Coopération dans les systèmes multi-robots: une contribution au maintien de la connectivité et de l'allocation dynamique des rôles

van Tuan Le

UNIVERSITÉ de CAEN/BASSE-NORMANDIE
U.F.R. de Sciences
ÉCOLE DOCTORALE SIMEM

Thèse

présenté par
Van Tuan Le

et soutenue le 6 octobre 2010
en vue de l'obtention du

Doctorat de l'université de Caen

Spécialité Informatique et Application
(Arrêté du 7 août 2006)

Coopération dans les systèmes multi-robots : Contribution au maintien de la connectivité et à l'allocation dynamique de rôles

Membres du jury

Rapporteurs :	Simon Lacroix	Directeur de Recherche CNRS, LAAS (Toulouse)
	Catherine Tessier	Maître de Recherche, Onera-DCSD (Toulouse)
Examinateurs :	Michel Occello	Professeur des Universités, Université Pierre-Mendes France (Grenoble)
	Olivier Simonin	Maître de Conférences, Université Henri Poincaré (Nancy)
Directeur :	François Bourdon	Professeur des Universités, Université de Caen-Basse Normandie
Encadrants :	Noury Bouraqadi	Maître Assistant, École des Mines de Douai
	Victor Moraru	Professeur, Institut de la Francophonie pour l'Informatique (Ha Noi)
	Serge Stinckwich	Maître de Conférences, Université de Caen-Basse-Normandie

Remerciements

Bien que je soit présenté comme l'unique auteur de ce mémoire, il y a plusieurs autres personnes dont les contributions ont été indispensables à l'accomplissement de cette thèse. Je n'en mentionne ici que quelques uns, faute de place.

J'ai eu beaucoup de chance de travailler avec mes encadrants : Noury Bouraqadi, Victor Moraru, et Serge Stinckwich. Ils se sont complétés à merveille. Cette thèse n'aurait pas été possible sans leur aide et leur patience. Je suis extrêmement reconnaissant pour leurs conseils avisés, leur accompagnement, leur soutien, ainsi que de le temps qu'ils m'ont accordé durant ces quatre années.

Je remercie sincèrement François Bourdon pour son rôle de directeur de thèse.

Je remercie Simon Lacroix et Catherine Tessier qui m'ont fait l'honneur d'être les rapporteurs de ce travail. Ma reconnaissance s'adresse également à Michel Occello et à Olivier Simonin qui ont bien voulu participer à ce jury.

Je remercie l'Agence Universitaire de la Francophonie (AUF) de m'avoir attribué un soutien financier sous la forme d'une bourse de formation à la recherche.

Je tiens à souligner l'appui technique et académique de l'Institut de la Francophonie pour l'Informatique (IFI) à Hanoi et de leurs personnels. Je remercie chaleureusement Ho Tuong Vinh de m'avoir accueilli dans le laboratoire MSI de l'IFI. La convivialité qui règne dans ce laboratoire a facilité mon intégration dans l'équipe recherche. C'était vraiment un honneur de pouvoir travailler dans le même laboratoire et d'être ami avec une personne que j'admire beaucoup : Alexis Drogoul. François Sempé est un autre chercheur de MSI que j'ai fréquenté pendant cette période et que je ne peux pas oublier. Je lui suis reconnaissant pour ses conseils, son soutien et sa par participation à de nombreuses réunions de la thèse.

En plus du soutien de l'IFI/AUF, le département Informatique et Automatique de l'École des Mines de Douai, a également joué un rôle important dans cette recherche en me fournissant un financement supplémentaire. Je remercie toutes les personnes que j'ai pu côtoyer pendant mes séjours à Douai : enseignants-chercheurs, techniciens, secrétaires. Je tiens à remercie Philippe Hasbroucq de m'avoir offert un environnement de recherche excellent où j'ai pu avoir la chance de travailler avec les chercheurs du département. J'ai pu ainsi beaucoup bénéficier de leur précieuses expériences. En particulier, j'ai pu profiter des discussions avec Arnaud Doniec. Je voudrais le remercier spécialement pour m'avoir donné confiance en mes résultats.

J'adresse enfin mes salutations et bénédictions à tous ceux qui m'ont soutenu à tout égards au cours de la réalisation de ce projet. Je tiens à remercier mon épouse, Linh, pour son soutien personnel et sa grande patience à tout moment. Ma mère, et mes frères m'ont donné, comme toujours, leur appui sans équivoque rendant tout remerciement insuffisant.

Ha Noi, le 18 octobre 2010.

Résumé

Face à la complexité des tâches à faire réaliser par un système multi-robots, l'approche la plus communément adoptée consiste à « diviser pour régner ». Il s'agit de décomposer la tâche complexe en sous-tâches, puis chacune de ces sous-tâches en sous-sous-tâches. Ce processus est répété jusqu'à arriver à des tâches élémentaires, qui peuvent être réalisées par des robots individuels. Cette démarche bien que séduisante à prime abord, présente l'inconvénient de ne pas être toujours facile à mettre en œuvre. En effet, dans la majorité des travaux existants, la décomposition d'une tâche complexe en tâches élémentaires est effectuée de manière plutôt ad hoc et liée au système robotique cible. Cela constitue en effet, un frein à la réutilisation du résultat de la décomposition – tant au niveau logiciel qu'au niveau logique – avec d'autres systèmes robotiques.

Dans cette thèse, nous proposons une solution qui permet à n'importe quelle collection de robots hétérogènes de s'organiser en équipes et sous-équipes et ce, en fonction à la fois des exigences de la tâche à réaliser, des robots disponibles et de leurs ressources. Notre approche basée sur la décomposition d'une tâche complexe en rôles, sépare les préoccupations du niveau de conception et du niveau d'implémentation. Ainsi, une même solution logique peut être (ré)utilisée sur des systèmes multi-robots avec des capacités variées. Une fois la tâche décomposée en rôles, nous sommes confrontés à un problème bien connu et encore non résolu : le problème général d'affectation optimale de rôles aux robots de manière efficace. Face à ce problème $\mathcal{NP}$-difficile, nous proposons des heuristiques basées sur le protocole Contract-Net pour affecter les rôles aux robots afin de former des coalitions. Chaque coalition se compose de robots coopérant de manière étroite pour effectuer une tâche unique.

L'affectation de rôles aux robots, ainsi que la coopération de ces derniers requièrt que les robots puissent communiquer de manière fréquente. Or, la connectivité du réseau de robots est un pré-requis de la communication. Nous proposons une solution originale à ce problème basée sur notre concept de « sensibilité à la connectivité ». Il s'agit de doter chaque robot d'une connaissance de la structure du réseau. Nous montrons qu'une connaissance partielle et locale à chaque robot, peut être exploitée pour maintenir la connectivité du réseau de manière distribuée et robuste. Chaque robot peut ainsi planifier localement ses déplacements sans mettre en péril la connectivité du réseau global. En effet, cette connaissance locale que représente la sensibilité à la connectivité peut être exploitée pour déterminer les nœuds (robots) et les connexions critiques du réseau de robots.

Mots-clés :

Intelligence artificielle répartie, Systèmes adaptatifs, Robots – Systèmes de commande

Abstract

Given the complexity of the tasks to be undertaken by a multi-robot system, the approach most commonly adopted is to "divide and conquer". It is to decompose the complex task into subtasks, then each of these sub-tasks into sub-sub-tasks. This process is repeated until we reach basic tasks that can be made by individual robots. This approach, although attractive at first glance, has the disadvantage of not always be easy to implement. Indeed, in the majority of existing work, the decomposition of a complex task into elementary tasks is performed in a rather ad hoc and dependent upon the target robot system. This is indeed a barrier to reuse the result of the decomposition – both software to the logic level with other robotic systems.

In this thesis, we propose a solution that allows any collection of heterogeneous robots to organize themselves into teams and sub-teams and this, according to both the requirements of the task at hand, robots available and resources. Our approach based on the decomposition of a complex task roles, separates the concerns of the design and implementation level. Thus, one logical solution may be (re)used on multi-robot systems with varying abilities. Once the task is decomposed into roles, we are facing a well-known, yet unresolved problem : the general problem of assignment of roles to robots effectively. Faced with this $\mathcal{NP}$-hard problem, we propose heuristics based on the Contract-Net protocol to assign roles to the robots to form coalitions. Each coalition is composed of robots cooperating tightly to perform a single task.

Assigning roles to robots, as well as the cooperation of the latter requires that the robots must be able to communicate frequently. As a result, the network connectivity of robots is a prerequisite for communication. We propose a novel solution to this problem based on our concept of "connectivity awareness". It is about equipping each robot with knowledge of the network structure. We show that a partial and local knowledge in each robot can be exploited for maintaining network connectivity in the robust distributed manner. Each robot can plan its own move locally without jeopardizing the overall network connectivity. Indeed, this local knowledge that is the connectivity awareness can be exploited so that each robot can determine whether the global network connectivity is robust to the its own network connectivity failure or not.

Keywords :

Distributed Artificial Intelligence, Adaptive Systems, Robots – Control Systems

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

1

Sommaire

L'un des objectifs finaux de l'utilisation des robots est de remplacer (ou au moins d'aider) les êtres humains dans la réalisation des certaines tâches. C'est particulièrement souhaitable pour les familles de tâches qualifiées en anglais de 3D pour *Dirty, Dangerous, or Dull*[1] [Takayama et al., 2008]. Il s'agit de tâches « sales » telles que le tri d'ordures, dangereuses comme l'exploration de l'intérieur d'une centrale nucléaire ou encore ennuyeuses tel que la surveillance d'un site. Cependant, force est de constater qu'il reste encore un certain nombre de défis techniques à surmonter avant que les robots puissent réaliser de telles tâches sans l'intervention d'opérateurs humains.

Cette thèse s'inscrit dans le contexte d'un projet de recherche qui vise à proposer un système d'aide à la décision pour l'organisation d'opérations de sauvetage après un désastre dans un environnement urbain. Dans ce cadre, nous nous sommes intéressés à la problématique de *coopération automatique* dans un système multi-robots. Plus concrètement, nous proposons des solutions aux deux questions sous-jacentes, à savoir :

1. Sales, Dangereuses, ou Ennuyeuses.

1. Comment maintenir la connectivité du réseau sans-fil constitué par les robots afin que ces derniers puissent communiquer ?
2. Comment allouer efficacement les tâches aux robots ?

Dans la suite de ce chapitre, nous détaillons le contexte et les motivations de ce travail de thèse. Puis, nous présentons brièvement nos contributions, avant de décrire l'organisation du mémoire.

1.1 Systèmes Multi-Robots

La robotique est une science complexe du fait qu'elle nécessite le concours de plusieurs disciplines comme l'électronique, la mécanique et le génie logiciel. Cette complexité est accentuée avec le passage aux systèmes multi-robots (SMRs) [Liu and Wu, 2001, Siciliano and Khatib, 2008]. Cette section est consacrée à ces systèmes et à la problématique de coopération qui découle de leur utilisation.

1.1.1 De l'homogénéité à l'hétérogénéité

La majorité des premiers travaux sur les systèmes multi-robots s'est intéressée aux systèmes homogènes constitués de robots identiques. Cependant, ces dernières années ont vu une évolution des travaux de la communauté robotique vers les systèmes multi-robots hétérogènes. Il s'agit de systèmes constitués de robots dotés de capacités différentes notamment au niveau physique : caractéristiques mécaniques, mode de locomotion, capteurs, actionneurs ... La citation suivante de Lynne Parker [Parker, 2003] résume bien cette tendance.

> « Les motivations fondamentales de la recherche en systèmes multi-robots sont [2] : 1) la capacité de résoudre des problèmes qui sont intrinsèquement distribués dans l'espace, dans le temps ou dans la fonctionnalité, 2) la capacité de résoudre des problèmes plus rapidement grâce au parallélisme, et 3) la capacité d'augmenter la robustesse des solutions par la redondance. Dans une proportion importante de la recherche sur systèmes multi-robots, les avantages du parallélisme, de la redondance des solutions distribuées dans l'espace et dans le temps sont obtenus par l'utilisation de robots homogènes, qui sont complètement interchangeables. Cependant, un nombre croissant de recherches essaye de répondre aux questions liées à l'utilisation de robots hétérogènes. Ces recherches impliquent généralement un nombre relativement faible de robots – peut-être de l'ordre d'une dizaine de robots ou moins. Même la recherche en robotique homogène est rarement expérimentée avec des équipes de plus de dix à vingt robots.
>
> ...
>
> ...les applications complexes futures qui exigent l'utilisation simultanée de grandes équipes avec plusieurs capteurs, qui ne peuvent pas être groupés sur un seul type de robot. Les robots devraient sans doute être conçus aussi avec des tailles plus petites, ce qui limiterait leur charge utile, ou bien rendrait certains capteurs nécessaires trop cher à dupliquer dans toute une équipe de plus de 100 robots. Cela conduit à la nécessité de permettre à de nombreux robots hétérogènes de travailler ensemble en coopération pour résoudre des applications intéressantes. »

2. Voir [Dias, 2004] pour plus d'avantages et de motivations pour l'utilisation des systèmes multi-robots.

1.1.2 Communication et coopération dans les systèmes multi-robots

Du point de vue général, comme conclut la citation de [Parker, 2003], une des questions centrales qu'on doit traiter dans un système multi-robots, quel que soit le domaine d'application, est comment faire coopérer efficacement les robots pendant une mission, d'une manière ou d'une autre, soit automatiquement soit, éventuellement, par l'intervention d'un opérateur externe. La coopération signifie que les robots doivent communiquer pour échanger des informations et coordonner leurs actions dans le but d'accomplir une mission commune globale [Arrichiello, 2006].

La coopération est le point clé pour exploiter le potentiel des systèmes multi-robots [Gerkey and Matarić, 2002]. La communication est une condition *préalable* et *indispensable* pour n'importe quel algorithme de coopération. La présence d'un canal de communication sûr et de débit suffisant permettrait de mettre en place un mécanisme de coopération sophistiqué et efficace.

La communication entre les robots dans une équipe peut être réalisée implicitement ou explicitement. La communication implicite, typiquement via l'environnement, est généralement accomplie par les actionneurs et les capteurs des robots. Cela limite à la fois la quantité de données transmises et le degré d'abstraction des informations ainsi échangées. Par conséquent, la communication implicite ne convient pas pour des mécanismes sophistiqués de coopération. Il faut recourir à la communication explicite. D'autre part, avec l'avancement des technologies de communication, les robots d'aujourd'hui sont équipés d'interfaces de communication sans fil haut-débit qui leur fournissent un moyen de communication sûr.

1.2 Contexte du travail de thèse

1.2.1 Projet AROUND

Cette thèse s'inscrit dans le cadre du projet AROUND (Autonomous Robots for Observation of Urban Networks after Disasters) [Boucher et al., 2009], dont l'ambition est de construire un *système complet* d'aide à la décision en temps réel pour la gestion des catastrophes naturelles dans les zones urbaines. Pour comprendre les composants différents du système présenté dans la figure 1.1, nous proposons d'analyser une mission de sauvetage typique [Tadokoro et al., 2000].

Le système robotisé AROUND se compose de 2 sous-systèmes :

1. Un niveau local avec le déploiement de robots autonomes mobiles pour aider et à plus long terme pour remplacer les secouristes. Ces robots sont censés être capables de s'auto-organiser pour réaliser les diverses activités de sauvetage comme le recueil d'informations sur les sites urbains sinistrés et pour établir un réseau pour la communication entre les secouristes humains.
2. Un niveau global qui contient un système d'aide à la décision spatiale (SDSS – Spatial Decision Support System).

Les flux de données entre ces niveaux sont bidirectionnels et peuvent fonctionner en mode entièrement automatique ou de manière semi-dirigée. Les données locales recueillies par les robots sont communiquées, analysées et fusionnées

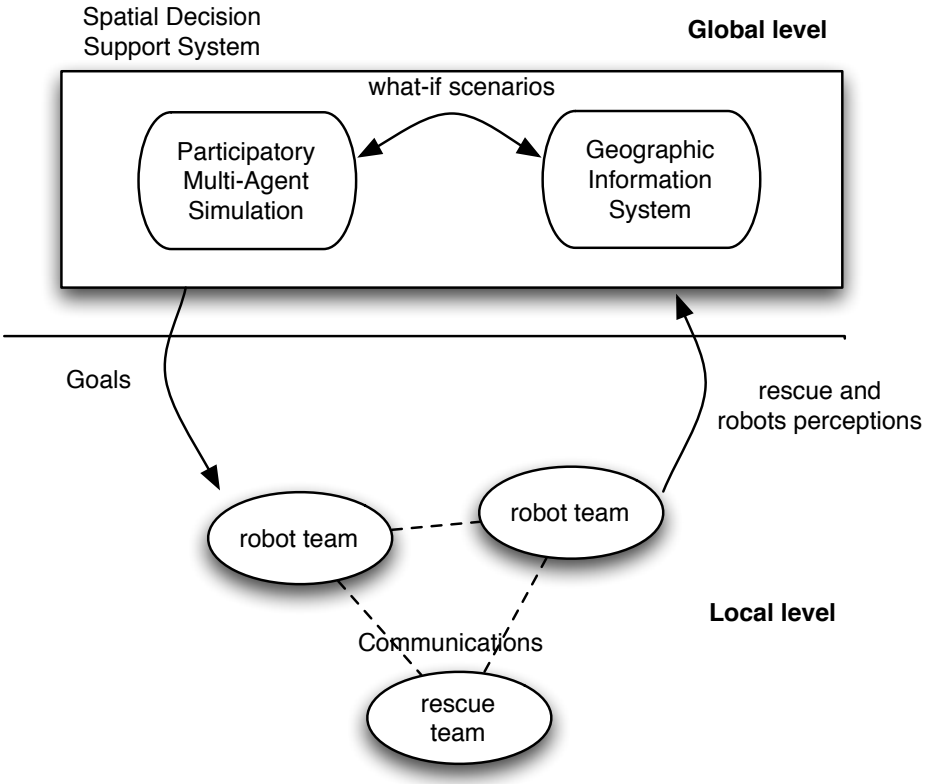


FIGURE 1.1 – *L'architecture du projet AROUND [Boucher et al., 2009]*

à chaque niveau intermédiaire. Ce processus assure l'acheminement des informations significatives aux décideurs en charge de l'organisation des secours. Le niveau global envoie (soit de manière automatique, soit à la demande des décideurs) au niveau intermédiaire des informations sur les objectifs de recherche. Ces informations sont distribuées ensuite aux robots.

Le travail de cette thèse est positionné au niveau local où interviennent les robots. Le recours aux robots est justifié pour minimiser trois paramètres : les délais de localisation des victimes, les risques pour les humains et les coûts financiers.

Après une catastrophe dans un site urbain, la tâche la plus importante est de localiser les victimes dans les plus courts délais possibles. Les opérations de secours traditionnelles impliquent des ressources humaines énormes : des équipes sur place et des coordinateurs à distance. Les équipes de sauvetage sont envoyées sur place pour explorer le lieu. Elles cherchent à localiser les victimes et, en même temps, cartographient la zone.

Les statistiques des taux de survie sur différents sites de tremblement de terre montrent qu'il est souhaitable de réaliser les sauvetages dans les 3 jours qui suivent le séisme. En effet, le taux de survie des victimes diminue sévèrement après 72 heures (appelé les 72 heures en or) [Tadokoro, 2005]. Dès lors, il est impératif d'avoir de nombreuses équipes qui collaborent pour explorer en parallèle les lieux du désastre.

Par ailleurs, une mission de sauvetage peut comporter des risques pour les secouristes humains. Par exemple, un séisme peut avoir des répliques et des pans de bâtiments endommagés peuvent s'écrouler. Ainsi, le remplacement des secouristes par les robots mobiles et autonomes permettrait de sauver des vies

sans risquer celles d'autres humains. Enfin, le dernier argument pour l'utilisation d'une flotte de robots pour les missions de ce type est financier. Le recours aux robots coûterait moins cher que de recruter, entraîner, acheminer et payer des secouristes humains.

1.2.2 Rôle des robots dans une mission de sauvetage – le cas de la compétition RoboCup Rescue

Afin de mieux illustrer le rôle que peuvent jouer les robots dans une mission telle que celles visées par le projet AROUND, nous nous appuyons sur ce qui se fait dans le cadre de la compétition RoboCup Rescue. La RoboCup™ [3] est une initiative internationale de recherche et d'éducation. Il s'agit d'une compétition visant à encourager la recherche dans le domaine de l'IA et de la robotique intelligente sur un problème standard où une vaste gamme de technologies peuvent être intégrées. La RoboCup regroupe plusieurs compétitions dont la RoboCup Rescue [4], qui a pour l'objectif de promouvoir la recherche et le développement dans l'utilisation des robots dans les différentes activités de sauvetage en environnement urbain (Urban Search And Rescue, USAR). La mission suivante est un exemple de ce qui doit être réalisé par les robots.

> « Un bâtiment s'est partiellement effondré en raison d'un tremblement de terre.
>
> Le commandant en charge des opérations de sauvetage dans la zone sinistrée, craignant un effondrement secondaire, a demandé aux équipes de robots de chercher immédiatement des victimes à l'intérieur du bâtiment.
>
> La mission assignée aux robots et à leurs opérateurs est de trouver des victimes, déterminer leur situation, leur état et leur emplacement et ensuite faire un rapport sous la forme d'un plan annoté de l'édifice et d'une feuille avec les informations sur les victimes.
>
> La section près de l'entrée de l'immeuble semble relativement intacte tandis que l'intérieur de la structure présente des degrés d'effondrement plus en plus éléve. Les robots doivent négocier les régions faiblement endommagées avant d'atteindre des obstacles plus difficiles et de gravats.

Comme la tendance actuelle de la robotique, la recherche dans le cadre de cette compétition s'appuie sur la coopération entre des robots hétérogènes dans un environnement hautement dynamique. Cela est illustré au travers du descriptif de la RoboCup Rescue.

> « Lorsqu'une catastrophe se produit, il faut minimiser les risques pour le personnel de recherche et de sauvetage, tout en augmentant les taux de survie des victimes, en déployant des équipes de robots permettant de :
>
> - négocier de manière autonome des compromis au travers des structures effondrées ;
> - trouver des victimes et déterminer leur état ;

3. http://www.robocup.org/
4. http://robotarenas.nist.gov/competitions.htm

- cartographier leurs emplacements ;
- acheminer la nourriture et les communications ;
- identifier des dangers ;
- placer des capteurs (acoustiques, thermiques, détecteurs de matières dangereuses, sismiques, etc.) ;
- fournir des étaiements structurels.

Idéalement, les robots devraient être capables de s'occuper de toutes les tâches définies ci-dessus. Cependant, cette vision est encore loin de l'état de l'art de la robotique. Aussi, les compétiteurs se concentrent pour le moment sur les trois premières tâches.

1.2.3 Hypothèses sur les capacités physiques des robots

Dans le cadre du projet AROUND, les robots sont censés aider voire même remplacer les humains dans les opérations de sauvetage. Dès lors, les robots sont supposés être autonomes dans la réalisation des différentes tâches qui leur sont assignées. Dans la suite de ce mémoire, le terme robot dénote une entité autonome mobile dotée de capacités de calcul.

Une hypothèse importante est que les robots dans le système sont hétérogènes en termes de capacités physiques et décisionnelles. En effet, de la même manière que différentes ONG [5] mettent actuellement en commun leur personnel, nous pensons que différents partenaires feront intervenir leurs robots, qui seront forcément différents. De plus, un même organisme peut se procurer des robots avec des caractéristiques différentes. Que ce soit pour réaliser des tâches différentes (exemple : robot serpent vs. robot araignée) ou bien tout simplement du fait de l'achat de générations/modèles différents d'un même robot.

En ce qui concerne la capacité de communication, nous faisons l'hypothèse que tous les robots sont équipés d'interfaces de communication. De plus, des robots doivent pouvoir automatiquement et dynamiquement mettre en place un réseau ad hoc.

1.3 Problématique

Nous présentons dans cette section la problématique abordée dans la thèse. Pour cela, nous partons d'un scénario applicatif lié à la robotique de sauvetage. Avec une démarche ascendante, nous déterminons les problèmes liés au scénario que nous utilisons pour introduire et justifier la problématique au cœur de cette thèse.

1.3.1 Scénario de sauvetage et les problèmes traités dans la thèse

Dans le cadre du projet AROUND et en nous inspirant de la RoboCup Rescue, nous définissons le scénario applicatif. Il servira de support du travail présenté dans cette thèse.

5. Organisations Non-Gouvernementales.

Après une catastrophe naturelle qui a eu lieu dans une zone urbaine, une flotte de robots hétérogènes (avec différentes capacités matérielles) est déployée sur le site pour participer aux opérations de secours. Parmi les robots disponibles, certains sont plus appropriés pour assister les victimes, alors que d'autres sont plus adaptés à localiser. Avant qu'une victime soit trouvée, les robots doivent s'organiser afin d'explorer une zone sinistrée. Dès qu'une victime est localisée, les robots s'organisent automatiquement pour prendre en charge la victime de manière efficace.

