial, de la dynamique du système dans la région courante R et de la politique locale appliquée. Celle-ci dépend elle même des valeurs des états définissant une entrée dans la région suivante. Une évaluation optimale des transitions et des récompenses impliquerait de calculer toutes les politiques locales optimales pour toutes les valeurs potentielles des états périphériques. Ces valeurs dépendent de l'ensemble des objectifs qu'il reste à réaliser dans toutes les régions.

Dans un cadre de ressources de calcul contraintes, les transitions et les récompenses abstraites sont simplement approximées sans aucun calcul de politiques locales. L'idée de l'approximation repose sur le fait que, le GO-MDP abstrait ne sert plus à sélectionner les meilleures politiques locales mais à approximer les valeurs des états périphériques à un GO-MDP local de façon à orienter la politique locale. Pour une région R, l'approximation est réalisée à partir de moyennes calculées sur l'ensemble des transitions et des récompenses au niveau du sous-MDP système $\langle R,\ S_d,\ A_d,\ t_d,\ r_d\rangle$, de l'ensemble des objectifs inclus dans la région $G_R = G \cap R$ et de leurs gains associés.

Dans le cadre d'une transition d'une région R à une autre R' par une action $target(R')$, la politique locale à R conduira l'agent sur un état frontière de R permettant de sortir en direction d'une autre région. En supposant une équi-probabilité d'atteindre un des états frontières de la région R, la transition abstraite peut être calculée de façon heuristique, vis à vis des actions maximisant les chances d'atteindre R'.

$$t_A(\ (R,H),\ target(R^t),\ (R',H')\ ) = \begin{cases} \frac{1}{|Frontier(R)|}\sum_{s_d\in Frontier(R),\ s'_d\in R'} t_d(s_d,\ a^*(s_d,R^t),\ s'_d) & \text{si} \quad H\neq H' \\ 0 & \text{sinon} \end{cases}$$

$$\text{avec} \quad a^*(s_d,R^t) = \underset{a\in A_A}{argmax}\left(\sum_{s^t_d\in R^t} t_d(s_d,\ a,\ s^t_d)\right)$$
$$\text{et} \quad Frontier(R) = \{s_d \mid s_d\in R, \quad \exists t_d(s_d,a,s'_d)>0, \quad s'_d\notin R\}$$

Inversement, lors de la réalisation d'une action $perform(R)$ à partir de la régions R, la transition abstraite est évaluée pour les actions maximisant les chances de rester dans la région R et finir les objectifs en cours. Nous considérons alors une équi-probabilité d'atteindre n'importe quel état frontière de la région R à un moment donné. Dans ces états frontières, nous supposons que la politique locale cherchera à atteindre des états dans la même région R.

$$t_A(\ (R,H),\ perform(R^t),\ (R',H')\ ) = \begin{cases} 0 & \text{si} \quad R\neq R' \text{ et } H\neq H' \\ 0 & \text{si} \quad R=R' \text{ et } H'\neq H\cup\{R\} \\ \frac{1}{|Frontier(R)|}\sum_{s_d\in Frontier(R),\ s'_d\in R'} t_d(s_d,\ a^*(s_d,R),\ s'_d) & \text{sinon} \end{cases}$$

Évaluation approximative des récompenses

De façon similaire aux transitions, il est possible d'évaluer approximativement la récompense obtenue dans une région R sur la base d'une moyenne sur les récompenses $r_m(R)$.

$$r_m(R) = \frac{1}{|R|}\sum_{s\in R}\left(\frac{\sum_{a\in A_{ds}} r_d(s,a)}{|A_{ds}|}\right)$$

$$\text{avec} \quad A_{ds} = \{a \mid \exists s'\in S_d, \quad s'\neq s, \quad t_d(s,\ a,\ s')>0\}$$

A_{ds} représente ici l'ensemble des actions disponibles depuis l'état $s\in S_d$. Ainsi, la récompense approchée pour atteindre une région voisine peut être grossièrement évaluée avec un nombre théorique β_t d'actions à réaliser pour sortir d'une région. De la même façon, nous proposons de définir le coût approché pour réaliser les objectifs d'une région en fonction d'un nombre théorique β_p d'actions nécessaires pour passer d'un objectif à un autre. La récompense d'atteindre les objectifs d'une région se calcule alors comme la somme des gains associés aux objectifs moins les coûts évalués.

$$r(\ (R,H),\ target(R')\ ) = \beta_t.r_m(R)$$

$$r(\ (R,H),\ perform(R)\ ) = \sum_{s\in G_R-H} gain(s) - \beta_p.r_m(R)$$

Cette approximation heuristique a été définie dans le cadre de robots mobiles planifiant leurs déplacements à partir d'une carte topologique. Le GO-MDP associé sera faiblement connexe, les actions et les probabilités de transitions connectent uniquement les positions proches. Les récompenses expriment, pour cette problématique, un coût proportionnel à la distance à parcourir en considérant toutes les distances dans un même ordre de grandeur. Les conséquences sur

l'optimalité de la solution par rapport à l'approximation choisie importe peu par rapport au gain de ressources de calcul effectué. En effet, la politique abstraite va servir uniquement à orienter les politiques locales en proposant un ordonnancement des régions pour la réalisation des objectifs qui pourra être localement remis en cause.

4.2.3 Résolution locale région par région

Dans les hypothèses initiales, la stratégie de contrôle du système impose des re-calculs de politiques en cours de mission. Ces re-calculs sont dus à l'actualisation des connaissances de la dynamique du système $\langle S_d,\ A_d,\ t_d,\ r_d \rangle$ et de l'actualisation de l'ensemble des objectifs G attribués à l'agent. Ces actualisations peuvent venir des perceptions propres de l'agent ou d'une communication avec d'autres agents du groupe.

Sur la base des valeurs approximatives calculées via le GO-MDP abstrait, la solution proposée consiste à ne calculer que la politique locale pour la région courante R. Ainsi, en cas d'actualisation de la connaissance, l'actualisation de la politique, avant un contrôle effectif, nécessite uniquement : un nouveau partitionnement, la résolution du GO-MDP abstrait approché et la résolution du GO-MDP local à la région R. La solution proposée approche une politique sur la base de la résolution séquentielle de 2 GO-MDPs (un abstrait et un local), construits avec k et $|G_R|$ objectifs (avec $k \simeq |G|/n^*$ régions et $|G_R| \simeq n^*$ objectifs par région) (Figure 4.4).

Orienter la politique locale

La politique abstraite est principalement utilisée pour orienter la politique locale (Figure 4.4). Pour rappel, l'équation de Bellman [Bellman 57] associant un gain espéré à chaque état en fonction de la politique appliquée, nécessite de connaître localement les valeurs de tous les futurs états atteignables et notamment ceux en dehors de la région courante. Cette évaluation permet en retour de calculer la politique à appliquer.

La résolution d'un GO-MDP local est donc conditionnée par la valeur des états atteignables dans les régions voisines en fonction de l'étape de la mission (objectifs atteints et non-atteints) (Figure 4.5). Concrètement la réalisation des objectifs d'une région va être orientée de façon à terminer dans un état facilitant le passage à la région suivante.

Initialiser la fonction de valeur

Pour rappel, la résolution locale d'un sous-MDP peut être réalisée en considérant une fonction de valeurs définie sur le MDP complet et en itérant uniquement sur les états de la région (Section 4.1.2). L'algorithme utilise alors une évaluation initiale V_{init} des états extérieurs à la région qui ne sera pas modifiée. Dans l'approche proposée ici, l'évaluation initiale découle des valeurs abstraites $V_A^{\pi_A^*}$ attachées à la politique π_A^* calculée pour le GO-MDP abstrait approché.

Les valeurs des états dans les régions voisines sont initialisées à 0 (neutre) tant que tous les objectifs de la région courante ne sont pas atteints. Ainsi, les actions locales de l'agent seront orientées vers la réalisation de tous les objectifs (en sous-entendant des gains positifs forts pour la réalisation des objectifs). Ensuite, la valeur d'un état voisin est associée à la valeur approximative de sa région pour l'étape de réalisation courante H (ensemble des régions dont les objectifs sont atteints) et en considérant que les objectifs de la région actuelle R seront aussi validés.

$$\forall s \in \mathit{Perifery}(R) \quad \begin{array}{ll} V_{init}(s) = V_A^{\pi_A^*}(R',\ H \cup \{R\}) & \text{si tous les objectifs de } R \text{ sont atteints} \\ V_{init}(s) = 0 & \text{sinon} \end{array}$$

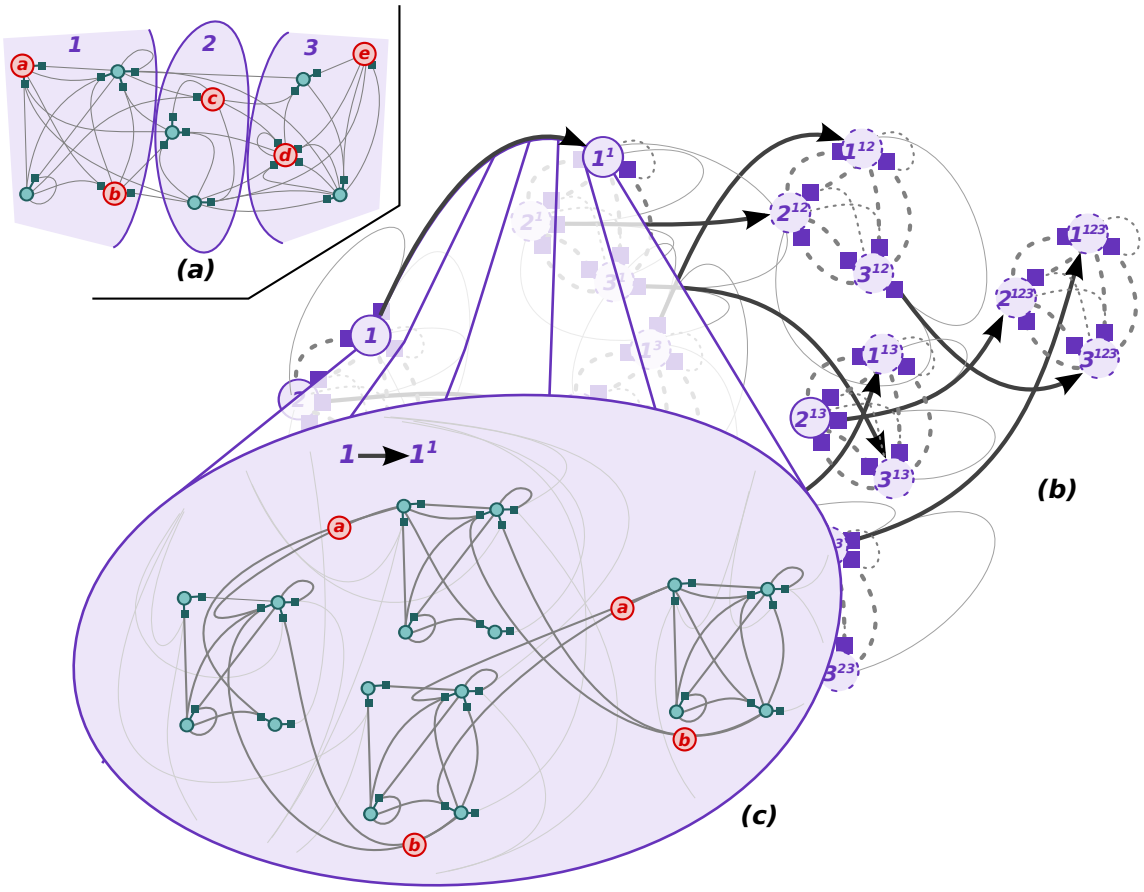


FIGURE 4.4 – Combinaison du GO-MDP abstrait (b) et d'un GO-MDP local (c) correspondant à l'état abstrait $(1, \emptyset)$. Ces 2 GO-MDPs dépendent des modifications réalisées sur le MDP système décomposé (a) et ses 5 objectifs $G = \{a,\ b,\ c,\ d,\ e\}$.

Ainsi, l'action abstraite, définissant une préférence sur l'ordre des régions à parcourir, peut être localement remise en cause. Par exemple, une résolution locale à la région 1 orientée par les deux régions 2 et 3 peut être initialisée avec une préférence sur la région 3 (Figure 4.5). La valeur abstraite pour $(3,\ \{1\})$ sera supérieure à la valeur de $(2,\ \{1\})$. La politique locale peut, elle, chercher à exécuter l'objectif a puis b et viser ensuite la région 2 (si le coût d'aller en 3 est supérieur à la différence des valeurs abstraites entre les régions 2 et 3.)

Malgré des valeurs initiales désavantageuses, le système peut changer de région involontairement avant que tous les objectifs locaux ne soient validés. En effet, l'agent peut choisir des actions avec une probabilité, même faible, de sortir de la région ou l'agent peut simplement ne pas avoir le choix (toutes les actions possèdent une probabilité de sortie). Dans une situation de sortie prématurée, il sera alors utile de ré-initialiser la politique en actualisant le partitionnement, le GO-MDP abstrait et le GO-MDP local courant en tenant compte des objectifs atteints ou non.

4.3 Validation statistique

L'approche hiérarchique proposée permet de résoudre de façon approchée un GO-MDP composé d'un nombre conséquent d'objectifs. Le GO-MDP permet, dans le cadre de nos travaux,

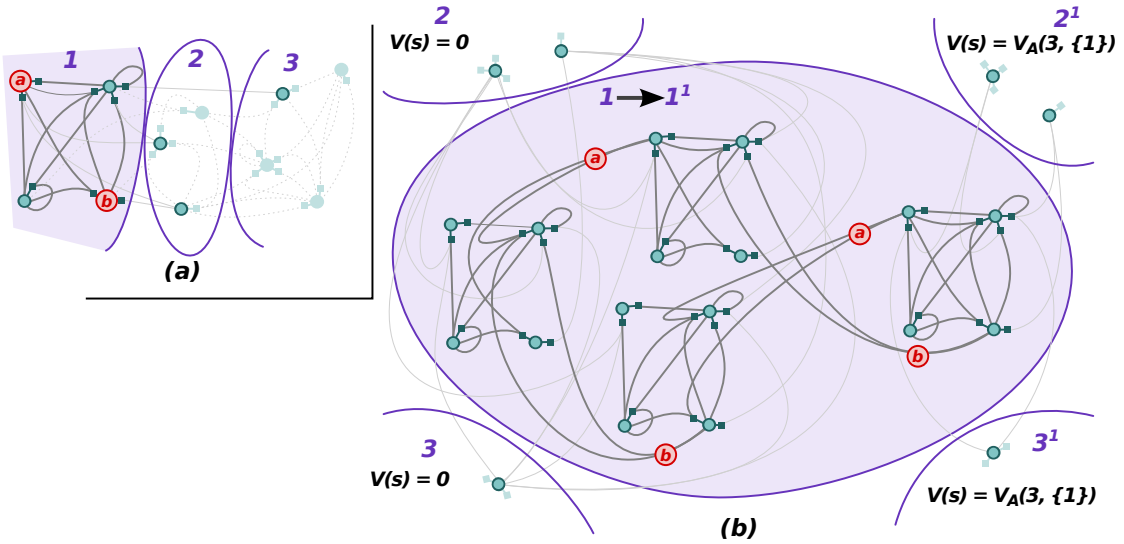


FIGURE 4.5 – Exemple de GO-MDP local (b) construit à partir de la région 1 et de l'ensemble des 2 objectifs $\{a, b\}$ (b). Dans la mesure où tous les objectifs locaux sont réalisés, il sera possible de changer de région en considérant les objectifs de la région 1 comme réalisés.

d'exprimer une problématique de robots explorateurs avec une allocation des objectifs entre les robots. Les objectifs représentent alors, à chaque instant, les frontières à repousser. La carte et les objectifs évoluent au fur et à mesure que l'exploration est réalisée.

La validation expérimentale permet de tester les heuristiques utilisées. Une évaluation qualitative cherche à évaluer les politiques construites notamment vis à vis de politiques optimales. D'autres séries d'expérimentations sont ensuite présentées pour évaluer la rapidité de l'approche pour construire la politique et, par conséquent, la possible utilisation de l'approche en cours de mission d'exploration en robotique mobile.

4.3.1 Problème considéré

Les robots sont contrôlés via l'architecture 2 axes (Chapitre 3). La planification des actions du robot se base sur sa connaissance modélisée dans la carte topologique. Le scénario proposé pour tester l'approche repose sur un robot leader qui a pour tâche de calculer une allocation des objectifs à visiter pour une flotte de 3 robots. L'allocation est basée sur l'espace partitionné et sert ensuite à chaque robot pour calculer sa politique individuelle (Figure 4.6)).

Ce scénario permet d'introduire des contraintes supplémentaires liées à une coordination simple de plusieurs agents. L'objectif consiste à valider l'approche dans un cadre de planification distribuée. Le calcul des politiques individuelles ainsi que le calcul de l'allocation (pour le leader) s'effectuent dans la partie "Délibération" en amont de la supervision du contrôle réactif du robot.

Dynamique du système

L'architecture 2 axes divise la problématique du contrôle de chaque robot sur un axe horizontal (connaissance-décision) et un axe vertical (réactif-cognitif). Le contrôle est ainsi divisé en 4 parties : perception (connaissance, réactif), représentation (connaissance, cognitif), déli-

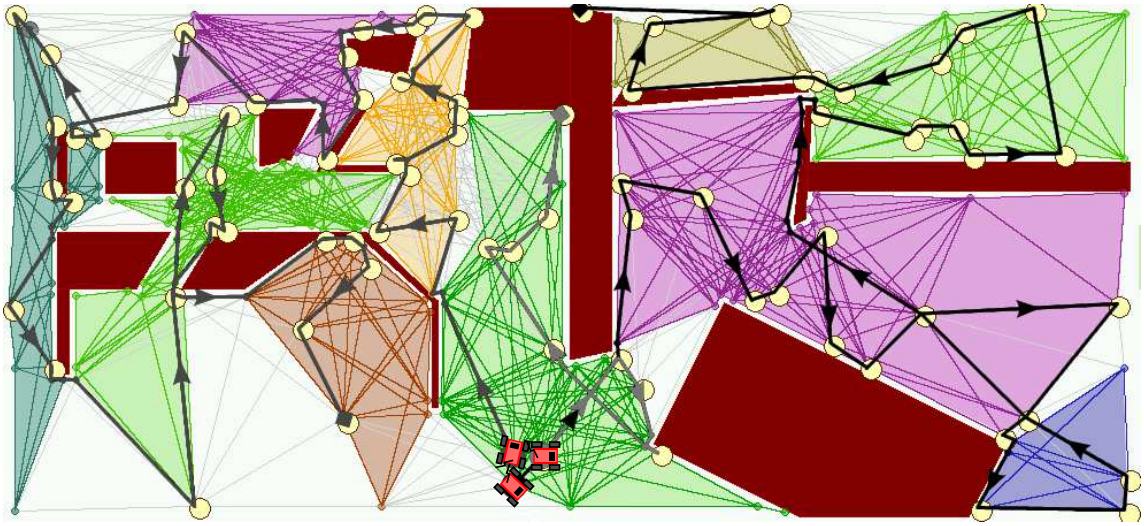


FIGURE 4.6 – Résolution hiérarchique, exemple de parcours empruntés par les 3 robots après répartition des régions par le leader et calcul individuel des politiques locales.

bération (décision, réactif) et contrôle réactif (décision, cognitif). Le calcul de la politique des déplacements intervient dans la partie délibération.

Le calcul de la politique s'effectue sur la base de la connaissance de la partie représentation pour superviser le contrôle réactif. Le contrôle réactif est défini par la réalisation de tâches fonctionnelles (atteindre une position particulière, longer un mur, traverser un carrefour...). La carte topologique organise une connaissance basée sur des positions clefs distantes les unes des autres et reconnaissables par la perception. Il est alors possible de parler de carte de faible densité (comparé à une représentation en grille d'occupation).

Le MDP modélisant la dynamique du système, exprime les possibilités de déplacements d'un robot dans son environnement à partir de sa connaissance individuelle. Chaque robot prend une décision de déplacement chaque fois qu'une position clef est atteinte. Un état s_d correspond alors à une position clef dans la carte. L'ensemble des actions A_d est construit par rapport à l'ensemble des connexions (chemins) dans la carte, lui-même dépendant des déplacements fonctionnels possibles. Enfin, les transitions et les récompenses découlent des déviations possibles du robot et des coûts des déplacements. Du fait de la faible densité, un appartement ou un petit quartier peut être représenté avec une centaine de positions clefs soit une centaine d'états (cf. section 3.4.1).

Allocation des tâches

Le problème de coordination résolu par le leader s'exprime uniquement en terme d'allocation des objectifs à visiter. En supposant une communication efficace en tous points de l'environnement, chaque fois que cela est nécessaire (et notamment à l'étape initiale de la mission multi-robots) le leader doit partitionner sa carte en régions et distribuer les objectifs par paquet entre les robots.

Une allocation *Alloc* est une partition de l'ensemble des objectifs d'une taille égale au nombre de robots présents n. Chaque élément H_{Ag} de l'allocation correspond à l'ensemble des régions contenant au moins un objectif à réaliser par le robot Ag actuellement dans la région R_{Ag}.

L'allocation optimale $Alloc^*$ est l'allocation qui maximise la somme des gains espérés de tous les robots du groupe. Soit, à partir de la politique π_A^* construite sur le GO-MDP abstrait :

$$Alloc^* = \underset{Alloc \in \mathbb{ALLOC}}{argmax} \left(\sum_{H_{Ag} \in Alloc} V_A^{\pi_A^*}(R_{Ag},\ H - H_{Ag}) \right)$$

$\mathbb{ALLOC}$ définit l'ensemble de toutes les allocations possibles et $H - H_{Ag}$ l'ensemble des régions objectifs non attribuées au robot Ag. En effet, la valeur d'une attribution pour un robot correspond à la valeur de son état courant où seuls ses objectifs sont encore à réaliser (soit $V_A^{\pi_A^*}(R_{Ag},\ H - H_{Ag})$ avec un robot dans la région R_{Ag}).

Dans le cadre des expérimentations mises en œuvre ici toutes les allocations possibles pour garder l'allocation optimale. Ce mécanisme est rendu possible du fait que le leader se contente d'attribuer les objectifs par paquet, région par région. La qualité de l'allocation dépend alors, principalement du partitionnement en régions.

4.3.2 Évaluation qualitative

Une évaluation visuelle effectuée sur une première série d'expérimentations permet de tirer quelques conclusions sur le partitionnement de la carte des déplacements. La carte est alors elle-même équivalente à un MDP modélisant la dynamique du système soit, les déplacements à réaliser. Trois exemples (Figure 4.7) permettent d'illustrer nos conclusions.