Il y a deux (sous-)problèmes principaux dans le scénario ci-dessus :

L'exploration est un problème commun à plusieurs applications robotiques. Du point de vue algorithmique, l'exploration consiste en un algorithme de coordination d'une flotte de robots autonomes mobiles pour localiser des points d'intérêt dans un territoire fini, inconnu *a priori*. Les robots construisent ainsi de manière collaborative une carte des zones visitées. Un mécanisme de coopération efficace doit minimiser les chevauchements entre les zones explorées par les différents robots. Il s'agit de trouver une solution qui satisfasse au mieux deux contraintes contradictoires. D'une part, les robots doivent s'éloigner les uns des autres autant que possible afin d'accélérer l'exploration. D'autre part, ils doivent rester en contact les uns avec les autres pour échanger les données collectées (fragments de cartes) et coordonner leurs déplacements.

L'assistance aux victimes requiert une stratégie de regroupement automatique des robots appropriés en équipes/groupes en fonction du besoin d'assistance. Un sous-ensemble de robots doit être sélectionné pour s'occuper de chaque nouvelle victime trouvée. Ces robots doivent être choisis sur la base de leur disponibilité et de leurs capacités. Par exemple, si la victime est coincée, il faut des robots capables de soulever/dégager les décombres et d'autres pour déplacer la victime. Si l'environnement est enfumé, il faut un robot capable de maintenir un masque à oxygène sur le visage de la victime.

Le point commun qui relie les deux problèmes d'exploration et d'assistance dans le scénario ci-dessus est la communication comme une *condition préalable*. La communication est indispensable pour la coopération entre les robots. Dans l'exploration, la communication sert à partager les données collectées par les robots dans l'équipe. Puis, quand les robots détectent un blessé, ils doivent communiquer pour coordonner leurs actions afin d'assister la victime en question. D'où le besoin d'un canal de communication fiable tout au long de la mission. C'est le premier problème abordé dans la thèse.

Le deuxième problème qui nous concerne est le problème de formation de coalitions. En effet, si l'on prend l'exemple de l'assistance aux victimes, différentes victimes peuvent être localisées pendant l'exploration. Différentes coalitions (équipes) de robots devront être formées pour secourir les différents blessés. Chaque coalition regroupera des robots ayant des caractéristiques complémentaires étant donné le contexte de la victime secourue. Les deux sections suivantes donnent plus de détails sur ces deux problèmes et les questions associées.

1.3.2 Maintien de la connectivité d'un réseau de communication robotique

Le problème du maintien de la connectivité consiste à assurer l'existence d'un canal de communication fiable tout au long de la mission. La difficulté réside dans le fait que le théâtre des opérations est une zone dévastée. Nous ne pouvons pas pré-supposer la disponibilité d'une infrastructure de communication opérationnelle et compatible avec les robots. Aussi, les robots doivent constituer et maintenir eux-même un réseau sans fil mobile.

Le problème de maintien d'un réseau de communication entre des équipements mobiles est connu dans la littérature sous l'acronyme MANET pour *Mobile Ad Hoc NETwork* [Perkins, 2001]. De multiples propositions d'algorithmes de routage ont été faites sur le sujet, mais elles font toutes l'hypothèse que les déplacements des nœuds du réseau sont aléatoires, i.e. non-contrôlés par les équipements [Abolhasan et al., 2003, Royer and Toh., 1999]. Or, dans le cas d'un système multi-robots, nous sommes face à des entités autonomes qui décident de leurs déplacements. Nous nous intéressons donc à la prise en compte de l'autonomie dans le traitement du problème du maintien de la connectivité d'un réseau de robots.

1.3.3 Formation dynamique et automatique de coalitions de robots

Le second problème sur lequel nous nous penchons dans cette thèse est connu sous le nom "formation de coalitions" [Horling and Lesser, 2005, Shehory and Kraus, 1998, Parker and Tang, 2006, Vig and Adams, 2007]. « Les coalitions sont, en général, orientées vers un but et sont de courte durée. Elles sont formées avec un but à l'esprit et se dissolvent quand ce besoin n'existe plus, la coalition cesse pour ses propres fins conçus, ou si une masse critique est perdue comme les agents qui en partent [Horling and Lesser, 2005] ». Nous nous intéressons donc à la question de l'organisation automatique (sans intervention humaine) de robots hétérogènes en équipes en fonction des tâches à réaliser, des robots disponibles et de leurs ressources.

Le problème peut être reformulé au travers de la question suivante : comment *sélectionner automatiquement* un groupe de robots parmi une flotte, pour contribuer à la réalisation d'une tâche donnée ? Il s'agit du problème d'allocation de tâches, bien connu aussi bien dans le monde de la robotique [Gerkey and Matarić, 2004a] que dans celui des systèmes multi-agents [Shehory and Kraus, 1998]. Ce problème a différentes gradations de difficulté suivant :

- le type de tâches : simples pouvant être réalisées par un seul robot, ou complexes, nécessitant la coopération de plusieurs robots,
- le type de robots : capables de traiter une ou plusieurs tâches en même temps,
- la stratégie d'optimisation par rapport à la configuration du système : localement à un instant donné, ou bien globalement en anticipant les évolutions futures du système.

Le problème est relativement simple à résoudre quand il s'agit d'attribuer 1 tâche à un 1 robot, avec des robots capables de traiter 1 seule tâche à la fois. Les difficultés surgissent quand il s'agit de tâches complexes avec des robots capables de contribuer à plusieurs tâches en même temps. Le problème est en-

core plus difficile quand on a affaire à des robots hétérogènes avec une stratégie d'optimisation globale.

1.4 Résumé des contributions

Les contributions de cette thèse portent naturellement sur les deux facettes des systèmes multi-robots introduites dans la section 1.3. Nous proposons donc des solutions pour, d'une part, le maintien de la connectivité, et d'autre part, pour la formation de coalitions. Comme décrit plus bas, ces propositions ont pour caractéristique d'être à la fois distribuées et génériques. Elles sont distribuées car l'effort en terme de calcul et de communication est réparti sur les robots. Elles sont génériques car elle ne sont pas liées à notre contexte applicatif, ni à un type de robot particulier.

1.4.1 Maintien décentralisé de la connectivité et robustesse du réseau

La première contribution majeure de cette thèse est une solution distribuée au problème du maintien de la connectivité. Cette proposition peut être divisée en deux parties. D'abord, les robots sont rendus *sensibles à la connectivité* du réseau qu'ils forment. Cela se traduit par l'acquisition par chaque robot d'une connaissance *partielle* sur les liens de communication qu'il peut avoir avec les autres robots au travers de ses voisins. La deuxième partie de notre solution consiste à utiliser cette sensibilité pour maintenir la connectivité. Nous démontrons que la connaissance partielle dont dispose chaque robot est suffisante pour qu'il puisse décider de ses déplacements sans rompre la connectivité globale du réseau.

Cette notion originale qu'est la sensibilité à la connectivité s'avère intéressante à plus d'un titre. Au delà de son utilisation pour maintenir la connectivité décrite plus haut, elle peut être aussi exploitée pour évaluer la robustesse du réseau formé par les robots. Ainsi, nous proposons un algorithme efficace pour évaluer la robustesse d'un réseau en détectant les éventuels nœuds (les robots) et liens de communication critiques, dont la disparition provoquerait la scission du réseau. À noter que ce dernier résultat peut être exploité non-seulement pour les systèmes multi-robots, mais pour tout autre réseau comme, par exemple, les réseaux de capteurs.

1.4.2 Coopération à base de rôles et formation des coalitions

Notre deuxième contribution majeure est une solution, sur la base du concept de rôle, au problème de coopération de robots. Nous définissons un rôle comme la description d'un comportement bien identifié. Plusieurs rôles sont en général nécessaires pour réaliser une tâche donnée. Nous proposons d'utiliser les rôles pour décrire les coalitions nécessaires à la réalisation des différentes tâches possibles dans une mission donnée. Cette description est de haut-niveau dans le sens où elle ne fait pas d'hypothèse sur les robots concrets qui vont former les coalitions et effectuer les tâches. Ainsi, elle peut être déployée *sans modification* sur différents systèmes avec des robots différents en nombre ou en caractéristiques. La seule condition est de disposer de robots ayant les ressources suffisantes et les compétences requises pour endosser les rôles.

En ce qui concerne l'affectation des rôles aux robots, notre solution est originale dans le sens où nous faisons l'hypothèse que les robots sont capables de jouer plusieurs rôles en même temps. Pour résoudre ce problème d'allocation, nous donnons d'abord une formalisation abstraite et générale du problème de formation de coalitions. Cette formalisation révèle que nous sommes face à un problème de type COP (Constraint Optimal Processing). Étant dans un contexte distribué, nous pouvons recourir à un algorithme DCOP (Distributed COP) [Modi et al., 2006, Petcu and Faltings, 2005, Petcu and Faltings, 2006, Ottens and Faltings, 2009] pour trouver des solutions optimales. Malheureusement, les problèmes COP sont $\mathcal{NP}$-difficiles en général. En outre, le nombre de messages échangés lors de la recherche d'une solution avec une approche DCOP croît généralement exponentiellement avec le nombre de robots. Nous proposons donc une alternative basée sur le protocole Contract-Net [Smith, 1980] pour former des coalitions en ligne (i.e. au fur et à mesure de l'apparition des tâches). Ce choix est un compromis raisonnable entre la qualité de la solution, le temps pour la trouver et le nombre de messages échangés par les robots.

1.5 Organisation du mémoire

Le présent mémoire se divise en quatre chapitres principaux.

Le chapitre 2 est dédié à un état de l'art sur le maintien de la connectivité des réseaux de robots. Notre contribution dans ce chapitre est la formalisation abstraite du problème général de maintien de la connectivité. Partant de cette formalisation, nous proposons une taxonomie des approches existantes pour le maintien de la connectivité.

Nous présentons ensuite, dans le chapitre 3, un état de l'art de la coopération dans les systèmes multi-robots. Dans ce cadre, nous discutons du lien entre les concepts de tâche et de rôle. Nous montrons ainsi les limites de la décomposition par tâche de mission robotique et de l'intérêt d'utiliser les rôles. Néanmoins, l'introduction des rôles ne résout pas pour autant le problème de déploiement d'une mission sur une flotte de robots. Le problème d'allocation de rôles est identique à celui de l'allocation de tâches. Les paramètres identifiés dans la taxonomie de Gerkey et Mataric [Gerkey and Matarić, 2004b] se retrouvent dans les deux problèmes. Nous montrons ainsi que le problème traité dans cette thèse est le cas le plus difficile. Il s'agit du cas de missions avec des tâches nécessitant la coopération de plusieurs robots hétérogènes capables d'endosser plusieurs rôles simultanément.

Dans le chapitre 4, nous présentons notre solution pour le maintien de la connectivité dans un réseau formé par une flotte de robots. Nous décrivons notre algorithme pour rendre les robots sensibles à la connectivité du réseau. Puis nous montrons comment utiliser cette sensibilité pour la vérification décentralisée et efficace de la connectivité robuste, ainsi que pour le maintien de la connectivité.

Nous présentons ensuite, dans le chapitre 5, notre solution pour la coopération dans un système multi-robots. Sur la base du concept de rôle, nous montrons comment décrire des organisations génériques, déployables sur des systèmes avec des robots différents en nombre et en caractéristiques. Nous présentons deux approches pour affecter automatiquement ces rôles aux robots tout en garantissant un certain degré d'optimalité de l'allocation. La première est une

approche descendante où les rôles sont assignés a priori aux robots. La seconde est une approche ascendante où les rôles sont assignés de manière paresseuse au fur et à mesure de l'apparition des tâches.

Le chapitre 6 est consacré à la validation expérimentale des résultats validés théoriquement dans les chapitres précédents. Nous commençons par décrire le simulateur discret multi-robots que nous avons développé. Ensuite, nous illustrons son utilisation pour les différentes solutions que nous proposons. Nous obtenons ainsi des mesures expérimentales qui sont meilleures que nos résultats théoriques. En effet, nos résultats théoriques ont porté sur le pire cas, alors que les résultats expérimentaux représentent une moyenne et sont donc plus représentatifs des cas généraux.

Le mémoire se conclut par une synthèse des travaux que nous avons réalisés dans le cadre de cette thèse. Nous dressons le bilan de nos propositions et nous listons quelques pistes pour des perspectives qui nous semblent intéressantes.

Première partie

État de l'art

État de l'art sur le maintien de la connectivité

2

Sommaire

L'existence d'un réseau de communication pendant une mission est un prérequis pour de nombreuses applications multi-robots en général, et plus particulièrement pour le problème de coopération entre les robots dont nous avons parlé précédemment dans l'introduction. On fait l'hypothèse (cf. section 1.2.3, page 7) ici que les robots disposent de moyens de communication sans fils (e.g. Wi-Fi). On va alors s'intéresser à la problématique de maintien de la connectivité entre eux.

Afin de réaliser un état de l'art pour cette problématique, nous examinons un certain nombre de travaux similaires puis nous proposons une taxonomie pour les comparer.

2.1 De la communication à base d'un réseau mobile ad hoc sans fil au maintien de la connectivité dans les systèmes multi-robots

Dans les systèmes multi-robots, les robots ont besoin de communiquer pour coopérer effectivement. Afin d'atteindre un degré élevé de flexibilité et d'autonomie, la communication entre les robots dans un système multi-robots devrait reposer sur des technologies de communication sans fil. De plus, la technologie employée doit permettre aux robots de s'organiser automatiquement en réseau, afin d'être opérationnels dès qu'ils sont mis en place sans aucune administration centralisée. En plus de l'auto-configuration du réseau, la technologie de communication doit être capable de s'adapter aux mobilités des robots pendant leur mission. Un réseau avec de telles caractéristiques est connu sous le nom réseau ad hoc mobile (MANET – Mobile Ad hoc NETwork, en anglais) [Perkins, 2001]. Ces caractéristiques rendent les MANETs extrêmement flexibles, faciles à déployer. Pour cette raison, l'utilisation des MANETs pour la communication entre les robots dans des applications de sauvetage, où on ne peut pas raisonnablement exiger l'existence d'une infrastructure de communication, est extrêmement adéquate.

Dans un MANET, chaque nœud a la possibilité de se déplacer de façon autonome. Quand des nœuds sont en déplacement, *les services de transmission de données* sont capables de s'adapter à ces changements. À cause de cette hypothèse de mobilité, et du fait qu'il n'y a pas d'administration centralisée dans les MANETs, un réseau de ce type peut se retrouver assez souvent partitionné. Cette situation n'est pas souhaitée dans plusieurs applications multi-robots où l'existence permanente d'un canal de communication entre tous les robots est nécessaire, comme dans l'exploration multi-robots [Diosdado, 2006, Rooker and Birk, 2005, Burgard et al., 2005, Simmons et al., 2000, Rooker and Birk, 2007]. Nous nous trouvons devant un défi : comment coordonner les robots dans leur déplacement de sorte que la connectivité du réseau ne soit pas compromise ? Ce problème a été identifié récemment dans la communauté robotique et fait l'objet de nombreux travaux.

Avant de donner une définition plus rigoureuse du problème, nous proposons de spécifier la signification du terme « maintien de la connectivité du réseau » dans un système multi-robots. D'une manière informelle, nous souhaitons proposer un algorithme qui permet aux robots de :

1. vérifier *localement* si le robot est bien connecté aux autres robots dans le réseau, et
2. s'assurer qu'un *éventuel déplacement*, ne causera aucune déconnexion dans le réseau. Si c'est le cas, le robot peut planifier et réaliser un déplacement en gardant en même temps la connectivité.

Intuitivement, on peut penser que les algorithmes de routage et de contrôle de la topologie proposés dans les MANETs pourraient être adaptés pour assurer les objectifs du maintien de la connectivité. Examinons plus précisément ces deux

problématiques pour comprendre leur relation avec le problème du maintien de la connectivité.

2.1.1 Routage dans les MANETs

La communication entre les nœuds dans un MANET est réalisée par le moyen d'interfaces sans fil avec des technologies différentes (par exemple, WiMAX, Wi-Fi, Bluetooth, etc. voir la figure 2.1). Les interfaces sans fil actuellement utilisées dans la plupart des MANETs en général, et dans les systèmes multi-robots en particulier, sont très limitées en terme de zone de couverture radio. Pour contourner cette limite, un nœud « ordinaire » dans un MANET est vu aussi comme un routeur qui fait suivre les données depuis et vers ses voisins. Grâce à cela, la zone de communication de l'ensemble du réseau est élargie considérablement.

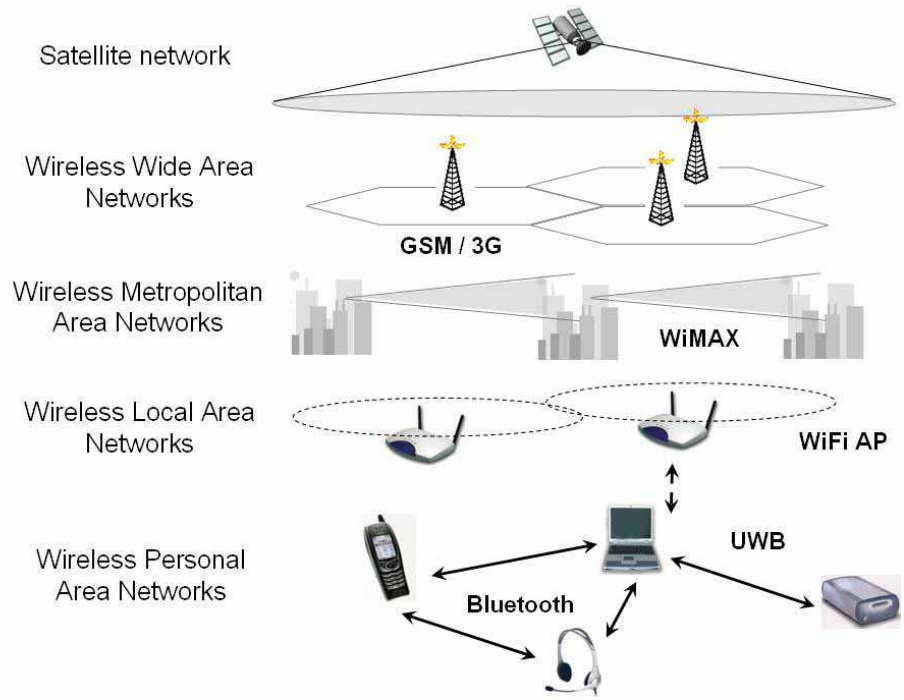


FIGURE 2.1 – *Un exemple d'interfaces de communication et les zones de couverture correspondantes [Bin and Choon, 2006]*

Cependant, cela fait apparaître un nouveau défi de recherche : les protocoles de routage dans les MANETs. Il s'agit de rédéfinir comment relayer correctement les données et comment faire le choix d'un nœud pour faire suivre les données. La conception d'un protocole de routage pour les MANETs est extrêmement difficile par rapport aux réseaux « conventionnels » (filaires). Parmi les difficultés que l'on peut rencontrer, citons : la dynamique de la topologie dans les MANET, le contrôle décentralisé, la fluctuation fréquente du signal, la gestion de l'énergie, etc. Malgré de nombreuses propositions existantes dans la littérature, la recherche sur le routage dans les MANETs reste toujours un domaine actif. Les auteurs dans [Abolhasan et al., 2003, Royer and Toh., 1999] fournissent une taxonomie et une comparaison des principaux protocoles actuels. Cette taxonomie repose sur les critères suivants :

Stratégie adaptative. Selon la stratégie adaptative, on peut classer les protocoles pour les MANETs en trois catégories. La première regroupe ce qui est généralement appelé « protocole pro-actif ». Ces protocoles tentent de maintenir les informations de routage avant qu'une requête de routage ne leur parvienne. Une deuxième approche, utilise des « protocoles réactifs » découvrent la route uniquement si on a besoin de relayer des données. Les protocoles du premier type permettent une meilleure performance, surtout une faible latence, mais ils consomment de la bande passante et de l'énergie. L'inconvénient des protocoles réactifs est une latence un peu plus élevée avant l'envoi de données. Une troisième catégorie, dite hybride, essaie de combiner les approches de deux types de protocoles précédents.

Routage plat ou hiérarchique. Dans les protocoles de routage plats tous les nœuds du réseau ont le même rôle. Le réseau n'est pas structuré. Tandis que dans les protocoles de routage hiérarchique ou en cluster, des nœuds différents peuvent jouer des rôles différents. Le fonctionnement des protocoles hiérarchiques peut être divisé en quelques phases distinctes. La première phase essaie de structurer le réseau en clusters. Au sein de chaque cluster il y a un nœud (appelé *tête du cluster*) qui joue le rôle de maître, les autres nœud jouant un des autres rôles : passerelle ou nœud ordinaire. Les protocoles hiérarchiques de routage sont supposés avoir une meilleure performance et être évolutifs, mais ils demandent des efforts pour la formation des clusters et pour leur maintien.

Routage basé sur la position géographique. Ce paradigme de routage a le grand avantage de ne pas avoir à maintenir une table de routage et d'éviter d'inonder le réseau grâce à la connaissance de la direction à suivre pour atteindre la destination. En outre, ce protocole ne souffre pas de la surcharge de communication liée aux informations de contrôle (pour établir et maintenir la table de routage). Néanmoins, des informations sur la position géographique des nœuds sont nécessaires. On a besoin d'un matériel supplémentaire comme un GPS [1] pour la localisation des nœuds.

Stratégie de sélection de routage. La manière la plus usuelle pour trouver des routes dans les protocoles de routage pour les MANETs est de choisir le chemin le plus court en terme de nombre de nœuds intermédiaires traversés. Le chemin de routage pourrait être choisi en se basant aussi sur d'autres critères. Ces critères pourraient être combinés en une valeur unique par une fonction d'utilité qui évalue les différents facteurs tels que la stabilité du lien, la puissance du signal, etc.

Un protocole de routage dans un MANET est donc en charge de trouver, de manière active ou pro-active, une route pour acheminer les données d'un nœud source vers un nœud cible *à condition que cette route existe*. Si une telle route ne peut pas être trouvée, les protocoles de routage n'ont aucune influence sur la mobilité des robots afin de rétablir cette route. À partir de l'étude de ces travaux, nous concluons que le routage est différent de la problématique du maintien de la connectivité. L'objectif du dernier est d'assurer qu'une telle route entre deux nœuds du réseau existe à chaque instant.

1. Global Positioning System

2.1.2 Contrôle de la topologie

Dans le contexte des réseaux sans fil ad hoc et/ou de capteurs, le contrôle de la topologie consiste à allumer ou éteindre certains nœuds du réseau sans compromettre la connectivité globale afin d'économiser de l'énergie pour prolonger la durée de vie du réseau [Rajaraman, 2002].

Les interfaces sans fil peuvent se trouver généralement dans quatre états : (1) émission, (2) réception, (3) arrêt, et (4) veille. Normalement, la consommation d'énergie est égale en mode de réception et en mode arrêt. L'état d'émission et celui de veille consomment le plus et le moins d'énergie, respectivement. L'idée de base pour économiser l'énergie consiste à mettre les interfaces en hibernation, autant que possible quand il n'y a pas de données à transmettre ou à recevoir, tout en minimisant la perte de paquets et la latence de communication [Xu et al., 2001, Katz and Dao., 2001, Awerbuch and Peleg, 1992].

Autrement dit, étant donné une configuration initiale de réseau, le problème du contrôle de la topologie consiste à trouver l'ensemble minimal de connexions à garder. Les connexions qui n'appartiennent pas à cet ensemble peuvent être supprimées (désactivées), de sorte que la connectivité globale du réseau ne soit pas affectée. Donc, dans une certaine mesure, un algorithme de contrôle de la topologie peut être utilisé pour le but 1) de la section 2.1 (page 17) du problème du maintien de la connectivité. Cependant, ces algorithmes de contrôle de la topologie échouent pour le point 2). Ils ne permettent pas aux robots d'identifier si leurs déplacements peuvent se faire sans déconnexion du réseau. Par conséquent, nous avons besoin de réfléchir à une solution supplémentaire.

2.2 Modélisation de la connectivité dans les réseaux sans fil

La section précédente nous a permis de voir que ni le routage ni le contrôle de la topologie dans les systèmes ad hoc ne proposent une solution adéquate pour le problème du maintien de la connectivité dans les systèmes multi-robots.

Les protocoles de routage, ainsi que les protocoles de contrôle de la topologie font des hypothèses sur la mobilité des nœuds et sur la liberté de se déplacer sans contraintes : les nœuds peuvent rejoindre ou quitter le réseau d'une manière arbitraire. Cette hypothèse rend ces protocoles inutilisables pour le but du maintien de la connectivité dans les applications qui demandent une connectivité permanente.

Cette section établit un cadre formel pour nos discussions à venir sur le problème du maintien de la connectivité traité dans notre travail.

2.2.1 Modèle mathématique de la communication

Nous modélisons un système multi-robots connecté en réseau par un graphe *non-orienté* $G = (V, E)$, où V est l'ensemble des robots dans le réseau, $E = V \times V$. Il y a une arête $e = \{u, v\} \in E$ si et seulement si u et v peuvent *communiquer mutuellement*, c'est-à-dire le lien entre l'eux est bidirectionnel. Dans ce cas, nous

disons que u et v sont des voisins et l'arête est aussi définie comme un *lien de communication* entre les deux robots.