Dans cette approche, la partition en régions n'est pas une fin en soi et l'approche est évaluée sur la capacité du leader à définir une allocation optimale. Une seconde série d'expériences compare les allocations construites sur le GO-MDP abstrait après partitionnement avec les meilleures et les pires allocations possibles.

Évaluation rapide des partitions

Ainsi, dans des environnements plus ou moins structurés, il a été remarqué que l'algorithme de partitionnement glouton construit une partition proche d'une partition qui serait attendue par un opérateur humain. Dans des environnements structurés (Figure 4.7(c)) avec un nombre cohérent d'objectifs par région, le partitionnement glouton construit approximativement une région par pièce. C'est un bon résultat compte tenu du fait qu'il n'y a pas de calculs globaux sur les coupes dans la succession de choix réalisés par l'algorithme.

Inversement, dans un environnement sans obstacle (MDP fortement connexe), le partitionnement rencontre des difficultés pour construire des régions significatives (Figure 4.7(a)). Les régions sont "allongées" et se chevauchent les unes aux autres. Cela est notamment dû à la non-prise en compte des récompenses et coûts sur les transitions. La notion de distance est seulement construite sur les fortes ou faibles probabilités d'atteindre un état. Dans les exemples Figure 4.7, les transitions liées aux actions de déplacements sont définies de façon homogène quelle que soit la distance entre les positions. Ce phénomène (régions incohérentes), se réduit dans des environnements encombrés même avec des obstacles aléatoires (Figure 4.7(b)).

Efficacité du partitionnement et du GO-MDP abstrait sur l'allocation

La seconde série d'expériences compare la valeur de l'allocation des objectifs du leader (consécutif à un partitionnement glouton et à la résolution du GO-MDP abstrait) avec la valeur d'une allocation optimale. La valeur de la pire allocation est aussi calculée en témoin pour étalonner un score pour l'allocation du leader entre les 2 valeurs extrêmes.

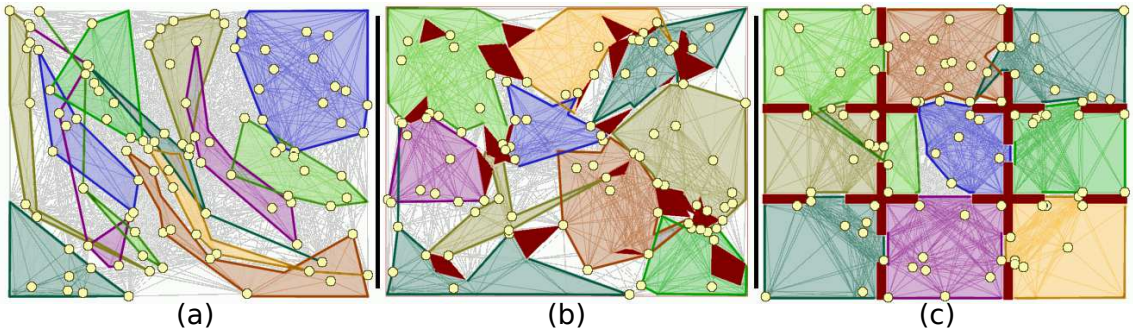


FIGURE 4.7 – Exemples de partitionnement glouton construits pour des environnements présentant des structures différentes : (a) grand espace libre, (b) obstacles aléatoires et (c) pièce de taille homogène. L'algorithme glouton est paramétré avec, comme position initiale, le point supérieur gauche.

La valeur d'une allocation équivaut à la somme des gains des objectifs moins la somme des déplacements probables de chaque robot en fonction de son attribution. Cette valeur est calculée en sommant les utilités espérées données par l'équation de Bellman associée aux politiques optimales globales référentes. Les politiques optimales globales sont calculées en considérant tous les objectifs attribués dans un GO-MDP unique (sans hiérarchisation : GO-MDPs abstrait et locaux). Ces politiques ne servent que pour l'évaluation des allocations dans cette série d'expériences.

$$value(G_0,\ \ldots,\ G_n) = \sum_{Ag \in [O,\ n]} V^{\pi^*}(s_{Ag}, G_{Ag}) \quad \text{avec } s_{Ag} \text{ l'état initial du robot } Ag$$

L'utilité espérée $V^{\pi^*}(s)$ dépend des coûts et récompenses qui seront statistiquement perçus par le robot. Dans le cadre des déplacements, les coûts sont ici calculés de façon proportionnelle à la distance entre les 2 positions ciblées par l'action. Le coût maximal est défini ici à 51.2 pour un déplacement virtuel sur la diagonale de l'environnement. Les gains pour visiter une position objectif sont posés à 1000 pour garantir qu'il est toujours intéressant de chercher à atteindre un objectif. Enfin, le partitionnement glouton est paramètré pour un idéal de 3 objectifs par région.

Les expériences sont réalisées avec une carte composée de zones hétérogènes (différentes densités d'obstacles), tel que présenté Figure 4.6. Les expériences sont classifiées en fonction du nombre total d'objectifs $|G|$ considérés. Ce nombre est limité à 12 à cause de la complexité de calculer l'allocation optimale via une énumération exhaustive de toutes les allocations possibles.

Pour chaque expérience, un score normalisé est calculé pour l'allocation définie par le leader et une allocation aléatoire. Le score correspond à la valeur résultant de la somme des politiques individuelles proportionnellement aux valeurs de la meilleure et de la pire allocation possible. Par exemple, une des expériences, dans la classe $|G| = 11$, qui a permit d'établir les moyennes (Figure 4.8), a donné les valeurs : 10913.8 (meilleur) ; 10822.9 (pire) ; 10872.9 (aléatoire) ; 10884.5 (leader) soit un score normalisé de 0.550 pour l'aléatoire et de 0.678 pour le leader.

La seconde série d'expériences est composée de 200 générations aléatoires d'objectifs pour chaque classe considérée (nombre paramétré d'objectifs). Ce volume permet une évaluation statistique de l'allocation réalisée sur la base d'un GO-MDP abstrait approximatif (Figure 4.8).

Les scores obtenus (Figure 4.8) montrent la difficulté, pour le leader de trouver l'allocation optimale sur la base du GO-MDP abstrait. La moyenne des scores est proche de 77% pour des

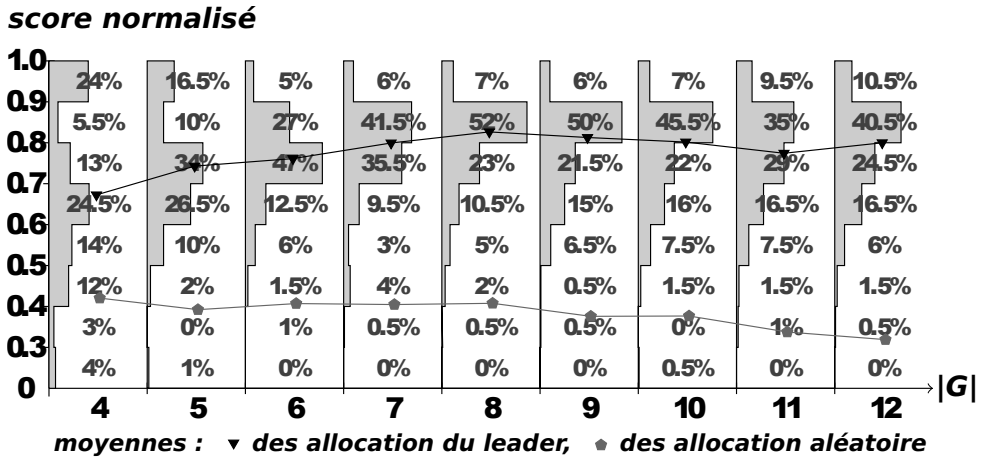


FIGURE 4.8 – Évaluation des allocations résultant d'un partionnement glouton. La population des scores obtenues par une allocation basée sur un GO-MDP abstrait approché est présentée en fonction du nombre d'objectifs considérés $|G|$. Par exemple pour $|G| = 8$: 7% des allocations construites ont eu un score entre 0.9 et 1.0 (avec 1.0 : allocation optimale) ; 52% entre 0.8 et 0.9, etc.

problèmes comprenant entre 6 et 12 objectifs ce qui induit autour de 2 à 4 régions. Cela reste une moyenne acceptable en considérant que le calcul de cette allocation est quasiment instantané. Cette moyenne est affectée négativement par un petit nombre d'expériences présentant de mauvais scores (< 0.5). Ces mauvais scores découlent d'un partitionnement inapproprié avec l'allocation des régions sur l'ensemble des 3 robots. En effet, le leader ne peut attribuer que des régions. Le nombre de ces expériences diminue avec l'augmentation du nombre d'objectifs (et le nombre de régions construites) pour se stabiliser à un score autour des 80% (entre la pire et la meilleure allocation possible). Ces dernières expériences correspondent à un ratio avec autant (ou un peu plus) de régions que d'objectifs par région.

4.3.3 Temps de calcul de la politique

L'efficacité de la solution s'évalue ensuite, par la capacité du partitionnement à contrôler la taille (en terme d'objectifs) et le nombre des régions construites. Ces paramètres vont permettre en retour, de contrôler la taille du GO-MDP abstrait et des GO-MDPs locaux.

L'approche est étudiée pour considérer, en cours de mission, un nombre important d'objectifs. La troisième série d'expériences évalue le temps nécessaire au calcul de la politique abstraite (partitionnement et résolution du GO-MDP abstrait) et pour allouer les régions entre les robots. Les expériences sont réalisées sur la base de : jusqu'à 120 objectifs, 3 robots et un nombre idéal de 10 (tailles et nombre de régions similaires) objectifs par région ($|G| \leq 120$, $n_r = 3$ et $n_i^* = 10$). La série d'expérimentations est générée par positionnement aléatoire des objectifs : 500 générations aléatoires pour les différents nombres d'objectifs $|G|$ considérés (Tableau 4.1).

Le tableau 4.1 et la figure 4.9 présente les résultats expérimentaux permettant de valider le bon contrôle du nombre de régions, du nombre d'objectifs par région et des temps de calcul associés. On peut noter un nombre déséquilibré d'objectifs par régions. La taille des GO-MDP et les temps de calcul augmentent exponentiellement avec le nombre de régions k construites (GO-MDP abstrait) et avec le nombre d'objectifs dans les régions (GO-MDP locaux). C'est un

$\|G\|$	nombre moyen de régions (k)	bornes moyennes sur la taille des régions	temps (s)	allocation la plus fréquente
5	1.13	$4.38 - 4.97$	0.059	$1 - 0 - 0$
20	3.41	$3.43 - 8.13$	0.047	$2 - 1 - 0$
40	5.6	$4.12 - 10.14$	0.053	$3 - 2 - 1$
60	7.71	$4.29 - 11.44$	0.116	$4 - 2 - 1$
80	9.43	$4.73 - 12.74$	0.339	$5 - 3 - 1$
100	10.63	$4.36 - 14.19$	0.897	$6 - 3 - 1$
120	12.42	$3.74 - 14.81$	4.95	$6 - 4 - 2$

TABLE 4.1 – Moyennes mesurées caractérisant la solution proposée par le leader. La dernière colonne permet de se représenter l'allocation type construite. Par exemple, avec 60 objectifs et autour de 7 régions, l'allocation type attribue 4 régions à un agent, 2 à un autre et 1 au dernier.

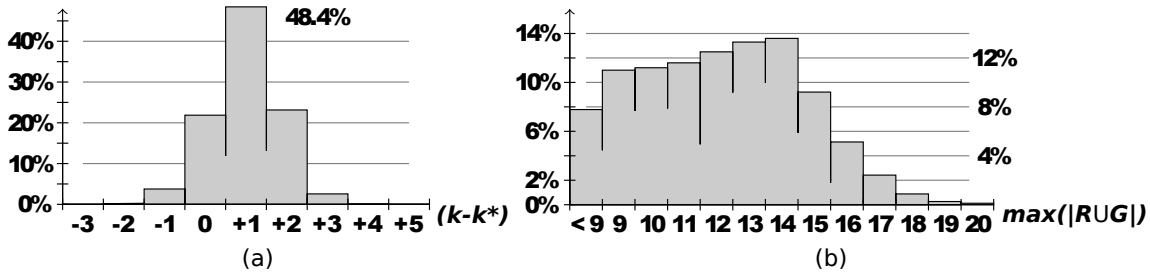


FIGURE 4.9 – Tailles des partitions gloutonnes créées. Les diagrammes (a) et (b) classifient les expériences en fonction : (a) de la différence entre le nombre construit k et le nombre idéal k^* de régions et (b) le nombre d'objectifs contenus dans la région la plus importante. Par exemple, (a) 48.4% des partitionnements ont construit une région de trop par rapport au nombre idéal k^* et (b) 13.5% des régions possèdent 14 objectifs.

résultat tout à fait cohérent sachant que la taille des GO-MDPs augmentent exponentiellement avec le nombre d'objectifs.

Les expériences comportant un faible nombre d'objectifs par rapport à la densité des obstacles (le nombre et leurs formes) conduisent à des partitions avec plus de régions qu'attendu ($k > k^*$) et des régions, voire toutes, avec un faible nombre d'objectifs ($|R \cup G| < n^*$ voire $max(|R \cup G|) < n^*$) (Figure 4.9). Ce résultat montre la souplesse du partitionnement vis à vis du nombre idéal d'objectifs par région et sa capacité à profiter de la configuration des obstacles pour passer d'une région à une autre.

On note aussi (Tableau 4.1) que l'allocation réalisée par le leader, sur la base de son GO-MDP abstrait, est déséquilibrée. Cela est normal étant donnée la configuration des obstacles (Figure 4.6). En effet, l'objectif du leader est de minimiser la somme des déplacements des robots, le critère employé ne cherche aucunement à équilibrer la charge de travail. Ces allocations correspondent à des missions où la consommation des ressources est critique. C'est-à-dire que l'objectif, pour le groupe de robots, consiste à réaliser un maximum de tâches avant que les batteries ou les réserves de carburant ne soient vides.

4.4 Conclusion

Le problème de planification par résolution d'un MDP large est soulevé avec une contrainte forte sur la rapidité des calculs mis en œuvre. Les approches par décomposition d'un processus décisionnel de Markov (DMDP) permettent d'accélérer la résolution d'un MDP large. Ces approches nécessitent alors un partitionnement de l'ensemble des états couplés au calcul de plusieurs politiques par région. Une résolution optimale d'un MDP décomposé orienté par des objectifs (GO-DMDP) est difficilement envisageable dans la mesure où même une simple énumération des états est fastidieuse.

L'approche présentée ici propose une solution *ad hoc* à un problème multi-objectifs, permettant de focaliser les calculs sur seulement quelques politiques locales, au fur est à mesure que les régions sont atteintes. Le partitionnement glouton d'un MDP définissant la dynamique du système (sans intégrer les objectifs) permet de construire une hiérarchie de GO-MDPs. Un GO-MDP abstrait approché, construit par rapport à l'ensemble des régions permet d'évaluer grossièrement les valeurs de gains espérés par région. Ces valeurs orientent ensuite la politique locale calculée pour la région dans laquelle le robot est positionné.

Les séries d'expériences ont montré que l'approche permet de calculer des politiques acceptables dans un temps cohérent. Dans un cadre multi-robots, ces politiques individuelles permettent une allocation rapide des objectifs sans perte importante sur la qualité de la coordination. La vitesse de convergence vers la solution est garantie par un partitionnement glouton offrant un bon contrôle statistique sur la taille de la partition et sur le nombre d'objectifs contenus dans chaque région.

Deux défis ont été surmontés, à savoir, résoudre certains MDPs qui ont la particularité d'être de très grande taille, tout en permettant une résolution qui puisse être réalisée en ligne. En effet, la décomposition en sous-MDPs proposée est caractérisée par des temps de résolution courts pour une faible dégradation sur la qualité des solutions produites.

Orientations futures

La résolution des MDPs orientés par des objectifs est une problématique intéressante pour des approches hiérarchiques approchées. Le modèle est suffisamment riche pour exprimer simplement des problématiques variées, pour lesquelles l'espace d'états est "non-énumérable".

Dans la continuité directe de ce travail, il est possible d'étudier, de façon plus approfondie, les conséquences du partitionnement sur la politique individuelle des robots. L'idée repose sur une évaluation comparative des techniques de partitionnement par rapport aux valeurs des politiques construites. Je ne suis pas, pour ma part, convaincu que le coût supplémentaire en ressources de calcul, d'une optimisation de la partition soit justifié par de réelles améliorations de la politique résultante. Cependant, cette supposition reste à vérifier.

Le GO-MDP couplé à une résolution hiérarchique a été étudié ici, dans le cadre d'une application particulière en robotique mobile. Une approximation plus fine du GO-MDP abstrait pourrait permettre de considérer des problématiques plus générales. La question de caractériser rapidement les politiques locales possibles nécessite, à mon sens, d'évaluer les transitions coupées en tenant compte de l'intérêt possible des actions, donné par les récompenses associées.

A plus long terme, j'aimerais chercher à optimiser les politiques locales en modélisant les transitions entre les régions de façon plus fine. Je pense qu'une remise en cause de l'heuristique du "tout ou rien" est indispensable à une amélioration de la politique globale. Cette heuristique force le contrôle à atteindre tous les objectifs d'une région avant de passer à la suivante même si la région est amenée à être traversée plusieurs fois ou que plusieurs objectifs sont proches mais

séparés par la partition. D'un autre côté, c'est principalement cette heuristique qui permet une construction des décisions rapides et les méthodes proposées ne devront pas alourdir de manière trop importante les calculs nécessaires à la détermination de la solution.

Chapitre 5

Coordination multi-agents par enchères

Une modélisation par un processus décisionnel de Markov orienté par des objectifs (Goals Oriented MDP : GO-MDP) permet à chaque agent/robot de calculer sa politique pour réaliser les objectifs attribués individuellement. Dans le cadre de missions multi-agents où la coordination des actions repose sur une allocation d'objectifs (ou de tâches), l'autonomie du groupe dépend de la capacité des agents à calculer une allocation permettant une réalisation efficace de la mission. Pour des missions muti-robots l'efficacité peut se mesurer en général par l'énergie consommée ou le temps nécessaire à la réalisation totale de la mission. En d'autres termes, l'efficacité correspond à une minimisation de la somme des déplacements de tous les agents ou de la somme des déplacements de l'agent avec la charge maximale.

La valeur d'une allocation dépend donc des politiques individuelles. Cependant, les politiques individuelles sont, elles-mêmes, fonction d'une allocation. Le calcul de l'allocation d'objectifs individuels et le calcul des politiques sont donc fortement liés. Le principe repose généralement sur des tests successifs d'allocations candidates jusqu'à converger vers une allocation optimale ou, à défaut, sur une allocation sous-optimale acceptable. Chaque test d'allocation candidate implique d'évaluer simultanément les politiques individuelles correspondantes. Cette évaluation peut être coûteuse et elle repose sur des données individuelles. Ainsi, de nombreuses approches distribuées s'intéressent au problème de l'allocation d'objectifs, de tâches ou de ressources [Davis 83, Gerkey 02, Burgard 05, Tovey 05, Dias 06, Matignon 12].

Dans les principales approches distribuées, les agents partagent une allocation commune et leurs valeurs d'efficacité individuelles sont calculées, elles, de façon distribuées. Il est alors nécessaire de mettre en place un protocole générant des allocations candidates communes. Les protocoles de ventes aux enchères proposent aussi un cadre théorique pour allouer des objets entre plusieurs agents en concurrence. Le principe repose sur des ventes où chaque agent peut enchérir sur la valeur du/des objets. Le/les objets sont ensuite attribués à l'agent (ou à la combinaison d'agents) offrant la/les enchères les plus intéressantes. Pour une coordination multi-agents, les objets peuvent définir des objectifs, des tâches ou encore des ressources à allouer entre les agents.

Les particularités liées à la coordination d'agents reposent essentiellement sur le fait que les valeurs des objets sont corrélées et que les agents ne sont pas réellement en concurrence. En effet, il est souvent difficile pour un agent d'évaluer, de façon unique, une tâche, un objectif ou une ressource alors que son intérêt dépend des autres tâches, objectifs ou ressources qui seront attribués à l'agent. Par contre, un groupe coopératif peut (et a tout intérêt à) remettre en cause autant de fois que nécessaire, l'allocation d'un objet. Dans ce cadre de travail, la proposition développée dans ce chapitre repose sur un protocole de coordination basé sur une succession de ventes aux enchères simultanées SSAC (Succesion of Simultaneous Auction for Coordination).

L'idée consiste à définir des règles de fonctionnement d'une vente permettant à chaque itération de remettre en cause l'attribution de chacun des objets. Il devient alors théoriquement possible de réaliser simultanément plusieurs commutations sur l'allocation en construction de façon à converger rapidement vers une allocation optimale localement.

Ce chapitre est constitué d'une première section présentant un état de l'art des approches par ventes pour la coordination multi-agents. La section suivante détaille le protocole de coordination proposé consistant en une succession de ventes aux enchères simultanées. L'accent sera mis sur la garantie de la convergence vers une allocation sans *quiproquo* et qui fasse consensus entre les agents. Enfin la section 3 présente les expériences mises en place pour valider l'approche dans le cadre de l'application robotique visée correspondant en un problème de multi-voyageurs de commerce avec actualisation possible en cours de missions. Enfin, ce chapitre se terminera par une conclusion et une discussion sur les orientations futures possibles propres aux idées développées ici, pour la coordination distribuée.

L'idée de coordination par ventes simultanées présentées dans ce chapitre a été publiée aux Journées Francophones des Systèmes Multi-Agents 2011 (JFSMA) [Lozenguez 11b]. Une version plus aboutie regroupant notamment les dernières expérimentations est actuellement en cours de soumission pour une conférence internationale.

5.1 Fondement théorique, allocation de tâches

Le challenge d'une planification distribuée consiste à donner à chaque agent, les moyens de calculer sa politique individuelle de façon à permettre au groupe d'atteindre ses objectifs. Une telle approche s'oppose à une modélisation centralisée où les récompenses sont définies pour le groupe en fonction de l'action jointe des agents. Une action jointe correspond à l'ensemble des actions individuelles réalisées en même temps. Citons notamment la modélisation par Processus Décisionnel de Markov Partiellement Observable et Décentralisé (Dec-POMDP) (en l'occurrence ce sont les décisions et les observations qui sont décentralisées, pas le modèle) [Bernstein 00, Goldman 04, Beynier 06, Canu 11]. Une telle modélisation impose, *a priori*, de raisonner sur les actions réalisées de façon simultanée et donc sur la conjonction des états de tous les agents en respectant leurs capacités perceptives. L'espace requis pour développer un modèle générique centralisé explose alors de façon combinatoire avec l'ajout de nouveaux agents.