Soit $|V| = n$ le nombre de robots dans le réseau. $deg(v)$ est appelé le degré d'un nœud v qui est défini par le nombre de ses voisins (c'est-à-dire le nombre des liens dont ce nœud est une extrémité).

On notera que le graphe est dynamique parce qu'il peut évoluer à travers le temps grâce à la mobilité des nœuds[2]. Le figure 2.2 donne un exemple d'un tel graphe.

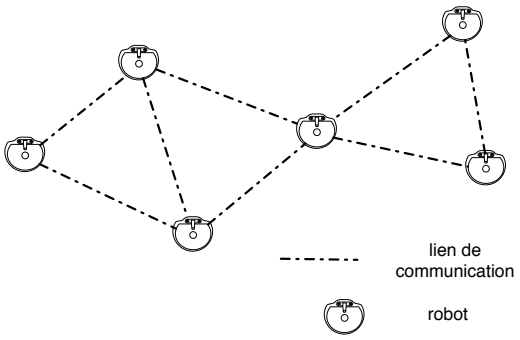


FIGURE 2.2 – *Un exemple du graphe de communication dans un système des robots en réseau*

Définition 2.1 (Chemin de communication) *Dans le graphe G, une séquence de nœuds sans cycle prenant part au processus de la transmission des données à partir de u à v est appelé un* chemin de communication $p(u, v)$ *(ou simplement p s'il n'y a pas de confusion).*

La représentation sous forme d'un vecteur est utilisée pour définir un chemin $p(v_1, v_n) = (v_1, v_2, \ldots, v_n)$ avec $\{v_i, v_{i+1}\} \in E, \forall i = \overline{1, n-1}$.

Étant donné un nœud k et un chemin p, nous écrivons $k \in p$ ou $k \notin p$ pour dire que le nœud k appartient ou non au chemin p. Par définition, toute arête $e \in E$ est un chemin de communication.

Définition 2.2 (Cycle ou Circuit) *Un chemin de communication fermé qui commence et se termine à v est appelé un cycle ou un circuit (de communication), il est noté* $C(v)$.

2.2.2 Connectivité simple

Définition 2.3 (Graphe connexe) *Un graphe G est dit connexe si et seulement si* $\forall u, v \in V$, *il existe un chemin* $p(u, v)$.

Selon cette définition, la façon la plus triviale pour déterminer la connectivité d'un graphe est de vérifier si, pour chaque nœud dans le réseau, il existe un chemin de communication entre lui et tous les autres nœuds. Évidement, cette solution naïve est très coûteuse en terme de calcul. Cependant, nous avons des propriétés basées sur l'arbre couvrant et le graphe Laplacien qui donnent les solutions les plus efficaces pour la détermination de la connectivité.

2. Le terme *nœud* est utilisé pour indiquer un robot dans ce document. Les termes nœud et robot sont utilisés de manière interchangeable.

Arbre couvrant

Définition 2.4 (Arbre couvrant) *Un graphe déduit de G, $G^* = (V, E^*)$, où $E^* \subseteq E$, est un arbre couvrant, noté $\mathcal{ST}$ [3], si les conditions suivantes s'appliquent :*
- *il n'y a pas de circuit dans G^*.*
- *$|E^*| = n - 1$.*

Corollaire 2.1 *Le graphe G est connexe si et seulement s'il existe un $\mathcal{ST}$ déduit de G.*

On peut toujours trouver un $\mathcal{ST}$ dans un graphe connexe. Si on arrive à construire un $\mathcal{ST}$ pour un graphe quelconque, alors il est connexe. Un $\mathcal{ST}$ est un graphe connexe minimal en terme de nombre d'arêtes. Le corollaire 2.1 donne un sens pratique de cette approche dans le maintien de la connectivité : au lieu de prouver qu'il existe un chemin de communication entre toutes les paires d'éléments dans E, on peut simplement construire un $\mathcal{ST}$. Plusieurs travaux sur le maintien de la connectivité exploitent cette propriété des graphes connexes. Ces algorithmes essayent de construire et de maintenir un arbre couvrant [4] dans le réseau.

Le graphe Laplacien et son spectre

Définition 2.5 (Graphe Laplacien) *Étant donné un graphe $G = (V, E)$, la matrice laplacienne du graphe G, $L = (l_{i,j})_{n \times n}$ est définie par :*

$$l_{i,j} = \begin{cases} deg(v_i) & si\ i = j \\ -1 & si\ i \neq j\ et\ (v_i, v_j) \in E \\ 0 & sinon \end{cases} \tag{2.1}$$

Définition 2.6 *Soit L une matrice $n \times n$. On dit que λ est la valeur propre de L si il existe un vecteur de R^n U non nul tel que $LU = \lambda U$. On dit alors que U est le vecteur propre de L associé à la valeur propre λ.*

Pour trouver les valeurs propres de la matrice L, il est nécessaire de résoudre l'équation 2.2.

$$\Delta_n(L) = \begin{vmatrix} l_{1,1} - \lambda & l_{1,2} & \cdots & l_{1,n} \\ l_{2,1} & l_{2,2} - \lambda & \cdots & l_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ l_{n,1} & l_{n,2} & \cdots & l_{n,n} - \lambda \end{vmatrix} = 0 \tag{2.2}$$

Le $\Delta_n(L)$ est le déterminant de L. Il y a exactement n solutions pour le déterminant d'une matrice laplacienne, donc, une matrice $L_{n \times n}$ a n valeurs propres associées. On a la propriété suivante.

Lemme 2.1 (Connectivité algébrique [Godsil and Royle, 2001]) *Soit l'ensemble des valeurs propres ordonnées de la matrice laplacienne L : $\lambda_1(L) \leq \lambda_2(L) \leq \ldots \leq \lambda_n(L)$, on a $\lambda_1(L) = 0$ avec le vecteur propre correspondant $\mathbb{1}$. De plus, $\lambda_2(L) > 0$ si et seulement si le graphe G est connexe.*

3. Spanning Tree
4. Notez que pour le but de maintenir la connectivité, on n'a pas besoin de construire un arbre couvrant minimal[Graham and Hell, 1985].

La valeur propre $\lambda_2(L)$ représente la propriété globale de la connectivité. De plus, si on rajoute un poids à chaque arête, la valeur $deg(v_i)$ de l'élément $l_{i,i}$ de L est la somme de tous les poids (mesurée par une fonction de distance entre les nœuds, par exemple) des arêtes connectant v_i avec ses voisins.

Cela est la propriété la plus intéressante du graphe Laplacien. Cependant, comme nous n'avons pas utilisé ces notions dans le reste de notre travail sur le maintien de la connectivité, nous ne détaillons ici pas plus ces notions (voir [Godsil and Royle, 2001] pour plus de détails).

2.2.3 Connectivité robuste

Dans de nombreuses applications qui fonctionnent principalement dans des environnements hostiles, on désire assurer que le mauvais fonctionnement d'un nœud ou la coupure d'un lien du réseau ne provoque pas des interruptions dans la communication. Une telle propriété est appelée connectivité robuste. Cette contrainte est beaucoup plus forte que la connectivité « simple » discutée jusqu'ici.

Définition 2.7 (Les nœuds et les liens critiques) *Un nœud $v \in V$, ou un lien $e \in E$ du graphe G est critique si son retrait du graphe va partitionner le graphe en au moins deux sous-graphes connexes. Sinon, il est non-critique.*

Définition 2.8 (La robustesse de la connectivité) *Un réseau sans fil possède une connectivité robuste, si et seulement si $\forall v \in V$, v est non-critique, et $\forall e \in E$, e est non-critique.*

Soit $N(v)$ l'ensemble des voisins du robot v, et $\mid N(v) \mid = m$, nous avons le théorème suivant sur la robustesse de la connectivité de réseau.

Théorème 2.1 *Un nœud $v \in V$ dans un graphe G connexe est non critique si et seulement si $\forall n \in N(v)$, il existe au moins un circuit $C(v)$ tel que $n \in C(v)$.*

Démonstration. Supposons que l'on peut trouver un circuit $C(v)$ constitué d'un chemin de communication commençant et se terminant par v. Après la suppression de v, tous les nœuds voisins n de v sont encore connectés les uns aux autres par ce chemin. Par conséquent, le nœud est non critique.

Réciproquement, si v n'est pas critique dans V, après son retrait du graphe, le graphe reste toujours connecté. C'est-à-dire, nous devons être capable de trouver un chemin $(n_1, v_1, v_2, ..., v_j, n_2, v_l ..., v_h, n_m)$, où $v_i \in V$, et $v_i \notin N(v), n_j \in N(v)$. Le circuit $C(v)$ est obtenu en ajoutant simplement v pour le début et la fin de ce chemin. Ceci conclut la preuve. □

2.3 Taxonomie des algorithmes du maintien de la connectivité dans les systèmes de robots en réseau

Dans cette section, nous proposons une taxonomie des algorithmes connus du maintien de la connectivité dans les systèmes de robots en réseau. À notre connaissance, cette taxonomie est un des premiers travaux de ce type.

Nous commençons la taxonomie par la présentation de l'hypothèse sur le modèle de communication local mise en œuvre dans les travaux existants, ainsi que dans notre travail qui sera présenté dans le chapitre 4.

2.3.1 Modèle de voisinage et capacité requise pour les robots

Le point de base commun dans presque tous les travaux étudiés [Ahmadi and Stone, 2006b, Zavlanos and Pappas, 2007, Spanos and Murray, 2004, Zavlanos and Pappas, 2008, Almeida et al., 2008, Stump et al., 2008, Hsieh et al., 2008, Das et al., 2009, Schuresko and Cortés, 2009, Schuresko, 2009] est de déterminer la relation de voisinage selon laquelle la communication entre les robots pourrait avoir lieu.

Ce modèle théorique de communication (appelé le *graphe de disques* [Clark and Colbourn, 1990]) où le voisinage est défini en fonction de la distance euclidienne d_e entre les robots est largement accepté dans la littérature qui vise le problème du maintien de la connectivité.

Le modèle de graphe de disques *Tous les robots disposent de la même portée de communication $range_{comm}$ et deux nœuds sont des voisins si et seulement si la distance euclidienne d_e entre eux est inférieure à $range_{comm}$.*

Selon ce modèle, 2 robots doivent être dans la zone de communication l'un de l'autre pour être capables de communiquer. La zone de communication d'un robot est définie par un cercle avec le rayon de $range_{comm}$ centré sur le robot. Dans le cadre du modèle de communication sous forme de graphe mentionné dans la section 2.2.1, page 20, pour les deux nœuds $u, v \in V$, nous avons $e(u,v) \in E \Leftrightarrow d_e(u,v) \leq range_{comm}$.

Bien que ce modèle de communication ne tienne pas compte des obstacles dans l'environnement pouvant réduire considérablement la distance de communication effective [5], il a l'avantage de simplifier les études. De plus, la plupart des algorithmes de coordination couplés avec le maintien de connectivité ne changent pas si on utilise un meilleur modèle pour représenter la zone de communication.

Afin qu'un robot puisse déterminer la zone de communication et déterminer si un déplacement peut causer la déconnexion avec un voisin, des informations sur sa localisation et sur la position de ses voisins sont requises.

Capacité de localisation *Chaque robot devrait être capable de connaître sa position actuelle ainsi que celles de ses voisins immédiats* [6].

En pratique, les robots peuvent connaître leur position au moyen d'un matériel spécialisé comme le GPS, ou en appliquant d'autres méthodes de localisation, par exemple le SLAM [7] [Durrant-Whyte and Bailey, 2006].

5. Cela dit les obstacles peuvent être, en fait, partiellement modélisés/simulés avec une distance plus grande.

6. En fait, les informations sur la localisation des voisins immédiats peuvent être échangées parce que les robots sont capables de communiquer selon la première hypothèse. Il suffit donc que les robots déterminent leur propre localisation, cependant, nous faisons cette hypothèse par souci de simplicité de la présentation.

7. Simultaneous Localisation And Mapping

L'utilisation du modèle de graphe de disques et de l'hypothèse sur la capacité de localisation sont largement acceptés dans la recherche sur le maintien de la connectivité dans les réseaux de robots mobiles. En outre, le modèle de graphe de disques implique que les robots soient identiques du point de vue de la capacité de communication. Pour simplifier notre présentation et sans perdre la généralité, nous faisons aussi l'hypothèse que les robots sont homogènes du point de vue de leurs capacités de déplacement et de localisation.

2.3.2 Les critères de classification

Nous essayons, dans cette section, de définir les critères qui serviront à faire une distinction de ces solutions. Voici les critères que nous proposons :

- *Le niveau de décision* : approche centralisée ou distribuée.
- *La généralité d'une solution* : *de circonstance* ou ad hoc vs. générale.
- *Le niveau de propagation* des informations sur les positions de robots et
- *le modèle mathématique* sous-jacent utilisé pour le maintien de la connectivité.

Le premier critère a pour but de distinguer la décision concernant le maintien de la connectivité, si elle est prise par un entité centrale ou non. Ainsi, nous avons une classe des approches centralisées et une autre classe des approches décentralisées.

Le deuxième critère concerne le niveau d'application des solutions. Si une solution est développée dans le contexte spécifique aux applications cibles, alors l'algorithme du maintien de la connectivité n'est pas séparé par rapport aux différentes préoccupations applicatives (classe des solutions de circonstance). Par conséquente, ces solutions ont besoin d'être modifiées avant de pouvoir être utilisées dans un autre contexte. Afin de répondre à cette exigence de généralité dans l'application d'une solution, le problème du maintien de la connectivité doit être attaqué de façon indépendante par rapport au contexte des applications cibles (la classe des solutions générales).

Le troisième critère concerne le niveau de propagation des informations sur les positions de robots. Dans les travaux existants sur le maintien de la connectivité, un robot a besoin des informations sur les positions des autres robots dans le réseau (voir la section 2.3.1 – page 24). Dans certaines approches, on n'échange que les informations locales, dans d'autres, les robots ont besoin des informations sur tous les robots dans le réseau.

Le dernier critère fait référence au modèle mathématique utilisé pour le maintien de la connectivité.

La table 2.1 résume notre analyse des travaux existants dans la littérature.

Comme montré dans la table 2.1, nous avons regroupé les travaux étudiés en trois classes principales en fonction de 4 critères identifiés auparavant. Le reste de cette section est structuré suivant ces 3 classes qui reflètent les critères 1 et 4. En effet, nous pensons qu'il n'est pas pertinent de discuter le niveau de décision, la classification d'une approche comme centralisée ou décentralisée semble être suffisante.

Pour le troisième critère, nous trouvons que le niveau de propagation des données sur les positions de robots est en forte relation avec le modèle mathématique sous-jacent utilisé pour le maintien de la connectivité : une propagation glo-

TABLE 2.1 – *résumé des travaux du maintien de la connectivité.*

Classification	travaux représentatifs	niveau de décision	niveau de propagation	modèle mathématique
de circonstance	[Vazquez and Malcolm, 2004]	distribuée	globale	graphe complet
	[Notarstefano et al., 2006]	distribuée	globale	graphe algébrique
	[Sheng et al., 2006b]	distribuée	globale	distance de proximité
	[Rooker and Birk, 2007]	centralisée	globale	graphe complet
	[Stump et al., 2008]	centralisée	globale	graphe algébrique
connectivité algébrique	[Zavlanos and Pappas, 2005, Srivastava and Spong, 2008]	centralisée	globale	matrice d'adjacence
	[Gennaro and Jadbabaie, 2006, Zavlanos and Pappas, 2007, Zavlanos and Pappas, 2008, Micheal et al., 2009, Schuresko, 2009]	distribuée	globale	graphe algébrique
basé sur $\mathcal{ST}$	[Diosdado, 2006, Schuresko, 2009]	distribuée	locale / partielle	$\mathcal{ST}$

bale dans la connectivité algébrique et des échanges locaux dans la connectivité basée sur l'arbre couvrant $\mathcal{ST}$. C'est la raison d'avoir regroupé les deux dernières critères ensemble dans notre taxonomie : les solutions dérivées de la connectivité algébrique avec la propagation globale, les solutions basées sur l'arbre couvrant $\mathcal{ST}$ avec la propagation locale des positions des robots.

2.3.3 Les solutions *de circonstance*

Dans les premiers travaux intégrant les systèmes multi-robots et le réseau ad-hoc, les solutions pour maintenir la connectivité étaient étroitement couplées aux applications, et ont été conçues de manière ad hoc[8]. Un exemple typique est l'exploration d'un environnement inconnu par une équipe de robots qui communiquent afin de cartographier conjointement le terrain [Vazquez and Malcolm, 2004, Sheng et al., 2006b, Rooker and Birk, 2007].

Afin de garder la connectivité de manière décentralisée, les robots d'explo-

8. Nous classifions ces travaux comme des solutions de circonstance, car dans leur forme originale, les auteurs ne considèrent pas les préoccupations hors du contexte de l'application cible. Cependant, pour certains travaux étudiés ici, il est facile de généraliser leur solution afin d'être utilisable dans d'autres applications, comme [Vazquez and Malcolm, 2004] par exemple.

ration dans [Vazquez and Malcolm, 2004] analysent la topologie complète du réseau pour reconnaître les liens critiques. S'il y a un lien critique, la tâche consistant à maintenir le lien est prioritaire sur l'exploration. Les robots doivent périodiquement échanger des messages contenant l'information sur la topologie complète du réseau pour la détection des liens critiques.

L'algorithme d'exploration proposé par [Rooker and Birk, 2007], qui est dérivé de l'algorithme d'exploration *basé sur la frontière*[9][Yamauchi, 1997] a pour but d'assurer qu'aucun robot d'exploration ne va perdre la connexion avec le système pendant l'exploration. Leur algorithme est totalement centralisé. Il existe un serveur central qui collecte toutes les informations nécessaires pour calculer les déplacements prochains des robots à chaque pas d'exploration. Les travaux de [Stump et al., 2008] ont pour objectif de contrôler le déplacement d'un groupe de robots d'exploration tout en maintenant la connectivité avec un robot stationnaire dans un environnement fermé. La contrainte pour la connectivité est calculée de manière centralisée (connectivité algébrique). Comme le cas de la solution proposée par [Vazquez and Malcolm, 2004], nous considérons ce travail appartenant à la classe des solutions ad hoc car l'algorithme est développé pour une application spécifique et pour maintenir la connexion avec une station fixe.

L'algorithme d'exploration de [Sheng et al., 2006b] est distribué mais il a besoin des informations globales pour le maintien de la connectivité. À chaque phase de planification, les robots échangent leur enchère sur la frontière à visiter. L'enchère de chaque robot est couplée avec une fonction d'utilité heuristique appelée *la mesure de proximité*[10] pour mesurer la proximité entre les robots. Cette solution n'assure pas toujours la connectivité dans le sens où la mesure de proximité ne donne que la préférence dans la sélection de la zone à explorer. Cette approche ad hoc est acceptable pour l'exploration dans laquelle les robots peuvent travailler en sous-réseaux separés. Elle n'est pas appropriée pour des applications qui exigent une connectivité permanente.

2.3.4 Le maintien de la connectivité algébrique

La lemme 2.1 (page 22) suggère que, si on peut contrôler les robots dans le système de sorte que la valeur propre $\lambda_2(L)$ soit toujours supérieure à 0, alors le graphe est bien connexe. Cette idée fonde la base mathématique de nombreuses approches de maintien de la connectivité [Zavlanos and Pappas, 2005, Notarstefano et al., 2006, Zavlanos and Pappas, 2007, Zavlanos and Pappas, 2008, Micheal et al., 2009, Schuresko, 2009].

Dans [Zavlanos and Pappas, 2005, Notarstefano et al., 2006] le calcul de la valeur propre pour le maintien de la connectivité est centralisé, les autres algorithmes utilisent un mécanisme de réplication des positions de tous les robots sur chaque robot individuel pour décentraliser la procédure.

La proposition de [Notarstefano et al., 2006] permet aux robots de déterminer leurs déplacements d'une manière décentralisée, mais en maintenant une topologie fixe (la relation voisinage entre les robots ne change pas). Le travail de [Zavlanos and Pappas, 2005] utilise la valeur propre λ_2 d'une façon centralisée.

9. Frontier-based exploration algorithm
10. En anglais : Nearness Measure

Puis cette attribution est utilisée dans un algorithme d'enchère pour contrôler la suppression d'un lien depuis le graphe. La procédure centralisée est ensuite décentralisée dans [Zavlanos and Pappas, 2007, Zavlanos and Pappas, 2008, Micheal et al., 2009]. [Schuresko, 2009, Schuresko and Cortés, 2009] proposent un ensemble d'algorithmes distribués pour propager les positions des robots dans le réseau afin que chaque robot calcule son déplacement de sorte que la valeur propre λ_2 ne soit jamais inférieure à un seuil pré-déterminé.

Remarquons que le calcul de la valeur de $\lambda_2(L)$ demande des informations globales sur tous les nœuds dans réseau. Le surcoût de communication est assez élevé. De plus, il est possible que les informations sur les positions des robots ne soient pas cohérentes à cause de la latence de communication dans le réseau et des déplacement des robots.

2.3.5 Le maintien de la connectivité basé sur l'utilisation d'un Arbre Couvrant

Comme cela a été remarqué dans le corollaire 2.1 (page 22), la connectivité est assurée si nous arrivons à construire et maintenir un $\mathcal{ST}$ dans le système. On retrouve ce principe dans [Diosdado, 2006, Schuresko, 2009], où les algorithmes pourraient être découpés en les étapes suivantes :

1. Construire un $\mathcal{ST}$ qui détermine la relation père-fils entre deux robots.
2. Réaliser des déplacements de sorte que le $\mathcal{ST}$ soit conservé (maintenir la lien entre le père et le fils).

[Diosdado, 2006] proposent BERODE2 [11], un outil de simulation d'exploration multi-robots sous la contrainte de la connectivité du réseau sans fil. Les robots doivent maintenir la connectivité pour synchroniser les données collectées pendant l'exploration. Au début de la mission d'exploration, un robot est choisi pour être la racine du $\mathcal{ST}$. Puis le $\mathcal{ST}$ est construit par un algorithme distribué basé sur l'algorithme de Dijkstra [Cormen et al., 1990]. Par la suite toute connexion entre les robots avec leurs voisins est classifiée comme *connexion de contrôle* si ce lien appartient au $\mathcal{ST}$ ou *connexion normale* sinon. On oblige les robots de ne maintenir que les connexions de contrôle. Pendant l'exploration, un robot peut jouer un des quatre rôles : *Explorer*, *Maintainer*, *Recoverer*, et *Pusher*. Les rôles gouvernent le comportement de chaque robot et assurent le choix d'un déplacement pour que le $\mathcal{ST}$ soit préservé. Le $\mathcal{ST}$ peut être modifié en fonction d'évolutions du réseau pendant la mission des robots. On va appeler dans ce cas le $\mathcal{ST}$ l'arbre couvrant *adaptatif* ou *évolutif*, pour faire la différence avec le $\mathcal{ST}$ fixé au début implémenté dans l'ancêtre de BERODE2 [12].

Comme les solutions mentionnées dans la section 2.3.3, l'algorithme implémenté dans BERODE2 ne sépare pas bien les préoccupations à l'égard du maintien de la connectivité des spécificités de l'application. Ce désavantage est supprimé dans la solution proposée par [Schuresko, 2009], qui est aussi basé sur un $\mathcal{ST}$ *évolutif* pour le maintien de la connectivité.

L'idée de base de cette approche est que chaque robot maintient la connexion avec un robot père dans le $\mathcal{ST}$. Dans le but de préserver la connectivité, chaque

11. BEhavioural ROle DEcentralized
12. BERODE [Diosdado, 2006]

robot conserve une estimation de sa profondeur vers la racine du $\mathcal{ST}$. Cette profondeur estimée sera utilisée pour déterminer la relation père-fils dans le $\mathcal{ST}$. Avant de réaliser un déplacement, les robots doivent éventuellement choisir un nouveau père pour maintenir la connexion entre eux et leur père. Le $\mathcal{ST}$ est évolutif dans le sens où les robots peuvent changer leur père, donc changer la topologie du $\mathcal{ST}$.

2.3.6 Résumé sur les études menées

À remarquer que dans notre analyse, il manque une comparaison formelle de la complexité de la communication. En effet, aucun auteur cité ne donne une analyse formelle à cet égard quelle que soit l'approche qu'il utilise, sauf les auteurs de [Notarstefano et al., 2006, Schuresko, 2009, Schuresko and Cortés, 2009] qui déclarent que cela est une perspective de recherche. À notre avis, c'est un sujet théorique très difficile principalement à cause de la dynamique dans la topologie du réseau et il mérite une étude approfondie.

Malgré ce manque, ce résumé nous suggère que, du point de vue du passage à l'échelle, l'approche basée sur le $\mathcal{ST}$ est le meilleur choix : elle est décentralisée contrairement à l'approche algébrique où la propagation globale de localisation des robots est nécessaire une fois que le $\mathcal{ST}$ est construit. L'approche basée sur le $\mathcal{ST}$ ne demande que des informations locales, par exemple la localisation des voisins. Pourtant, le désavantage de l'approche basée sur $\mathcal{ST}$ est que nous devrons choisir un nœud pour être la racine du $\mathcal{ST}$. C'est-à-dire que tous les robots devraient être au courant sur ce choix et ils ne commencent la mission qu'après que le $\mathcal{ST}$ soit construit.

Un avantage des algorithmes basés sur le graphe laplacien est que le calcul de la valeur propre peut donner un sens plus significatif dans les cas où les arêtes du graphe sont couplées avec les poids [13]. Pour ce dernier cas, le problème peut être formulé comme un problème d'optimisation (maximisé) de sa valeur λ_1 [Schuresko and Cortés, 2009].

Dans le contexte de notre travail et dans le cadre du problème que nous avons formulé, nous ne nous intéressons pas à la maximisation de la connectivité dans le sens donné par la valeur propre λ_1.

Un point important à noter est qu'aucun travail sur la connectivité parmi ceux étudiés ne s'intéresse à la connectivité robuste. Cela est du fait que, d'un côté, les modèles mathématiques utilisés pour la connectivité ne capturent pas intuitivement la propriété de la robustesse de la connectivité. D'autre côté, la connectivité simple est suffisante pour la plupart des applications. Cependant, une telle approche peut être plus intéressante dans les systèmes robotiques qui fonctionnent dans des environnements hostiles.

Conclusion du chapitre

L'utilisation de la technologie MANET comme support de communication dans les systèmes des robots mobiles est prometteuse. Pour conclure, en supposant l'autonomie des nœuds, les MANETs ne disposent pas des moyens pour

13. Les poids peuvent être la distance entre les robots voisins, la puissance du signal, etc.

contrôler la topologie et la connectivité dans le réseau. La situation est bien différente dans les réseaux de robots mobiles où il est possible de contrôler les déplacements de robots. Les robots peuvent planifier leurs déplacement de sorte que la connectivité ne soit pas compromise.

Dans ce chapitre, nous avons proposé une taxonomie qui est, à notre connaissance, une des premières de ce type. La taxonomie classifie les travaux existants en trois classes basées sur des critères que nous croyons importants. Cette classification relève les points forts et les points faibles de chaque approche, ce qui nous permet d'identifier la démarche à suivre dans la recherche des solutions pour notre problème.

Coopération dans les systèmes multi-robots

3

Sommaire

Nous présentons dans ce chapitre, un état de l'art de la coopération dans les systèmes multi-robots. Dans ce cadre, nous discutons du lien entre les concepts de tâche et de rôle. Nous montrons ainsi les limites de la décomposition par tâche des missions robotiques et de l'intérêt d'utiliser les rôles. Néanmoins, l'introduction des rôles ne résout pas pour autant le problème de déploiement d'une mission sur une flotte de robots. Le problème d'allocation de rôles est identique à celui de l'allocation de tâches. Les paramètres identifiés dans la taxonomie de Gerkey et Mataric [Gerkey and Matarić, 2004b] se retrouvent dans les deux problèmes. Nous montrons ainsi que le problème traité dans cette thèse est le cas le plus difficile. Il s'agit du cas de missions avec des tâches nécessitant la

coopération de plusieurs robots hétérogènes capables d'endosser plusieurs rôles simultanément.

3.1 Scenario de référence : pousser une boîte avec des robots hétérogènes

Une des tâches à réaliser lors des opérations de sauvetage est l'assistance aux victimes. Cependant, les capacités des robots actuels sont encore loin d'être suffisantes pour secourir de manière autonome les blessés. Néanmoins, du point de vue de la coopération, un parallèle peut être fait entre cette tâche complexe et une tâche plus simple où des robots hétérogènes doivent collaborer pour pousser une boîte vers une destination précise. Nous présentons ci-dessous ce scénario que nous utilisons pour illustrer et discuter l'état de l'art dans le reste du chapitre.

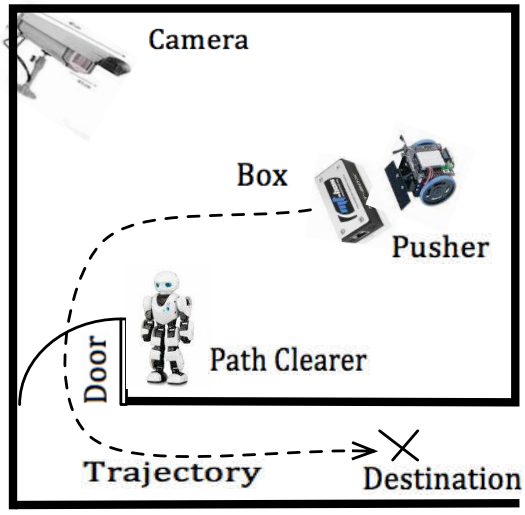


Figure 3.1 – *Un exemple de tâche coopérative*

Cette tâche coopérative consiste à déplacer une boîte vers la destination selon une trajectoire connue a priori. Pour mener à bien cette tâche, nous disposons de deux robots mobiles autonomes, à savoir un **Pusher** et un **Path Clearer**. En outre, il y a une caméra montée sur le mur qui est connectée à un ordinateur. À partir des images capturées par la caméra, l'ordinateur est capable de localiser des objets dans la pièce. Le **Pusher** qui n'a ni des capteurs visuels ni des capacités de localisation peut, comme son nom l'indique, pousser la boîte. Il a besoin pour cela d'être instruit en temps réel sur sa localisation et sa position relative par rapport à la boîte. La porte de la pièce est toujours fermée. Le **Path Clearer** doit donc l'ouvrir et la maintienir ouverte pendant que le **Pusher** pousse la boîte vers le couloir. En tant que tel, le groupe de ces trois robots [1] fonctionne bien jusqu'au moment où le **Pusher** a déplacé la boîte hors de la zone de couverture de la caméra. L'ordinateur n'est alors plus en mesure de fournir des instructions de localisation au **Pusher**. Ainsi, afin de poursuivre la réalisation de la tâche, l'ordinateur doit être remplacé. Le **Path Clearer**, qui dispose de caméras, est à ce titre un bon candidat pour guider le **Pusher**.

1. Sur la base de la définition donnée dans la section 1.2.3, l'ordinateur est considéré comme étant un robot fixe, doté d'un capteur visuel (la caméra).

3.2 Aspects liés à la coopération

Dans son travail de thèse, Arrichiello [Arrichiello, 2006] a montré que dans un système de robots coopérant il est nécessaire de s'intéresser aux trois aspects suivants : *les tâches*, *le mécanisme de coopération*, et *la performance du système*. Dans cette section, nous discutons de la coopération et de chacun de ces aspects.

3.2.1 Coopération émergente vs coopération intentionnelle

Dans les systèmes multi-robots, deux types de coopérations peuvent être distinguées : 1) la coopération émergente et 2) la coopération intentionnelle [Gerkey and Matarić, 2002].

La coopération émergente est habituellement mise en œuvre dans la robotique en essaim [2] [Parker, 2008b]. Cette approche s'inspire des observations de sociétés biologiques – en particulier les fourmis, les abeilles et les oiseaux – pour développer des comportements similaires dans les équipes de robots. Dans de tels systèmes, la coopération n'est pas explicitement définie. Mais, elle *émerge* des interactions locales entre les robots eux-mêmes et entre les robots et l'environnement selon des règles simples. Une des grandes forces de cette approche est qu'elle permet aux robots d'opérer dans des situations imprévues et de s'adapter aux conditions changeantes. La coopération émergente a couramment été appliquée à des groupes de robots homogènes. Elle s'appuie sur une redondance des compétences et des ressources à travers le groupe pour obtenir une bonne performance globale comme observé dans les sociétés biologiques [Gerkey and Matarić, 2002]. La difficulté de cette approche est qu'il est difficile d'identifier les comportements individuels des robots, parce qu'ils sont très éloignés de la tâche globale à réaliser. Un autre problème est le nombre de robots qui doit être important pour que l'émergence ait lieu.

Le mécanisme de coopération intentionnelle implique que les robots coopèrent par le biais d'échanges d'informations et de négociations directement liées à la tâche à accomplir. Ce type de coopération nécessite donc une communication explicite. Il permet de mieux exploiter les capacités d'équipes de robots hétérogènes en les exploitant pour accomplir des actions complémentaires directement liées à la tâche à réaliser. Ces actions sont obtenues en décomposant la tâche correspondante. Cette démarche descendante est plus simple à mettre en œuvre que la démarche ascendante de la coopération émergente [Gerkey and Matarić, 2004a]. En effet, l'approche "diviser pour régner" est bien connue et maîtrisée par les concepteurs. De plus, elle convient même pour un petit nombre de robots. C'est pour ces raisons que nous nous focalisons dans ce travail sur cette approche. Il faut cependant garder en tête que la coopération intentionnelle est plus difficile à utiliser pour de grandes flottes de robots du fait de la combinatoire. Une autre difficulté liée à la coopération intentionnelle est qu'il est nécessaire de gérer explicitement l'adaptation du système aux changements de l'environnement ou à la défaillance des robots.

2. Swarm robotics

3.2.2 Mission, tâches, sous-tâches

L'idée de base dans de nombreux travaux en robotique est connue comme le paradigme *diviser-puis-attribuer*[3] [Gerkey and Matarić, 2002, Dias, 2004, Kalra et al., 2004]. Il s'agit de décomposer une mission robotique en un ensemble de tâches (reliées). Puis, ces tâches sont à leur tour décomposées en sous-tâches. Cette subdivision se poursuit jusqu'à ce que l'on arrive à des sous-tâches *élémentaires*. Une (sous-)tâche est dite élémentaire si elle est directement réalisable par un robot seul.

Une fois la décomposition réalisée, il faut répartir les (sous-)tâches élémentaires sur les robots. Puis, les robots vont réaliser ces (sous-)tâches, et ainsi accomplir la mission. Le degré de coordination des actions des robots dépend du degré d'interdépendance des (sous-)tâches. Cette interdépendance doit donc être explicitement exprimée dans la définition des (sous-)tâches.

Dans le reste de ce mémoire, nous utilisons le terme *tâche* pour indiquer une *partie atomique et indépendante* d'une mission. Par « atomique et indépendante », nous voulons dire qu'une telle tâche n'est liée à aucune autre tâche. La réalisation d'une tâche est indépendante de la réalisation des autres tâches de la mission.

La coordination est, par contre, nécessaire pour la réalisation d'une tâche. En effet, une tâche se décompose en un ensemble de sous-tâches élémentaires interdépendantes. Un robot qui travaille sur une de ces sous-tâches élémentaires doit coopérer avec au moins un autre robot réalisant une autre sous-tâche élémentaire.

Le scénario dans le figure 3.1 (page 33) décrit une mission qui ne comporte qu'une unique tâche. Cette tâche comporte trois sous-tâches : 1) la localisation, 2) pousser la boîte, et 3) garder la porte ouverte. Ces trois sous-tâches sont interdépendantes : le robot qui travaille sur la sous-tâche 2 dépend des robots en charge des sous-tâches 1 et 3.

3.2.3 Mécanisme de coopération

La façon dont une mission est décomposée en tâches, l'ordre d'exécution de ces tâches et le niveau de synchronisation des actions des robots pendant l'exécution des sous-tâches élémentaires représentent ensemble la logique du mécanisme de coopération.

Une méthode très utilisée pour répartir les sous-tâches élémentaires sur les robots est l'appel d'offre [Dias et al., 2006]. Il emploie le protocole de réseau contractuel (en anglais : Contract-Net Protocol – CNP), inventé par [Smith, 1980]. Par le moyen du CNP, les robots négocient entre eux afin d'optimiser une fonction d'utilité (voir la section 3.2.4). Le processus de négociation implique généralement des communications explicites entre des robots sur les sous-tâches à réaliser. Etant donné une sous-tâche, chaque robot disposant des capacités requises pour la réaliser émet une offre qui dépend de ses ressources. La détermination du robot « gagnant » est faite généralement de manière gloutonne en fonction des profits estimés.

Le CNP est largement mis en œuvre dans la robotique [Parkes and Ungar, 2000, Brunet et al., 2008, Choi et al., 2009] grâce au compromis entre la répartition des

3. divide-then-allocate

charges sur les robots et la décision centralisée. Presque toutes les applications robotiques qui sont déjà validées reposent sur une variante ou une autre du CNP pour l'attribution des tâches.

3.2.4 Performance du système et fonction d'utilité

La répartition des sous-tâches élémentaires entre des robots doit être réalisée de manière à ce que le système multi-robots soit le plus performant possible pour réaliser une tâche et plus généralement la mission. La performance du système peut être représentée par des caractéristiques comme, par exemple, le temps d'exécution de la mission, la complexité algorithmique, la robustesse et la tolérance aux pannes. Elle peut dépendre de la structure globale du système, par exemple, de la stratégie de décomposition des tâches ou des caractéristiques de la communication, etc. Idéalement, tous ces facteurs devraient être pris en compte dans l'évaluation de la performance du système. Cependant, une telle quantité est souvent difficile à mesurer lors de l'exécution du système. Par conséquent, on utilise l'utilité comme estimation de la performance [Gerkey and Matarić, 2004a].

L'utilité a pour origine l'économie dans laquelle le terme fait référence à la satisfaction relative reçue de la consommation d'une marchandise ou d'un service. Aujourd'hui, ce concept est couramment exploité dans la recherche opérationnelle, dans la théorie de jeux et dans la coordination robotique. L'idée sous-jacente est qu'un membre d'une société est le plus à même de mesurer son aptitude pour réaliser une action et la traduire en un nombre comparable avec les mesures des autres membres de la société. Cette mesure est nommée diversement même dans la même domaine, comme l'habilité, le coût, l'évaluation, ou bien le revenu perçu pour la réalisation d'une tâche. Cette idée est reprise dans la coopération robotique, comme le montre l'excellente synthèse sur l'allocation des tâches [Gerkey and Matarić, 2004a, Gerkey and Matarić, 2004b] .

3.3 Approches à base de rôle

3.3.1 Rôle vs. Tâche

Dans la robotique, les deux termes *tâche* et *rôle* réfèrent tout deux aux résultats d'une approche de « *diviser pour mieux régner* » pour la résolution de problèmes de manière coopérative. La répartition des tâches et l'attribution de rôled sont les problèmes d'associer ces sous-tâches aux robots de manière optimale. Les problèmes restent les mêmes quel que soit le terme, tâche ou rôle, utilisé. Ainsi, les deux termes sont utilisés de façon interchangeable dans la littérature [Gerkey and Matarić, 2004a, Gerkey and Matarić, 2004b]. Toutefois, il existe certaines différences !

Le dictionnaire Webster [4] définit le rôle en tant que :

> (a) une fonction ou une partie réalisée en particulier dans une opération ou du processus, (b) un profil social du comportement attendu généralement déterminé par le statut d'un individu dans une société donnée.

4. http://www.merriam-webster.com/dictionary

Tandis qu'une tâche est : une partie d'un travail et doit souvent être terminée dans un certain délai.

Selon ces définitions, la notion de tâche devrait être utilisée comme une unité de travail, alors que la notion de rôle devrait être utilisée pour décrire la fonction que le robot « joue » au sein de l'équipe. Le rôle que le robot prend définit les tâches qu'il peut/doit ou ne peut/doit pas réaliser.

Dans [Campbell and Wu, 2007], les auteurs donnent un exemple de distinction entre rôle et tâche : « considérons une équipe comprenant n robots. Si l'objectif de l'équipe est de récupérer N mines marquées, il est possible d'assigner une mine à chaque robot et ensuite dire quelque chose qui ressemble à : *la tâche du robot X est de récupérer la mine Y*. Il pourrait également être présenté d'une manière différente : *le rôle du robot X est* Récupérateur-De-Mine. Dans la première déclaration, la tâche est utilisée pour se référer à la pièce de travail effectif qui est attribuée au robot X, alors que le rôle est utilisé dans la seconde pour décrire le 'personnage' que le robot X joue dans l'équipe ».

Dans la communauté multi-agents, les rôles ont été utilisés pour décrire les « fonctions » joués par un agent, un modèle de comportement nommé ou tout simplement une *série de tâches à exécuter*. Une définition simple de rôle le décrit comme *ensembles de tâches liées* [Thomas and Williams, 2005].

Selon [Odell et al., 2003b], les sociétés doivent employer des motifs de comportement pour exister. Le rôle fournit de tels motifs pour les individus qui l'endossent. De manière opérationnelle, un rôle est définit comme « une classe qui définit un répertoire comportemental normatif d'une entité » [Odell et al., 2003b].

Enfin, comme indiqué dans [Gerkey and Matarić, 2004b], les rôles ont des responsabilités s'étendant dans le temps, quant aux tâches, elles sont transitoires par nature.

Pour conclure, un rôle définit un ensemble d'actions connexes ou comportements. Les rôles représentent aussi la relation entre les différentes entités dans un système. Par conséquent, les rôles décrivent naturellement mieux la coopération intentionnelle entre les individus que les tâches. Nous pensons donc que les approches basées sur les rôles sont plus appropriées pour les problèmes coopératifs. Dans de nombreux domaines, il est plus naturel de penser à une solution composée d'entités interactives avec des rôles distincts qu'à des tâches [Dias, 2004].

3.3.2 Des tâches aux rôles

L'approche consistant à décomposer une mission en tâches et sous-tâches souffre du fait qu'elle est, selon nous, fortement liée au système multi-robots cible. En effet, la frontière entre une tâche complexe nécessitant la coopération de plusieurs robots et une tâche élémentaire réalisable par un seul robot varie en fonction des compétences des robots disponibles. Par exemple, nous avons décomposé la mission de la section 3.1 en trois tâches élémentaires : localiser la boîte, ouvrir la porte et pousser la boîte. Ces tâches correspondent avec les capacités des robots décrits du scénario. Supposons maintenant que nous disposons d'un unique robot sophistiqué, doté de moyens de localisation et capable de garder la porte ouverte tout en poussant la boîte. Dans ce cas, la décomposition

de la mission débouche sur une unique tâche élémentaire : déplacer la boîte de l'intérieur de la pièce à l'extrémité du couloir.

Cette dépendance entre la description de la mission et la configuration du système multi-robots rend la conception de solutions difficile. En effet, les concepteurs ne peuvent pas décomposer une mission sans connaître les robots cibles. Or, cette connaissance n'est pas disponible dans tous les projets et encore moins dans des situations telles que celles du sauvetage robotisé. Les robots qui sont utilisés ne sont connus qu'au moment de l'intervention. Conditionner la description de la mission à l'arrivée des robots introduirait un retard inacceptable, puisqu'il faut agir en urgence pour sauver des vies.

Une alternative à laquelle nous adhérons, est basée sur concept de *rôle*. Il permet de concevoir des solutions génériques en décomposant les missions en rôles. Ainsi, les concepteurs d'applications robotiques peuvent développer des solutions, tant au niveau algorithmique qu'au niveau logiciel, pour résoudre les tâches indépendamment des capacités disponibles de tout système de robot en question. Au lieu de décomposer une tâche (complexe), en sous-tâches, puis en sous-sous-tâches, on considère chaque robot comme une entité dans organisation dans laquelle il coopère à base de ses rôles. Dans le cas de notre mission de déplacement d'une boîte, les concepteurs déboucheraient sur 3 rôles :

- **Pusher** – correspond au comportement : pousser une boîte.
- **Path Clearer** – correspond au comportement : écarter les obstacles mobiles.
- **Pilot** – correspond au comportement : localiser des entités et de diffuser des informations de guidage.

Ces trois rôles peuvent être répartis sur 3 robots différents. Ils peuvent aussi être endossés par un même robot.

3.4 Conception des systèmes coopératifs à base de rôles

Dans la démarche de développement à base de rôle, au lieu de décomposer l'application en sous-tâches, les développeurs d'abord définissent des rôles différents, indiquent les comportements associés à chaque rôle, ainsi que l'interaction entre les acteurs jouant ces rôles. Ainsi, le système est une organisation sociale avec des robots qui jouent des rôles. Il est donc nécessaire d'identifier ces rôles. Il existe des méthodes qui servent à ce but, comme le modèle AGR [Gutnecht, 2001], le modèle Moise [Hannoun et al., 2000, Hübner et al., 2005] dans le monde du multi-agents.

En robotique, l'approche basée sur les rôles a connu de nombreux succès comme en témoignent dans la littérature [Stone and Veloso, 1999, Candea et al., 2001, Emery et al., 2002, Parker, 2003, Howard et al., 2006, Agüero et al., 2006, Certo et al., 2007, McMillen and Veloso, 2007, Parker, 2008a]. Dans le cadre de la RoboCup™ [5], [Stone and Veloso, 1999], [Candea et al., 2001], [McMillen and Veloso, 2007], ont utilisé les rôles comme mécanisme fondamental pour exprimer la coopération dans une équipe de robots footballeurs. [Chaimowicz et al., 2004] ont proposé une architecture pour la coordination et l'allocation des tâches aux robots pour des applications multi-robots exigeant une coopération étroite telles que le transport collectif. Toutes ces approches ont

5. http://www.robocup.org/

la même idée fondamentale que les rôles sont des abstractions « enveloppant » les entités effectives. Les rôles décrivent les compétences individuelles dans le contexte d'un processus de coopération intentionnelle.

Dans ce qui suit, nous présentons en détail le modèle AGR [Gutnecht, 2001] des systèmes multi-agents, et les travaux [Stone and Veloso, 1999, Parker, 2003, Certo et al., 2007] dans la robotique. Nous avons choisi ces approches comme représentatives car elles sont très proches de notre but de décrire la coopération de manière explicite.

3.4.1 AGR : Agent–Groupe–Rôle

Dans la communauté multi-agents, les rôles sont un moyen efficace pour la séparation des préoccupations dans les approches *par le haut* pour concevoir des agents coopératifs [Odell et al., 2003b]. L'AGR (Agent-Groupe-Rôle) est un modèle organisationnel qu'on peut utiliser comme modèle de référence pendant la phase de conception des systèmes multi-agents [6].

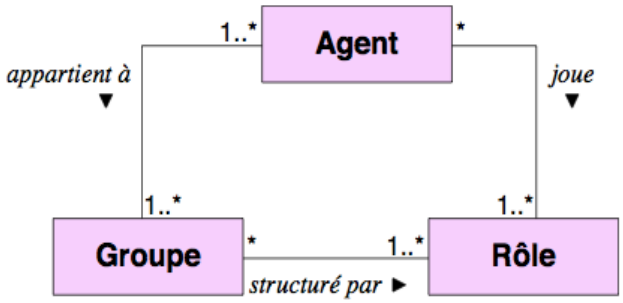


FIGURE 3.2 – *Le méta-modèle UML représentant les trois concepts centraux de l'AGR [Gutnecht, 2001]*

Comme son nom le suggère, le modèle AGR introduit trois concepts centraux : l'Agent, le Groupe et le Rôle. Un agent est *une entité autonome et communicante*. L'ensemble des agents collaborant à l'accomplissement de buts communs constitue une organisation.

Au sein d'une organisation, les agents se répartissent en groupes. Un groupe est donc *un ensemble d'agents* ou plus exactement un sous-ensemble de l'organisation. La notion de groupe permet la partition du système en sous-ensembles d'agents capables de communiquer entre eux et de collaborer (figure 3.2). Un rôle est *une représentation abstraite des fonctionnalités d'un agent*. Un rôle définit localement au sein d'un groupe la manière dont l'agent devra agir. Un agent peut endosser plusieurs rôles et ainsi faire partie de groupes différents.

Enfin, la notation de « plateau de fromages » est proposée pour décrire la conception d'application comme, par exemple, l'organisation de l'Agence de Voyage montrée par la figure 3.3. Dans le diagramme, un groupe est représenté

6. Soulignons qu'AGR n'est qu'une solution parmi d'autres dans la conception des systèmes multi-agents. Il existe de nombreux autres modèles organisationnels [Hannoun et al., 2000, Hübner et al., 2005], dont certains sont directement dérivés de l'AGR comme AGRE (AGR + Environment) [Ferber et al., 2005], AGREEN (AGRE Enhanced) [Báez-Barranco et al., 2007], ou AGRS (AGR + Service) [Mansour, 2007]. Cependant, dans le but d'illustrer les méthodes de conception des applications à base de rôles, seule introduction de l'AGR est suffisante, vu que ce qui nous intéresse est le problème d'allocation de rôles aux robots.