Une première solution consiste en une co-évolution des politiques individuelles [Chades 02, Nair 05]. A chaque itération, un agent actualise sa politique en considérant les politiques des autres agents du groupe comme fixes puis il envoie sa solution individuelle aux autres. Les agents affinent successivement leurs politiques individuelles jusqu'à ce qu'il n'y ait plus d'amélioration possible. Cette approche permet de converger vers une solution localement optimale mais reste difficile à mettre en place. A chaque itération, l'agent doit modifier les valeurs de son modèle individuel (un POMDP), le résoudre puis communiquer sa politique définie par $|B|$ actions avec B, l'ensemble des états individuels (de croyances pour un POMDP). Cette solution n'est donc pas applicable dans le cadre d'une problématique orientée par plusieurs objectifs où le nombre d'états du système augmente exponentiellement par rapport au nombre d'objectifs.

Par contre, une approche par co-évolution de politiques, comme avec l'algorithme JESP (Joint Equilibrium-based Search for Policies) [Nair 05] permet de traiter des problèmes de coordination générique. Dans un cadre de voyageurs de commerce, il est possible de considérer la problématique sous la forme d'une allocation de tâches. Une approche par recherche de consensus [Ren 08] consiste à définir un ensemble de variables d'interaction (tâches/ressources à allouer, sens de circulation, rôles, rendez-vous à définir, etc.). Ensuite, la problématique de coordination se divise

en : affectation des variables d'interaction et planification distribuée. Concrètement ici, les variables d'interaction correspondent aux objectifs à réaliser individuellement par les agents. Ainsi, sur la base d'une allocation d'objectifs entre les agents, chaque agent va pouvoir calculer, de façon distribuée, sa propre politique sans se soucier davantage des actions choisies par les autres agents.

5.1.1 Protocole de ventes aux enchères

L'allocation par ventes aux enchères repose sur les procédés issues des problématiques liées au commerce. Une vente aux enchères consiste simplement à allouer un objet au plus offrant de façon à réguler l'offre et la demande. Avec une prix d'appel bas qui sera surenchéri, ces protocoles optimisent les chances de trouver un acquéreur et les gains du vendeur. Cependant, en fonction du protocole, les gains du vendeur seront limités par l'enchère en seconde position, l'enchère maximale de l'acquéreur le plus intéressé ne sera pas forcément atteinte.

Dans le cadre de l'allocation de tâches, les tâches correspondent aux objets à vendre et les agents sont mis en concurrence pour les réaliser. En règle générale, une tâche doit forcément être allouée et l'allocation doit permettre de minimiser les coûts liés à sa résolution. Ainsi, chaque tâche va être attribuée à l'agent avec un coût de réalisation minimal.

Principe de fonctionnement

La coopération basée sur les ventes aux enchères est largement utilisée en robotique [Dias 06] dans la mesure où cette coopération peut s'exprimer sous la forme d'une allocation de tâches et/ou de ressources. Dans "Contract net" [Davis 83] ou "Murdoch" [Gerkey 02] les ventes reposent sur une distribution de rôles dynamiques. Un objet peut être mis en vente par n'importe quel agent, celui-ci devient le "manager de l'objet". Les autres agents sont alors des clients potentiels, ils vont enchérir une fois avec leurs valeurs liées à leurs coûts respectifs pour réaliser la tâche (e.g. consommation de ressources ou de temps). L'objet est ensuite alloué à l'agent proposant la valeur la plus intéressante.

L'évaluation d'une tâche peut s'effectuer sur la base d'une valeur nominale positive de gain. L'enchère sur une tâche pour un robot est calculée par la soustraction de son coût d'exécution à sa valeur nominale [Dias 06]. Une seconde solution consiste à considérer une vente inversée, l'objet est alloué à l'agent proposant l'enchère minimale calculée uniquement sur les coûts [Tovey 05].

L'agent acquéreur peut alors réaliser la tâche ou la remettre en vente. En considérant que chaque robot réalise une tâche à la fois, la remise en vente peut découler d'une acquisition d'une seconde tâche plus intéressante [Davis 83]. Plusieurs ventes peuvent ainsi s'effectuer en même temps avec plusieurs robots managers d'objets différents.

Rapidité et communications limitées

En considérant une vente par objet, ces protocoles permettent une coordination rapide ne nécessitant que peu de communication. En effet, l'allocation d'un objet implique l'envoi d'un premier message à tous les agents pour mettre en vente une tâche et définir les rôles, l'envoi d'une valeur par agent au manager et enfin l'envoi d'un message confirmant l'attribution de l'objet au meilleur agent.

D'autre part, ces protocoles peuvent être mis en œuvre par sous-groupe. C'est-à-dire qu'il n'est pas obligatoire qu'un groupe soit au complet pour attribuer une tâche. Ainsi, sur la base d'un manager principal distribuant des tâches à réaliser, celui-ci n'est pas dans l'obligation d'attendre le retour de tous les agents et plusieurs tâches peuvent être allouées pendant que d'autres sont

réalisées [Davis 83]. La distribution dynamique des rôles permet aussi de redistribuer les tâches à réaliser entre sous-groupes d'agents lorsque ceux-ci se croisent dans l'environnement. Il en résulte des mécanismes multi-agents simples à mettre en œuvre pour l'allocation de tâche unitaire entre des robots hétérogènes dans des scénarios d'exécution de tâches en parallèle.

Limitations

Dans le cadre de ventes aux enchères pour la coordination d'agents coopératifs, on est peu sujet aux problématiques liées au respect des données privées ou, plus largement, à l'optimisation de gains individuels au détriment des autres [P. R. Milgrom 82]. Les enchères peuvent se faire en toute transparence. Par contre, les valeurs des tâches peuvent être interdépendantes. C'est-à-dire que la valeur d'une tâche pour un agent dépend de l'allocation des autres tâches.

Dans un cadre où l'agent exécute les tâches au fur et à mesure qu'il s'en porte acquéreur [Davis 83], l'optimalité du système dépend alors de l'ordonnancement des ventes. En effet, avec un système de 2 robots et 2 tâches, il est facile de converger vers une allocation optimale localement. Supposons que le robot 1 propose des valeurs homogènes et toujours supérieures aux valeurs du robot 2 et que le robot 2 a une forte préférence pour la tâche A plutôt que pour la tâche B, alors, l'allocation optimale attribue la tâche A au robot 2 et la tâche B au robot 1. Pourtant, si la tâche A est mise en vente en premier, le robot 1 va s'en porter acquéreur et le robot 2 sera forcé de réaliser la tâche B.

Il est aussi possible de chercher à allouer plusieurs tâches pendant une même phase de coordination. Ainsi, le robot 1 peut remettre en vente la tâche A après avoir acquis la tâche B avec une différence sur les gains individuels importante. Ainsi, chaque tâche peut potentiellement être revendue autant de fois qu'il y a d'agents pour s'en porter acquéreur et la remettre en cause par une acquisition suivante. Le problème de ventes successives se complique encore davantage si chacun des agents peut se porter acquéreur de plusieurs tâches avant de les réaliser. Dans cette situation, la valeur unitaire de chaque tâche dépend généralement de l'ensemble des tâches attribuées à l'agent. Ces ventes sont identifiées dans la littérature comme des ventes d'objets corrélés ou avec synergies [Krishna 96, Rothkopf 98, Berhault 03, Vries 03, Tovey 05, Dias 06].

5.1.2 Ventes d'objets corrélés

La gestion des valeurs corrélées entre les objets est un sous-problème lié à des ventes aux enchères multiples. Ce problème découle de la synergie pouvant exister entre plusieurs objets en vente. Concrètement, pour la réalisation de tâches localisées dans l'environnement, si plusieurs tâches proches les unes des autres sont allouées à un même robot, les coûts de déplacements pourront être mutualisés (Figure 5.1). Plus un agent possède de tâches dans un groupe de tâches proches les unes des autres, plus la valeur de chacune de ces tâches sera intéressante pour lui.

La corrélation entre les objets peut aussi s'exprimer de façon inverse : l'acquisition d'un object initialement intéressant pour un agent peut s'avérer regrettable au fur et à mesure que l'agent s'attribue d'autres objets. La difficulté consiste alors à proposer des protocoles de ventes permettant aux agents d'optimiser la construction d'une allocation avec des groupes de tâches profitant de synergies intéressantes.

Ventes combinatoires

Le principe des ventes combinatoires repose sur la possibilité d'exprimer des enchères sur un groupe d'objets [Rothkopf 98, Vries 03]. Un manager met simultanément en vente plusieurs objets et chacun des clients propose une enchère sur un sous-ensemble des objets. Le manager

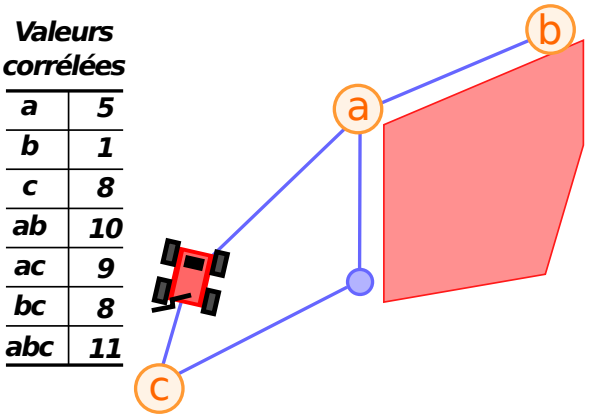


FIGURE 5.1 – Exemple d'une évaluation corrélée. Ici, l'intérêt pour la tâche b est fortement différente si le robot possède ou non la tâche a. Inversement, si la tâche c, réalisée seule, est intéressante, elle ne possède aucune synergie avec les autres tâches.

de la vente sélectionne alors la combinaison d'enchères entre les agents qui maximise la somme des valeurs. Le manager peut ainsi sélectionner un agent acquéreur par objet ou groupe d'objets en respectant les synergies exprimées.

Cette solution repose sur de nombreux calculs individuels. En effet, chaque agent doit calculer les valeurs de tous les sous-ensembles d'objets (soit $2^{|objets|}$ valeurs, exemple Figure 5.1). De plus, pour permettre au manager de sélectionner l'allocation optimale, chaque agent doit envoyer l'intégralité de son vecteur de valeurs impliquant une charge importante de communication. De son côté, le manager doit explorer l'ensemble des vecteurs valeurs pour chercher l'allocation optimale.

L'intérêt de ces approches apparaît dans une utilisation où les agents détectent et n'expriment que les synergies sur des groupes d'objets critiques de façon à borner le nombre de valeurs exprimées individuellement [Rothkopf 98, Vries 03]. Dans un cadre robotique, une solution basée sur des ventes combinatoires permet d'exprimer une synergie entre plusieurs tâches à réaliser [Berhault 03]. Cependant, une solution par ventes combinatoires reste limitée à une allocation de quelques tâches ou en considérant une limite sur le nombre d'objets pouvant être vendus simultanément.

Ventes séquentielles *versus* simultanées

Dans le cadre de ventes de plusieurs objets, le principe de ventes aux enchères séquentielles repose sur l'allocation successive des objets. Elles s'opposent à des ventes simultanées où tous les objets sont vendus individuellement et en même temps. Les ventes simultanées permettent d'accélérer le processus de vente dans la mesure où toutes les ventes sont réalisées sur une même durée. Par contre, l'évaluation des valeurs corrélées d'un objet est plus difficile si d'autres objets sont aussi en cours d'attribution.

Il est alors possible de revendre des objets après acquisition [Davis 83, Tovey 05] chaque fois que la valeur d'un des objets est remise en cause par l'acquisition d'autres objets. Il se pose alors la question : quand remettre en vente un objet ? Concrètement, même si la valeur d'un objet semble toujours intéressante pour l'agent le possédant, rien ne lui permet de garantir qu'un autre agent ne pourrait pas surenchérir sur la base de la nouvelle allocation.

Au pire, il faut remettre en vente systématiquement, de façon séquentielle ou simultanée, tous les objets jusqu'à trouver une allocation stable. C'est-à-dire que, quel que soit l'objet remis en vente, les valeurs proposées n'impliquent aucun changement sur les attributions individuelles. Cette solution peut nécessiter un nombre important de ventes aux enchères (un nombre borné si on considère qu'il n'est pas possible de tester 2 fois, une même allocation).

Optimalité des solutions

La succession de ventes permet d'exprimer des valeurs individuelles en synergie avec l'attribution courante de l'agent. Ainsi, au fur et à mesure que l'allocation est actualisée, les attributions individuelles peuvent être affinées et les valeurs échangées seront proches des valeurs pour l'allocation finale. Cependant, dans la mesure où chaque objet est évalué individuellement et où il est impossible de garantir les acquisitions futures pour évaluer un objet, ces approches ne permettent de converger que vers des solutions optimales localement. En effet, les remises en vente (simultanées ou séquentielles) successives ne permettent que des optimisations locales de l'allocation courante.

Le protocole séquentiel ne permet d'exprimer qu'une synergie sur l'allocation courante avec ou sans l'objet actuellement en vente ; soit une synergie à plus ou moins 1 objet. Cependant, certaines synergies peuvent apparaître par l'ajout ou la suppression simultanée de plusieurs objets. Par exemple, le robot 1 peut avoir des valeurs intéressantes pour le triplé de tâches A, B et C mais des valeurs insignifiantes pour chacune de ces tâches prises séparément. Le robot 1 sera alors, *a priori*, incapable d'acquérir deux des objets lui permettant d'exprimer sa synergie sur le triplé.

Dans un cadre de ventes simultanées non combinatoires, les valeurs d'enchères expriment un intérêt sur les objets pris individuellement. Si le calcul de ces intérêts unitaires peut prendre en compte de possibles synergies dans l'attribution visée par l'agent, l'allocation ne respectera pas forcément ces synergies [Rothkopf 98, Vries 03]. A la remise en vente suivante, il est alors nécessaire de proposer des règles d'ajustement pour proposer de nouvelles enchères et converger vers une allocation en accord avec les valeurs communiquées par les agents.

5.1.3 Distributivité du critère d'efficacité

Les protocoles de ventes aux enchères s'effectuent par évaluation individuelle des objets en vente. Une coordination basée sur cette approche permet une allocation des tâches et des ressources de façon à optimiser les comportements individuels. Cependant, l'efficacité du comportement collectif ne s'exprime pas forcément par la simple somme des comportements individuels mais par leurs combinaisons. Par exemple, la rapidité de réalisation d'une mission multi-agents de plusieurs tâches indépendantes, dépend uniquement de l'agent ayant la plus forte charge de travail (de l'agent qui prendra le plus de temps pour réaliser ses tâches). D'autre part, au delà de l'allocation des tâches et des ressources, le choix des actions individuelles a aussi un impact sur le groupe et l'environnement. La gestion du trafic, par exemple, relève d'une évaluation des chemins en fonction de leurs fréquentations. Une allocation de tâches en vue d'optimiser un trafic ne s'exprimera pas uniquement en terme de minimisation individuelle des distances à parcourir mais nécessitera de tenir compte des chemins parcourus par tous les autres agents.

Les enchères traditionnelles consistent à mettre les agents en concurrence et ne permettent pas directement d'exprimer ces problématiques. Il est alors nécessaire de mettre en place une intelligence collective artificielle [Matarić 95, Tumer 00] pour modéliser les interactions possibles et modifier les comportements individuels en vue d'une amélioration de l'efficacité du comportement

collectif. A partir des approches par ventes aux enchères, il existe 2 principales approches : modifier les règles permettant au(x) manager(s) d'attribuer un ou plusieurs objets, et/ou proposer une évaluation individuelle altruiste des objets en vente.

L'évaluation d'une coordination par l'allocation de tâches et de ressources peut être calculée selon plusieurs critères. L'évaluation peut privilégier l'économie de ressources, le temps d'exécution et/ou l'équité de la charge de travail. Il en découle trois principales formules [Tovey 05] : optimiser (maximiser ou minimiser) la somme des coûts/récompenses individuelles, optimiser l'agent délivrant les moins bons résultats ou optimiser la moyenne des coûts/récompenses par tâche et par attribution.

Optimisation simple

Optimiser la somme des coûts/récompenses individuelles consiste à chercher l'allocation $Alloc^* = (G_0, \ldots, G_n)$ qui maximise ou minimise les valeurs individuelles $VA_{Ag}(G_{Ag})$ liées à l'attribution des tâches G_{Ag} à chacun des n agents (Figure 5.2). Cette valeur peut être calculée par l'équation de Bellman dans le cadre de l'utilisation de processus décisionnels de Markov [Burgard 05, Spaan 10, Capitan 12] :

$$Alloc^* = \underset{Alloc=\{G_0,\ldots,G_n\}}{argopt} \sum_{Ag} VA_{Ag}(G_{Ag})$$
$$\text{avec} \quad argopt = argmin \text{ ou } argmax$$

L'optimisation de la somme repose sur le calcul des intérêts individuels. Cette solution est compatible avec des enchères traditionnelles. Les valeurs d'enchères $interet_{Ag}(G_{Ag},\ g)$ de chaque tâche g peuvent être calculées en comparant l'attribution courante avec ou sans cette tâche (Figure 5.2).

$$interet_{Ag}(G_{Ag},\ g) = VA_{Ag}(G_{Ag} + g) - VA_{Ag}(G_{Ag} - g)$$

G_{Ag}	VA_{Ag}	$Intérêt_{Ag}$ a	b	c
	0	5	1	8
a	5	5	5	4
b	1	9	1	7
c	8	1	0	8
ab	10	9	5	1
ac	9	1	2	4
bc	8	3	0	7
abc	11	3	2	1

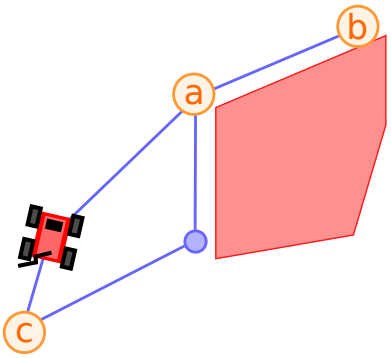


FIGURE 5.2 – Exemple d'intérêt unitaire pour la tâche a en fonction de l'allocation des tâches G_{Ag} pour un robot mobile. Ici par exemple, si le robot possède ac l'intérêt de garder a (1) ou d'ajouter b (2) est faible et l'intérêt lié à c (4) est moyen.

Dans le cas où la tâche g est attribuée à l'agent Ag ($g \in G_{Ag}$), $interet_{Ag}(G_{Ag},\ g)$ représente le coût de libérer cette tâche. Dans le cas contraire, $interet_{Ag}(G_{Ag},\ g)$ représente le gain d'attribuer cette tâche à Ag.

L'allocation optimale correspond bien à une allocation optimisant les valeurs d'enchères individuelles par tâche. En effet, s'il existe des valeurs d'enchères impliquant un changement d'attribution, alors il existe un gain sur la somme des valeurs individuelles et l'allocation n'est pas

optimale. En revanche, une allocation stable (pas d'échange unitaire possible) ne garantit pas d'être optimale. En effet, il pourrait exister une meilleure allocation résultant de plusieurs commutations simultanées d'objets entre les agents.

Critères plus globaux

Optimiser le moins bon comportement des agents ou d'optimisation de l'équité sur la charge de travail implique une évaluation basée sur la comparaison des valeurs des agents les uns par rapport aux autres. Ces deux critères peuvent être formulés comme suit :

– maximiser les gains minimaux :

$$Alloc^* = \underset{Alloc=\{G_0,\ldots,G_n\}}{argmax} \left(\min_{Ag}(\ VA_{Ag}(G_{Ag}) \ ) \right)$$

– minimiser les écarts à la moyenne :

$$Alloc^* = \underset{Alloc=\{G_0,\ldots,G_n\}}{argmin} \sum_{Ag} \left| \frac{\sum_{Ag_j} V_{Ag_j}(G_{Ag})}{n} - V_{Ag}(G_{Ag}) \right|$$

Ici, une valeur d'enchère est fonction de l'allocation de l'agent et de l'allocation sur l'ensemble des autres agents. Dans le premier cas, l'enchère est fonction de l'impact de l'attribution sur le "pire" agent et dans l'autre cas sur la moyenne des n agents. Dans une vente aux enchères traditionnelle, chaque agent n'est pas, *a priori*, en possession des informations requises au calcul de son enchère.

Ces enchères impliquant une efficacité du comportement collectif, différent de la simple somme des comportements individuels. Il est possible d'adapter les règles d'attribution et les règles de calcul des enchères en conséquence. En effet, le manager d'une vente peut centraliser les informations lui permettant un choix sur un critère global qui peut être basé sur une évaluation individuelle particulière des enchères [Tovey 05].

Ainsi, dans le cadre d'une optimisation du "pire" agent dans un protocole de ventes successives, il est possible pour le manager d'allouer la tâche en favorisant l'agent possédant la charge minimale. Dans ce cadre la, chaque agent enchérit non pas par son intérêt lié à la réalisation de la tâche, mais avec la valeur de son attribution totale $VA_{Ag}(G_{Ag} + g)$ incluant sa charge actuelle G_{Ag} plus la tâche en discussion g. Avec un même système du calcul des enchères, le manager peut aussi favoriser une répartition équitable des valeurs individuelles.

Une autre solution consiste à répercuter un coût/récompense sociale dans l'évaluation individuelle d'une tâche. Le principe repose sur la comparaison des intérêts individuels et des intérêts collectifs de façon à construire une enchère altruiste. Chaque agent doit alors être en mesure d'évaluer l'impact de ses choix sur le groupe. Une telle approche repose généralement sur le modèle décisionnel permettant d'évaluer la tâche. Dans le cadre de processus décisionnels de Markov par exemple, il est possible de définir une fonction de récompense en intégrant un facteur social [Boussard 08, Beynier 06].

5.2 Coordination par succession de ventes aux enchères simultanées

La coordination d'agents coopératifs par ventes aux enchères repose généralement sur 2 particularités principales :

– l'allocation des tâches et des ressources sont corrélées,
– les agents ne sont pas "réellement" en concurrence.

L'évaluation de biens corrélés est difficile, elle nécessite de détecter les synergies entre les tâches ou les ressources et de planifier les futures attributions possibles. Cependant, les agents du groupe ne sont pas en concurrence et leurs objectifs communs, lors de l'allocation, consistent à optimiser l'efficacité du groupe. Il est alors possible de remettre en cause successivement l'allocation de façon à converger vers une solution localement optimale.

Dans le cadre de cette thèse, on s'intéresse à construire une allocation efficace tout en parallélisant au mieux les ressources de calcul. L'objectif sous-jacent consiste à améliorer la vitesse de convergence sur la solution finale. Cette vitesse est directement liée au nombre d'allocations candidates testées avant d'atteindre une allocation stable. C'est-à-dire que l'allocation est en équilibre avec les valeurs communiquées, que plus aucun échange de tâche/ressource ne permet d'amélioration.

Les approches courantes basent leur coordination sur des ventes aux enchères séquentielles sans poser le problème de la vitesse de convergence du fait d'un faible nombre de tâches à attribuer [Dias 06]. Pourtant, de telles ventes n'impliquent l'échange que d'une unique tâche ou d'une unique ressource entre deux agents à chaque itération. Dans un cadre de ventes simultanées, plusieurs ventes peuvent être réalisées en même temps. Cependant, cette approche nécessite des règles de fonctionnement particulières pour enchérir et attribuer les objets corrélés.

Un protocole basé sur une succession de ventes aux enchères simultanées est proposé pour allouer un ensemble d'objectifs à réaliser individuellement entre les agents du groupe. Le protocole est initialisé avec une allocation qui est successivement améliorée en effectuant plusieurs échanges d'objectifs simultanés entre les agents. L'objectif consiste à permettre à des agents de se répartir plusieurs dizaines de tâches/ressources en ne testant qu'un nombre restreint d'allocations. En effet, chaque allocation testée engendre son lot de calculs pour actualiser les valeurs d'enchères. Ces calculs nécessitent un temps non négligeable, notamment s'il y a un nombre conséquent de tâches.

5.2.1 Évaluation des enchères

L'évaluation individuelle d'un objectif dépend du coût des actions mises en œuvre par la politique optimale pour la réalisation de cet objectif. Les processus décisionnels de Markov (MDPs) permettent alors de modéliser un problème d'optimisation des gains espérés pour un agent agissant sous incertitude. L'évaluation d'un objectif peut être donnée par l'équation de Bellman attachée à un MDP orienté par des objectifs (Goal-Oriented MDP, GO-MDP). Ainsi, l'évaluation individuelle tient compte du gain lié à la réalisation de chaque objectif moins les coûts en ressources statistiquement requises.

Cette évaluation individuelle peut être comparée à une évaluation sociale de la politique de façon à privilégier un comportement collectif efficace. Dans le cadre de nos travaux, on s'intéresse à minimiser la charge de travail maximale pour optimiser la vitesse de réalisation de la mission. Dans cette approche, l'évaluation sociale doit pouvoir être effectuée par l'agent lui-même, avec ses données propres, de façon à ce que les valeurs d'un agent ne soient pas dépendantes des valeurs des autres agents dans une boucle nécessitant plusieurs communications et actualisations.

Intérêts individuels

Sur la base d'un GO-MDP $\langle S,\ A,\ t,\ r\rangle$ construit sur l'ensemble de tous les objectifs G, un agent Ag est en mesure d'évaluer ses intérêts individuels espérés pour la réalisation de tous les

objectifs de G. Cette valeur est donnée par sa politique optimale π^* et pour son état initial s_{Ag}. L'état initial s_{Ag} d'un agent Ag est construit relativement à l'état dynamique initial sd_{Ag} et un ensemble des objectifs accomplis initialement vide :

$$V_{Ag}^{\pi^*}(s_{Ag}) = r(s_{Ag},\ \pi^*(s_{Ag})) + \sum_{s' \in S} t(s_{Ag},\ \pi^*(s_{Ag}),\ s')V(s') \quad \text{avec } s_{Ag} = (sd_{Ag},\ \emptyset)$$

Cette valeur correspond à la somme des récompenses probablement perçues lorsque le robot suit sa politique individuelle optimale. Dans le GO-MDP, ces récompenses sont définies comme la somme des gains liés à la réalisation des objectifs et des récompenses ou des coûts des actions mises en œuvre. Pour l'agent Ag, évaluer une attribution consiste alors à ne considérer que les objectifs qui lui sont attribués G_{Ag}. La valeur d'une attribution $VA_{Ag}(G_{Ag})$ peut alors être simplement calculée en considérant, dans l'état initial s_{Ag}, que tous les objectifs qui ne lui sont pas attribués $(G - G_{Ag})$ sont réalisés.

$$VA_{Ag}(\ G_{Ag}\ ) = V_{Ag}^{\pi^*}(s_{Ag}) \quad \text{avec } s_{Ag} = (sd_{Ag},\ G - G_{Ag})$$

Ainsi, l'intérêt individuel $interet_{Ag}$ calculé pour un objectif g résulte de la différence de valeur entre les attributions incluant ou non l'objectif ($G_{Ag} + g$ ou $G_{Ag} - g$). Dans le cadre où l'objectif g appartient déjà à l'attribution de l'agent ($g \in G_{Ag}$), l'intérêt individuel exprime la perte liée à la suppression de cet objectif de son attribution.

Optimisation de la répartition des tâches

L'intérêt individuel permet de mettre en place des ventes aux enchères optimisant la somme des coûts liés à la réalisation des objectifs. Les travaux dans lesquels s'inscrivent cette thèse sont motivés par une optimisation du temps nécessaire à la réalisation de la mission. Dans la mesure où la fonction de récompense liée aux actions exprime une durée de réalisation, le critère à optimiser consiste à minimiser les coûts de l'agent ayant la charge maximale (la somme probable maximale des coûts). La somme probable des coûts, avec des valeurs espérées issues d'un GO-MDP, se calcul en retranchant les gains liés à la réalisation de ses objectifs :

$$Alloc^* = \underset{Alloc=\{G_0,\ldots,G_n\}}{argmax} \left(\min_{Ag} \left(VA_{Ag}(G_{Ag}) - \sum_{g \in G_{Ag}} gain(g) \right) \right)$$

Dans l'approche proposée, on s'intéresse au calcul d'enchères altruistes permettant de ne pas complexifier le protocole de vente : sans modifier la règle d'attribution (au plus offrant) et sans partager plus d'informations que les valeurs d'enchères. L'idée repose sur une évaluation calculée comme la somme de l'intérêt individuel et d'un intérêt collectif visant à homogénéiser les charges de travail. Cet intérêt collectif doit alors pouvoir être calculé par chaque agent, à partir de ses connaissances propres.

Enchères altruistes

Le principe heuristique proposé repose sur une comparaison de charges individuels à une valeur de charge moyenne pré-établie. Le gain espéré individuel lié à la réalisation d'un objectif est : soit minoré en cas de surcharge de son attribution ; soit majoré si l'agent possède une faible charge de travail. Ainsi, l'agent va chercher à optimiser son attribution de façon à avoir une charge de travail raisonnable.

La valeur altruiste VAA_{Ag} d'une attribution G_{Ag} correspond à la valeur individuelle $VA_{Ag}(G_{Ag})$ moins le coût social $CS_{Ag}(G_{Ag})$ de surcharger ou souscharger son attribution. Le coût social est évalué à partir d'une constante *oc* représentant le coût d'opportunité de réaliser des objectifs en plus ou en moins, par rapport au nombre moyen d'objectifs par agent (avec des objectifs d'importances équivalentes). Ainsi, un agent avec 1 objectif en plus ou en moins par rapport au nombre moyen d'objectifs aura une pénalité de *oc*. Un agent avec 2 objectifs en plus ou en moins aura une pénalité de *oc* pour le premier et de $2 * oc$ pour le second et ainsi de-suite.

$$VAA_{Ag}(G_{Ag}) = VA_{Ag}(G_{Ag}) - CS_{Ag}(G')$$

$$\text{avec} \quad CS_{Ag}(G') = oc \sum_{j=0}^{\left|\frac{|G|}{n} - |G'|\right|} j \quad = \frac{oc}{2} \left|\frac{|G|}{n} - |G'|\right| \left(\left|\frac{|G|}{n} - |G'|\right| + 1\right)$$

$$\begin{aligned} interetAltruiste_{Ag}(g) &= VAA_{Ag}(\ G_{Ag} + g\ ) - VAA_{Ag}(\ G_{Ag} - g) \\ &= \begin{cases} interet_{Ag}(g) + oc * \left|\frac{|G|}{n} - |G_{Ag} + g|\right| & \text{si } g \in G_{Ag} \\ interet_{Ag}(g) - oc * \left|\frac{|G|}{n} - |G_{Ag} + g|\right| & \text{sinon} \end{cases} \end{aligned}$$

Ainsi, chaque agent peut évaluer ses enchères en ne connaissant que le nombre d'agents du groupe n et le coût d'opportunité *oc* lié à un déséquilibrage de l'attribution. Ainsi, plus l'agent possède d'objectifs plus il est difficile pour lui de s'en voir attribuer de nouveaux. Inversement, moins il possède d'objectifs, plus il est favorisé. Cette formulation permet quand même de surcharger une attribution dans la mesure où les gains personnels sont supérieurs à la somme des coûts d'opportunités liées au déséquilibre. En effet, équilibrer la charge de travail ne correspond pas directement à équilibrer l'allocation d'objectifs mais à équilibrer les ressources mises en œuvre.

Le coût d'opportunité peut être défini en fonction de l'utilisation de façon à privilégier plus ou moins l'équité dans l'attribution. Il peut être évalué le plus proche possible d'un intérêt moyen pour tous les agents. Le coût d'opportunité peut être calculé proportionnellement à la différence maximal *cm* entre les intérêt individuels pour réaliser une des tâches quelle que soit l'attribution des agents :

$$oc = noc * cm \quad \text{avec } noc \text{ le coût d'opportunité normalisé.}$$

Une valeur $noc = 0$ implique une optimisation de la somme des intérêts individuels alors qu'une valeur $noc = 1$ empêchera tout déséquilibre dans l'allocation.

5.2.2 Protocole par succession de ventes simultanées

Sur la base des GO-MDP individuels et d'un coût d'opportunité déterminé, chaque agent est capable de calculer ses enchères pour chaque objectif vis à vis d'une allocation donnée. Le protocole par succession de ventes aux enchères simultanées SSAC (Succession of Simultaneous Actions for Coordination) permet de définir un cadre pour que les agents échangent leurs valeurs et actualisent itérativement l'allocation *Alloc* jusqu'à converger vers une allocation optimale localement.

Le SSAC se divise en 5 étapes (Figure 5.3) : (1) ouverture, (2) identification des objets, (3) évaluation individuelle, (4) actualisation de l'allocation et (5) fermeture. Ces étapes sont réalisées de façon synchronisée entre les agents. Les étapes (3) évaluation individuelle et (4) actualisation de l'allocation peuvent être répétées successivement autant de fois que nécessaire pour converger vers une allocation qui fasse consensus entre les agents.

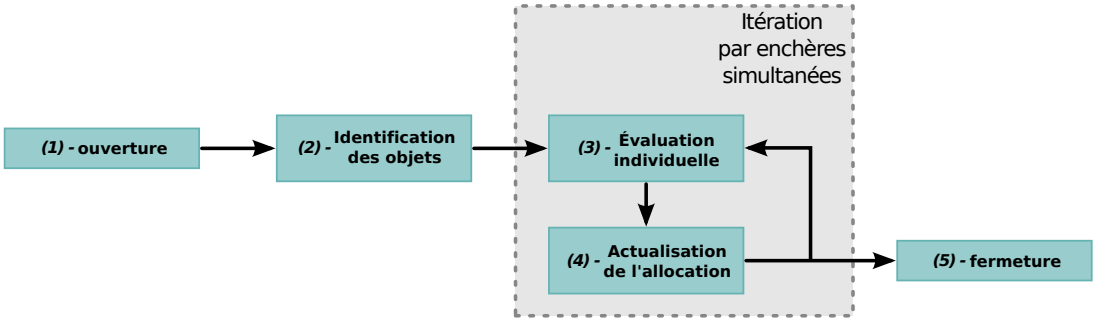


FIGURE 5.3 – Enchaînement des étapes du protocole par succession de ventes aux enchères simultanées pour la coordination (SSAC).

Initialisation du protocole

Un SSAC est ouvert (1) à la demande d'un agent qui devient le manager de la vente. Cette étape consiste à faire l'inventaire des agents présents et à les identifier de façon unique. Les agents présents regroupent l'ensemble des agents capables de communiquer efficacement de façon directe ou indirecte avec le manager. Dans le cadre d'une communication contrainte (e.g. communication parasitée nécessitant plusieurs envois de chaque message et donc un certain temps), cette étape peut servir à identifier le réseau permettant les meilleures communications entre agents et à mettre en place une transmission efficace des messages entre le manager et les autres agents.

Le rôle du manager n'est pas obligatoire dans la mesure où les règles de fonctionnement et d'allocation sont déterministes et connues de tous. Avec une communication efficace entre tous les agents, chacun peut actualiser l'allocation commune sur la base des enchères communiquées entre tous. Cependant, l'utilisation du rôle de manager permet d'identifier ce qui est commun de ce qui est distribué indifféremment sur chacun des agents.

L'étape (2) d'identification des objets permet à chaque agent de mettre en vente les objets en sa possession et de prendre connaissance de ceux des autres. Les objets doivent être tous identifiés de façon unique et sans ambiguïté entre les agents. Dans notre étude, les objets en vente correspondent aux objectifs qui ne sont pas encore réalisés. Le manager définit et communique une allocation *Alloc* initiale des objets entre les agents. Celle-ci peut correspondre simplement à l'attribution des objets avant l'ouverture du protocole SSAC ou être tout simplement aléatoire ou vide (aucun objet attribué à aucun agent). Il est aussi possible de définir une hiérarchie entre les agents pour attribuer un objet à l'agent de rang supérieur dans les cas indécidables où plusieurs enchères égales et maximales sont proposées pour un objet.

Itération dans SSAC

Une itération correspond à la réalisation successive des étapes (3) évaluation et (4) actualisation de l'allocation. Ces étapes sont effectuées par l'ensemble des agents de façon synchronisée. En effet, il est important qu'il n'y ait pas d'ambiguïté sur l'allocation *Alloc* partagée entre les agents. Aussi, l'actualisation ne peut être effectuée que lorsque toutes les évaluations sont terminées et communiquées.

À l'étape (3) évaluation, chaque agent *Ag* évalue ses enchères pour chacun des objets, afin qu'il puisse être supprimé ou ajouté à son attribution courante. Dans le cadre d'enchères corrélées, les valeurs sont dépendantes de l'allocation courante *Alloc* des objets entre les agents.

Pour l'allocation d'objectifs, les enchères peuvent être calculées à partir des politiques optimales individuelles via une modélisation par des GO-MDPs.

Il est alors possible de construire une fois pour toute un "gros" GO-MDP sur l'ensemble de tous les objectifs G ou plusieurs plus petits GO-MDPs en considérant uniquement les attributions atteignables à l'itération suivante. C'est-à-dire les attributions incluant l'attribution courante G_{Ag} de l'agent Ag plus un objectif parmi les objectifs qui ne lui sont pas attribués (soit $|G-G_{Ag}|$ GO-MDP avec $|G_{Ag}|+1$ objectifs). Chaque agent calculera donc, à chaque étape (3), les $|G-G_{Ag}|$ politiques lui permettant d'évaluer ses enchères.

Les valeurs d'enchères sont communiquées au manager et l'étape (3) évaluation, sera terminée lorsque le manager à la connaissance de toutes les valeurs (une valeur par objet et par agent). Nous rappelons qu'il est toujours considéré une communication parfaite est sans perte. À ce moment là, le manager actualise l'allocation *Alloc* (étape (4)). L'allocation est actualisée en effectuant plusieurs commutations successives. Une commutation consiste à changer l'attribution d'un objet entre l'ancien et le nouveau propriétaire proposant une enchère supérieure et actuellement maximale. Les commutations sont réalisées dans l'ordre décroissant d'importance : en sélectionnant en priorité les objets avec les différences d'enchères maximales. Enfin, l'ensemble des commutations respecte la contrainte forte d'une unique modification d'attribution par agent. Une fois toutes les commutations détectées, le manager envoie à chaque agent, la nouvelle allocation *Alloc* pour que chacun ré-évalue ses enchères (étapes (3)).

Dans une exécution sans manager, les enchères de l'étape (3) sont envoyées en broadcast à tous les agents. Chaque agent attend de recevoir toutes les valeurs avant d'actualiser l'allocation selon les règles communes, (étapes (4)). Une fois l'allocation actualisée, l'agent peut ré-évaluer ses enchères (étape (3)) et ainsi de suite. La synchronisation des agents se fait avec l'envoi et la réception de la dernière valeur d'enchère, chaque itération est définie par l'envoi de $n|G|$ messages de la forme (agent, objet, valeur).

Conditions d'arrêt

Si aucune actualisation sur l'allocation n'est réalisée à l'étape (4), le protocole SSAC peut être fermé (étape finale (5)). Un message de dé-connexion est envoyé à chacun des agents par le manager pour leur permettre de partir réaliser leur part de la mission, c'est-à-dire atteindre l'ensemble des objectifs individuellement attribués. Dans un cadre sans manager simple, chaque agent construit l'allocation totale, sur tous les agents, chacun est donc en mesure d'évaluer la condition d'arrêt, même s'il n'est pas concerné par les actualisations.

Le protocole SSAC verrouille les agents pendant son processus. En effet, ce processus est conçu pour des missions avec des phases ponctuelles de communication efficace. Les actions des agents doivent alors garantir de maintenir la communication jusqu'à la fermeture du protocole SSAC. Cependant, lors d'une rupture de communication, les agents peuvent conserver la dernière allocation communiquée et reprendre la procédure avec les agents toujours en capacité de communiquer.

Ce protocole permet d'effectuer plusieurs échanges d'objets simultanés entre les agents et de distribuer le calcul de la politique jointe dans un cadre de coordination basée sur l'allocation de tâches/ressources. La distribution du processus est limitée par le besoin de synchroniser l'allocation partagée *Alloc* à l'étape (3). De plus, ce protocole (au même titre que des ventes aux enchères traditionnelles) permet à plusieurs agents hétérogènes de se coordonner. En effet, les GO-MDPs individuels peuvent être définis sur la base d'objectifs communs identifiés mais avec des dynamiques propres à chaque agent.

5.2.3 Caractéristiques du protocole SSAC

Le protocole SSAC consiste à optimiser successivement l'allocation courante jusqu'à ce que plus aucune optimisation ne soit possible. Dans un premier temps, nous nous intéressons à la convergence du protocole sur une allocation optimale localement. Cette convergence permet de garantir notamment que le protocole ne boucle pas indéfiniment sur une succession de mêmes allocations testées. Dans un second temps, les limites théoriques sont étudiées pour chercher à caractériser le déroulement du protocole SSAC, de l'utilisation des GO-MDPs et de l'introduction du coût d'opportunité sur l'allocation construite.

Convergence

La convergence est garantie par l'étape (4) actualisation de l'allocation. Les règles d'actualisation et notamment la contrainte d'une unique modification par agent, permettent, d'une itération sur l'autre, d'augmenter la somme des intérêts individuels.

Dans les faits, chaque commutation de propriétaire pour un objet donné est basée sur un gain sur les valeurs d'enchères. Ce gain correspond alors à une amélioration de la valeur altruiste de la nouvelle allocation par rapport à l'ancienne. En effet, une commutation signifie que l'intérêt altruiste pour le nouveau propriétaire est supérieur à la perte pour l'ancien propriétaire et la somme des intérêts altruistes individuels du groupe est donc augmentée.

Cependant, la convergence est garantie uniquement si une seule modification est effectuée par agent et par itération dans SSAC. En effet, le calcul d'une enchère est défini par rapport à l'allocation de l'itération précédente et en considérant qu'une unique modification. Ce mécanisme ne garantit donc pas une amélioration dans le cas où plusieurs commutations sont réalisées en même temps. Dans les faits, chaque modification de l'attribution d'un objet pour un agent remet en cause les valeurs d'enchères pour tous les autres objets.

En revanche, l'actualisation de l'allocation étape (4) peut inclure plusieurs commutations d'objets entre des agents différents. Il est alors possible de faire autant de modifications d'attribution qu'il y a de paires d'agents dans le groupe d'agents. Ainsi, en théorie, augmenter le nombre d'agents n'a qu'un faible impact sur le nombre d'itérations dans SSAC. Il est aussi attendu qu'une augmentation du nombre d'agents avec un nombre de tâches fixe accélère la coordination par SSAC avec des enchères altruistes visant une équi-répartition de la charge de travail. Ses hypothèses sont formulées avec une réserve sur la possibilité de communiquer instantanément les $n|G|$ messages entre les n agents.

Qualité de l'allocation finale

La convergence permet de garantir l'arrêt du protocole sur une solution optimale localement. En effet, l'amélioration de l'allocation d'une itération sur l'autre empêche le protocole SSAC de visiter deux fois une même solution et, d'autre part, il y a un nombre fini d'allocations possibles. De plus, l'allocation finale est optimale par rapport à l'ensemble des allocations visitées et en considérant une seule commutation pour l'attribution d'un des objets. En effet, les règles d'actualisation favorisent la commutation optimale et tant qu'il existe une meilleure allocation à une commutation près, les étapes d'évaluation (3) et d'actualisation (4) sont relancées.

Cependant, la commutation d'un unique objet par agent couplée à des enchères favorisant un équilibre dans la distribution des charges est un frein à la convergence vers une solution efficace. En effet, avec un coût d'opportunité fort, le protocole SSAC s'arrête sur la première allocation équilibrée proposée. Les enchères altruistes calculées par les agents ne leur permettent pas de libérer un objet pour en prendre un autre à l'itération suivante. Une première solution peut

consister à permettre des actualisations incluant 2 modifications par agent. Une telle solution repose alors sur l'évaluation combinatoire d'enchères d'ordre 2. Chaque agent va enchérir relativement à son intérêt d'ajouter ou de retirer 1 ou 2 objets à son attribution ($|G|$ objets et $2^{|G|}$ couples d'objets). Une telle solution implique le calcul et la communication de $(|G| + 2^{|G|})$ enchères à chaque itération. De plus, le processus d'actualisation de l'allocation courante est aussi alourdi avec une donnée initiale de $n(|G| + 2^{|G|})$ valeurs d'enchères.

Une seconde solution consiste à faire varier l'importance de l'intérêt collectif par rapport à l'intérêt individuel d'une itération sur l'autre. L'idée repose sur une introduction progressive de la contrainte de partage équilibré des charges. Avec un coût d'opportunité oc initialisé à 0, les agents vont privilégier leurs intérêts individuels. En augmentant progressivement la valeur de oc jusqu'à sa valeur idéale oc^*, les actualisations successives permettront de passer progressivement, d'une maximisation de la somme des intérêts individuels vers une maximisation de la somme des intérêts altruistes. Cependant, la convergence vers la solution finale optimale localement sera garantie qu'à partir du moment où la valeur de oc est à son niveau idéal oc^*.