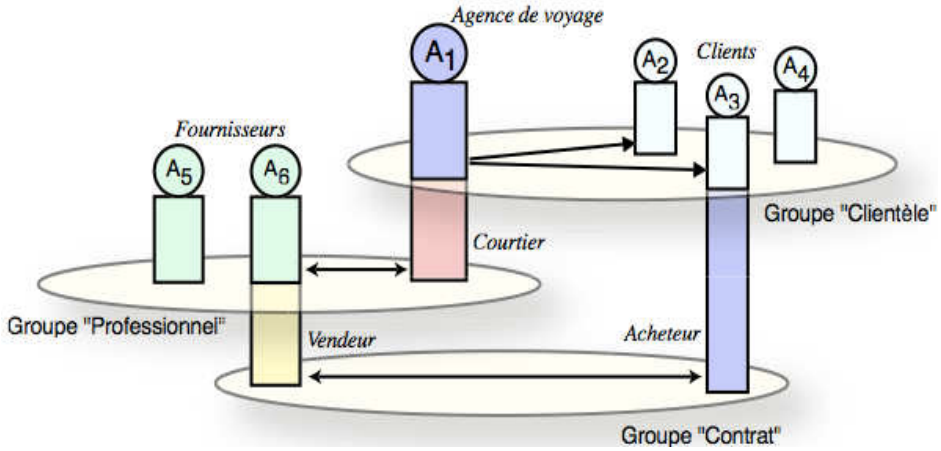


FIGURE 3.3 – *La notation « plateau de fromages » pour décrire l'organisation de l'Agence de Voyage [Gutnecht, 2001].*

comme un ovale qui ressemble à une planche. Les agents sont représentés comme des quilles qui se dressent sur la planche et passent parfois au travers du bord quand ils appartiennent à plusieurs groupes.

Dans AGR, les robots correspondent à des agents physiques, situés dans l'environnement. Ces agents peuvent jouer plus d'un rôle en même temps. Une coalition de robots correspond à un groupe d'agents.

Une fois les rôles définis, la question est comment affecter les rôles aux agents/robots appropriés en fonction de leurs capacités matérielles et logicielles avec un certain degré d'optimalité (par exemple en maximisant le somme totale des fonctions d'utilité – cf. la section 3.2.4, page 36). Malheureusement, cette question n'est pas traitée dans AGR. La figure 3.4 montre un exemple d'allocation de rôle dans MadKit [Gutknecht and Ferber, 2000], l'implantation de référence du modèle AGR. Dans cet exemple, nous pouvons voir que l'allocation des rôles aux agents est faite « manuellement » lors de la conception. Le code de l'agent référence déjà les rôles qu'il doit jouer.

Comme nous avons pu voir, AGR ne fournit aucune méthode d'allocation automatique des rôles aux agents. L'allocation de rôle dans d'autres travaux sur les systèmes multi-agents comme [Colman and Han, 2007, Dastani et al., 2003, Odell et al., 2003a] est similaire. Cela n'est pas satisfaisant pour développer un système capable de s'organiser automatiquement en fonction des tâches, de l'environnement ainsi que des ressources et des défaillances des robots (c.f. 1.3.3, page 9).

3.4.2 Attribution dynamique de rôles pour des robots footballeurs

[Stone and Veloso, 1999] sont des pionniers dans l'emploi des concepts sociaux qu'ils ont mis en œuvre pour leur équipe de robots footballeurs dans le cadre de la RoboCup. En fait, ils affirment que cette structure est appropriée non seulement pour le domaine de RoboCup, mais aussi pour toutes les applications de type *periodic team synchronization* (PST) où : « il y a une équipe d'agents autonomes qui collaborent à la réalisation d'un objectif commun à long terme. »

Selon l'approche de Stone et Veloso, une équipe de type PST est organisée selon une formation. Une formation comprend un ensemble de rôles, qui sont

```
public class Hello extends Agent {
        String myCommunity="myCommunity";
        String myGroup="myGroup";
        String myRole="myRole";

        boolean alive = true;

        public void activate(){
                println("Hello I'm an agent !");
                // create a distributed group
                int r = createGroup(true, myCommunity, myGroup, null, null);
                if (r != 1)
                        alive =false;
                else
                        requestRole(myCommunity, myGroup, myRole);
        }

        public void live()
        {
                println("Hello world...");
                while(alive){
                        Message m = waitNextMessage();
                        handleMessage(m);
                }
        }

        void handleMessage(Message m){
                // You should describe here the agent's behavior
                // upon reception of a message
        }

        public void end()
        {
                println("\t That's it !!! Bye ");
                pause(2000); // just to be abloe to see the last message..
        }
}
```

FIGURE 3.4 – *Le programme Helloworld avec MadKit [Gutknecht and Ferber, 2000] pour illustrer comment un groupe est créé, et un agent joue un rôle au sein du groupe (les instructions dans le rectangle).*

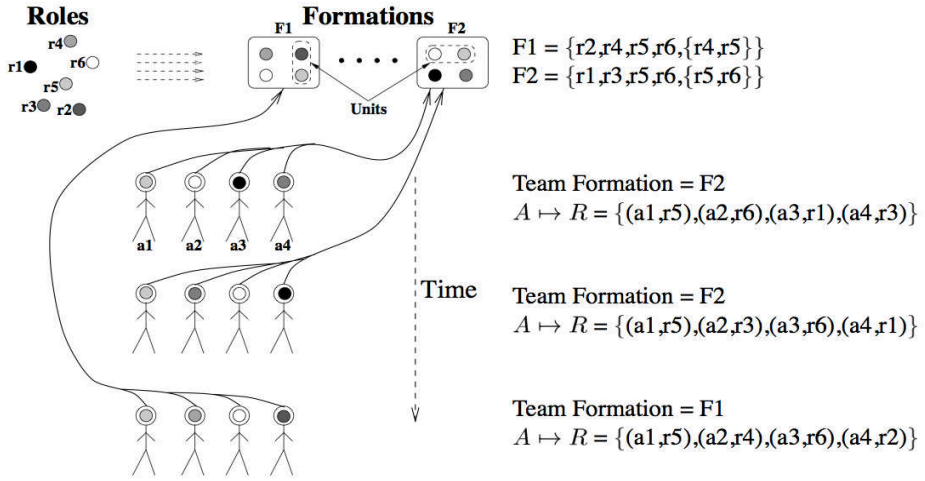


FIGURE 3.5 – *[Stone and Veloso, 1999] Une équipe d'agents avec commutation des rôles et des formations dans le temps. Différents rôles sont représentés par des cercles différemment grisés. Les formations se recoupent éventuellement collections des rôles. Les unités au sein des formations sont indiquées dans un cadre en pointillé.*

regroupés en unités (cf. le figure 3.5). Les définitions de tous les rôles, les formations et les unités sont connues de tous les agents. Ces informations sont stockées, sur chaque agent, dans une structure appelée *locker-room agreement*. Le rôle actuel d'un agent est indiqué par un cercle ombré (voir figure). Une flèche lie l'agent à sa formation actuelle.

Chaque rôle est pris par un seul agent, et un agent ne peut prendre qu'un seul rôle. Par conséquent, le nombre de rôles dans chaque formation doit être égal au nombre d'agents.

Les rôles peuvent être rigides, en précisant tout le comportement d'un agent. A contrario, ils peuvent être souples, en laissant une certaine autonomie à l'agent qui remplit le rôle. Un rôle est une spécification de comportements internes et externes d'un agent. Pour chaque comportement, il y a un ensemble de conditions d'activation. Quand ces conditions sont vérifiées, le robot peut décider de déclencher le comportement. Cette décision prend en compte l'état interne du robot et celui de son environnement.

Le rôle qu'un agent va jouer au sein d'une formation est déterminé de façon fixée. Un agent ne change que son rôle si l'équipe passe d'une formation à une autre. Dans leur équipe RoboCup, Stone et Veloso ont mis en œuvre trois stratégies pour passer à une nouvelle formation et donc changer le rôle de chaque robot.

Formation statique : la formation est fixée par le «locker-room agreement» et ne change jamais. Dans cette stratégie, le rôle que le robot joue est déterminé avant de faire partie de l'équipe et ne change pas pendant sa participation.

Changement de formation pendant l'exécution : lors de rendez-vous prédéterminés dédiés à la synchronisation entre les agents[7], l'équipe décide du changement de formation à effectuer. Cette décision est prise sur la base d'indicateurs de la performance accessibles par tous les membres de l'équipe.

Changement de formation à base de communication : un membre d'un groupe de robots décide que l'équipe devrait changer de formation et communique la décision à ses coéquipiers.

Chaque agent dans l'équipe a connaissance de la formation choisie. Partant de là, il déduit le rôle qu'il doit jouer selon l'information stockée dans le locker-room agreement. Cependant, comme nous l'avons déjà mentionné, la grande limitation de cette approche est que le nombre de robots et de rôles sont identiques. Chaque robot endosse un unique rôle et chaque rôle n'est joué que par un seul robot à la fois.

3.4.3 Stratégie commune pour faire coopérer les robots de sauvetage simulés

Proposé par Kitano [Kitano and Tadokoro, 2001], l'environnement simulé de RoboCup Rescue se compose d'une cité virtuelle qui a subit une grande catastrophe. Dans cet environnement dynamique, des agents intelligents et hétérogènes, doivent doivent coordonner leur actions visant à sauver des gens et des biens. Les agents sont de six types différents : les pompiers, les policiers, les

7. Comme par exemple pendant la pause d'un match de foot.

ambulanciers et trois centres correspondant à chacun des trois types d'agents précédents. Les pompiers sont responsables de l'extinction des incendies, les policiers sont là pour ouvrir les routes bloquées et les équipes d'ambulanciers sont en charge de dégager les blessés. La coopération est réalisée entre tous ces agents à travers les agents-centres.

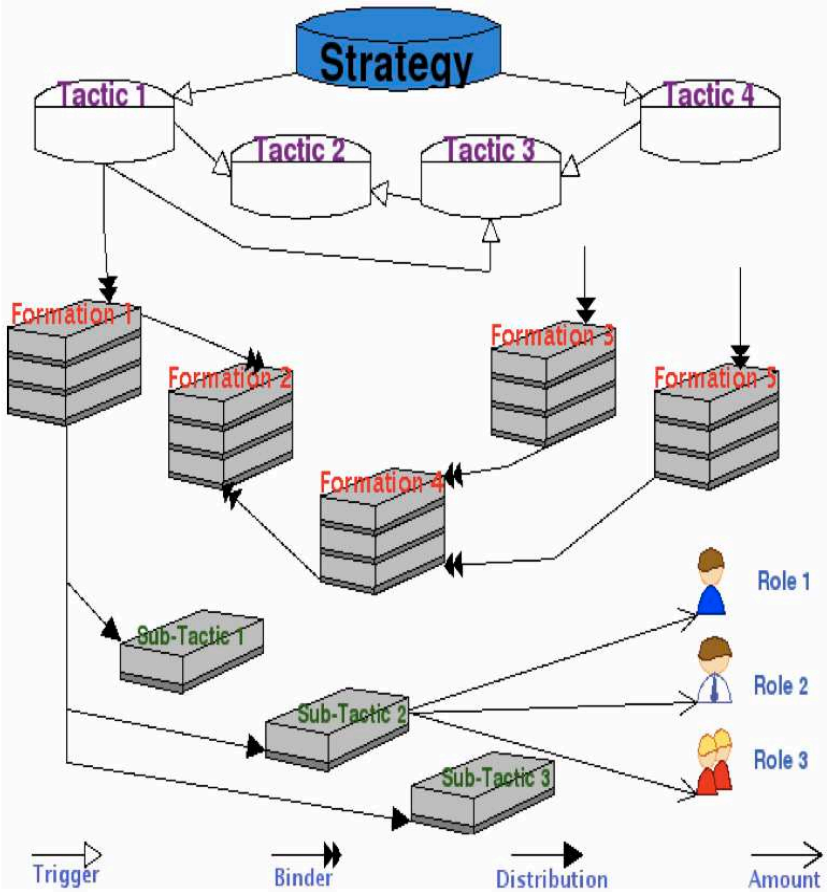


FIGURE 3.6 – *Le schéma de concepts stratégiques [Certo et al., 2007]*

[Certo et al., 2007] proposent une couche stratégique multi-buts et multi-domaines, adaptable pour coordonner différents agents dans le cadre décrit dans le paragraphe précédent. Il s'agit d'une démarche hiérarchique pour décrire une solution en partant du niveau le plus général (un jeu, une mission de sauvetage) et descendre jusqu'au niveau concret (rôles à assigner aux robots). La couche de stratégie définit la structure du système (Figure 3.6). Pour une mission de sauvetage, il y a des *stratégies* différentes qui définissent la coopération générale. Une stratégie est définie selon une structure hiérarchique (Figure 3.6). Cette dernière se compose de *tactiques*, qui regroupent des *formations* alternatives, elles-mêmes composées de *sous-tactiques* qui regroupent plusieurs rôles.

Pour choisir une tactique, les conditions prédéfinies par les concepteurs doivent être satisfaites. De même, des conditions pour activer une formation alternative au sein de la tactique courante devraient être spécifiées par les concepteurs. Une formation est une structure de haut niveau qui regroupe tous les agents avec l'intention de les affecter à des sous-tactiques. L'affectation est réalisée en utilisant des agents du même type ou qui ont les mêmes objectifs immédiats, ou les deux. Au niveau individuel, chaque agent doit adopter un rôle qui régit son comportement.

La couche stratégique – connue par tous les agents (comme le locker-room agreement [Stone and Veloso, 1999]) – permet la gestion des agents homogènes ou hétérogènes, et la gestion centralisée ou décentralisée de la stratégie. On pourrait donc considérer que cette couche stratégique de coordination pour une équipe qui regroupe des agents avec des objectifs communs comme une solution de la structure de travail d'équipe PST [Stone and Veloso, 1999]. Cette extension a été appliquée avec succès dans le domaine de RoboCup Rescue Simulation [Kitano and Tadokoro, 2001] et RoboCup Soccer Simulation [Reis and Lau, 2001]. Contrairement à la structure d'équipe de travail de [Stone and Veloso, 1999], la couche stratégie de [Certo et al., 2007] vise à la coopération des agents hétérogènes. Cependant, l'allocation des rôles est laissée ouverte, car « une attribution optimale de rôle dépend des conditions du scénario comme la proximité des objectifs, des positions des agents, etc. Basé sur ce fait, le modèle ne spécifie pas de méthode. » (cf. [Certo et al., 2007]).

3.4.4 Synthèse sur l'utilisation des rôles en robotique

Notre étude de l'état de l'art de la conception des systèmes coopératifs montre un consensus sur l'idée d'organiser les agents dans une structure organisationnelle hiérarchique autour du concept de rôle. C'est le cas aussi bien dans les systèmes multi-agents, que dans plusieurs travaux robotiques. Indubitablement, cela montre que le rôle est un moyen puissant pour faire coopérer les robots hétérogènes, et donc, les approches basées sur le rôle sont tout à fait faisables.

Cependant, notre étude montre également qu'il manque un mécanisme d'attribution automatique et générale des rôles aux robots avec un certain degré d'optimalité [Parker, 2003, Certo et al., 2007]. D'autre part, dans les applications robotiques (comme dans [Candea et al., 2001]) où le problème d'allocation optimale de rôles aux robots est traité, chaque robot ne peut jouer qu'un rôle ; par conséquent, le nombre de rôles est strictement égal au nombre des robots. De plus, cette proposition concerne des équipes de football, donc avec un nombre de robots toujours inférieur à 11 (hors gardien). Pour les systèmes multi-robots de grande échelle qui nous intéressent, l'allocation de rôles proposée par [Stone and Veloso, 1999, McMillen and Veloso, 2007] est évidemment insuffisante.

3.5 Allocation de tâches et formation de coalitions

Nous avons vu dans la sections précédente que le problème d'allocation de rôles aux robots a été laissé ouvert dans les travaux que nous avons examinés. Cependant, à partir de notre discussion de la section 3.3.1, nous pouvons dire que, du point de vue algorithmique, la nature de ce problème est exactement la même que celle de l'allocation des tâches [Gerkey and Matarić, 2004a]. Nous étudions donc en détail dans cette section l'état de l'art du problème d'allocation de tâches dans les systèmes multi-robots. Nous nous intéressons notamment aux travaux sur la formation de coalitions. Il s'agit en effet d'une classe particulière du problème d'allocation de tâches où plusieurs robots doivent collaborer pour accomplir chaque tâche.

3.5.1 Allocation de tâches dans les systèmes multi-robots

Le problème d'allocation de tâches (*MRTA* [8]) traite la question la plus fondamentale dans les applications robotiques coopératives : comment attribuer les tâches aux robots pour atteindre une solution avec une certaine qualité, c'est-à-dire quelles sont les options qui sont meilleures que d'autres, et comment les choisir ? Quoique partageant le même principe de « *par le haut* » [9], ou « *diviser pour régner* », les détails de chaque approche peuvent varier en fonction du nombre de robots, de la nature des tâches, des relations entre les tâches, etc. [Gerkey and Matarić, 2004a] ont définit une taxonomie pour l'allocation des tâches dans la robotique. Cette taxonomie MRTA est reconnue comme une « norme » *de facto* dans la robotique [Parker, 2008b].

Taxonomie MRTA de Gerkey et Mataric

Gerkey et Mataric [Gerkey and Matarić, 2004b] classifient les problèmes MRTA selon trois critères :

1. *La capacité des robots à traiter des tâches :* Il y a ceux capables d'exécuter plusieurs tâches en même temps, dénotés par l'abréviation *MT* (Multi-Task), et ceux qui ne peuvent travailler que sur une seule tâche à un moment donné, dénotés comme *ST* (Single-Task).
2. *Le nombre de robots requis pour chaque tâche :* Il existe deux types de tâches : les tâches mono-robot *SR* (Single-Robot) et les tâches multi-robots *MR* (Multi-Robot). Une tâche *SR* nécessite exactement un robot pour l'accomplir, tandis qu'une tâche *MR* nécessite que plusieurs robots travaillent de concert pour la réaliser.
3. *La stratégie d'optimisation de l'allocation des tâches :* L'allocation de tâches est optimisée sur la base d'une fonction d'utilité. Deux types de stratégies peuvent être distinguées pour cette optimisation. Il y a d'une part celles dites *IA* (Instantaneous Assignment), qui optimisent localement dans le temps la répartition des tâches. Cette répartition est faite étant donnée la valeur de la fonction d'utilité au moment de la décision. D'autre part, il y a les stratégies dites *TA* (Time-extended Assignment), qui optimisent la répartition en se projetant dans le futur. Pour ce faire, elles estiment l'utilité cumulée à long terme (typiquement une fenêtre de temps à venir).

En utilisant la taxonomie MRTA, un problème d'allocation de tâches est désigné par trois acronymes à deux lettres correspondants. Chaque acronyme correspond à une valeur de l'un des trois critères précédents. Par exemple, *SR-ST-IA* désigne le problème d'allocation de tâches le plus simple. En effet, dans ce problème, chaque tâche nécessite un unique robot. De plus, chacun des robots utilisés est mono-tâche. Enfin, l'allocation est faite avec une stratégie d'optimisation prenant la valeur instantanée de la fonction d'utilité.

Nous terminons cette présentation en indiquant qu'il y a 8 (2^3) combinaisons possibles et donc familles possibles de problèmes d'allocation de tâches. Il s'agit de *ST-SR-IA*, *ST-SR-TA*, *ST-MR-IA*, *ST-MR-TA*, *MT-SR-IA*, *MT-SR-TA*, *MT-MR-IA*, et *MT-MR-TA*. Leur description est détaillée dans [Gerkey and Matarić, 2004b].

8. Multi-Robot Task Allocation
9. top-down

Complexité du problème d'allocation de tâche et évaluation des solutions

En terme de complexité de calcul, la classe de problème *SR-ST-IA* (Single-robot task, single-task robot, instantaneous assignment) est la plus facile. Les autres sont beaucoup plus difficiles car ils sont tous $\mathcal{NP}$-difficiles. Par conséquent, seul pour la variante *SR-ST-IA*, qui est en effet une instance du problème d'allocation optimale de travail [10] bien connu depuis longtemps [Gale, 1960], il existe des solutions polynomiales. Pour les autres, nous devons nous satisfaire de solutions approximatives.

Comme la recherche en robotique évolue et les tâches à traiter deviennent de plus en plus complexes, travailler uniquement avec des tâches *SR* se révèle être une simplification excessive. Dans de nombreux domaines, les problèmes avec des tâches du type *MT*, nécessitant une équipe de robots par tâche sont fréquents. Ainsi, la classe de problème *MR-ST-IA* (Multi-robot task, single-task robot, instantaneous assignment) a attiré ces dernières années beaucoup d'attention dans la communauté robotique [Chaimowicz et al., 2004, Tang and Parker, 2005a, Tang and Parker, 2005b, Parker and Tang, 2006, Vig and Adams, 2006, Tang and Parker, 2007, Vig and Adams, 2007].

Pour évaluer les solutions, nous utilisons les trois critères suivants :

- *la complexité du calcul* pour arriver à une solution.
- *la communication* qui est évaluée par le nombre de messages échangés.
- *la qualité de la solution.* Hormis le problème *SR-ST-IA* qui a des solutions optimales, les autres problèmes ont des solutions de plus ou moins bonne qualité. Afin de mesurer cette qualité, il est courant d'associer un facteur de compétitivité. Ainsi, pour un problème de maximisation, un algorithme est appelé α-compétitif si, pour une entrée, il trouve une solution dont l'utilité n'est jamais inférieure à $\frac{1}{\alpha}$ de l'utilité optimale.

Avec ces critères, Gerkey et Mataric ont analysé quelques approches représentatives des problèmes *MR-ST-IA*. Nous reprenons leur présentation dans le tableau 3.1.

Formation de coalitions croisées et taxonomie MRTA

Les problèmes de la classe *MR-ST-IA* sont souvent appelés problèmes de formation de coalitions [Tang and Parker, 2005b, Parker and Tang, 2006, Vig and Adams, 2007]. Ce terme vient des systèmes multi-agents. Nous nous intéressons dans cette thèse (voir section 1.3.3, page 9) à une variante de ce problème : la formation de coalitions *croisées* . En utilisant la taxonomie de Gerkey et Mataric, la formation des coalitions croisées correspond aux deux problèmes *MR-MT-IA* ou *MR-MT-TA*. En effet, une tâche nécessite une coalition de plusieurs robots pour être résolue (Multi-Robot task). De plus, les robots sont supposés être capables de jouer plus d'un rôle en même temps (Multi-Task robot) et peuvent donc participer à plusieurs coalitions (d'où l'adjectif *croisées*).

Les deux classes *MR-MT-IA* et *MR-MT-TA*, dont notre problème est une instance, ne sont pas encore étudiées dans la littérature (notre solution présentée dans le chapitre 5 est la première !). Les raisons sont, d'une part, qu'elles sont

10. Optimal Assignment Problem

TABLE 3.1 – *Résumé de la complexité de calcul et de la communication, ainsi que de la qualité des solutions des approches sur l'affectation en-ligne de tâches [Gerkey and Matarić, 2004a]*

Nom	Calcul par tâche	Communication par tâche	Qualité de la solution
MURDOCH [Gerkey and Matarić, 2002]	$O(1)$ par enchérisseur $O(n)$ par commissaire-priseur	$O(n)$	3-compétitive
enchères au premier prix [Dias, 2004]	$O(1)$ par enchérisseur $O(n)$ par commissaire-priseur	$O(n)$	au moins 3-compétitive
Dynamique affectation des rôles [Chaimowicz et al., 2004]	$O(1)$ par enchérisseur $O(n)$ par commissaire-priseur	$O(n)$	au moins 3-compétitive

$\mathcal{NP}$-difficiles, et, d'autre part, que « les robots qui sont capables de réaliser plusieurs tâches en parallèle sont encore au delà de l'état de l'art [Parker, 2008b]. » Cependant, il y a des tâches sensorielles et de calcul qui correspondent assez bien à ces deux modèles, comme le scénario dans [Gerkey and Matarić, 2004a] :

> Considérons la répartition des tâches de surveillance d'une équipe de robots dans un immeuble de bureaux. Chaque robot patrouille dans une partie fixe de l'immeuble. En raison des capacités de calcul et/ou des limitations sensorielles, chaque robot ne peut détecter simultanément qu'un nombre limité d'événements environnementaux (par exemple : personne suspecte, fumée, porte ouverte). Étant donné un ensemble d'événements à chercher, et des connaissances sur l'endroit dans le bâtiment où chaque événement est susceptible de se produire, quel robot devrait être assigné pour chercher quel événement ?

La section suivante présente notre étude des travaux existants les plus proches du problème de formation des coalitions croisées traité dans cette thèse (un problème *MR-MT-IA*). Ce sont les solutions pour les problèmes de la famille *MR-ST-IA*. Les tâches nécessitent plus d'un robot pour être résolues, mais contrairement à notre problème, les robots ne peuvent réaliser qu'une seule tâche à la fois. Enfin, l'allocation de tâches est optimisée localement dans le temps.

3.5.2 Familles d'approches pour la formation de coalitions dans les systèmes multi-robots

Le problème de formation des coalitions qui est $\mathcal{NP}$-difficile, a été profondément étudié dans la communauté multi-agents. Les solutions connues sont des algorithmes heuristiques [Shehory and Kraus, 1998].

Une approche naïve consisterait à appliquer les solutions développées pour des systèmes multi-agents sur des systèmes multi-robots. Cependant, selon [Vig and Adams, 2007], à cause des différences entre un agent logiciel et un robot situé dans un environnement physique, ces solutions ne sont pas applicables directement. Un exemple de ces différences est que les capacités des robots sont plutôt statiques car elles dépendent fortement de leur configuration matérielle. Les échanges de capacités comme dans le cas des agents logiciels ne sont pas possibles entre des robots.