SSAC asynchrone

La synchronisation des étapes du protocole SSAC peut constituer une seconde limitation sur la vitesse du calcul de l'allocation. L'actualisation de l'allocation est conditionnée par l'évaluation des enchères par tous les agents. Cette condition impose aux agents d'attendre, à chaque itération, l'agent ayant le temps de calcul plus important.

Des différences entre les agents peuvent exister sur le plan des ressources de calcul ou de la charge de calcul résultante des modifications dans les attributions. Par exemple, dans le cadre d'un GO-MDP actualisé à chaque étape évaluation (3), l'ajout de nouveaux objectifs demande une actualisation de la politique pour les nouvelles attributions atteignables. Les charges de calculs pour l'actualisation des politiques dépendent alors de la taille des attributions individuelles.

Aussi, notre approche consiste à étendre le protocole de successions de ventes aux enchères simultanées (SSAC) sans manager en désynchronisant les étapes (3) et (4). Supprimer la contrainte de synchronisation implique de possibles malentendus sur l'allocation des objectifs entre les agents. Le protocole SSAC asynchrone est basé sur un mécanisme de verrouillage des objets de façon à limiter les incohérences entre les agents.

A chaque instant et pour chaque objet $g \in G$, l'agent Ag avec l'enchère maximale verrouille l'objet en cachant sa valeur réelle et en communiquant une enchère maximale inatteignable. Chacun des agents communique donc une enchère maximale pour l'ensemble des objets de son attribution et son intérêt réel pour les autres objets. Pour l'agent Ag, si un second agent propose une enchère pour g supérieure à sa valeur réelle, l'agent Ag libère son objet et communique à nouveau son intérêt réel. Le second agent pourra alors, à son tour, verrouiller l'objet g.

Ce mécanisme garantit qu'un objet ne peut être attribué qu'à un et un seul propriétaire (une hiérarchie entre les agents permet de désambiguïser les situations avec des enchères maximales identiques). La fin du processus d'allocation des objets est détectée lorsqu'ils sont tous attribués et que les agents sont stables (les enchères communiquées sont cohérentes avec leurs attributions).

5.3 Évaluation du protocole SSAC

Le protocole de coordination par succession de ventes aux enchères simultanées (SSAC) permet à un groupe d'agents coopératifs d'atteindre un consensus sur une allocation des tâches. A chaque itération, chaque agent évalue unitairement toutes les tâches par rapport à son attribution courante. L'évaluation distribuée des tâches peut être réalisée à l'aide de processus décisionnels

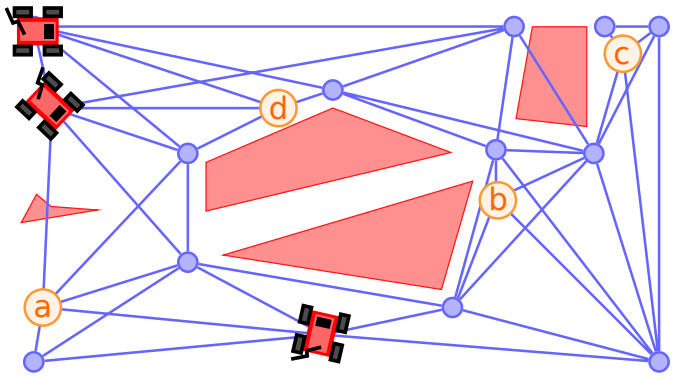


FIGURE 5.4 – Problématique : illustration avec 3 robots partageant une même *Road-Map* et 4 positions objectifs $G = \{a,\ b,\ c,\ d\}$.

de Markov orientés par des objectifs (GO-MDPs). Ainsi, l'allocation définissant l'attribution de chaque agent, peut être actualisée itérativement de façon à optimiser les politiques liées aux tâches à réaliser individuellement.

L'objectif de ce protocole est de converger rapidement et avec efficacité vers l'allocation finale. Une coordination par ventes aux enchères s'oppose à une solution par co-itération sur les politiques individuelles basée sur une communication totale de toutes les actions réalisées par tous les agents. Le protocole SSAC s'inspire de cette approche en cherchant à alléger les ressources nécessaires au calcul d'une solution par rapport à une coordination par une séquence ventes unitiares (ventes séquentielles). L'efficacité d'une solution peut alors être mesurée en comparant les valeurs des allocations, c'est-à-dire la somme des gains altruistes espérés individuels. Pour rappel, ces gains sont donnés par l'équation de Bellman appliquée aux politiques des GO-MDPs individuels et aux coûts sociaux associés (Section 5.2.1).

5.3.1 Problème considéré

Les expérimentations réalisées pour valider l'approche se rapportent, dans le cadre de cette thèse, à la problématique d'une flotte de robots missionnée pour visiter un ensemble de positions objectifs (Figure 5.4). Ces positions sont définies dans une carte topologique de façon à s'intégrer naturellement dans l'architecture PRDC (Perception, Représentation, Décision et Contrôle) appliquée à chacun des robots.

L'exécution de la mission multi-robots est divisée en 2 types de phases : des phases de coordination et des phases de réalisation des objectifs individuels. Le protocole SSAC est utilisé dans les phases de coordination, entre tous les robots qui sont en capacité de communiquer. Une phase de coordination particulière sert à initialiser la mission en partageant tous les objectifs entre tous les robots.

D'autres phases de coordination peuvent être requises durant l'exécution si la connaissance d'un des robots est modifiée de façon à remettre en question son attribution (e.g. un chemin bloqué) et qu'il est en capacité de communiquer avec d'autres robots. Une remise en question de l'attribution peut être détectée, par exemple, si l'intérêt d'un des objectifs (via l'équation de Bellman) est modifié au-delà d'un certain seuil, par rapport à la dernière phase de coordination. Dans cette situation, les objectifs non encore atteints des agents communiquants peuvent être ré-alloués.

On s'intéresse ici, principalement à la résolution de phases de coordination sans expérimenter la réalisation de la mission de façon globale. Dans cette problématique, l'ensemble des objets à attribuer correspond aux positions objectifs à visiter. Le protocole doit alors, permettre aux robots de se distribuer plusieurs objectifs en cours de mission.

Représentation

La partie "représentation" de l'architecture PRDC (Section 3.2.3.0) des robots est principalement défini par une *Road-Map* $\langle W,\ P\rangle$ (Figure 5.4), soit un graphe où les nœuds représentent des positions clefs reconnaissables (Way-points) et les arcs représentent les chemins (Paths) permettant de se déplacer d'une position à une autre. Chaque arc $p \in P$ est défini par une position initiale w_p et intègre des informations complémentaires pour évaluer les probabilités de déviation $d_p(w)$ et le coût c_p de déplacement.

$$P = \big\{(w_p, d_p, c_p),\quad w_p \in W,\quad c_p \in \mathbb{R}^-,\quad d_p : W \to [0,1]\ \big\}\quad \text{avec}\quad \sum_{w\in W} d_p(w) = 1$$

La *Road-Map* modélise les possibilités de déplacements d'un robot dans son environnement de façon équivalente au tuple ⟩états, actions, transitions, récompenses⟨ utilisé dans un MDP. Les états correspondent à la détection d'une position clef w_p et les actions sont définies par rapport à l'ensemble des chemins existants. La fonction de transition est donnée par la déviation d_p qui retourne la probabilité d'atteindre une position en suivant le chemin p depuis w_p. Enfin, la fonction de récompenses est donnée par le coût associé à chaque chemin.

Politique individuelle

Un MDP individuel orienté par des objectifs (GO-MDP) $\langle S,\ A,\ t, r\rangle$ est défini par et pour chaque robot Ag à partir de sa *Road-Map* courante et des objectifs qui lui sont attribués G_{Ag}. Les états sont construits relativement aux positions W et à une mémorisation des objectifs individuels atteints. Les actions correspondent aux chemins dans la carte. La dynamique du GO-MDP est donnée par les fonctions de déviation de chaque chemin d_p et en augmentant, de façon cohérente, l'ensemble des objectifs atteints. Enfin, la fonction de récompenses est calculée sur la base des coûts de déplacements plus un gain important perçu chaque fois qu'un nouvel objectif est atteint.

Les GO-MDPs individuels permettent à chaque robot de calculer sa politique de déplacements optimale (vis à vis de la carte topologique) lui permettant d'atteindre ces objectifs attribués. La taille $|S|$ du GO-MDP et la difficulté de le résoudre explosent avec le nombre d'objectifs considéré $|G|$ ($|S| > 2^{|G|}$). De ce fait, plutôt que de travailler sur un unique GO-MDP construit sur l'ensemble de tous les objectifs G, plusieurs petit GO-MDPs sont construits par un agent Ag, avec un GO-MDP pour chaque sous-ensemble d'objectifs testé G_{Ag}. En effet, avec des allocations homogènes, la taille des attributions reste restreinte. De plus, seules quelques modifications sont nécessaires pour passer d'un GO-MDP (sur G_Ag) à un autre avec un ensemble d'objectifs G'_Ag équivalent à un objectif prés ($|G_Ag \cap G'_Ag| = |G_Ag \cup G'_Ag| - 1$).

Pour actualiser le GO-MDP individuel, dans le cadre de la suppression d'un objectif, il est alors possible de considérer le sous-GO-MDP construit sur les états ou l'objectif supprimé est considéré atteint. Aucune actualisation de la politique n'est alors requise. Dans le cadre de l'ajout d'un objectif, l'ensemble des états, des transitions et des récompenses de l'ancien GO-MDP correspondent aux états, aux transitions et aux récompenses du nouveau GO-MDP où l'objectif ajouté est considéré atteint. Ainsi, il est possible de se restreindre à un calcul de la politique sur la moitié des états du nouveau GO-MDP. D'autre part, le calcul de cette moitié de politique peut être initialisé avec l'autre moitié de la politique.

Flotte hétérogène

La solution SSAC avec évaluation par GO-MDP permet, même si ce n'est pas le cas dans nos expérimentations, de considérer une flotte de robots hétérogènes. En effet, chaque robot peut posséder une *Road-Map* différente avec des chemins, des déviations et des coûts particuliers. Ainsi, il est possible d'imaginer travailler avec une flotte de robots composée de robots terrestres à roues ou à pattes, de robots amphibies ou encore de drônes. En effet, certains robots pourront être mieux adaptés ou avoir un handicape vis à vis de la réalisation de certains objectifs. Notons que l'hétérogénéité des robots peut aussi venir d'une connaissance ayant évoluée différemment pendant leurs déplacements passés.

Les seules restrictions concernent la capacité des robots à identifier les mêmes objectifs et à définir une fonction de récompense cohérente. En effet, le protocole SSAC ne permet pas aux agents de se coordonner si les positions objectifs référencées par chaque agent ne correspondent pas aux mêmes positions dans l'environnement. De plus, les gains perçus par la réalisation d'un objectif doivent être les mêmes d'un robot à un autre et les valeurs des coûts liées aux déplacements doivent être cohérentes entre les agents. Elles peuvent, par exemple, être définies sur une même métrique, comme une consommation de ressources temporelles en secondes.

5.3.2 Résultats par comparaison avec une allocation optimale

Une première série d'expériences cherche à valider l'efficacité de l'utilisation du protocole SSAC synchronisé par rapport à une allocation optimale. L'efficacité est calculée en comparant la somme des gains espérés de chaque robot obtenue par l'allocation SSAC, l'allocation optimale, une allocation aléatoire et la pire allocation.

Les expériences comprennent 3 robots évoluant dans une des 2 cartes considérées (Figure 5.5) et entre 2 et 13 objectifs. Le calcul de l'allocation optimale par une énumération exhaustive limite considérablement la taille des problèmes pouvant être traités. En effet, pour 13 objectifs, il existe 3^{13} allocations possibles et potentiellement entre $50 * 2^{13}$ et $70 * 2^{13}$ (en fonction de $|W|$) états dans le GO-MDP de chaque robot.

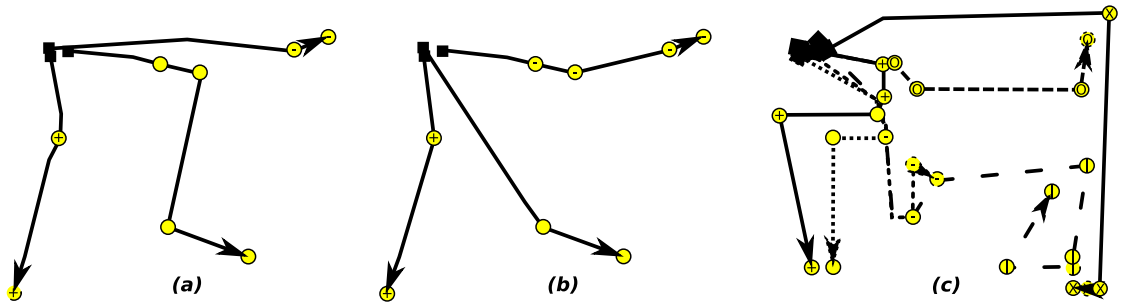


FIGURE 5.5 – Les 2 *Road-Map*s utilisées pour les expériences : (a)(b) avec quelques obstacles et (c) l'environnement de type labyrinthe. (a) et (c) présentent des allocations construites avec SSAC et (b) une allocation optimale.

Les robots possèdent les mêmes caractéristiques de déplacements. Le coût associé à chaque chemin est défini par la somme des distances entre la position initiale et les positions atteignables proportionnellement aux probabilités de les atteindre. Le coût maximal théorique, correspondant à l'action de traverser l'environnement par sa diagonale, est de 51.2. La récompense d'atteindre un objectif est définie à 1000 de façon à garantir son attrait, quelle que soit la position du robot

dans l'environnement. Enfin, le coût d'opportunité normalisé est défini par 0 ou 0.1 (soit un coût d'opportunité de 5.12) en fonction des expériences.

L'expérience se limite à un coût normalisé à 0.1 pour encourager une allocation balancée des tâches sans forcer un équilibre parfait. En effet, si l'ensemble des tâches à accomplir est aligné sur une même route il peut être plus intéressant qu'un seul agent les réalise toutes. Ceci dit, les expériences réalisées ici, cherchent à évaluer une allocation construite pour un critère donnée et non pas le critère lui-même. L'utilisation de 2 coûts d'opportunités différents permet d'avoir des résultats sur 2 critères.

Résultats Qualitatifs

Chaque expérience est classée en fonction de la carte utilisée, du nombre d'objectifs et de la valeur du coût d'opportunité. Une série d'expériences est lancée avec 200 générations aléatoires sur la position des objectifs pour chaque classe d'expérience (48 classes). Chaque expérience consiste à relever la somme des gains altruistes espérés pour l'allocation SSAC, une allocation aléatoire, l'allocation optimale et la pire allocation. Un score est ensuite calculé comme le pourcentage entre la pire et la meilleure allocation.

Par exemple, une des expériences dans la classe (labyrinthe, $|G| = 4$ et $oc = 0$), qui a permis d'établir les moyennes (Tableau 5.1), a retourné les valeurs : 3891.11 (meilleure allocation) ; 3836.41 (pire allocation) ; 3890.84 (allocation SSAC) et 3845.74 (allocation aléatoire). L'allocation SSAC n'a pas été optimale, elle a produit un score de 99.5% (et $17,0\%$ pour l'allocation aléatoire). Le tableau 5.1 présente les moyennes des scores obtenus pour chaque classe d'expériences ainsi que le pourcentage d'expériences où SSAC a convergé sur une allocation optimale.

Les scores obtenus permettent de valider que l'allocation construite par SSAC est très proche d'un allocation optimale dans des problèmes de tailles restreintes. De plus, nous avons observé seulement 2 expériences sur les 9600 avec un score inférieur à 70%. D'un autre côté, la probabilité de trouver une allocation optimale diminue lorsque la probabilité de converger vers des optimums locaux augmente (cf. différences entre Figure 5.5(a) et (b)). C'est notamment vrai dans l'environnement de type labyrinthe où l'amélioration de certaines allocations nécessite plusieurs commutations simultanées des objectifs entre les robots.

Dans, plusieurs expériences avec un coût d'opportunité normalisé de 0.1, le protocole SSAC synchronisé montre des difficultés à échanger un objectif pour pouvoir en prendre un autre de valeur supérieure, à l'itération suivante, lorsque le robot a une charge dans la moyenne. Dans cette situation, la valeur sociale liée à sa charge est optimale, et les évaluations effectuées par le robot ne lui permettent pas de détecter qu'un autre des objectifs serait plus adéquat avec son attribution courante dans de commutations futures possibles. Cela se traduit par une baisse des scores enregistrés dans les classes d'expériences où $noc = 0.1$.

5.3.3 Problèmes de tailles supérieures et situations réelles

L'objectif de l'approche consiste à traiter, avec des robots réels, des problèmes caractérisés par une taille ne permettant pas forcément une résolution optimale en-ligne. Une seconde série d'expériences a été réalisée pour valider la capacité du protocole SSAC asynchrone à coordonner une flotte allant jusqu'à 10 robots ($n = 10$) et un nombre de tâches proportionnel à la taille de la flotte.

Ensuite, le protocole SSAC asynchrone a été testé pour coordonner 3 robots réels communiquant par Wifi. Ces expériences permettent de valider l'utilisation de SSAC dans l'architecture

TABLE 5.1 – Scores obtenus par SSAC comparativement à une solution optimale. Les scores moyens et minimaux (min) expriment la distance (en %) entre la pire allocation possible (0.0) et la meilleure possible (100.0). Enfin, il est relevé le pourcentage d'expérience où SSAC a produit une allocation de score optimal (pourcentage d'optimal).

$\|G\|$	quelques obstacles, $noc = 0$				quelques obstacles, $noc = 0.1$			
	Aléatoire	SSAC			Aléatoire	SSAC		
	Score Moyen	Score Moyen	Min	pourcentage optimals	Score Moyenne	Score Moyenne	Min	pourcentage d'optimal
2-3	42.1	99.5	85.8	89.5	49.7	99.6	80.0	92.5
4-5	40.6	99.0	77.5	80.3	47.4	99.0	47.4	69.3
6-7	37.5	98.9	83.3	71.8	54.6	99.2	82.6	47,8
8-9	36.8	98.4	86.1	63.8	64.7	99.2	86.5	45.0
10-11	34.1	98.1	72.7	57.0	71.3	99.3	87.6	44.5
12-13	32.0	98.4	83.9	55.7	76.0	99.2	91.6	37.0

$\|G\|$	labyrinthe, $noc = 0$				labyrinthe, $noc = 0.1$			
	Aléatoire	SSAC			Aléatoire	SSAC		
	Score Moyen	Score Moyen	Min	pourcentage optimals	Score Moyenne	Score Moyenne	Min	pourcentage d'optimal
2-3	40.8	99.7	89.0	97.8	48.3	99.5	78.7	86.8
4-5	44.6	99.7	77.8	85.5	50.9	99.7	88.8	63.0
6-7	39.1	99.5	81.4	73.6	54.8	99.2	75.8	42.0
8-9	37.9	99.3	85.5	67.2	54.1	99.0	75.0	34.0
10-11	36.0	99.0	85.1	53.8	58.8	98.8	82.8	32.3
12-13	36.2	98.8	83.2	44.5	64.9	98.8	86.8	23.0

PRDC et de vérifier que les résultats expérimentaux sur des robots réels correspondent aux valeurs moyennes relevées lors des simulations.

Augmentation du nombre d'agents et d'objectifs

Dans le cadre de la seconde série d'expériences (jusqu'à 10 robots, $n = 10$), le coût d'opportunité normalisé est fixé à 0.1 (soit $oc = 5.12$ avec un coût maximal théorique de 51.2) pour équilibrer l'allocation des tâches. Cela implique aussi que les tailles des GO-MDPs individuels seront limitées. Ainsi, les expériences de cette série comprennent jusqu'à $n = 10$ robots et une moyenne de 4 objectifs par robot ($|G| = 4 * n$). L'allocation optimale serait alors a chercher parmi 10^{40} possibilités et potentiellement 2^{40} états sur les GO-MDPs.

La figure 5.5(c) présente des résultats expérimentaux pour 6 robots et 24 objectifs dans l'environnement labyrinthique. Les chemins dessinés des robots représentent leurs déplacements probables basés sur les probabilités les plus fortes des fonctions de déviations.

En utilisant le protocole SSAC asynchrone, nous avons relevé, pour chaque robot Ag, le nombre de modifications qu'il a réalisées sur son attribution des tâches. Une modification sur l'attribution intervient si le robot propose une enchère maximale sur un objectif libre ou s'il déverrouille un de ses objectifs. Chaque modification entraîne une actualisation du GO-MDP personnel, des intérêts altruistes des objectifs et donc, des valeurs communiquées. Dans les faits, à certains moments de la phase de coordination, chaque robot peut itérer dans le vide pour simplement attendre que les autres robots se stabilisent ou libèrent des objectifs.

La série d'expériences se compose de 200 expériences par taille de flotte considérée ($n \in [2,\ 10]$). Chaque expérience se différencie par une génération aléatoire des positions des objectifs dans un des deux environnements tests. 2 moyennes sont calculées sur l'ensemble des 200 expériences : une moyenne du nombre de modifications pour tous les robots et une moyenne en considérant uniquement le robot ayant le plus grand nombre de modifications (Fig. 5.6). En effet, c'est le robot qui totalisera le plus de modifications qui déterminera la vitesse de convergence.

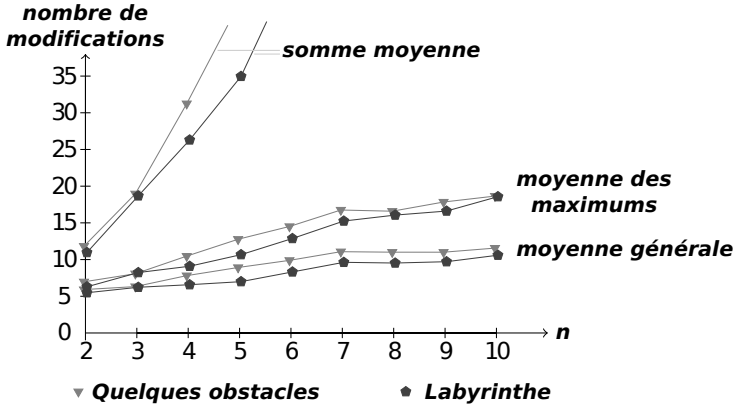


FIGURE 5.6 – Nombres moyens de modifications individuelles effectuées par les robots, donnés en augmentant la taille du problème considéré (n robots et $4 * n$ objectifs).

Il est possible de conclure que seulement quelques actualisations, par rapport au nombre total d'objectifs, sont nécessaires pour converger vers une allocation optimale d'ordre 1, c'est-à-dire optimale à une commutation près. Le nombre moyen de modifications par robot augmente de façon linéaire en pente douce, avec le nombre total d'objectifs et en considérant un nombre idéal fixe d'objectifs par agent (ici 4).

Cela signifie que, statistiquement, le processus de chaque robot utilise un nombre de commutations quasiment proportionnel au nombre attendu d'objectifs par robot ($|G|/n$ dans le cas d'allocations équilibrées). Le nombre moyen de commutations est multiplié par 2 alors que les nombres d'objectifs et d'agents sont multipliés par 5. Seulement quelques remises en cause d'attributions d'objectif sont nécessaires pour converger vers l'allocation finale.

Expériences sur des robots réels

Le protocole SSAC asynchrone a été testé pour coordonner 3 robots mobiles réels (Figure 5.7, Vidéos [9]) dans un environnement sans obstacle (Vidéo 1) et un environnement de type urbain (Vidéo 2). La carte initiale est similaire en termes de taille et de coût de déplacement aux cartes utilisées en expérimentation virtuelle ($\simeq 51$ metres sur la diagonale et une densité de points clefs équivalante). La communication entre les robots et l'ordinateur de l'opérateur est réalisée par Wifi. A partir d'une carte initiale, l'opérateur initialise la position et l'orientation des robots et 12 points objectifs. Ces informations sont envoyées indifféremment aux robots qui doivent se coordonner puis visiter les points objectifs. Le protocole SSAC est donc lancé en première phase avant tout déplacement. Le coût d'opportunité normalisé est également fixé à 0.1 pour favoriser un équilibre sans le forcer.

FIGURE 5.7 – (a) un des 3 robots Pioneer, (b) une vue prise du robot et (c) une vue aérienne de la zone expérimentale (Google).

Cette expérimentation permet essentiellement de valider l'intégration du SSAC dans l'architecture 2 axes. Le protocole SSAC asynchrone a permis aux robots de coordonner leurs politiques en s'allouant mutuellement les objectifs à visiter. Les politiques individuelles construites via les GO-MDPs supervisent ensuite les déplacements des robots entre les positions clefs successives à atteindre en environnement de type "intérieur, sans obstacle" (Figure 5.8) ou "extérieur, encombré" (Figure 5.9). Ces déplacements sont rendus possibles par la cohérence entre la carte topologique et les capacités sensori-moteur des robots (cf. Chapitre 3). Les attributions sont alors cohérentes et permettent une réalisation efficace de la mission multi-robots.

Les nombres relevés de modifications ($\simeq 7$) sur les attributions lors de la phase de coordination correspondent aux nombres relevés en expérimentations virtuelles (Figure 5.6) et l'allocation finale est trouvée après une moyenne de 16 secondes (avec une programmation non totalement optimisée). Les robots sont équipés d'un ordinateur portable standard (3 CPU à 2.50GHz et 3.5

9. R-Discover : www.greyc.fr/node/1629

FIGURE 5.8 – Expérience de coordination et navigation dans un environnement sans obstacle.

FIGURE 5.9 – Expérience de coordination et navigation dans un environnement urbain.

Go Ram). La communication Wifi est aussi mise à contribution pour monitorer sur l'ordinateur de l'opérateur, la connaissance des robots (leurs positions supposées, l'allocation des objectifs et leurs réalisations). L'ordinateur de l'opérateur permet de visualiser les échanges d'objectifs effectués lors de la phase de coordination.

5.3.4 Comparaison avec une résolution séquentielle

Les premières expériences réalisées avec SSAC asynchrone permettent de valider l'utilisation de ce protocole dans le cadre de coordinations multi-robots ponctuelles. Le protocole permet de converger rapidement vers une solution optimale localement de bonne qualité. L'objectif de cette seconde phase de validation consiste à comparer une coordination par SSAC avec une coordination par ventes aux enchères séquentielles.

Pour ces expériences, un protocole de ventes séquentielles est mis en place sur la base des 5 étapes de SSAC : ouverture, inventaire, évaluation, actualisation et fermeture. La différence consiste à n'évaluer qu'un seul objectif à chaque itération. L'actualisation de l'allocation ne s'effectue alors que relativement à cet objectif. A chaque itération, le manager met en vente un objectif et l'attribue à un robot. L'attribution n'est définitive qu'à la fin du protocole. En effet, les objets sont remis en vente tant que le manager n'a pas effectué une boucle complète sur l'ensemble des objectifs sans qu'il n'y ait eu de commutation.

Ces expérimentations ont été réalisées dans un environnement virtuel de type labyrinthe (Figure 5.5(c)) en considérant une flotte de 6 robots. Le coût d'opportunité normalisé est de nouveau fixé à 0.1 pour encourager, sans forcer, une allocation aux charges équitables. Le nombre d'objectifs est augmenté progressivement de 2 à 24 objectifs soit un idéal de 4 objectifs par robot. L'objet de ces expériences consiste alors à comparer l'efficacité et la vitesse de convergence des deux approches.

Résultats bruts

Les expériences réalisées se découpent en 4 séries en fonction de l'approche : séquentielle ou par succession de ventes simultanées (SSAC) et de l'utilisation ou non d'une allocation initiale aléatoire. 200 expérimentations par série et par nombre d'objectifs considérés $|G|$ sont générées en positionnant aléatoirement les objectifs dans le labyrinthe. Chaque série est lancée sur un même jeu d'expériences. Pour comparer les deux approches, nous avons relevé le nombre de ventes et

TABLE 5.2 – Moyenne des résultats obtenus par ventes simultanées (SSAC) et par ventes séquentielles sans allocation initiale

	Coordination par ventes séquentielles :					Coordination par SSAC :				
	nombre de		valeur de	charge		nombre de		valeur de	charge	
$\|G\|$	ventes	commut.	l'allocation	min.	max.	ventes	commut.	l'allocation	min.	max.
2	4.0	2.0	1934.3	0.0	-41.4	2.4	2.4	1934.3	0.0	-41.4
4	8.4	4.3	3880.5	0.0	-54.4	3.3	5.7	3880.6	0.0	-54.3
6	13.3	6.5	5845.3	-0.0	-63.3	3.9	8.8	5845.2	-0.0	-63.1
8	19.0	9.0	7812.6	-1.2	-67.6	5.2	12.0	7812.5	-1.2	-67.1
10	25.2	11.6	9782.3	-3.7	-70.6	6.7	16.4	9783.1	-4.3	-70.0
12	33.1	14.4	11776.7	-8.6	-71.9	8.0	20.8	11775.9	-10.0	-71.1
14	41.4	17.3	13751.4	-13.6	-76.1	9.8	26.1	13750.5	-16.2	-75.2
16	49.1	19.7	15730.1	-16.0	-77.5	11.0	30.2	15727.7	-18.4	-78.2
18	57.3	22.7	17725.6	-20.1	-78.8	12.4	34.8	17721.3	-23.6	-77.5
20	62.3	24.8	19710.5	-23.3	-79.7	13.6	38.5	19706.1	-26.4	-80.5
22	72.7	27.3	21698.3	-24.3	-80.0	15.2	44.0	21691.4	-28.0	-80.0
24	80.7	29.9	23693.6	-27.0	-83.3	16.7	48.4	23687.0	-30.9	-82.8

de commutations sur les attributions effectuées, la valeur des allocations construites (somme des intérêts altruistes Section 5.1.3) ainsi que les coûts des déplacements entre le robot de charge minimale et le robot de charge maximale.

Le tableau 5.2 regroupe les moyennes relevées pour l'exécution des 2 protocoles dans un cadre sans allocation initiale. Dans ce cadre, les protocoles considèrent initialement qu'aucun objectif n'appartient à un agent. Ainsi, dans les premières itérations, les objectifs seront simplement attribués au robot proposant un intérêt altruiste maximal. Avec $|G|$ objectifs, il faudra au minimun $2|G|$ ventes au processus séquentiel pour attribuer tous les objectifs et valider l'attribution (si celle-ci est stable). Il faudra aussi plusieurs itérations au protocole SSAC pour créer une première allocation totale. En effet, le protocole ne permet d'ajouter qu'un et un seul objectif par agent et par itération et uniquement à l'agent proposant un intérêt altruiste maximal. La construction de la première allocation est donc freinée si un même agent propose les valeurs maximales pour plusieurs objets.

Sans surprise, un rapide parcours des tableaux permet de valider, que le processus séquentiel nécessite plus de ventes que le processus SSAC. Par contre, le processus séquentiel a recours à moins de commutations. De plus, les valeurs des allocations et des charges minimales et maximales sont similaires. La comparaison entre les valeurs des charges minimales et des charges maximales permet de noter des déséquilibres entre les charges des robots. Ces déséquilibres sont dans un ordre de grandeur similaire au coût de traverser l'environnement par sa diagonale (51.2). Ces déséquilibres sont justifiés par la configuration des objectifs dans le labyrinthe. En effet, l'équité (paramétrée à 0.1) ne justifie pas que tous les robots traversent tout l'environnement jusqu'aux extrémités les plus lointaines. Un seul robot ira à l'objectif le plus lointain générant de fait un déséquilibre.

En première analyse

Une première analyse des résultats plaide plutôt en faveur d'un processus d'allocation séquentiel. En effet, on remarque que cette approche permet de converger vers une solution meilleure et en effectuant moins de commutations entre les attributions, soit moins d'actualisations individuelles pour les robots. Cependant, les gains sur les valeurs des allocations produites sont minimes.

En se basant sur les expériences effectuées comparativement à une allocation optimale et à la pire allocation (3 robots et jusqu'à 13 objectifs dans l'environnement labyrinthique), il est possible de considérer 51.2 (coût de déplacement maximal théorique) comme borne minimale pour la différence entre la pire et la meilleure allocation possible. C'est-à-dire que l'on considère que les détours engendrés par la pire allocation consistent à traverser entièrement l'environnement pour chacune des tâches. Il est possible d'observer que, dans les faits, cette borne est largement surévaluée du fait de la densité des objectifs dans l'environnement. Malgré cette différence maximale grossière de 51.2 par tâche (en défaveur de SSAC), les gains moyens pour le protocole séquentiel ne sont que de 7.3% (pour 12 à 24 objectifs et sans allocation initiale).

Le nombre de ventes mises en place est naturellement inférieur avec un protocole de ventes simultanées. Cependant, la plupart des ventes orchestrées par un protocole séquentiel sont "gratuites". Elles n'engendrent pas de commutation sur les attributions et donc aucune actualisation de politique. Seule l'enchère pour l'objet en cours doit être évaluée et envoyée. En se plaçant sur un critère cherchant à minimiser les communications, le protocole par ventes séquentielles est préférable (Figure 5.10). En effet, chaque vente engendre un volume de n messages (agent, objet, valeur) avec n robots en mode séquentiel et $n|G|$ messages dans le cadre d'un SSAC. Pour 24 objectifs à allouer et sans allocation initiale, cela représente en moyenne, un volume $484,2$ $(80.7 * 6)$ messages contre $2404,8$ messages $(16.7 * 6 * 24)$ soit, approximativement, 5 fois plus.

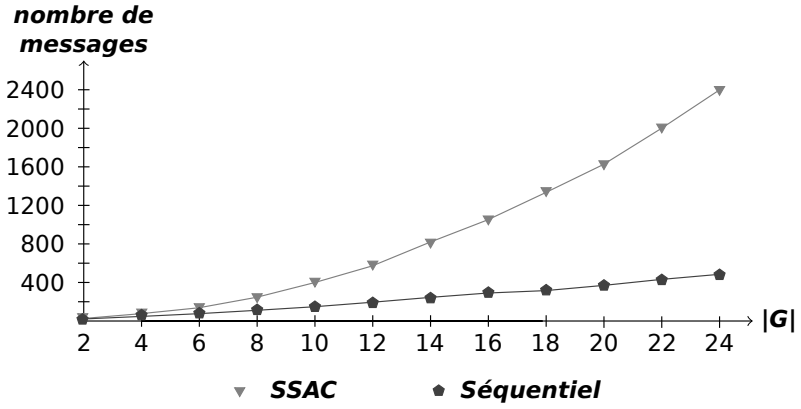


FIGURE 5.10 – Protocole séquentiel *versus* simultané : charge sur les communications.

Intérêt du SSAC

Cependant, nous plaçons nos travaux dans un cadre de communications quasi-gratuites comparativement aux ressources de calcul nécessaires pour l'actualisation des politiques individuelles. L'objet de l'étude consiste à montrer la capacité du protocole SSAC synchronisé à mieux paralléliser les ressources de calcul et, par conséquent, la vitesse de convergence. Les étapes d'actualisation des valeurs d'intérêt altruiste sont réalisées de façon distribuée. Les temps nécessaires à ces étapes sont alors similaires qu'il y ait 1 ou n agents pour actualiser ses valeurs.

L'évaluation de la vitesse de convergence des 2 approches nécessite une analyse plus fine des données relevées. En effet, en considérant comme critique les étapes d'actualisation de politiques et d'évaluation des nouvelles valeurs d'intérêt altruiste, la vitesse de convergence dépend du nombre de commutations successives. Dans un protocole séquentiel, toutes les commutations sont par nature, successives et concernent un ou deux robots. Les autres robots attendent alors passivement la fin des calcules réalisés par leurs camarades. Dans le cadre de SSAC plusieurs

commutations peuvent être réalisées en même temps, et davantage de robots pourront actualiser leurs politiques en parallèle. C'est alors le nombre total de ventes qui informe du nombre de commutations successives réalisées par le manager.

Ainsi, si l'on compare le nombre de commutations réalisées par le protocole séquentiel avec le nombre de ventes réalisées par SSAC (Figure 5.11), il est possible d'observer un gain intéressant sur la vitesse de convergence vers l'allocation finale optimale localement. On observe ainsi un gain de 42.9% en faveur des ventes simultanées.

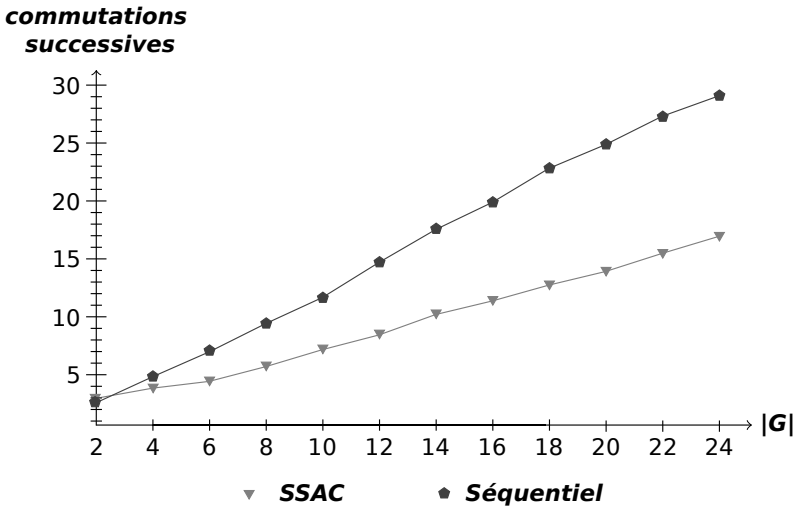


FIGURE 5.11 – Protocole séquentiel *versus* simultané : vitesse de convergence.

En conclusion de cette série d'expérimentations, il est possible d'affirmer que SSAC permet une meilleure parallélisation des ressources de calcul. SSAC, en s'appuyant sur plusieurs commutations d'objectifs simultanées, converge plus rapidement (en terme d'itération) vers une solution optimale localement. Or, diminuer les allocations successivement testées permet aux robots d'effectuer moins de calculs de politiques, ce qui est fortement intéressant lorsque ces calculs sont difficiles à réaliser.

Avec une allocation initiale aléatoire

Le tableau 5.3 présente les moyennes des relevés effectués pour les séries initialisées avec une allocation aléatoire et ici aussi, des robots partant d'une même zone. De nouveau, l'allocation aléatoire est la même pour les expériences avec le protocole séquentiel et avec SSAC. L'allocation aléatoire est définie de façon à équilibrer le nombre d'objectifs par robot. Chaque commutation correspond alors au retrait d'un objectif de l'attribution d'un agent au profit d'un autre.

On remarque alors que l'initialisation d'une allocation aléatoire permet, dans la mesure où les robots partent d'une même zone de l'environnement, de réduire le nombre de commutations et de ventes utiles à la convergence des 2 protocoles sur la solution finale. En revanche, les expériences confortent les premières conclusions tirées à partir des expériences sans allocation initiale. Cependant, les différences entre les deux approches sont plus faibles.

Les moyennes sur les valeurs des allocations construites sont proches avec un écart très faible, de nouveau en faveur du protocole par ventes séquentielles. Avec aussi, un écart réduit, le nombre de commutations successives montre que SSAC continue à converger plus rapidement. Ceci dit, le

TABLE 5.3 – Moyenne des résultats obtenus par ventes simultanées (SSAC) et par ventes séquentielles avec une allocation initiale aléatoire

	Coordination par ventes séquentielles :					Coordination par SSAC :				
	nombre de		valeur de	charge		nombre de		valeur de	charge	
$\|G\|$	ventes	commut.	l'allocation	min.	max.	ventes	commut.	l'allocation	min.	max.
2	2.4	0.4	1934.3	0.0	-41.4	1.4	0.4	1934.3	0.0	-41.4
4	6.5	1.8	3880.6	0.0	-54.2	2.3	1.7	3880.6	0.0	-54.3
6	11.1	2.9	5845.3	-0.0	-62.8	2.9	2.8	5845.2	-0.0	-63.1
8	16.8	4.8	7812.6	-1.2	-67.1	3.9	4.7	7812.5	-1.2	-66.9
10	24.2	7.2	9782.8	-4.2	-70.4	5.3	7.5	9783.2	-4.4	-69.7
12	31.3	8.1	11776.0	-9.8	-71.8	6.2	9.5	11775.8	-10.6	-71.2
14	39.7	10.5	13749.6	-15.1	-76.7	7.7	12.5	13749.8	-15.7	-76.1
16	49.7	13.1	15729.0	-18.5	-77.6	8.8	15.3	15728.5	-18.3	-77.9
18	58.7	14.3	17722.8	-21.6	-79.7	9.5	16.7	17723.4	-22.9	-78.0
20	67.5	16.0	19708.5	-24.9	-81.6	10.8	19.6	19708.1	-25.7	-81.0
22	76.8	18.2	21690.1	-26.2	-81.5	12.0	22.7	21689.6	-27.0	-81.1
24	85.5	19.6	23680.0	-30.1	-84.7	13.4	25.4	23686.1	-30.3	-83.1

gain sur le nombre moyen d'allocations successivement testées reste plus que honorable à 31.6% pour 6 robots et 24 objectifs.

5.4 Conclusion

Coordonner les actions de plusieurs agents est un problème difficile. Sur cette problématique, les protocoles par ventes aux enchères montrent de bons résultats pour allouer des tâches/ressources entre les agents. Les enchères découlent de planifications individuelles de façon à attribuer chaque tâche ou ressource à l'agent proposant l'intérêt le plus fort. Cependant, les tâches et les ressources sont interdépendantes. Ainsi, les valeurs d'enchères proposées par un agent varient en fonction de son attribution, de quelles tâches ou ressources il possède déjà.

Le protocole SSAC présenté dans ce chapitre permet à un groupe d'agents de converger vers une allocation des tâches (ou ressources) en respectant les synergies existantes entre les valeurs de ces tâches (ou ressources). Ce protocole repose sur une succession de ventes aux enchères simultanées. Ainsi, à chaque itération, l'ensemble des tâches/ressources peut être remis en question sur la base de l'allocation résultant des itérations précédentes. En respectant des règles strictes sur l'actualisation de l'itération, il est possible de garantir que SSAC converge vers une allocation finale optimale localement en un nombre fini d'itérations.

Dans notre étude, nous nous sommes focalisés sur l'allocation de tâches. L'intérêt de chaque tâche, à chaque itération, est alors évalué via l'utilisation de processus décisionnels de Markov orientés par des objectifs (GO-MDPs). De façon à équilibrer la charge de travail entre les agents, l'intérêt individuel donné par l'évaluation de Bellman est minoré d'un coût social. Ce coût social est paramétrable et permet de représenter les déséquilibres existants entre les attributions, en fonction des coûts mis en œuvre pour les réaliser.

Enfin, la solution proposée est évaluée conformément à un cadre applicatif multi-robots. Les résultats obtenus témoignent de la bonne qualité des allocations construites comparativement aux allocations optimales. De plus, SSAC, appuyé d'une évaluation par GO-MDP, supporte une montée en charge du nombre d'agents et d'objectifs pour atteindre des flottes moyennes de robots ($2-10$ robots) avec plusieurs objectifs par robot ($2-40$ objectifs). Enfin, une comparaison avec un protocole séquentiel permet de bien mettre en avant les avantages et inconvénients d'utiliser plusieurs commutations simultanées à chaque itération. Pour un sur-coût sur les communications,

SSAC permet effectivement d'accélérer la convergence vers des solutions optimales localement et quasi-équivalentes à des solutions obtenues par des ventes séquentielles.

Orientations futures

Dans le cadre des travaux que nous avons réalisés, nous nous sommes concentrés sur la validation d'une approche SSAC et GO-MDP. Les expériences réalisées ont montré des limites et appellent des améliorations pour accélérer encore la convergence ou la qualité de la solution construite. D'autres expérimentations mériteraient d'être développées sur la base d'un panel plus large d'environnements et de situations initiales et avec des heuristiques différentes pour l'actualisation de l'allocation dans SSAC. Les heuristiques peuvent être définies par un parcours aléatoire des objectifs ; de façon à favoriser l'agent avec une charge minimale ; en autorisant plusieurs commutations par agent, dans les premières itérations ; etc..

Dans un second temps, il pourrait être intéressant d'initier une réflexion sur la question : Quelle dose d'altruisme ? L'évaluation individuelle d'une distance à une charge de travail théorique idéale permet aux agents d'altérer leurs intérêts individuels au profit du groupe. Cependant, le profit du groupe n'est pas directement lié à une équité totale entre les agents mais à la minimisation de la charge maximale. Ce dernier critère (contrairement à une charge homogène) est difficilement distribuable, à chaque instant l'agent de charge maximale doit être identifié et mis au centre des discussions. L'idée reposerait alors sur une étude comparative des coordinations en fonction du degré d'altruisme injecté dans chaque agent (le coût d'opportunité dans l'approche proposée ici) pour optimiser la convergence et la qualité de la coordination résultante.