Par ailleurs, la formulation du problème de formation des coalitions dans les systèmes multi-agents est souvent statique, dans le sens où l'ensemble de tâches, ainsi que l'ensemble des agents, sont connus a priori. Quant aux systèmes multi-robots, peu d'applications admettent une allocation en une seule fois (par exemple au début de la mission). L'allocation doit prendre en compte le dynamique de l'environnement, ainsi que les ressources de l'équipe de robots comme la tâche d'observation des objets mobiles étudiée dans [Parker, 1999] où les tâches peuvent changer pendant l'exécution du système [Gerkey and Matarić, 2002]. Cette dynamique de l'environnement nécessite une dynamique dans la formation des coalitions. Il existe deux variantes : la formation répétitive et la formation en-ligne.

Formation répétitive

L'algorithme de base est le suivant :

1. S'il y a des robots qui sont encore sans occupation, alors essayer de former une coalition pour une tâche non-résolue de manière optimale.
2. Retirer les robots de l'éventuelle coalition déjà en charge de la tâche en question.
3. Confier la tâche à la nouvelle coalition.
4. Revenir à l'étape 1 (Répéter indéfiniment).

Cet algorithme est mis en œuvre dans RACHNA [Vig and Adams, 2007] et dans une extension d'ASyMTRe-D [Tang and Parker, 2007] (voir la section 3.5.3). Il partage le même principe avec le BLE (Broadcast of Local Eligibility) mis en œuvre dans [Werger and Matarić, 2001] pour résoudre l'allocation répétitive *ST-SR-IA*.

Formation en-ligne

La formation en-ligne est destinée à un cas particulier de la classe de problèmes *MR-ST-IA* : les tâches sont supposées apparaître dynamiquement une par une. Une coalition est formée à l'apparition d'une nouvelle tâche. Cette hypothèse est très réaliste dans les applications robotiques. Une telle situation peut avoir lieu très souvent dans des environnements hautement dynamiques, tels que la robotique de sauvetage. Le squelette de l'algorithme est le suivant.

1. S'il y a une nouvelle tâche, alors former une coalition pour la résoudre.
2. Revenir à l'étape 1 (Répéter indéfiniment).

Cet algorithme a été implémenté dans [Chaimowicz et al., 2004] (c.f. section 3.5.3). Nous pouvons le considérer comme une variante de l'algorithme mis en œuvre dans MURDOCH [Gerkey and Matarić, 2002] pour le problème *SR-ST-IA*.

3.5.3 Travaux dans la robotique sur la formation de coalitions

Nous présentons ici trois travaux qui nous semblent représentatifs de la formation des coalitions. Parmi les trois travaux, seule l'allocation en-ligne de rôles de [Chaimowicz et al., 2004] a été évaluée formellement par [Gerkey and Matarić, 2004b] (cf. le tableau 3.1). Pour les deux autres, leurs auteurs les ont évalué à l'aide de simulations. C'est pour cette raison que nous ne pouvons pas effectuer une comparaison à base des critères introduits dans la section 3.5.1 (page 46). Enfin, il est intéressant de noter que ces travaux, comme toutes les autres propositions sur la formation des coalitions, ont recours à un système à base d'enchères pour l'allocation des tâches.

Allocation en-ligne de rôles pour le transport collaboratif

[Chaimowicz et al., 2004] définissent un rôle comme une fonction qu'un ou plusieurs robots remplissent pour l'exécution d'une tâche coopérative. Chaque robot prend un rôle quand certaines conditions internes et externes sont satisfaites. Le rôle sert à définir le comportement du robot, y compris l'ensemble des contrôleurs utilisés par le robot, les informations qu'il envoie et reçoit, et comment il va réagir à la présence d'événements dynamiques et inattendus. Le rôle est donc le concept central dans leur approche pour la tâche de transport collaboratif. Une équipe de robots cherche des objets dispersés sur le terrain. Les robots doivent collecter ces objets et les déplacer vers un endroit prédéfini. Pour chaque objet trouvé, il faut une coalition de robots. Un seul robot de la coalition joue le rôle de *leader*, qui participe au transport de l'objet et au calcul du chemin à suivre. Les autres membres jouent tous le rôle de *follower*. Ils participent au transport de l'objet en suivirant le chemin calculé par le *leader*.

Le mécanisme de formation dynamique des coalitions est le suivant. Lorsqu'un robot détecte un objet, il prend le rôle *leader*, puis lance l'appel de participation pour compléter la formation de la coalition. D'autres robots qui reçoivent l'appel répondent avec un « prix », puis ils attendent la confirmation d'attribution du rôle de *follower* dans la coalition. Sur la base de la règle « premier arrivé–premier servi » ainsi qu'en fonction des prix (enchère), le *leader* recrute des *followers* jusqu'à ce qu'il y ait assez de robots pour déplacer l'objet.

Une fois engagé dans une coalition, un robot ne la quitte qu'à la terminaison de la tâche. Mais les robots d'une coalition peuvent échanger leurs rôles pendant la durée de vie de la coalition. En effet, il se peut que le *leader* ne soit plus en mesure de calculer le chemin correctement. La coalition ne progresse alors plus dans l'accomplissement de la tâche. Dans ce cas, le *leader* peut échanger son rôle avec un robot *follower* pour pouvoir mener la tâche à bien.

Formation des coalitions dans RACHNA

[Vig and Adams, 2007] proposent des heuristiques pour la formation des coalitions, similaires à celles issues des travaux de [Shehory and Kraus, 1998] pour des systèmes multi-agents. Cependant, comme indiqué par Vig et Adams, la formation de coalition de Shehory et Kraus est réalisée de manière statique. De plus, leurs algorithmes heuristiques fonctionnent bien lorsque la taille des coalitions est limitée. Quand les tâches exigent des coalitions d'une taille plus grande, la solution peut être arbitrairement mauvaise. Le système RACHNA a été conçu pour surmonter ces limitations.

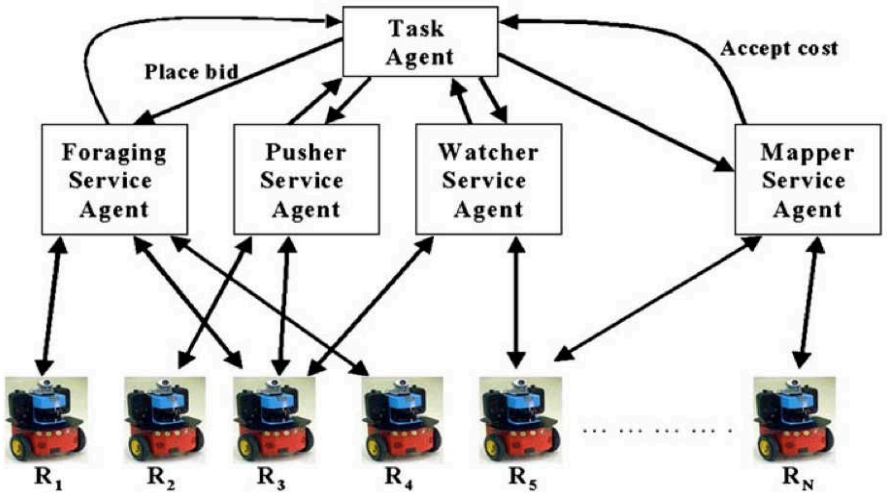


FIGURE 3.7 – *Un exemple de mise en œuvre de* RACHNA *[Vig and Adams, 2007]*

Dans le système RACHNA (voir le figure 3.7), il y a des *Service Agents* qui gèrent un (sous-)ensemble de robots qui fournissent chacun un service donné (Foraging Service Agent, Pusher Service Agent, etc.). Il y a un *Task Agent* par tâche. Pour former une coalition, un *Task Agent* propose offre un *salaire* aux *Service Agents* qui gèrent les robots fournisseurs des services requis. Chaque *Service Agents* qui accepte le salaire proposé, sélectionne parmi les robots qu'il gère et leur assigne des rôles dans la coalition. Ainsi, contrairement à d'autres travaux, le processus d'enchère dans RACHNA est renversé. C'est le *Task Agent* qui fixe un prix pour acheter un service particulier, alors que les *Service Agents* décident seulement s'ils acceptent de vendre les services des robots qu'ils gèrent au *Task Agent*.

Notons qu'ici aussi, à un moment donné, un robot ne peut participer qu'à une seule coalition.

Synthèse automatisée des solutions avec ASyMTRe-D

Parker et Tang introduisent une extension de la théorie des schémas [Arkin, 1987, Lyons and Arbib, 1989] pour permettre le partage de capteurs entre robots [Parker et al., 2005]. Cette extension est implémentée dans ASyMTRe-D [Tang and Parker, 2005a, Parker and Tang, 2006]. L'idée fondamentale d'ASyMTRe-D est de considérer les fonctionnalités fournies par les robots comme un ensemble de « schémas ». Il y a des schémas de perception, des schèmas de moteurs et des schémas de communication qui sont préprogrammés dans le robot au moment de la fabrication. Un schéma est caractérisé par des

entrées et des sorties de certains type d'information. Deux schémas peuvent être connectés si leurs entrées et leurs sorties correspondent. Ainsi, les schémas peuvent être connectés de manière autonome à l'intérieur ou à travers des robots sur la base des flux d'informations nécessaires pour accomplir une tâche. Avec la capacité de se connecter dynamiquement lors de l'exécution, plusieurs solutions peuvent être configurées de nombreuses façons pour résoudre la même tâche ou peut être reconfigurées pour résoudre une nouvelle tâche.

Face à une tâche multi-robots, l'approche dans ASyMTRe-D permet à des schémas différents de se connecter de façon autonome et dynamique en fonction de l'état de l'équipe de robots au moment de l'exécution, au lieu d'utiliser des connexions pré-définies. Ainsi, au lieu de représenter l'abstraction d'une solution possible pour une tâche comme un ensemble de compétences nécessaires, ASyMTRe-D la représente comme un ensemble de schémas connectés (Figure 3.8).

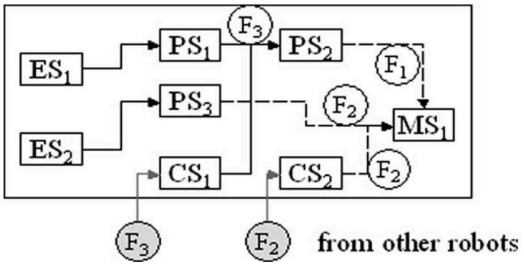


FIGURE 3.8 – *Un exemple de la façon dont les schémas sont reliés pour accomplir une tâche [Parker and Tang, 2006]*

Bien que ASyMTRe-D fournit un mécanisme pour générer une solution de travail coopératif d'une équipe de robots hétérogènes, cette proposition est limitée à une seule coalition à la fois. Néanmoins, une extension de ASyMTRe-D a été proposée dans [Tang and Parker, 2007], dans lequel un mécanisme d'enchères est utilisé pour former plusieurs coalitions. Cependant, un robot ne peut pas participer à plus d'une coalition.

ASyMTRe-D est intéressant car il permet la spécification de solutions pour résoudre une tâche au niveau du schéma de façon indépendante du système concret de robots. Le problème est que l'ASyMTRe-D est basé sur des schémas bas-niveau. Cela constitue une restriction, en ne fournissant qu'une unique implantation par solution.

Conclusion du chapitre

L'étude présentée dans ce chapitre montre que le rôle est un excellent moyen pour concevoir les systèmes coopératifs. Imaginer une coalition composée de rôles pour résoudre une tâche permet aux concepteurs de rester à un haut niveau d'abstraction lors de la conception de la solution. Ainsi, la conception peut se faire en ignorant les robots qui seront utilisés. Cette approche permet de concevoir des stratégies qui permettent aux équipes de robots de se configurer et de manière autonome en équipes et sous-équipes en fonction des objectifs courants de la mission, des robots disponibles et de leurs ressources.

L'utilisation de robots hétérogènes capables de jouer plusieurs rôles en même temps (et donc participer à plus d'une coalition), rend le problème de formation

des coalitions $\mathcal{NP}$-difficile. En effet, il est sans solution satisfaisante comme le montre l'état de l'art. Nous proposons dans le chapitre 5 une réponse qui, sans être idéale, permet garantir un certain niveau d'optimalité.

Deuxième partie

Contributions

Sensibilité à la connectivité 4

Sommaire

Dans ce chapitre nous fournissons une solution pour déterminer (de manière distribuée) si un déplacement de robot peut potentiellement causer la déconnexion d'un robot quelconque avec le reste de l'équipe. L'algorithme que nous proposons accepte un ensemble de mouvements possibles en entrée, et retourne un déplacement qui ne rompt pas la connectivité. Notre proposition se découpe en deux parties : (i) rendre les robots conscients de la connectivité de réseau, nous appelons cette connaissance « la sensibilité à la connectivité », et (ii), employer cette information afin de planifier les tâches des robots sans compromettre la connectivité.

Pour le point i), nous proposons un nouvel algorithme distribué qui sera exécuté sur chaque robot pour le rendre sensible à la connectivité. Notre solution au point ii) comprend deux algorithmes qui exploitent la sensibilité à la connectivité dans deux applications. Le premier algorithme consiste en la vérification

de la robustesse d'un réseau sans fil, cet algorithme présente l'avantage d'avoir un coût de communication très bas en comparaison avec les travaux existants. Le deuxième algorithme utilise la sensibilité à la connectivité dans la planification de déplacement des robots pour maintenir la connectivité.

Tous ces algorithmes sont décentralisés et nous fournissons également des bases théoriques pour en faire l'analyse et prouver un certain nombre de propriétés.

4.1 Formalisation du problème du maintien de la connectivité

4.1.1 Planification du déplacement

Considérons un système de n robots en réseau $\mathcal{R} = \{R_1, R_1, \ldots, R_n\}$. Ces robots fonctionnent et sont coordonnés par un algorithme de coordination quelconque. L'environnement est hautement dynamique : d'une part par la nature de la mission, d'autre part par la modification de l'environnement par les robots. L'algorithme de coordination doit donc spécifier les instants où les robots doivent échanger les données nécessaires pour la coordination (par exemple, mettre à jour le modèle interne du monde, ou synchroniser les informations collectées par les robots). Ces échanges sont nécessaires pour que les robots prennent en compte le changement de l'environnement. Comme dans les travaux similaires [Burgard et al., 2005, Sheng et al., 2006b, Rooker and Birk, 2007, Schuresko, 2009, Schuresko and Cortés, 2009], nous proposons l'hypothèse suivante de discrétisation du temps.

Hypothèse de discrétisation du temps Supposons que les robots évoluent dans un temps discrétisé $T = \{t_0, t_1, \ldots, t_l\}$. À l'instant $t_i \in T$, le robot prend en compte les modifications de l'environnement et planifie les actions à réaliser pendant l'intervalle $[t_i, t_{i+1}]$. Nous supposons également que les horloges des robots sont synchronisées tout au long de la mission.

Définition 4.1 (Déplacement planifié du robot) *Un déplacement planifié m^{R,t_i} pour le robot R à l'instant $t_i \in T$ est un déplacement que le robot R est capable d'accomplir pendant l'intervalle $[t_i, t_{i+1}]$.*

Nous visons à développer une solution *générique* pour le maintien de la connectivité. Pour cela, nous séparons les préoccupations liées au maintien de la connectivité de l'algorithme de coordination pour le contrôle de déplacement d'une flotte de robots. Nous avons la définition suivante.

Définition 4.2 (Algorithme de coordination sous-jacent pour le contrôle du déplacement) *L'algorithme de coordination sous-jacent pour le contrôle du déplacement appliqué au robot R doit produire un ensemble de déplacements possibles $M_R = \{m_1^{R,t_i}, m_2^{R,t_i}, ..., m_k^{R,t_i}\}$ à l'instant $t_i \in T$ pour le robot R.*

Dans la pratique, un ordre de préférence sera appliqué à ces actions selon des critères spécifiques à chaque application.

4.1.2 Problème du maintien de la connectivité

Définition 4.3 (Problème du contrôle distribué des déplacements avec maintien de la connectivité) *Étant donné l'ensemble M_R des déplacements planifiés au moment t, le problème de contrôle distribué de mouvement avec maintien de la connectivité consiste à fournir une procédure distribuée à exécuter sur chaque robot R pour trouver un $m_i^{R,t} \in M_R$ tel qu'après la réalisation de $m_i^{R,t}$, le graphe du réseau reste connexe.*

Selon cette définition du problème, les préoccupations concernant le maintien de la connectivité sont bien séparées de celles de la planification des déplacements des robots. L'avantage de cette formulation est que nous pouvons étudier le problème du maintien de la connectivité indépendamment du déplacement (voir définition 4.2, page 57). Par conséquent, la solution trouvée pourra être réutilisée dans d'autres applications.

Généralement, nous pouvons donner un squelette d'une solution décentralisée au problème du contrôle de déplacement des robots, couplé avec le maintien de la connectivité comme dans l'algorithme 1. L'algorithme est à exécuter sur chaque robot pour un intervalle $[t_i, t_{i+1}]$.

Algorithme 1: Squelette de la solution du maintien de la connectivité

début

Étape 1 : exécuter l'algorithme de coordination afin de trouver l'ensemble M des déplacements planifiés (cf. définition 4.1);

Étape 2 : sélectionner un $m \in M$ en prenant compte le maintien de la connectivité;

Étape 3 : effectuer le déplacement sélectionné m;

Étape 4 : communiquer avec les voisins en fonction des applications.

fin

L'avantage de la séparation des préoccupations dans cet algorithme est évident. Cela nous permet de « laisser » à part les étapes 1, 3 et 4 [1] pour nous intéresser à l'étape de sélection d'un déplacement (c'est-à-dire l'étape 2).

4.2 Éléments de base de la sensibilité à la connectivité

4.2.1 Nœud de référence

Partant d'un réseau connexe, l'objectif principal du maintien de la connectivité est de garantir la connectivité globale permanente du réseau. Dans ce contexte, nous prenons un nœud r, et nous l'appelons le *Nœud de Référence* [2]. Entre le nœud r et n'importe quel nœud u dans le réseau, il devrait alors exister un chemin de communication $p(u, r)$. La définition 2.3, page 21, de la connectivité d'un graphe représentant un système multi-robot peut être reformulée comme suit.

Définition 4.4 (Graphe Référentiellement Connexe) *Etant donné un Nœud de Référence $r \in V$, le graphe G est appelé connexe* référentiellement *si et seulement si $\forall u \in V$, il existe un chemin $p(u, r)$.*

1. Rappelons que nous regardons ces problèmes comme les préoccupations spécifiques liées aux application diverses. Quant au maintien de la connectivité, les solutions pourraient être trouvées indépendamment de ces problèmes.

2. Dans le contexte d'un réseau de robots où chaque robot est un nœud, et vice versa, nous utilisons de façon interchangeable le terme nœud et robot, ainsi que le terme Nœud de Référence, ou Robot de Référence.

Dans notre approche décentralisée, cette définition équivalente sur la connectivité d'un graphe apporte un grand avantage. Si les robots dans le réseau établissent un consensus sur un Nœud de Référence, la tâche de maintien de la connectivité est alors considérablement réduite. Car dans cette situation chaque robot ne doit être sûr que de la connexion avec le Nœud de Référence, et pas avec tous les autres robots du réseau. C'est l'idée fondamentale dans notre approche du maintien distribué de la connectivité du réseau.

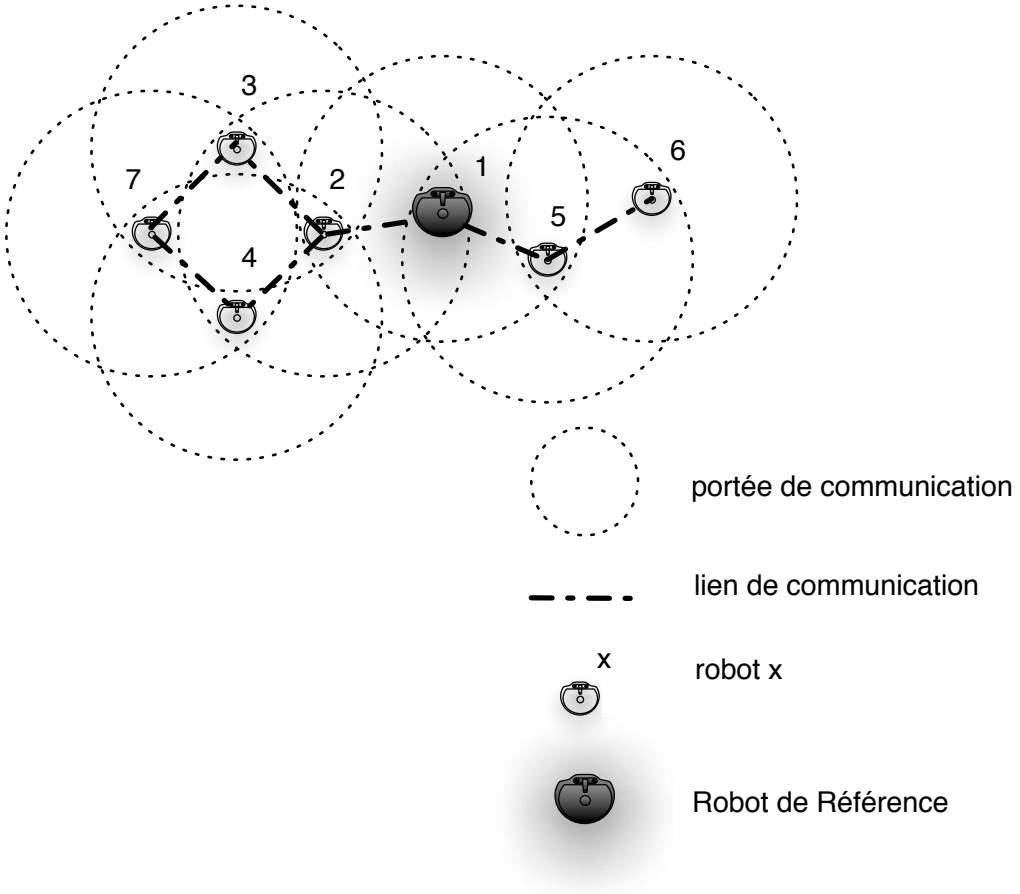


FIGURE 4.1 – *Exemple de robots en réseau où le robot 1 est choisi comme Nœud de Référence*

De manière générale, la première chose à faire dans notre solution du maintien de la connectivité est de définir un Nœud de Référence « fixé », puis chaque robot de l'équipe doit, tout en accomplissant sa tâche, rester en contact avec *au moins un robot voisin ayant un chemin vers le Nœud de Référence*. Concrètement, dans le système illustré à la figure 4.1, si le robot 1 est choisi comme Nœud de Référence, alors le robot 5 et le robot 2 doivent maintenir leurs connexions avec le robot 1. Le robot 6 doit rester en contact avec le robot 5. Le robot 3 et le robot 4 doivent se déplacer sans rupture des liens entre eux et le robot 2. Le robot 7 doit maintenir une connexion soit au robot 3 soit au robot 4. Ainsi, la connectivité du système sera assurée. En tant que tel, la charge du maintien de la connectivité de réseau est répartie entre les robots.

Autrement dit, le choix d'un Nœud de Référence permet de répartir équitablement la tâche du maintien de la connectivité sur les robots individuels. Cette distribution est réalisée par la localisation d'une propriété globale en vue des individus dans le réseau.

4.2.2 Chemin d'accès

Définition 4.5 (Robot d'Accès et Chemin d'Accès) *Étant donné le Robot de Référence r et deux robots voisins différents $u, v \in V$, v est appelé un robot d'accès de u si et seulement si il existe une arête $e\{u, v\} \in E$ et il existe un chemin $p(v, r)$ tel que $u \notin p$. Nous appelons p un* chemin d'accès.

La sensibilité à la connectivité est définie comme une connaissance sur le robot de référence et sur les robots d'accès.

Cette connaissance permet de maintenir la connectivité *du point de vue local* de chaque robot. Car un robot sait lesquels de ces voisins sont importants à suivre pour rester connecté au robot de référence, et ainsi à l'ensemble des autres robots. Par exemple, dans la figure 4.1 ci-dessus, le robot 5 sait qu'il a besoin de maintenir la connexion avec le robot 1, et le maintien du lien $(5, 6)$ est de la responsabilité du robot 6.

4.2.3 Tableaux de connectivité

La sensibilité à la connectivité pour un robot donné est matérialisée sous la forme d'un *Tableau de Connectivité* (ou désormais simplement le Tableau tout court s'il n'y a pas de confusion) stocké sur chaque robot. Ce tableau contient un ensemble de chemins d'accès qui représentent une vue partielle sur la connectivité du réseau. Des exemples de ces tableaux sont donnés dans la table 4.1, page 60.