A plus long terme, il serait intéressant d'adapter le protocole SSAC à des problèmes de coordination plus généraux. Actuellement, notre approche par ventes aux enchères ne permet que d'allouer parmi des agents, des tâches et des ressources qui sont réalisées ou consommées de façon individuelle. Pour traiter des problématiques plus larges, le processus de coordination doit permettre d'affecter des variables partagées quels que soient leurs domaines de variations. Concrètement pour des robots mobiles, les variables partagées peuvent représenter des dates et lieux de rendez-vous, des sens de circulation, règles d'usage ou encore de positionnement de panneaux. L'idée repose alors sur une augmentation de l'expressivité des messages échangés entre les agents dans SSAC pour que chacun puisse communiquer son intérêt sur telles ou telles affectations de variables partagées. Les enchères ne reflètent plus l'intérêt d'ajouter ou de supprimer un objet à une attribution mais traduisent l'intérêt d'associer des valeurs aux variables partagées. On bascule alors sur des systèmes de votes pondérés où chaque modification affecte potentiellement tous les agents.

Conclusion et perspectives

Les travaux réalisés dans le cadre de cette thèse visent à l'élaboration d'une stratégie coopérative pour la mise en œuvre d'une flotte de robots mobiles dans un milieu ouvert et encombré. Nous nous sommes intéressés aux missions visant la réalisation de plusieurs tâches indépendantes les unes des autres. Que la définition des tâches soit dynamique (e.g. exploration) ou non (e.g. navigation), il est difficile de modéliser le problème dans sa globalité pour raisonner et planifier le comportement des robots dans leurs détails et jusqu'à la réalisation totale de la mission.

L'approche défendue ici repose sur plusieurs niveaux d'abstractions de façon à permettre individuellement de prendre des décisions locales, tout en considérant un horizon lointain. Ces décisions sont motivées par des phases ponctuelles de coordination où les robots sont en mesure de se répartir la réalisation de leur mission, par un processus distribué. Les contributions présentées dans cette thèse s'inscrivent autour de 3 thèmes : un contrôle divisé sur plusieurs niveaux ; une planification hiérarchisée des déplacements multi-objectifs et des processus distribués pour l'allocation de tâches. Ceci a pour objectif de réduire la complexité du probléme tout en s'appuyant sur un outil mathématique efficace, sur des Processus Décisionnels de Markov (MDPs) distribués.

Nous avons vu, dans un premier chapitre, les difficultés spécifiques aux robots mobiles. Les vitesses ou les accélérations doivent être définies à chaque instant pour générer des déplacements sécurisés et pour garantir d'atteindre successivement des cibles distantes. L'exercice consiste, à partir d'une perception, à localiser les éléments de l'environnement par rapport au robot et à définir les commandes en conséquence. L'approche proposée est positionnée sur une coordination par planification distribuée de déplacements haut niveau basés sur un contrôle réactif fonctionnel.

La problématique d'une planification multi-robots multi-tâches a été ensuite présentée au chapitre 2, dans le cadre d'une mission d'exploration. Cette problématique peut être modélisée comme un multi-voyageurs de commerce dynamique, évoluant dans un environnement incertain. L'objectif consiste à planifier les déplacements minimaux des robots de façon à ce que toutes les positions frontières connues à un instant donné, soient explorées. La définition des tâches (les positions frontières à explorer) peut alors être dynamique. Elle évolue au fur est à mesure que de nouveaux passages ou obstructions sont détectés. De plus, il existe une incertitude sur la localisation et les déplacements des robots. Ainsi, pour un état donné des connaissances, il est proposé une modélisation de l'évolution d'un robot mobile à l'aide d'un processus décisionnel de Markov orienté par des objectifs (GO-MDP). Cependant, la taille des problèmes exprimés dans ce modèle ne permet de traiter qu'un petit nombre d'objectifs, pendant la mission.

Le chapitre 3 présente l'architecture proposée pour des robots explorateurs. Cette architecture est basée sur une connaissance individuelle et topologique. Elle permet de séparer les difficultés liées à la mise en œuvre du contrôle/commande des robots par une décomposition en 4 parties : Perception, Représentation, Décision et Contrôle. Les expérimentations réalisées dans un environnement urbain expérimental ont montré qu'il était possible de planifier les déplacements des robots sur la base d'une carte topologique (Road Map) de faible densité pour définir des prises de décisions uniquement sur des positions clefs de l'environnement. La navigation entre ces positions clefs distantes est reléguée au niveau réactif offrant de bonnes garanties tout en utilisant des processus peu coûteux en ressources de calculs.

L'utilisation de cartes de faible densité devrait alors permettre une planification des déplacements vis à vis de mission complexe. Nous nous sommes intéressés dans le chapitre 4, aux situations où un nombre important d'objectifs (positions) doivent être atteints. Une modélisation par une collection de GO-MDPs individuels implique une explosion combinatoire du nombre d'états potentiels de chacun des robots. Il est proposé ici une approche innovante sur la base d'un partitionnement rapide de la carte couplé à une planification hiérarchique approchée. La prise de décision s'effectue alors sur plusieurs niveaux :

- un niveau global, induisant un certain ordonnancement des régions
- un niveau local, pour atteindre les objectifs au sein de la région courante tout en considérant les régions suivantes.

Ainsi, il a été montré par des expériences empiriques que le partitionnement en régions permet à un leader d'allouer les tâches par paquet. Il a aussi été montré que chaque robot d'une flotte est alors en mesure de calculer ponctuellement une politique de bonne qualité lui permettant de réaliser les tâches qui lui ont été attribuées.

Enfin, dans le dernier chapitre, nous avons présenté un protocole de coordination par allocation de tâches s'appuyant sur une planification distribuée. Le protocole de coordination par succession de ventes aux enchères simultanées, développé ici, permet à un groupe de robots de converger vers une allocation des tâches en respectant les synergies existantes entre les valeurs de ces tâches. En effet, les tâches sont inter-dépendantes entre elles et les valeurs d'enchères proposées évoluent au fur et à mesure que l'allocation est actualisée vers une solution optimale localement. Il a été montré que le protocole converge sur des solutions intéressantes tout en améliorant la parallélisation des calculs, par rapport à un protocole purement séquentiel.

Perspectives

Pour chacun des thèmes pour lesquels des contributions ont été apportées, il a été relevé des perspectives qui peuvent être traitées à plus ou moins long terme (se référer aux sections "Conclusion" des chapitres 3, 4 et 5). En effet, ce travail de thèse a été réalisé sur 3 ans et sur un champ assez large. Les contributions apportées mériteraient, chacune d'elles, des investigations plus approfondies pour garantir leur efficacité sur des cadres applicatifs plus génériques. Les perspectives proposées peuvent être regroupées en 3 principaux objectifs :

- mieux caractériser les modules de contrôle/commande et de perception des robots mobiles,
- orchestrer plus de fonctionnalités,
- valider de façon plus large, les processus délibératifs employés.

Les 2 premières thématiques visent une amélioration des capacités de nos robots. La dernière thématique cherche davantage à exporter et continuer les travaux réalisés, dans un cadre plus générique.

Une meilleure caractérisation des modules

L'efficacité de la supervision des déplacements des robots dans un environnement dépend principalement des modules de perception, des modules de contrôle et de la connaissance que l'on a de ces modules. Une augmentation de l'autonomie des robots repose principalement sur la mise en place de modules ayant des sémantiques partagées plus fortes (e.g. reconnaissance des éléments environnants : véhicule, robot, humain...). Cela demande alors un travail important sur la normalisation des entrées/sorties possibles pour les 4 parties de l'architecture PRDC, de façon à produire une supervision haut niveau, à la fois générique et adaptée.

La carte topologique (Road-Map) se trouve au centre de ce travail de normalisation. La Road-Map exprime les capacités fonctionnelles des robots pour être utilisée comme entrée par les processus délibératifs. Sur ce thème, une perspective importante concerne la capacité de cartographier un environnement de façon hybride, c'est-à-dire en profitant de jeux d'informations à la fois topologiques et métriques. Le challenge consiste à profiter au mieux de données issues de modules raffineurs perceptifs variés tout en garantissant des cartes de faible densité et partageables entre les robots.

Enfin, les processus décisionnels doivent s'adapter pour apporter une supervision du contrôle qui soit précise et efficace tout en conservant des temps de calcul acceptables. Dans la mesure où la donnée initiale (Road-Map) s'enrichit (e.g. connaissances sur le trafic), les prises de décision peuvent s'effectuer à partir d'un ensemble d'informations plus large. L'enrichissement des données risque de conduire à des problèmes de taille toujours plus importante, toutes les améliorations apportées doivent alors être intégrées dans les phases de coordination, sans diminuer l'horizon de planification.

Orchestrer plus de fonctionnalités

L'enrichissement des fonctionnalités des robots mobiles peut s'effectuer aussi sur un champ exécutif plus large : rouler/marcher, suivre, doubler, manipuler des objets ou toute autre forme d'interaction. Le besoin en Intelligence Artificielle embarquée va s'intensifier au fur et à mesure que les robots deviendront multi-fonctionnels. Il faut alors montrer que les approches décisionnelles proposées sont à la fois génériques et adaptatives.

Les solutions doivent pouvoir être utilisées quels que soient les types de robots et doivent pouvoir répondre à des scénarios variés. Cela suppose ici que la Road-Map soit ouverte de façon à permettre, au besoin, l'ajout d'informations spécifiques sur les nœuds et les arcs. Il serait intéressant aussi de chercher à exprimer la synergie d'actions fonctionnelles réalisées en même temps (e.g. se déplacer et manipuler). Les processus décisionnels individuels devront alors être adaptés pour intégrer des scénarios riches (e.g. le déplacement d'objets dans des environnements multi-agents implique l'optimisation des déplacements et du trafic).

Enfin, dans la mesure où l'approche permet d'appréhender des robots différents avec des fonctionnalités plurielles, une coordination fine des comportements individuels sera sûrement requise. Exprimer la coordination de robots par une simple allocation de tâches et de ressources est dans ce cas restrictif. Une autre perspective majeure consiste à s'orienter vers un protocole de coordination basé sur une succession d'échanges d'intérêts, permettant des recherches de consensus plus génériques. Les consensus peuvent alors intégrer la mise en place de rendez-vous,

l'attribution de rôles (e.g. leader/serviteur, acteur/observateur, etc.) ou encore la définition de règles d'usage (e.g. priorité à droite).

Une validation plus large des processus délibératifs employés

Les approches développées dans le cadre de cette thèse sont ancrées dans un cadre multi-robots mobiles. Pour la partie planification, elles s'appuient essentiellement sur une modélisation par processus décisionnels de Markov. Ce modèle est caractérisé notamment par sa généricité. Il est alors légitime de s'interroger sur la capacité de l'approche par résolution hiérarchique et de l'approche par succession de ventes aux enchères simultanées à s'appliquer efficacement à des problèmes différents (e.g. la gestion de l'énergie dans les grilles intelligentes).

De façon plus globale, il serait intéressant de mettre en place une validation plus générique des approches décisionnelles qui ont été proposées. L'idée serait de partir d'une problématique multi-agents exprimée de façon générique, sans contextualiser l'agent et son environnement. Ainsi, il serait possible d'évaluer l'efficacité des deux approches développées quelle que soit la dynamique du système, si celui-ci peut s'exprimer sous la forme d'une collection de processus décisionnels de Markov orientés par des objectifs (GO-MDPs) (un par agent).

Il serait donc possible d'orienter les travaux futurs vers une résolution hiérarchique et distribuée de problèmes multi-agents génériques. Dans un premier temps, cela consisterait à fusionner la résolution individuelle hiérarchique avec le processus de coordination par ventes aux enchères. Dans un second temps, appuyé sur les réflexions de recherche de consensus plus génériques, le travail consiste à pouvoir exprimer des variables partagées autres que l'allocation d'objectifs. L'approche pourrait alors être étendue pour résoudre, de façon hiérarchique et distribuée, des problèmes multi-agents nécessitant la synergie d'actions réalisées en même temps par plusieurs agents.

Bibliographie

[Abdel-Illah Mouaddib 07] Maroua Bouzid Abdel-Illah Mouaddib Mathieu Boussard. *Towards a Formal Framework for Multi-Objective Multi-Agent Planning.* In Autonomous Agents and Multi-Agent Systems, 2007. 63

[Aberdeen 03] D. Aberdeen. *A (Revised) Survey of Approximate Methods for Solving Partially Observable Markov Decision Processes.* National ICT Australia, technical report, 2003. 61

[Adouane 09] L. Adouane. *Hybrid and Safe Control Architecture for Mobile Robot Navigation.* In Autonomous Robot Systems and Competitions, 2009. 12, 77

[Alami 98] R. Alami, R. Chatila, S. Fleury, M. Ghallab & F. Ingrand. *An Architecture for Autonomy.* Journal of Robotics Research, vol. 17, no. 4, pages 315–337, 1998. 71, 72, 75, 79

[Arkin 87] Ronald C. Arkin. *Motor schema-based mobile robot navigation.* In International Conference on Robotics and Automation, pages 264–271, 1987. 18

[Astrom 65] K.J. Astrom. *Optimal control of Markov decision processes with incomplete state estimation.* Journal of Mathematical Analysis and Applications, vol. 10, pages 403–406, 1965. 61

[Barnard 94] S. T. Barnard & H. D. Simon. *Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems.* Concurrency : Practice and Experience, vol. 6, no. 2, pages 101–117, 1994. 107

[Barraquand 92] J. Barraquand, B. Langlois & J.-C. Latombe. *Numerical potential field techniques for robot path planning.* IEEE Transactions on Systems, Man and Cybernetics, vol. 22, pages 224–241, 1992. 18, 33

[Bellman 57] R. Bellman. *A Markovian Decision Process.* Journal of Mathematics and Mechanics, vol. 6, pages 679–684, 1957. 40, 41, 43, 44, 54, 55, 101, 114

[Bellman 70] R. E. Bellman & L. A. Zadeh. *Decision-Making in a Fuzzy Environment.* Management Science, vol. 17, pages B141 – B164, 1970. 55

[Belouaer 11] L. Belouaer, M. Bouzid & A.-I. Mouaddib. *Spatial knowledge in planning language.* International Conference on Knowledge Engineering and Ontology Development, 2011. 27

[Benzerrouk 10] Ahmed Benzerrouk, Lounis Adouane, Laurent Lequievre & Philippe Martinet. *Navigation of multi-robot formation in unstructured environment using dynamical virtual structures.* In IROS, pages 5589–5594, 2010. 13, 18

[Benzerrouk 11] A. Benzerrouk. *Architecture de contrôle hybride pour systèmes multi-robots mobiles.* PhD thesis, Université Blaise Pascal, 2011. 18

[Berhault 03] M Berhault, H. Huang, P. Keskinocak, S. Koenig, W. Elmaghraby, P. Griffin & A Kleywegt. *Robot exploration with combinatorial auctions.* In International Conference on Intelligent Robots and Systems, volume 2, pages 1957–1962, 2003. 34, 35, 53, 67, 128, 129

[Bernstein 00] D. S. Bernstein, S. Zilberstein & N. Immerman. *The Complexity of Decentralized Control of Markov Decision Processes.* In 16th Conference on Uncertainty in Artificial Intelligence, 2000. 48, 59, 62, 63, 126

[Beynier 05] A. Beynier & A.I. Mouaddib. *A polynomial algorithm for decentralized Markov decision processes with temporal constraints.* In Autonomous Agents and Multi-Agent Systems, pages 963–969, 2005. 79

[Beynier 06] A. Beynier & A.I. Mouaddib. *An iterative algorithm for solving Constrained Decentralized Markov Decision Processes.* In The 31st National Conference on Artificial Intelligence, pages 1089–1094, 2006. 60, 63, 126, 132

[Beynier 10] A. Beynier & A.-I. Mouaddib. *A Rich Communication Model in Opportunistic Decentralized Decision Making.* In International Conference Intelligent Agent Technology, pages 133 – 140, 2010. 66

[Bichot 10] Charles-Edmond Bichot & Patrick Siarry. Partitionnement de graphe. Hermes-Lavoisier, 2010. 106, 107

[Bichot 11] Charles-Edmond Bichot & Patrick Siarry. Graph partitioning. Wiley-ISTE, 2011. 105

[Boussard 07] M. Boussard, M. Bouzid & A.-I. Mouaddib. *Multi-criteria decision making for local coordination in multi-agent systems.* In International Conference on Tools with Artificial Intelligence, pages 87–90, 2007. 63, 64, 79

[Boussard 08] M. Boussard, M. Bouzid & A.-I. Mouaddib. *Vector Valued Markov Decision Process for robot platooning.* In European Conference on Artificial Intelligence, pages 925–926, 2008. 66, 132

[Boutilier 96] Craig Boutilier. *Planning, Learning and Coordination in Multiagent Decision Processes.* In Yoav Shoham, editeur, 6th Conference on Theoretical Aspects of Rationality and Knowledge, pages 195–210. Morgan Kaufmann, 1996. 40

[Boutilier 99] Craig Boutilier, Thomas Dean & Steve Hanks. *Decision-theoretic planning : Structural assumptions and computational leverage.* Journal of Artificial Intelligence Research, vol. 11, pages 1–94, 1999. 100

[Brooks 86] R. Brooks. *A robust layered control system for a mobile robot.* Journal of Robotics and Automation, vol. 2, pages 14–23, 1986. 9, 11, 77

[Buffet 08] O. Buffet & O. Sigaud. Processus décisionnels de markov en intelligence artificielle. Lavoisier, hermes science edition, 2008. 40, 44

[Bui 93] T. N. Bui & C. Jones. *A Heuristic for Reducing Fill-In in Sparse Matrix Factorization.* In 6th Conference on Parallel Processing for Scientific Computing, pages 445–452, 1993. 107

[Burgard 05] W. Burgard, M. Moors, C. Stachniss & F. Schneider. *Coordinated Multi-Robot Exploration.* IEEE Transactions on Robotics, vol. 21, 2005. 14, 15, 34, 39, 53, 54, 56, 57, 60, 66, 73, 79, 125, 131

[Campion 96] G. Campion, G. Bastin & B. Dandrea-Novel. *Structural properties and classification of kinematic and dynamic models of wheeled mobile robots.* IEEE Transactions on Robotics and Automation, vol. 12, pages 47–62, 1996. 6

[Canu 11] Arnaud Canu & Abdel-Illah Mouaddib. *Dynamic Local Interaction Model : framework and algorithms.* In International Conference on Autonomous Agents and Multiagent Systems, (Workshop MSDM), 2011. 64, 126

[Canudas de Wit 92] C. Canudas de Wit & O.J. Sordalen. *Exponential stabilization of mobile robots with nonholonomic constraints.* IEEE Transactions on Automatic Control, vol. 37, pages 1791–1797, 1992. 9

[CAO 97] Y. U. CAO, A. S. FUKUNAGA & A. B. KAHNG. *Cooperative Mobile Robotics : Antecedents and Directions.* Autonomous Robots, vol. 4, pages 7–27, 1997. 33

[Capitan 12] J. Capitan, M. T. J. Spaan, L. Merino & A. Ollero. *Decentralized multi-robot cooperation with auctioned POMDPs.* In International Conference on Robotics and Automation, pages 3323–3328, 2012. 34, 131

[Cassandra 94] A. Cassandra, L. Kaelbling & M. Littman. *Acting optimally in partially observable stochastic domains.* In National Conference on Artificial intelligence, 1994. 60, 61

[Cassandra 98] A. Cassandra. Exact and approximate algorithms for partially observable, markov decision pocesses. Thèse de doctorat, Brown University, 1998. 61

[Chades 02] I. Chades, B. Scherrer & F. Charpillet. *A Heuristic Approach for Solving Decentralized-POMDP : Assessment on the Pursuit Problem.* In ACM symposium on applied computing, pages 57–62, 2002. 33, 34, 35, 48, 60, 63, 126

[Chardaire 07] P. Chardaire, M. Barake & G. P. McKeown. *A PROBE-Based Heuristic for Graph Partitioning.* IEEE Transaction on Computers, vol. 56, no. 12, pages 1707–1720, 2007. 107

[Chatila 95] Raja Chatila. *Deliberation and reactivity in autonomous mobile robots.* Robotics and Autonomous Systems, vol. 16, pages 197–211, 1995. 71, 72

[Choset 95a] Howie Choset & J. Burdick. *Sensor Based Planning, Part I : The Generalized Voronoi Graph.* In International Conference on Robotics and Automation, pages 1643–1648, 1995. 22

[Choset 95b] Howie Choset & J. Burdick. *Sensor Based Planning, Part II : Incremental Construction of the Generalized Voronoi Graph.* In International Conference on Robotics and Automation, pages 1649–1655, 1995. 22

[Choset 01] H. Choset & K. Nagatani. *Topological simultaneous localization and mapping (SLAM) : toward exact localization without explicit localization.* Transactions on Robotics and Automation, vol. 17, pages 125–137, 2001. 29, 33, 53

[Cormen 01] Thomas H Cormen, Clifford Stein, Ronald L Rivest & Charles E Leiserson. Introduction to algorithms. MIT Press, 2001. 20

[Courbon 09] J. Courbon, Y. Mezouar & P. Martinet. *Autonomous navigation of vehicles from a visual memory based on the use of generic camera model.* IEEE Transaction on Intelligent Transportation System, vol. 10, pages 392–402, 2009. 11, 29, 77, 79

[Davis 83] R. Davis & R. G. Smith. *Negotiation as a Metaphor for Distributed Problem Solving.* Artificial Intelligence, vol. 20, pages 63–109, 1983. 125, 127, 128, 129

[Dean 95] Thomas Dean, Shieu hong Lin & Shieu hong Lin. *Decomposition Techniques for Planning in Stochastic Domains.* In 14th International Joint Conference on Artificial Intelligence, 1995. 42, 100, 102, 103, 104

[Dias 06] M. B. Dias, R. Zlot, N. Kalra & A. Stentz. *Market-Based Multirobot Coordination : A Survey and Analysis.* Proceedings of the IEEE, vol. 94, pages 1257–1270, 2006. 125, 127, 128, 133

[Dibangoye 09] J. Dibangoye, B. Chaib-Draa, A.-I. Mouaddib & G. Shani. *Topological Order Planner POMDP.* In International Joint Conference of Artficial Intelligence, 2009. 64

[Dudek 96] G. Dudek, M. Jenkin, E. Milios & D. Wilkes. *A taxonomy for swarm robots.* Autonomous Robots, vol. 3, pages 375–397, 1996. 17

[Duhaut 07] Dominique Duhaut, Elian Carrillo & Sébastien Saint-Aime. *Avoiding deadlock in multi-agent systems.* In International Conference on Systems, Man and Cybernetics, pages 1642–1647, 2007. 13

[Elfe 87] A. Elfe. *Sonar-based real-world mapping and navigation.* Robotics and Automation, vol. 3, pages 249–265, 1987. 11, 23, 26, 27, 28, 30, 33, 53, 77, 78, 81

[Engelson 92] S. P. Engelson & D. V. McDermott. *Error correction in mobile robot map learning.* In International Conference on Robotics and Automation, pages 2555–2560, 1992. 30