TABLE 4.1 – *Exemples des tableaux de connectivité de quelques robots dans la Figure 4.1, le robot 1 est le Robot de Référence.*

(a) Robot 3

Robot d'accès	Chemin d'accès
2	$(2, 1)$

(b) Robot 7

Robot d'accès	Chemin d'accès
3	$(3, 2, 1)$
4	$(4, 2, 1)$

La section suivante présente notre algorithme pour construire le tableau de connectivité. Par souci de simplification de l'explication, nous faisons l'hypothèse que la topologie du réseau ne change pas pendant l'exécution de l'algorithme. En fait, tous les résultats présentés peuvent être retrouvés sans cette hypothèse. Ainsi dans la section 4.4, le maintien du tableau de connectivité dans la présence de la mobilité est introduit. Les simulations avec des robots se déplaçant présentés dans le chapitre 6 valident également notre approche. Nous supposons qu'au début d'une mission, les robots sont à proximité les uns des autres et forment un réseau connexe, par conséquent, ils peuvent communiquer au moyen d'un protocole de routage. En plus, les messages envoyés par un nœud sont reçus correctement pendant une durée de temps finie (un *pas de simulation*) par *tous ses voisins*. Chaque nœud connaît son identifiant (ID) et les identifiants de tous ses voisins. L'identifiant d'un robot est unique au niveau du système. Les préoccupations principales pour fournir la sensibilité à la connectivité du réseau

se résument donc aux problèmes de sélection d'un Robot de Référence et de la construction des tableaux de connectivité.

4.3 Construction de la sensibilité de connectivité

Afin de rendre les robots sensibles à la connectivité, nous construisons des tableaux de connectivité. Après avoir choisi au préalable un Nœud de Référence, nous calculons des chemins d'accès et en les propageant dans le réseau, les tableaux de connectivité des différents robots peuvent être mis en place.

4.3.1 Choix d'un Nœud de Référence

Le choix d'un Nœud de Référence peut être dépendant de l'application spécifique. Il peut être effectué sur la base de plusieurs critères tels que le niveau d'énergie, le nombre de voisins, les besoins en matériel, etc. Dans certaines situations, les rôles des robots permettent plus facilement de faire un choix. Par exemple, dans des systèmes de type leader/followers, le leader est un bon candidat pour devenir le nœud de référence. Un autre exemple est celui d'une équipe de robots qui nécessite de maintenir la connectivité avec une station fixe. Celle-ci peut être naturellement choisie pour être le nœud de référence.

Nous proposons une méthode simple pour la sélection d'un Nœud de Référence pour le cas le plus général, où les robots ne sont pas nécessairement connectés sur le même réseau. Lors du démarrage d'un robot, si ses voisins sont pas affiliés à aucun réseau avec un robot de référence, et son ID est plus grand que celui de tous ses voisins, alors le robot considère qu'il est le Nœud de Référence, puis il lance le processus de construction des Tableaux de Connectivité décrit dans la section 4.3.2 suivante. Si non, il s'attache à un réseau existant, ou attend à s'attacher son robot voisin dont l'ID est plus grand que le sien. Bien évidement, il se peut que plusieurs robots se déclarent comme Nœuds de Référence. La fusion de ces « sous-réseaux » est décrite dans la section 4.4 (page 65).

Une autre option pour pour déterminer le Nœud de référence consiste à employer un mécanisme d'enchère pour le sélectionner sur la base des critères quelconques arbitraires fixés par le concepteur du système multi-robots.

4.3.2 Construction des Tableaux de Connectivité

Une fois choisi, le Robot de Référence diffuse à tous ses voisins directs un message `New-Access-Path`, avec le chemin d'accès composé de son propre identifiant. Nous considérons deux stratégies pour propager cette information aux autres robots afin qu'ils construisent leurs propres tableaux de connectivité : 1) retransmission simple et, 2) retransmission avec filtrage.

Retransmission simple des informations de connectivité

À la réception d'un message `New-Access-Path`, le robot extrait le chemin d'accès (voir algorithme 7). Au moyen de cette information, il est capable de

Algorithme 2: Algorithme à exécuter lors de la réception d'un message `New-Access-Path` (version simple).

Entrées : Le message `New-Access-Path` M
Sorties : Le Tableau de Connectivité T mis à jour

```
1 début
2     p ← le chemin d'accès dans M;
3     si this.id() ∉ p alors
4         ajouter le chemin d'accès p au tableau de connectivité T ;
5         mettre à jour et faire suivre le message à tous les voisins;
6     fin
7 fin
```

vérifier si ce message a déjà été reçu précédemment ou non. Cette étape de vérification assure que le chemin ne comprend pas de boucle. Si le message a déjà été traité, il est simplement ignoré. Sinon, le robot va ajouter le chemin dans son tableau et retransmet le message `New-Access-Path` (mis à jour avec son propre ID ajouté au chemin d'accès) à ses voisins. Le message se répand donc progressivement dans le réseau. Tant que les nœuds sont accessibles depuis le Nœud de Référence, ils recevront le message et construiront leur propre Tableau de Connectivité.

Nous allons maintenant montrer la validité du processus de construction de tableaux de connectivité tel que défini dans l'algorithme 7. Nous faisons les propositions suivantes.

Théorème 4.1 *Le processus de construction des tableaux de connectivité est correct et il n'y pas de chemin d'accès avec boucle. Ce processus se terminera au bout de $l^{ième}$ l'étape, où l est la longueur du chemin d'accès le plus long dans le réseau en terme de nombre de sauts* [3].

Démonstration. Une boucle se produit dans le réseau quand un message qui a été traité par un robot i revient à celui-ci et ce dernier continue à le traiter. La ligne 3 de l'algorithme 7 veille à ce que les boucles soient filtrées.

Le processus de construction de tableaux de connectivité est initié par le Robot de Référence et se propage dans le réseau, étape par étape.

Après k étapes, les nœuds qui sont à une distance de *k-sauts* du Nœud de Référence sont capables de construire des chemins composés de $k-1$ nœuds intermédiaires. Comme l est la longueur en terme de nombre de sauts du chemin d'accès le plus long, au bout de l étapes tous les robots auront construit leur tableaux et les boucles seront filtrées. □

Théorème 4.2 *Dans le pire des cas (i.e. lorsque le graphe est complet), la complexité spatiale pour le stockage d'un tableau de connectivité et la complexité en nombre de messages de l'algorithme 7 est $O(n!)$.*

Démonstration. Lorsque le graphe est complet, les robots dans le réseau sont tous voisins les uns des autres. Un robot a donc $n-1$ robots voisins. Pour le robot i, le

3. "hop" en anglais [Perkins, 2001]

message délivré par le Robot de Référence est reçu (et un chemin sera ajouté dans le Tableau) par l'intermédiaire de tous les chemins acycliques. Dans un graphe complet, il y a $(n-2)![1+\frac{1}{1!}+\frac{1}{2!}+...+\frac{1}{(n-2)!}]$ tels chemins. La complexité de l'espace pour sauvegarder le tableau de connectivité est donc $O(n!)$

En ce qui concerne le nombre de messages, le Robot de Référence envoie d'abord le message à $n-1$ robots. À l'étape suivante, tous les $n-1$ robots vont transmettre le message à tous leurs voisins, c'est pourquoi il y a $n-1$ messages. Un message cessera d'être transmis quand il revient vers le nœud qui l'a envoyé auparavant. Ainsi, à l'étape k, il existe $n-k-1$ robots qui continuent d'envoyer des messages. En conséquence, nous avons au total $1+(n-1)!$ messages ou, en d'autres termes, la complexité des messages de l'algorithme est $O(n!)$. □

Retransmission des informations de connectivité avec filtrage

Dans l'algorithme 7 présenté sur la page 62, les robots retransmettent tous les chemins acycliques qu'ils reçoivent. Cette stratégie construit sur chaque robot une vue complète de la connectivité, mais au prix d'un grand nombre de messages lorsque le réseau est très dense.

Il est à noter que les informations stockées dans le Tableau de Connectivité ont pour objectif d'aider à reconnaître les robots d'accès parmi les robots voisins. Les chemins d'accès servent à éviter les boucles dans la retransmission de messages et à gérer la mobilité dans les systèmes multi-robot (voir la section 4.4 page 65). Par ailleurs, nous remarquons qu'une fois qu'un robot, à savoir R_s, fait suivre le premier chemin qu'il a reçu, en provenance du robot voisin R_o, alors tous les robots voisins du robot R_s (excepté R_o) propagent le chemin d'accès ainsi obtenu. Ainsi, tous ces robots considèrent R_s comme un de leurs robots d'accès sauf ceux qui sont référencés dans le chemin. À partir de ce moment, il ne reste que le robot R_o qui ne considère pas le robot R_s un robot d'accès. Donc le robot R_s ne retransmet qu'un autre chemin d'accès qui ne contient pas R_o afin de se rendre lui-même le robot d'accès du dernier. Par conséquent, si un robot peut devenir un nœud d'accès pour l'ensemble de ses voisins, il n'a besoin de transmettre que deux chemins d'accès[4].

Dans cette nouvelle version (algorithme 13), les robots stockent encore tous les chemins qu'ils reçoivent, mais ne transmettent que deux messages seulement.

Considérons le réseau décrit dans la figure 4.2. Le Robot de Référence 1 diffuse un message avec le chemin d'accès (1). Le robot 2 ajoute ce chemin dans son tableau de connectivité (table 4.2a) et le transmet par diffusion du chemin d'accès $(2,1)$. Les quatre voisins du robot 2 (c'est-à-dire, les robots 1, 3, 4 et 5) reçoivent ce message. Mais le robot 1 ne tient pas compte de ce message car il est déjà dans le chemin. Les robots 3, 4 et 5 mettent à jour leurs tableaux de connectivité (table 4.2b, 4.2c et 4.2e). Puis, chacun de ces trois robots avise ses voisins sur le chemin d'accès. Supposons que robot 3 est le premier à émettre la notification. Robot 2 ignore le message, mais le robot 5 ajoute le chemin d'accès $(3,2,1)$ à son tableau d'accès. Depuis ce moment, le robot 5 n'avise plus ses voisins sur

4. Seulement deux retransmissions ! Car la retransmission d'un seul chemin n'est pas suffisante pour conclure que tous les robots d'accès figurent dans le tableau et il n'est pas nécessaire de faire plus de deux retransmissions, parce qu'à partir de la deuxième retransmission, tous les robots d'accès sont présents dans le tableau.

Algorithme 3: Algorithme à exécuter lors de la réception d'un message `New-Access-Path` (version avec filtrage).

Entrées : Le message `New-Access-Path` M
Sorties : Le Tableau de Connectivité T mis à jour

```
1  début
2      p ← le chemin d'accès dans M;
3      si this.id() ∉ p alors
4          ajouter p dans le tableau de Connectivité T ;
5          si déjà fait suivre 2 messages alors
6              retourne ;
7          si c'est le premier message à faire suivre
           OR (premierAccesseur.id() ∉ p) alors
8              premierAccesseur ← l'expéditeur de M ;
9          fin
10         marquer le chemin comme "fait suivre";
11         mettre à jour et transmettre le message;
12     fin
13 fin
```

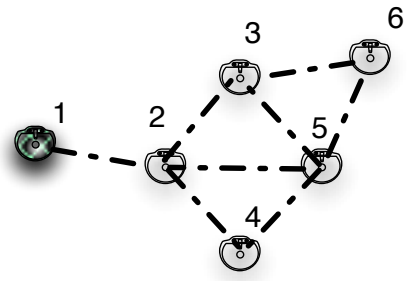


FIGURE 4.2 – *Dans cet exemple, le robot* 1 *est le Noeud de Référence. Tous les robots (sauf le robot* 6 *qui transmet deux messages) n'ont besoin de faire suivre qu'un seul message pour les tableaux de connectivité complets.*

les mises à jour qu'il recevra, car tous les chemins d'accès passent par le robot 2. Par conséquent, robot de 5 n'a pas besoin de transmettre toutes nouvelles voies d'accès. À la fin, les tableaux de connectivité des robots sont tels que dans la table 4.2.

Pour cette version de l'algorithme de retransmission des messages, il y a un champ supplémentaire dans les tableaux pour indiquer si le chemin a été retransmis. Cette information sera utilisée pour gérer la mobilité des robots et sera présentée dans la section 4.4, page 65. Nous voyons que seul le robot 6 transmet deux messages, les autres robots n'ont besoin de faire suivre qu'un seul message [5].

5. Notez qu'il est possible avoir des différences dans les tableaux de connectivité pour un même réseau au niveau de chemin d'accès selon l'ordre réel de la réception des messages.

TABLE 4.2 – *Les tableaux de connectivité des robots dans l'exemple de la figure 4.2.*

(a) *Le tableau du robot* 2

Robot d'accès	Chemin d'accès	Retransmis
1	(1)	oui

(b) *Le tableau du robot* 3

Robot d'accès	Chemin d'accès	Retransmis
2	(2, 1)	oui
5	(5, 2, 1)	non
6	(6, 5, 2, 1)	non

(c) *Le tableau du robot* 4

Robot d'accès	Chemin d'accès	Retransmis
2	(2, 1)	oui
5	(5, 2, 1)	non

(d) *Le tableau du robot* 5

Robot d'accès	Chemin d'accès	Retransmis
2	(2, 1)	oui
3	(3, 2, 1)	non
4	(4, 2, 1)	non
6	(6, 3, 2, 1)	non

(e) *Le tableau du robot* 6

Robot d'accès	Chemin d'accès	Retransmis
3	(3, 2, 1)	oui
5	(5, 2, 1)	oui

Théorème 4.3 *La complexité des messages de l'algorithme 13 est de* $2n - 1$*, et il y a* $2d$ *chemins d'accès dans le Tableau de Connectivité, où d est le nombre de robots voisins.*

Démonstration. À partir de l'algorithme 13, nous constatons que chacun des $n - 1$ robots envoie au plus deux messages, et le robot de référence envoie un seul message. Ainsi, il y a au maximum $2(n - 1) + 1$ messages – et ceci est la limite supérieure pour le nombre de messages envoyés. En retour, pour stocker, un robot peut recevoir au maximum 2 messages de chaque voisin avec un chemin d'accès, donc la taille du Tableau de Connectivité est au plus $2d$. □

Dans le reste de ce mémoire, sauf explicitement indiqué, l'algorithme 13 sera utilisé pour construire des tableaux de connectivité.

4.4 Prendre en compte la mobilité

La topologie du réseau de robots peut changer, d'une part, du fait des modifications de l'environnement et, d'autre part, des déplacements des robots liés à leur mission. Le Robot de Référence (ou un des robots d'accès) peut également tomber en panne à cause de diverses raisons (dommages physiques, décharge de la batterie, etc ...). Cette dynamique pose le problème du maintien du tableau de

Algorithme 4: Algorithme à exécuter lors de la réception d'un message `New-Neighbor`.

Entrées : Le message `New-Neighbor` M
Sorties : Le Tableau de Connectivité T mis à jour

```
1  début
2      pour chaque p ∈ M faire
3          si this.id() ∉ p alors
4              ajouter p dans le tableau de Connectivité T ;
5              si pas encore fait suivre 2 messages alors
6                  si c'est le premier message à faire suivre
                   OR (premierAccesseur.id() ∉ p) alors
7                      premierAccesseur ← l'expéditeur de M ;
8                  fin
9                  marquer le chemin comme "fait suivre";
10                 mettre à jour et transmettre le message;
11             fin
12         fin
13     fin
14 fin
```

connectivité des robots pour assurer sa cohérence avec la topologie du réseau. Cette section présente notre proposition pour résoudre ce problème.

4.4.1 Changement de voisins

Il y a deux situations qui peuvent rendre obsolètes les informations dans le tableau : (i) le robot peut disposer de nouveaux voisins ou (ii) des robots peuvent être déconnectés après leur déplacement ou à cause d'une défaillance d'un robot voisin. Dans ces cas, les robots doivent prendre les actions appropriées afin de mettre à jour leurs tableaux de connectivité.

Avoir des nouveaux voisins Si un robot détecte qu'il y a un robot qui entre dans sa zone de couverture (grâce aux services de réseau sous-jacent), le robot envoie un message `New-Neighbor` à ce nouveau voisin[6]. Le message `New-Neighbor` contient les *deux chemins que le robot a retransmis*. Le récepteur de ce message y extrait les chemins d'accès et les ajoute dans son tableau, chemin par chemin comme dans le cas où il reçoit le message `New-Access-Path` traité par l'algorithme 13, page 64. La procédure (présentée par l'algorithme 14) à exécuter dans ce cas est simplement une version modifiée de l'algorithme 13.

Un robot détecte la disparition d'un de ses voisins Il va supprimer les informations concernées de son tableau, c'est-à-dire tous les chemins d'accès passant

6. Notez que si un robot détecte un nouveau voisin, alors son nouveau voisin le détecte également, c'est-à-dire que ce dernier envoie aussi un message `New-Neighbor`.

par le robot disparu. Ensuite, si un des chemins supprimés a été déjà transmis, le robot diffuse un message `Link-Broken` à ses voisins. Le message contient les informations sur le lien interrompu (ID du robot disparu) et au plus deux chemins d'accès valides alternatifs. Les robots qui reçoivent un message `Link-Broken` insèreront dans leurs tableaux de connectivité le chemin d'accès fourni, après avoir supprimé tous les chemins d'accès obsolètes. Encore une fois, si l'un des chemins supprimés a été déjà transmis, le robot diffuse un message `Link-Broken` à ses voisins.

Pour illustrer cela, reprenons l'exemple de la figure 4.2 (page 64) où nous supposons que le lien $\{5,2\}$ est rompu. Le robot 5 détecte qu'il n'est plus en contact direct avec le robot 2 et retire le chemin $(2,1)$ de son tableau. Comme le robot 5 trouve qu'il ne reste qu'un message déjà retransmis, il insère le chemin $(4,2,1)$ ou $(3,2,1)$ (qui n'a pas été retransmis avant) comme paramètre du message `Link-Broken` à envoyer. À la réception de ce message, les robots $3,4$ et 6 mettront à jour correctement leurs tableaux.

4.4.2 Reprise après la défaillance du Robot de Référence

Le mécanisme de mise à jour décrit dans la section 4.4.1 assure que le Tableau de Connectivité est maintenu de manière cohérente avec la configuration réelle du réseau en présence de la mobilité. Mais qu'est-ce qui se passerait si le Robot de Référence ne fonctionnait plus ? Il faudrait alors sélectionner un nouveau Robot de Référence.

Les robots peuvent détecter la défaillance du robot de référence (ou la déconnexion de ce dernier) si, juste après une mise à jour lors de la réception d'un message `Link-Broken`, il s'avère que leurs tableaux de connectivité sont vides. Dans ce cas, le mécanisme de sélection automatique du robot de référence présenté dans la section 4.3.1 (page 61) sera utilisé pour rétablir une situation cohérente.

Partitionnement du réseau Dans le réseau présenté dans la figure 4.3, l'échec de n'importe quel robot parmi les robots $1,2$, et/ou 5 va partitionner le réseau en deux ou plusieurs sous-réseaux. Ces nœuds sont identifiés comme des nœuds critiques (voir la définition 2.7 dans la section 2.2.2, page 21). Supposons, par exemple, que le robot 1 ne fonctionne plus, alors le mécanisme de récupération va reconstituer le réseau en deux composantes : la première est constituée des robots $2,3,4,7$, et les robots $5,6$ seront regroupées dans un autre sous-réseau.

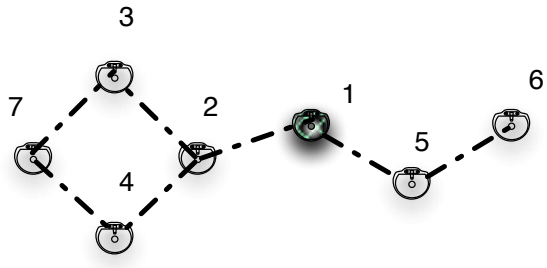


FIGURE 4.3 – *Reseau de robots en réseau où les robots* 1, 2 *et* 5 *sont des nœuds critiques.*

Fusion de sous-réseaux Après un partitionnement, il peut arriver que les robots de deux sous-réseaux peuvent se rapprocher les uns des autres. Dans ce cas, ces sous-réseaux doivent à nouveau se regrouper. Notez que tous les robots dans un réseau connaissent l'ID du Robot de Référence avec qui ils s'affilient. Lorsque les deux groupes sont à portée de communication, les robots à la frontière échangerons leur tableaux avec les informations sur la référence, et le sous-réseau dont le Robot de Référence ayant l'*id* plus petit sera affiliée à l'autre. L'algorithme est identique à l'algorithme 14.

4.5 Robustesse de la connectivité

4.5.1 Robustesse tirant parti de la sensibilité à la connectivité

Dans cette section, nous montrons les relations qu'entretiennent les notions de sensibilité à la connectivité et la robustesse de la connectivité de réseau. Pour un graphe G connexe (référentiellement), avec le Robot de Référence r, on note avec $A(u)$ l'ensemble des robots d'accès, et $N(u)$ l'ensemble des robots voisins du robot u. Notez que $A(u) \subseteq N(u)$ par définition.

On a alors les propriétés suivantes.

Théorème 4.4 *Un nœud $u \in V, u \neq r$, est critique si et seulement si $\exists n_i \in N(u)$ tel que $n_i \notin A(u)$.*

Démonstration. Par définition, si le nœud u est critique, son retrait de G déconnecterait au moins un nœud quelconque $n_i \in N(u)$ depuis le Robot de Référence. C'est-à-dire, le nœud n_i a le nœud u comme son accesseur unique. Par conséquent, le nœud u ne pourrait pas considérer n_i comme son accesseur. On en conclut que $n_i \notin A(u)$.

À l'inverse, si $\exists n_i \in N(u)$ tel que $n_i \notin A(u)$, et le nœud u n'est pas critique, alors selon le théorème 2.1 (page 23), il doit exister un circuit $C(u)$. Sans perdre en généralité, on peut supposer que c'est le nœud n_1 qui n'est pas dans $A(u)$, et on ré-écrit le circuit sous la forme de $(u, n_1, v_1, v_2, ..., v_j, n_2, v_l ..., v_h, n_m, u)$, où $v_i \in V$, et $v_i \notin N(n), n_j \in N(u)$. Notez que ce chemin pourrait ne pas comprendre le Nœud de Référence. Par contre, il doit y avoir un nœud $u_k \in A(u)$ tel qu'on puisse trouver le chemin de communication vers le Nœud de Référence (car le graphe est connexe), et ce chemin ne contient pas de cycle. Donc, on prend le chemin $(n_1, u, n_m, ..., n_{m-1}, ..., n_k)$. Ce dernier nous ramène à conclure que u est un accesseur de n_1, où n_1 appartient à $A(u)$. On arrive à une contradiction car le nœud n_1 n'est pas dans $A(u)$. □

Remarque 4.1 *Dans la démonstration ci-dessus, on a utilisé le théorème 2.1 (définit à la page 23) pour prouver l'hypothèse {$\exists u_i \in N(u) \Rightarrow u$ est critique}. En effet, on peut le prouver indépendamment du théorème 2.1.*

Démonstration. Si $\exists n_i \in N(u)$ tel que $n_i \notin A(u)$, supposons que le nœud u n'est pas critique, puis une fois qu'il a été enlevé, le nœud n_i reste encore connecté, ce qui signifie que ce nœud n_i a un nœud d'accès autre que u. Ainsi, le nœud u peut considérer que le nœud n_i est un de ses nœuds d'accès. Autrement dit, on a $n_i \in A(u)$, ce qui conduit à une contradiction avec l'hypothèse que $v \notin A(u)$. Par conséquence, le nœud u est critique. □

Cette remarque nous permet de démontrer les propriétés concernant la robustesse de la connectivité en n'utilisant que la conception de la connectivité référentielle et du Nœud de Référence comme le cas du théorème 4.5. Cela a pour but de montrer que les deux théorèmes 4.4 ci-dessus et le théorème 2.1 sont équivalents.

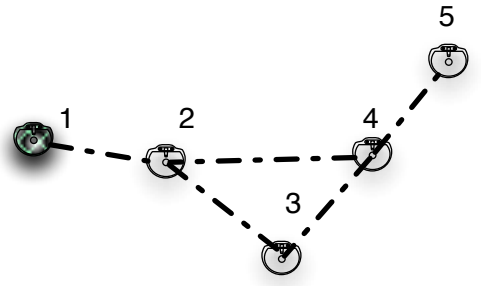


FIGURE 4.4 – *Le nœud* 1 *est le Nœud de Référence. Les nœuds* 2 *et* 4 *sont critiques, mais le lien* $\{2,4\}$ *n'est pas critique.*

Théorème 4.5 *Un lien $e(u,v) \in E$ est critique si et seulement si $A(u) = \{v\}$ ou $A(v) = \{u\}$.*

Démonstration. Si $e(u,v) \in E$ est un lien critique, alors lorsque le lien est rompu, l'un des deux nœuds u ou v sera déconnecté du Nœud de Référence. Si le nœud déconnecté est u, alors u ne doit avoir aucun robot d'accès autre que v ($A(u) = \{v\}$). De même, si le nœud v est déconnecté, alors $A(v) = \{u\}$.

Au cas où $A(u) = \{v\}$, alors la rupture de ce lien séparera les nœuds qui sont connectés au Nœud de Référence à travers u de la sous-composante avec le Nœud de Référence. Par conséquent, le lien $e(u,v)$ est un lien critique. De même pour le cas où $A(v) = \{u\}$. □

À partir des deux théorèmes 4.4 et 4.5, on obtient les corollaires suivants.