[Ferber 95] Jacques Ferber. Les systèmes multi-agents, vers une intelligence collective. InterEditions, iia edition, 1995. 6, 40, 45

[Fierro 01] R. Fierro, A.K. Das, V. Kumar & J.P. Ostrowski. *Hybrid control of formations of robots.* In International Conference on Robot and Automation, volume 41, pages 157–162, 2001. 17

[Fikes 71] R.E. Fikes & N. J. Nilsson. *STRIPS : A new Approach to the Application of Theorem Proving to Problem Solving.* Artificial Intelligence, vol. 2, no. 3-4, pages 189–208, 1971. 40, 41, 42

[Foka 07] Amalia F. Foka & Panos E. Trahanias. *Real-time hierarchical POMDPs for autonomous robot navigation.* Robotics and Autonomous Systems, vol. 55, no. 7, pages 561–571, 2007. 54, 60, 61, 73, 77, 79

[Franchi 07] A. Franchi, L. Freda, G. Oriolo & M. Vendittelli. *A Randomized Strategy for Cooperative Robot Exploration.* In International Conference on Robotics and Automation, pages 768–774, 2007. 34, 35

[Garey 74] M. R. Garey, D. S. Johnson & L. Stockmeyer. *Some simplified NP-complete problems.* In 6h Symposium on Theory of Computing, pages 47–63, New York, NY, USA, 1974. ACM. 105

[Gerkey 02] B. P. Gerkey & M. J. Mataric. *Sold ! : auction methods for multirobot coordination.* Transactions on Robotics and Automation, vol. 18, 2002. 33, 34, 125, 127

[Goldman 04] C. V. Goldman & S. Zilberstein. *Decentralized Control of Cooperative Systems : Categorization and Complexity Analysis.* Journal of Artificial Intelligence Research, vol. 22, 2004. 63, 126

[Guestrin 02] C. Guestrin, D. Koller & R. Parr. *Multiagent planning with factored mdps.* Advances in information processing systems, vol. 2, pages 1523–1530, 2002. 63

[Guitton 09] Julien Guitton, Jean-Loup Farges & Raja Chatila. *Cell-RRT : Decomposing the environment for better plan.* In International Conference on Intelligent Robots and Systems, pages 5776–5781, 2009. 22

[Guo 02] Y. Guo & L. E. Parker. *A distributed and optimal motion planning approach for multiple mobile robots.* In International Conference on Robotics and Automation, volume 3, pages 2612–2619, 2002. 33, 34

[Gustavi 08] T. Gustavi & X. Hu. *Observer-Based Leader-Following Formation Control Using Onboard Sensor Information.* IEEE Transactions on Robotics, vol. 24, pages 1457–1462, 2008. 18

[Hagen 92] L. W. Hagen & A. B. Kahng. *A new approach to effective circuit clustering.* In Internationnal Conference on Computer Aide Design, pages 422–427, 1992. 107

[Hanna 02] H. Hanna & A.-I. Mouaddib. *Task selection as decision making in multiagent systems.* In Autonomous Agents and MultiAgent Systems, pages 616–623, 2002. 63

[Haralick 92] R. M. Haralick & L. D. Shapiro. Computer and robot vision. Addison-Wesley, 1992. 26

[Hart 68] P. E. Hart. *A Formal Basis for the Heuristic Determination of Minimum Cost Paths.* Transactions on Systems Science and Cybernetics, vol. 4, pages 100–107, 1968. 20, 42

[Hébert 96] Patrick Hébert, Stéphane Betgé-Brezetz & Raja Chatila. *Probabilistic Map Learning : Necessity and Difficulties.* Lecture Notes in Computer Science, vol. 1093, 1996. 26

[Howard 60] R.A. Howard. Dynamic programming and markov processes. MIT Press, 1960. 56

[Huang 01] Q. Huang, K. Yokoi, S. Kajita, K. Kaneko, H. Arai, N. Koyachi & K Tanie. *Planning walking patterns for a biped robot.* Transactions on Robotics and Automation, vol. 17, pages 280–289, 2001. 9

[Kalman 60] Rudolph Emil Kalman. *A New Approach to Linear Filtering and Prediction Problems.* Transactions of the ASME–Journal of Basic Engineering, vol. 82, pages 35–45, 1960. 26

[Karam 06] Nadir Karam, Frederic Chausse, Romuald Aufrere & Roland Chapuis. *Localization of a Group of Communicating Vehicles by State Exchange.* In International Conference on Intelligent Robots and Systems, pages 519–524, 2006. 14

[Karami 10] A.-B. Karami, L. Jeanpierre & A.-I. Mouaddib. *Human-robot collaboration for a shared mission.* In International Conference on Human-Robot Interaction, pages 155–156, 2010. 14

[Karypis 98] G. Karypis & V. Kumar. *Multilevel k-way Partitioning Scheme for Irregular Graphs.* Journal of Parallel and Distributed Computing, vol. 48, no. 1, pages 96–129, 1998. 107

[Kavraki 96] Lydia Kavraki, Petr Svestka, Jean claude Latombe & Mark Overmars. *Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces.* In IEEE International Conference on Robotics and Automation, pages 566–580, 1996. 22, 33

[Kernighan 70] B. W. Kernighan & S. Lin. *An Efficient Heuristic Procedure for Partitioning Graphs.* The Bell system technical journal, vol. 49, pages 291–307, 1970. 107

[Khatib 86] O. Khatib. *Real-Time Obstacle Avoidance for Manipulators and Mobile Robots.* The International Journal of Robotics Research, vol. 5, 1986. 18

[Korrapati 12] H. Korrapati, Y. Mezouar, P. Martinet & J. Courbon. *Image Sequence Partitioning for Outdoor Mapping.* In International Conference on Robotics and Automation, 2012. 27, 79, 83

[Krishna 96] V. Krishna & R. W. Rosenthal. *Simultaneous Auctions with Synergies.* Games and Economic Behavior, pages 1–31, 1996. 128

[Kuipers 91] B. Kuipers & Y.-T. Byun. *A Robot Exploration and Mapping Strategy Based on a Semantic Hierarchy of Spatial Representations.* Journal of Robotics and Autonomous Systems, vol. 8, pages 47–63, 1991. 29, 33, 77, 78, 79

[Kuipers 00] Benjamin Kuipers. *The Spatial Semantic Hierarchy.* Artificial Intelligence, vol. 119, pages 191–233, 2000. 27

[Lasdon 70] L. S. Lasdon. Optimization theory for large systems. Macmillan Company, 1970. 42

[Laumond 89] Jean-Paul Laumond, Thierry Siméon, Raja Chatila & Georges Giralt. *Trajectory planning and motion control for mobile robots.* Lecture Notes in Computer Science, vol. 391, 1989. 20, 77

[LaValle 00] S. M. LaValle & J. J. Kuffner. *Rapidly-exploring random trees : Progress and prospects.* In the Algorithmic Foundations of Robotics, 2000. 22

[Lenain 11] R. Lenain, B. Thuilot, O. Hach & P. Martinet. *High-speed mobile robot control in off-road conditions : a multi-model based adaptive approach.* In International Conference on Robotics and Automation, 2011. 10

[Lisien 03] Brad Lisien, Deryck Morales, David Silver, George Kantor, Ioannis Rekleitis & Howie Choset. *Hierarchical Simultaneous Localization and Mapping.* In Intelligent Robots and Systems, pages 448–453, 2003. 27

[Littman 95] M. Littman, T. Dean & L. Kaelbling. *On the complexity of solving Markov decision problems.* In 11th Conference on Uncertainty in Artificial Intelligence, pages 394–402, 1995. 59, 100

[Lozenguez 11a] G. Lozenguez, L. Adouane, A. Beynier, P. Martinet & A.-I. Mouaddib. *Map Partitioning to Approximate an Exploration Strategy in Mobile Robotics.* In Advances on Practical Applications of Agents and Multiagent Systems, volume 88, pages 63–72, 2011. 100

[Lozenguez 11b] Guillaume Lozenguez, Lounis Adouane, Aurélie Beynier, Philippe Martinet & Abdel-Illah Mouaddib. *Calcul distribué de politiques d'exploration pour une flotte de robots mobiles.* In Journées Francophone des Systémes Multi-Agent, 2011. 126

[Lozenguez 12a] G. Lozenguez, L. Adouane, A. Beynier, P. Martinet & A.-I. Mouaddib. *Interleaving Planning and Control of Mobiles Robots in Urban Environments Using Road-Map.* In International Conference on Intelligent Autonomous Systems, 2012. 100

[Lozenguez 12b] G. Lozenguez, L. Adouane, A. Beynier, A.-I. Mouaddib & P. Martinet. *Map Partitioning to Approximate an Exploration Strategy in Mobile Robotics.* Multiagent and Grid Systems, vol. 8, pages 275–288, 2012. 70

[Lumelsky 90] V. Lumelsky & A. Stepanov. *Path Planning Strategies for a Point Mobile Automaton Moving Amidst Unknown Obstacles of Arbitrary Shape.* Autonomous Robot Vehicles, 1990. 16, 70

[Madani 99] 0. Madani, S. Hanks & A. Condon. *On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems.* In National Conference on Artificial Intelligence, pages 541–548, 1999. 61

[Martin 06] J. G. Martin. *Spectral techniques for graph bisection in genetic algorithms.* In Genetic and Evolutionary Computation Conference, pages 1249–1256, 2006. 107

[Mataric 92] Maja J. Mataric. *Integration of Representation Into Goal-Driven Behavior-Based Robots.* Transactions on robotics and automation, vol. 8, pages 304–312, 1992. 29

[Matarić 95] M. J. Matarić. *From Local Interactions to Collective Intelligence.* In The Biology and Technology of Intelligent Autonomous Agents, pages 275–295, 1995. 130

[Matignon 12] L. Matignon, J.P. Laurent & A.I. Mouaddib. *Distributed Value Functions for Multi-Robot Exploration.* In International Conference on Robotics and Automation, 2012. 14, 28, 30, 34, 35, 39, 53, 54, 56, 57, 60, 64, 66, 79, 125

[Merveilleux 10] P. Merveilleux, O. Labbani-Igbida & E. M. Mouaddib. *Free space detection using active contours in omnidirectional images.* In International Conference on Image Processing, pages 3533–3536, 2010. 26, 77, 81

[Morette 09] Nicola Morette. *Contribution à la navigation de robots mobiles : approche par modèle direct et commande prédictive.* PhD thesis, École doctorale Science et Technologie d'orleans, 2009. 73

[Morette 11] N. Morette, C. Novales, L. Josserand & P. Vieyres. *Direct Model Navigation issue shifted in the continuous domain by a predictive control approach for mobile robots.* In International Conference on Robot and Automation, pages 2566–2573, 2011. 20, 73, 77

[Mourioux 07] G. Mourioux, C. Novales & G. Poisson. *Control robot by a generic control architecture.* In IEEE International Conference on Intelligent Robots and Systems, pages 3050 – 3055, 2007. 73, 74, 76

[Naïm 07] P. Naïm, P. Wuillemin, P. Leray, O. Pourret & A. Becker. Les réseaux bayésiens. Eyrolles, 2007. 45

[Nair 03] R. Nair, M. Tambe, M. Yokoo, S. Marsella & D.V. Pynadath. *Taming decentralized pomdps.* In International Joint Conference on Artificial Intelligence, 2003. 48, 60, 63

[Nair 05] R. Nair, P. Varakantham, M. Tambe & M. Yokoo. *Networked Distributed POMDPs : A Synthesis of Distributed Constraint Optimization and POMDPs.* In National Conference on Artificial Intelligence, page 7, 2005. 60, 126

[Nehmzow 00] Ulrich Nehmzow & Carl Owen. *Robot Navigation in the Real World : Experiments with Manchester's FortyTwo in Unmodified, Large Environments.* Robotics and Autonomous Systems, vol. 33, pages 223–242, 2000. 27, 29, 30, 77

[P. R. Milgrom 82] R. J. Weber P. R. Milgrom. *A theory of auctions and competitive bidding.* Econometrica : Journal of the Econometric Society, 1982. 128

[Papadimitriou 87] C.H. Papadimitriou & J.N. Tsitsiklis. *The complexity of Markov decision process.* Mathematics of Operations Research, vol. 12, pages 441–450, 1987. 59, 61, 100

[Parker 93] L.E. Parker. *Designing control laws for cooperative agent teams.* In International Conference on Robotics and Automation, pages 582–587, 1993. 11

[Parr 98] Ronald Parr. *Flexible Decomposition Algorithms for Weakly Coupled Markov Decision Problems.* In 14th Conference on Uncertainty in Artificial Intelligence, pages 422–430, 1998. 105, 106

[Puterman 94] M. L. Puterman. Markov decision processes : Discrete stochastic dynamic programming. John Wiley & Sons, Inc., 1994. 44, 54, 56

[Puterman 05] M. L. Puterman. Markov decision processes : Discrete stochastic dynamic programming. John Wiley & Sons, Inc. (Deuxiéme édition), 2005. 56

[Ren 08] W. Ren & R. W. Beard. Distributed consensus in multi-vehicle cooperative control, theory and application. springer, 2008. 48, 67, 126

[Reynolds 87] Craig W. Reynolds. *Flocks, Herds, and Schools : A Distributed Behavioral Model.* Computer graphics and interactive techniques, page 13, 1987. 12, 14, 70

[Rooker 07] Martijn N. Rooker & Andreas Birk. *Multi-robot exploration under the constraints of wireless networking.* Control Engineering Practice, vol. 15, no. 4, pages 435–445, 2007. 15, 34

[Rosenblatt 95] Julio Rosenblatt. *DAMN : A Distributed Architecture For Mobile Navigation - Thesis Summary.* In Journal of Experimental and Theoretical Artificial Intelligence, pages 339–360, 1995. 77

[Rothkopf 98] M. H. Rothkopf, A. Pekec & R. M. Harstad. *Computationally Manageable Combinational Auctions.* Management Science, vol. 44, 1998. 128, 129, 130

[Sabbadin 02] Régis Sabbadin. *Graph partitioning techniques for Markov Decision Processes decomposition.* In 15th Eureopean Conference on Artificial Intelligence, pages 670–674, 2002. 105, 106

[Saranli 01] U. Saranli, M. Buehler & D. E. Koditschek. *RHex : A Simple and Highly Mobile Hexapod Robot.* International Journal of Robotic Research, 2001. 9

[Siegwart 05] Roland Siegwart & Illah R. Nourbakhsh. Introduction to autonomous mobile robots. MIT Press, 2005. 6, 9, 24, 76, 77, 89

[Sim 05] R. Sim & N. Roy. *Global A-Optimal Robot Exploration in SLAM.* In International Conference of Robotics and Automation, 2005. 62

[Simhon 98] S. Simhon & G. Dudek. *A global topological map formed by local metric maps.* In International Conference on Intelligent Robots and Systems, volume 3, pages 1708–1714, 1998. 30, 79

[Simmons 95] Reid Simmons & Sven Koenig. *Probabilistic Robot Navigation in Partially Observable Environments.* In International Joint Conference on Artificial Intelligence, pages 1080–1087, 1995. 24, 33

[Simmons 00] Reid Simmons, David Apfelbaum, Wolfram Burgard, Dieter Fox, Mark Moors, Sebastian Thrun & Hakan Younes. *Coordination for Multi-Robot Exploration and Mapping.* In National Conference on Artificial Intelligence, 2000. 14, 15, 33, 34, 39, 53

[Spaan 08] M. Spaan & F. Melo. *Interaction-driven Marcov games for decentralized multiagent panning under uncertainty.* In International Joint Conference on Autonomous Agents and Multiagent Systems, 2008. 63

[Spaan 10] M. T. J. Spaan, N. Goncalves & J. Sequeira. *Multirobot coordination by auctioning POMDPs.* In International Conference on Robotics and Automation, pages 1446–1451, 2010. 131

[Sutton 98] P. Sutton & A. Barto. Reinforcement learning : An introduction. MIT Press, Cambridge, MA, 1998. 56, 59, 100

[Taylor 11] Frederick Winslow Taylor. Principles of scientific management. Harper & brothers, 1911. 5

[Teichteil-Königsbuch 06] Florent Teichteil-Königsbuch & Patrick Fabiani. *Autonomous Search and Rescue Rotorcraft Mission Stochastic Planning with Generic DBNs*. In IFIP AI, pages 483–492, 2006. 54

[Thrun 98] S. Thrun. *Learning Metric-Topological Maps for Indoor Mobile Robot Navigation*. Artificial Intelligence, vol. 99, no. 1, pages 21–71, 1998. 11, 27, 28, 30, 33, 53, 78, 86

[Thrun 05] S. Thrun, W. Burgard & D. Fox. Probabilistic robotics. MIT Press, 2005. 24

[Tovey 05] C. Tovey, M. Lagoudakis, S. Jain & S. Koenig. *The Generation of Bidding Rules for Auction-Based Robot Coordination*. Multi-Robot Systems. From Swarms to Intelligent Automata, vol. 3, pages 3–14, 2005. 34, 125, 127, 128, 129, 131, 132

[Tumer 00] Kagan Tumer & David Wolpert. *Collective Intelligence and Braess' Paradox*. In National Conference on Artificial Intelligence, 2000. 130

[Ulrich 00] Iwan Ulrich & Illah Nourbakhsh. *Appearance-Based Place Recognition for Topological Localization*. In International Conference on Robotics and Automation, volume 2, pages 1023–1029, 2000. 77, 79

[Varakantham 09] P. Varakantham, J. Kwark, M. Taylor, J. Marecki, Scerri P. & M. Tambe. *Exploiting coordination locales in distributed pomdps via social model shaping*. In International Conference on Automated Planning and Scheduling, 2009. 63

[Vilca 12] J.-M. Vilca, L. Adouane & Y. Mezouar. *Robust On-line Obstacle Detection using Data Range for Reactive Navigation*. In 10th International IFAC Symposium on Robot Control (SYROCO'12), 2012. 77

[Vincent 08] R. Vincent, D. Fox, J. Ko, K. Konolige, B. Limketkai, B. Morisset, C. Ortiz, D. Schulz & B. Stewart. *Distributed multirobot exploration, mapping, and task allocation*. Annals of Mathematics and Artificial Intelligence, vol. 52, pages 229–255, 2008. 34, 35, 66

[Volpe 01] R. Volpe, I. Nesnas, T. Estlin, D. Mutz, R. Petras & H. Das. *The CLARAty architecture for robotic autonomy*. In Aerospace Conference, volume 1, pages 121–132, 2001. 71, 72, 73, 75, 79

[Vries 03] S. De Vries & R. V. Vohra. *Combinatorial Auctions : A Survey*. INFORMS Journal on Computing, vol. 15, no. 3, pages 284–309, 2003. 128, 129, 130

[West 01] Douglas B. West. Introduction to graph theory. Prentice Hall, 2001. 20

[Wooldridge 95] M. Wooldridge & N. R. Jennings. *Intelligent agent : Theory and practice*. The Knowledge Engineering Review, vol. 10, pages 115–152, 1995. 40, 45

[Wurman 08] P. R. Wurman, R. D'Andrea & M. Mountz. *Intelligent agent : Theory and practice.* ai magazine, vol. 29, 2008. 23

[Zlot 02] R. Zlot, A. Stentz, M.B. Dias & S. Thayer. *Multi-robot exploration controlled by a market economy.* In International Conference on Robotics and Automation, volume 3, pages 3016–3023, 2002. 33, 34, 39

Stratégie coopérative pour la mise en œuvre d'une flotte de robots mobiles dans un milieu ouvert et encombré

Les travaux réalisés dans le cadre de cette thèse visent à la mise en œuvre d'une flotte de plusieurs robots pour l'exploration d'un environnement ouvert et encombré. L'objectif consiste à proposer des stratégies permettant au groupe de robots d'être autonome dans le calcul des politiques décentralisées et dans la réalisation de leurs tâches respectives. La solution attendue doit permettre à chacun d'évoluer en étant affecté au minimum par une perte de communication. Les zones à visiter doivent alors être allouées entre les robots à chaque fois qu'il est nécessaire et que les robots sont en capacité de communiquer.

Dans un premier temps, il est proposé une architecture basée sur une représentation topologique sémantique de l'environnement de façon à générer des cartes de faible densité. L'objectif est double : alléger les charges de calcul pour la planification en diminuant les données utiles et garantir que les hypothèses mises en œuvre respectent les capacités structurelles des robots dans un environnement réel. La seconde contribution propose une planification approchée sur plusieurs niveaux de façon à résoudre, en cours de mission, une problématique qui s'apparente à un voyageur de commerce stochastique. La solution consiste à décomposer un processus décisionnel de Markov orienté par plusieurs objectifs de façon à s'affranchir d'une limite sur le nombre d'objectifs considérés. Enfin, distribuer cette solution pour une résolution multi-robots nécessite la mise en place de protocoles permettant d'allouer rapidement l'ensemble de tous les objectifs. Sur la base d'une évaluation individuelle des préférences corrélées sur les objectifs, la solution distribuée proposée converge rapidement sur un consensus à travers une succession de ventes aux enchères simultanées.

Les expérimentations réalisées visent à une évaluation statistique de la qualité des solutions produites ainsi que la possibilité de traiter des problématiques de taille réelle, en cours de mission. Elles permettent de valider les approches proposées dans un cadre multi-robots.

Mot-clefs : Intelligence artificielle répartie, Robots mobiles, Markov (Processus de), Décomposition (Méthode mathématique)

Cooperative strategies for a fleet of mobile robots moving in open clustered environment

The work presented in this thesis aims to implement a fleet of robots to explore an open and crowded environment. The objective is to propose strategies allowing the robots to be autonomous in the calculation of decentralized policies and in the implementation of their tasks. The expected solution should allow each robot to act without being affected by any loss of communication. A set of points to visit should be allocated between the robots whenever it is necessary and communication is possible.

As a first step, we proposed an architecture based on a topological representation of the environment to generate maps of low density. The aim is to reduce planification calculation costs by reducing the useful information and ensure that the assumptions respect the structural capacities of the robots in a real environment. The second contribution proposes an aproximate solution that helps in solving, during a mission, a problem that is similar to a stochastic traveling salesman. The solution is to decompose a multi-goal Markov decision process to overcome the limitation of the number of considered goals. Finally, distributing the resolution over several robots requires defining protocols to quickly allocate the set of goals. The proposed solution converges rapidly through successive simultaneous rounds of auctions based on an individual assessment of correlated preferences on the objectives.

The experiments aim to evaluate statistically the quality of the solutions produced and the ability to deal, online, with problems of real size. These experiments are used to validate the proposed approaches in multi-robot frameworks.

Keywords : Distributed artificial intelligence, Mobile robots, Markov process, Decomposition method