Corollaire 4.1 *Si un nœud $u \in V$, $u \neq r$, a un seul nœud d'accès v, alors v est critique.*

Le théorème 4.4 dit en fait que les nœuds autres que celui de référence ont l'information de leur propre criticité si et seulement si il existe un nœud qui n'est pas accesseur parmi leur nœuds voisins. En d'autres termes, le nombre de ses robots d'accès est plus petit que le nombre de ses voisins, et conformément au théorème 4.5 et au corollaire 4.1, si le robot est doté d'un seul robot d'accès, alors un robot sait que le lien entre lui avec son robot accès unique est critique et ce le dernier est également critique. Il est à noter que le théorème 4.5 permet d'éviter la détection de faux liens critiques reliant deux nœuds critiques (cf. la figure 4.4).

Le point le plus intéressant est que, sauf pour le Nœud de Référence, un nœud *est capable de déterminer lui-même* si il est critique ou pas. Pour le Nœud de Référence, comme il n'a pas de liste d'accesseurs, il ne peut pas déterminer lui-même s'il est critique ou non, mais selon le corollaire 4.1, ses voisins peuvent le savoir ! On en déduit le corollaire suivant.

Corollaire 4.2 *Le Nœud de Référence est critique si et seulement si le Nœud de Référence a plus d'un voisin et un de ses voisins a comme seul accesseur.*

Les théorèmes et les corollaires présentés dans cette section sont une base théorique solide pour la détection efficace des nœuds critiques. En outre, l'algorithme est applicable non seulement aux réseaux de robots mobiles, mais aussi à d'autres types de réseaux sans fil en général. Les sous-sections suivantes présentent l'algorithme proposé pour la détection des nœuds critiques dans le réseau sans fil de robots.

4.5.2 Compléter la liste des accesseurs

La stratégie de transmission simple (l'algorithme 7, page 62) génère une liste complète des robots d'accès pour tout robot autre que le Robot de Référence dans le réseau. Cela veut dire que nous pouvons appliquer directement les théorèmes et les corollaires présentés dans la section précédente pour la détection complètement décentralisée des nœuds critiques. Cependant, cet algorithme émet un nombre de messages très important lorsque le graphe du réseau est complet. Bien que le pire des cas n'arrive presque jamais dans la pratique, cette approche naïve est probablement inutilisable pour un réseau très dense. Par conséquent, cette version n'est applicable que pour des réseaux de très petite taille.

Dans la version avec filtrage (l'algorithme 13, page 64), le nombre de messages voyageant dans le réseau est considérablement réduit et l'algorithme possède une complexité de message de l'ordre de $O(2n)$. Toutefois, cette réduction du nombre de messages a un prix : on n'est pas assuré que la liste des robots d'accès soit complète pour tous les cas.

Par exemple, dans la figure 4.5, le robot 5 peut recevoir et transmettre les deux chemins $(2, 1)$ et $(3, 2, 1)$ avant la réception d'un chemin à travers le robot 6 de l'autre sens (avec les autres robots dans le réseau, ceux qui sont représentés par la ligne courbe et pointillée connectant le robot 1 avec le robot 6). Le robot 6 transmet les chemins $(6, 3, 2, 1)$ et $(6, 5, 2, 1)$ avant de recevoir les autres, ce qui entraîne le fait que tous les chemins en provenance de la direction du robot 6 vers le robot 5 n'atteignent jamais les robots 3 ou 4. Ainsi, ces deux robots ne transmettent qu'un seul message, ce qui rend la liste de robots d'accès de robot 2 incomplète.

L'algorithme 13 ne garantit donc pas que la liste d'accès soit complète[7]. Toutefois, il est nécessaire d'avoir des listes complètes afin d'appliquer les théorèmes présentés dans la section 2.2.2 (page 21) pour la détection des nœuds critiques.

La question concernant la solution pour la détection des nœuds critiques se résume au problème de compléter la liste d'accès construite par l'algorithme 13. La solution supplémentaire doit générer un nombre de messages inférieur à celui de l'algorithme 7.

Remarquons que si la liste de robots d'accès d'un nœud est incomplète, c'est parce que certains de ses voisins n'ont transmis qu'un seul message. Nous proposons d'utiliser un mécanisme similaire à la RREQ (Route Request) et RREP (Route Reply) dans le protocole de routage DSR [Johnson et al., 2001] pour compléter la liste d'accès. Après la réception du premier message `New-Access-Path`,

7. L'exigence de la complétude de la liste d'accès est nécessaire si on a pour objectif la détection exacte des nœuds critiques. L'algorithme 13 reste satisfaisant pour le but du maintien de la connectivité qui n'exige pas que la liste d'accès soit complète. Dans la section 6.2 du chapitre 6, nous présenterons l'évaluation empirique de cette incomplétude de la liste d'accès.

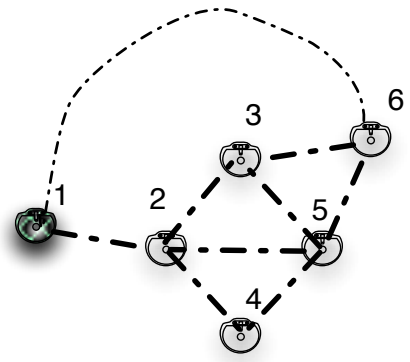


FIGURE 4.5 – *Un exemple de réseau où l'algorithme 13 peut construire des tableaux avec une liste incomplète de robots d'accès pour le robot* 2. *Le robot* 1 *est le Noeud de Référence.*

le robot va attendre pendant une certaine durée (définie par la constante `Complete-Waiting-Time-Out` [8]). Lorsque le temps est passé et si la liste n'a pas encore été achevée, le robot enverra un message `Complete-Request` à ses voisins dont il n'a reçu qu'un seul `New-Access-Path` message. Le message `Complete-Request` contient :

- *l'identifiant du message* qui sera utilisé pour éviter les doublons dans le traitement de la même requête. Si un robot reçoit un message, il va enregistrer l'identification de ce message pour éviter de le traiter dans l'avenir. Cette astuce permet d'éviter d'inonder le réseau avec la même requête.
- *Un chemin inverse* qui sera utilisé pour retourner le message quand un robot trouve un tel chemin. Ce chemin commence par l'identifiant de l'émetteur d'origine du message. Quand ce message est transmis par un robot, l'identifiant de l'expéditeur sera ajouté au chemin.

Dès la réception d'un message `Complete-Request`, si ce dernier a été déjà traité (le robot reconnaît cela par l'identification des messages déjà stockés), le message sera ignoré. Sinon, le robot cherche dans son tableau un chemin qui ne contient aucun nœud dans le chemin inverse. S'il est trouvé, le robot diffuse un message `Complete-Response` qui contient le chemin complémentaire. Sur la réception d'un message `Complete-Response`, un robot va ajouter le chemin d'accès complémentaire dans son tableau. Si son identifiant est dans le chemin inverse, il va supprimer ce chemin et le rajouter au chemin d'accès complémentaire. Enfin, il transmettra le message avec les chemins de mise à jour (à savoir le chemin d'accès complémentaire et le chemin inverse).

Reconsidérons le réseau de la figure 4.5 (page 71). Après que le temps d'attente soit passé, le robot 2 constate que ses voisins, le robot 3 et le robot 4, n'ont retransmis qu'un seul message `New-Access-Path`. Le robot 2 va alors envoyer un message `Complete-Request` à ces deux robots. Quand le robot 3 reçoit ce message de requête, il ne peut trouver aucun chemin dans son tableau pour remplir la liste d'accès, il diffuse le message au robot 5. Parce que le robot 5 trouve le chemin $(6, ..., 1)$ qui ne passe ni par le robot 2 ni par le robot 3, il diffuse un nouveau message `Complete-Response` au robot 3 et au robot 4. Ainsi, le message atteindra le robot 2 pour l'aider à compléter la liste d'accès. Cette solution fonctionne avec moins de messages supplémentaires en comparaison avec

8. Cette constante dépend du nombre de robots dans le réseau.

l'algorithme 7. Les expérimentations présentées dans la section 6.2 du chapitre 6 donnent une évaluation empirique du nombre de messages générés par cette proposition de détection des nœuds critiques.

4.5.3 Comparaison avec l'état de l'art

En terme de coût liés à la communication (i.e le nombre de messages échangés), notre algorithme est plus efficace que ceux proposés dans les travaux existants : à savoir l'approche classique centralisée de parcours en profondeur [9] de [Duque-Anton et al., 2000], les algorithmes distribués de [Ahmadi and Stone, 2006a, Sheng et al., 2006a] et l'algorithme heuristique localisé de [Jorgic et al., 2004].

Les travaux [Ahmadi and Stone, 2006a] utilisent un mécanisme d'échange de messages très proche du notre, mais leur algorithme souffre d'une complexité de $O(2n!)$. En outre, notre solution s'applique même lorsque les robots sont en mouvement, à l'opposée à celle d'Ahmadi et Stone qui ne s'applique que si la topologie du réseau est figée.

L'algorithme DMCC, (Detection Algorithm based on Midpoint Coverage Circle) proposé par [Sheng et al., 2006a], détermine d'abord si un nœud est critique dans une zone locale limitée (Midpoint Coverage Circle). Une fois qu'un nœud est suspecté d'être critique, l'algorithme essaie de trouver tous les chemins globaux entre le nœud et ses voisins afin d'arriver à une conclusion définitive sur la criticité globale du nœud. Ces approches ne passent pas à l'échelle (en terme de nombre de robots) en raison de la nécessité de faire un échange complet de la topologie du réseau. Une autre approche pour la détection des nœuds critiques basée sur les voisinages de *k-saut* est celle proposée par [Jorgic et al., 2004]. Dans cette approche, les nœuds qui sont voisins k-saut, échangent des informations afin de reconstruire une vision locale de la connectivité au lieu d'être conscients de la topologie du réseau entier. Ce travail est ensuite étendu dans [Das et al., 2009] à la procédure pour remédier à la criticité une fois détectée. Cette localisation a l'avantage d'éliminer le besoin d'information globale. Toutefois, jusqu'à 20% des nœuds sont déclarés faux positifs (i.e nœud détecté comme positif critique alors qu'il ne l'est pas).

En conclusion, l'avantage de notre approche sur ces travaux est la suivante : la sensibilité à la connectivité permet de vérifier de manière triviale si un réseau possède une connectivité robuste. Ceci se faisant avec un coût de communication très bas.

4.6 Maintien de connectivité utilisant les mécanismes de la sensibilité

La sensibilité à la connectivité donne une nouvelle perception sur la connectivité globale pour les nœuds dans le réseau. Le maintien de la connectivité basée sur la sensibilité peut être interprété pour chaque robot dans le réseau de la manière suivante : *étant donné un Robot de Référence, pour la préservation de la connecti-*

9. en anglais DFS (Depth First Search)

vité de réseau, les robots doivent maintenir les liens de communication avec leurs robots d'accès dans l'exercice de leurs tâches [Le et al., 2009a].

Autrement dit, la solution de maintien de la connectivité au problème formulé dans la section 1.3.1 (page 7) consiste à déterminer un déplacement planifié pour que cela ne viole pas la *contrainte locale* : après avoir effectué un déplacement $m_i^{R,t}$, le robot a toujours au moins un robot d'accès.

Nous pouvons raffiner un peu le squelette de la solution générale au problème du maintien de la connectivité donnée par l'algorithme 1, page 58. La raffinement concerne l'étape 2. Cette étape est raffinée avec deux sous-étapes comme dans l'algorithme 5 qui suit (l'algorithme à exécuter sur chaque robot pendant l'intervalle $[t_i, t_{i+1}]$.).

Algorithme 5: Squelette de la solution du maintien de la connectivité sur la base de la sensibilité.

1 **début**
2 | *Étape 1* : exécuter l'algorithme de coordination afin de trouver l'ensemble M des déplacements planifiés;
3 | *Étape 2-1* : **mettre à jour le Tableau de Connectivité**;
4 | *Étape 2-2* : **sélectionner un** $m \in M$ **en prenant en compte le maintien de la connectivité sur la base de la sensibilité**;
5 | *Étape 3* : effectuer le déplacement m sélectionné;
6 | *Étape 4* : envoyer les mises à jours aux voisins;
7 **fin**

L'étape 2-1 est déjà résolue avec la sensibilité à la connectivité. Il nous reste à proposer une solution pour l'étape 2-2.

4.6.1 Sélection simple d'un déplacement

On pourrait imaginer intuitivement qu'avec son mode de communication et ses capacités de localisation, un robot devrait pouvoir lui-même déterminer si un déplacement peut le séparer d'un autre robot voisin. Une solution pour l'étape 2-2 dans l'algorithme 5 sur la base de la sensibilité devrait simplement satisfaire la contrainte de sélection d'un déplacement d'un robot. La contrainte locale est qu'après la réalisation du déplacement choisi, le robot doit rester dans la zone de communication au moins d'un de ces robots d'accès.

Les robots étant supposés réaliser les déplacements en même temps à chaque intervalle de temps pendant la mission et afin d'être certain qu'un déplacement choisi ne met pas le robot hors de connexion avec un accesseur, nous définissons une portée de communication en sécurité $range_{sec}$, qui est plus petite que la portée physique de communication $range_{comm}$. L'idée est que, dans le pire cas où un robot choisit de maintenir la connexion avec un accesseur, et que ce dernier sélectionne de faire un déplacement dans la direction opposée, alors la connexion ne sera pas perdue.

Nous appelons cette zone *la zone de communication en sécurité* (Figure 4.6). Nous pouvons maintenant donner plus de détails sur la solution pour sélec-

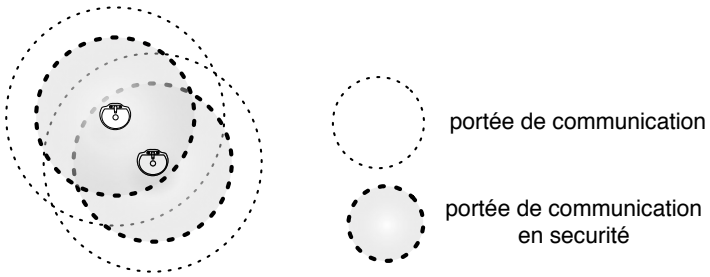


FIGURE 4.6 – *Portée de communication (le cercle gris clair plus grand) et zone de sécurité (le cercle plus petit et plus sombre)*

Algorithme 6: Sélection simple d'un déplacement

Entrées : L'ensemble M de déplacements planifiés triés en l'ordre de préférence

Sorties : Le premier déplacement planifié à actionner de sorte qu'il y ait au moins une connexion avec les accesseurs.

```
1  début
2      tant que M n'est pas vide faire
3          m ← M.soustraireLePremierElément();
4          pour chaque accesseur a du robot faire
5              si dist(a.pos(), m.pos()) ≤ range_sec alors
6                  retourne m ;
7          fin
8          retourne nulle ;
9      fin
10 fin
```

tionner un $m \in M$ en prenant en compte le maintien de la connectivité dans l'algorithme 6.

Dans la ligne 5 de l'algorithme, le robot vérifie si parmi les accesseurs, il en existe un avec lequel la distance entre eux est inférieure à $range_{sec}$. De cette manière, après la réalisation d'un déplacement qui va dans le sens opposé du déplacement du robot d'accès retourné, on est assuré que la connexion de réseau sera toujours assurée.

Remarquons que :

Remarque 4.2 *La stratégie simple de sélection d'un déplacement planifié établit une dépendance asymétrique entre un robot et ses robots d'accès : le choix d'un déplacement est fait indépendamment et par conséquent, les robots dépendent de leurs accesseurs pour sélectionner la position suivante. Plus simplement, les robots ont tendance à suivre leurs robots d'accès.*

La remarque 4.2 nous amène à penser que cette stratégie simple s'applique naturellement pour les applications de type leader/follower où le leader est choisi pour être le Robot de Référence et les autres robots le suivent sous réserve de rester connectés.

Comparaison avec d'autres algorithmes

Notre solution au problème du maintien de la connectivité sur la base de la sensibilité est générale dans le sens que la solution est indépendante d'une application spécifique. Cela veut dire que nous pouvons l'appliquer à diverses applications comme nous l'avons décrit précédemment (cf. la section 4.1, page 57 dans le chapitre 2). Si on examine les travaux existants tels que ceux présentés dans la section 2.3 (page 23), notre approche partage cet avantage avec les solutions basées sur l'utilisation d'un arbre couvrant ou du graphe Laplacien, même si elle n'appartient pas à la même classe de solutions.

En comparaison avec les algorithmes qui utilisent le graphe Laplacien [Gennaro and Jadbabaie, 2006, Zavlanos and Pappas, 2007, Zavlanos and Pappas, 2008, Micheal et al., 2009, Schuresko, 2009], notre algorithme est *totalement décentralisé*, et *il n'y a d'échanges d'information de localisation des robots qu'entre les voisins « directs »*. Cela est clairement un d'avantage, car un échange de l'information de localisation globale peut générer une charge de communication importante. Ainsi, notre proposition offre un meilleur passage à l'échelle comme pour les solutions basées sur l'arbre couvrant analysées dans la section 2.3.6, page 29.

Par rapport à l'utilisation du graphe Laplacien comme solution pour le maintien de la connectivité dans un réseau sans fil où la latence de la transmission est importante, notre solution s'adapte plus rapidement à l'évolution du système. Comme noté dans [Schuresko and Cortés, 2009], leur l'algorithme continue à fonctionner alors que les localisations des robots sont obsolètes et si les robots ne parviennent pas à choisir un déplacement, ils restent immobiles pendant quelques instants. Enfin, la sensibilité à la connectivité offre plus de flexibilité dans le choix du robot pour maintenir la connexion par rapport aux travaux basés sur des arbres couvrants comme [Diosdado, 2006, Schuresko, 2009]. C'est un point important qui distingue notre algorithme de maintien de la connectivité sur la base de la sensibilité par rapport à ces travaux. Les expérimentations de [Diosdado, 2006] montrent que les explorations multi-robots avec le maintien de connectivé basé sur un arbre couvrant adaptatif sont plus efficaces que si l'arbre couvrant est fixe. Selon nous, avoir plus de choix pour les robots d'accesseurs permet d'avoir une meilleure performance.

Avec l'approche orientée-expérimentation, les auteurs dans [Nguyen et al., 2004a, Nguyen et al., 2004b] ont proposé des stratégies et des techniques pour le maintien de la connectivité sur des robots réels. Pour maintenir la communication entre un robot explorateur avec une station fixe, le robot explorateur dans les expérimentations de [Nguyen et al., 2004a] emporte un ensemble de routeurs avec lui. Ainsi, lorsque le robot se déplace assez loin de la station ou du dernier routeur déployé, le robot déploie un nouveau routeur pris dans l'ensemble qu'il transporte. De cette façon, la communication entre la station et le robot est étendue. La proposition de [Nguyen et al., 2004a] tout comme notre proposition, met l'accent sur les moyens proactifs dans lesquels les robots individuels de l'équipe peuvent éviter les déconnexions entre les noeuds du réseau. La recherche de [Ulam and Arkin, 2004] aborde les aspects réactifs de la récupération de la communication. Ils proposent un certain nombre de comportements utilisés pour répondre à la question « comment les membres de l'équipe peuvent réagir en cas de déconnexions que ce soit entre certains nœuds ou à l'échelle de l'ensemble du réseau ? ». Bien que ces solutions prises séparément sont moins performantes

que notre proposition, nous pensons qu'elles peuvent être utilisées de manière complémentaire avec la sensibilité au contexte. Nous obtiendrions ainsi une solution globalement plus robuste.

Limitations de la sélection simple d'un déplacement

Le choix d'un robot d'accès avec lequel la connexion doit être maintenue peut avoir des conséquences sur : 1) la topologie du réseau, et 2) les arrangements topologiques des robots. Parmi les idées intuitives que nous pouvons utiliser pour faire ce choix, on a : 1) choisir le robot d'accès avec le chemin d'accès le plus court, et 2) prendre en compte des informations sur la stabilité et la puissance du signal pour ce choix. Ce sont les critères que nous pouvons utiliser pour améliorer la sélection d'un déplacement.

Sinon, pour l'algorithme 6 (page 74), nous avons mentionné (c.f. la remarque 4.2, page 74) que dans la sélection simple d'un déplacement, les robots ont tendance de suivre le Robot de Référence. De plus, la sélection simple peut causer le partitionnement du réseau si deux robots se choisissent mutuellement comme des robots d'accès à suivre. Ces sont les limitations de la sélection simple d'un déplacement. Dans le chapitre 6 sur la perspective de la thèse, nous donnons une démarche pour remédier à ces défauts.

Conclusion du chapitre

Dans ce chapitre nous avons présenté quelques bases théoriques dans l'élaboration de notre solution sur le maintien de la connectivité dans un réseau multi-robots. En fait, la sensibilité à la connectivité est, à notre avis, un concept original dans le domaine des réseaux ad hoc mobiles. Nous avons utilisé ce concept pour construire une représentation distribuée et locale de la connectivité. Les robots sont sensibles non seulement à la connectivité « simple », mais aussi à la connectivité robuste où la déconnexion d'un nœud ou d'un lien n'est pas critique pour la connectivité globale du réseau. Le travail présenté est original dans le sens où nous avons proposé une approche qui améliore la propriété globale du réseau à partir des interactions entre les robots voisins, qui ne disposent que des informations locales.

Le chapitre traite également le problème de la mise à jour des connaissances de chaque robot pour prendre en compte l'évolution/la dynamique du réseau et les états des robots (pannes, déconnexion, déplacement, regroupement, etc.).

Tenant compte que les robots disposent désormais d'une connectivité de réseau fiable et robuste, nous allons examiner dans le chapitre suivant le problème de la coopération entre les robots et plus précisément, le problème de formation des coalitions dynamiques dans un système multi-robots avec des entités hétérogènes.

Allocation de rôles à des robots multi-tâches

5

Sommaire

Parker [Parker, 2003] présente une solution pour effectuer une mission de surveillance et de reconnaissance à l'intérieur d'un bâtiment. Il s'agit de cartographier un étage dans un immeuble inconnu, pour rechercher des cibles ou des points d'intérêt, puis à les « protéger ». Elle requiert un nombre important de robots hétérogènes (plus de 100 robots).

Avec de telles contraintes, la stratégie de Parker consiste à faire prendre aux robots – en fonction de leurs capacités matérielles – quelques rôles parmi un ensemble prédéfini. Une fois qu'ils se sont vu assigner des rôles, les robots travaillent et interagissent en conséquence.

Pour un scénario comme celui-ci, les robots traitent les tâches appropriées à leurs rôles. Les rôles sont déterminés par une structure d'organisation conçue a priori. Nous parlons dans ce cas d'*organisation statique*.

D'un autre côté, dans le scénario d'application USAR présenté dans la section 1.3 (chapitre 1, page 7), la structure de l'organisation peut nécessiter un changement pour s'adapter à une nouvelle tâche. Une manière de procéder consiste à répartir les rôles sur les robots à chaque apparition d'une nouvelle tâche. Nous parlons alors d'*organisation émergente*.

Dans la suite de ce chapitre, nous présentons et comparons des solutions pour structurer un système multi-robots selon chacune de ces deux approches.

Dans un premier temps, nous introduisons une solution qui est une extension du modèle organisationnel AGR. Puis, nous proposons un algorithme de formation des coalitions qui illustre l'approche avec une organisation émergente. Dans les deux cas, nous faisons l'hypothèse que les robots sont en mesure de communiquer. Autrement dit, nous faisons l'hypothèse que notre solution de maintien de la connectivité réseau présentée dans le chapitre 4 est mise en œuvre.

5.1 Déploiement d'organisation « statique » sur un système multi-robots

Dans cette section, nous présentons notre solution pour déployer une organisation sur un système multi-robots. Comme nous le décrivons plus bas, cette organisation est totalement décrite de manière indépendante des robots. Ainsi, il est possible changer toute l'organisation pendant le déroulement d'une mission.

Notre point de départ est le modèle AGR (c.f. la section 3.4.1, page 39 du chapitre 3) développé dans la communauté des chercheurs s'intéressant aux systèmes multi-agents. Nous reprenons à l'identique la structure des organisations d'AGR et notamment l'idée de faire jouer aux robots des rôles au sein de groupes. Notre contribution est construite autour d'une *description de l'organisation* que nous fournissons aux robots pour qu'ils puissent raisonner dessus. Ainsi, les robots choisissent de façon autonome les rôles qui leur sont les plus appropriés (section 5.1.1), forment des groupes et forment ainsi l'organisation effective. Notre principal apport dans cette section est le protocole d'allocation de rôles qui peut être utilisé pour changer l'organisation globale dynamiquement (section 5.1.2). Rappelons qu'en la matière ni le modèle AGR lui-même, ni la plate-forme MadKit qui est son implémentation de référence n'abordent cette question.

5.1.1 Adaptation d'AGR

L'utilisation du modèle AGR en robotique collective requiert quelques adaptations prenant en compte les contraintes physiques du système. Le concepteur peut utiliser AGR pour exprimer de manière fonctionnelle le système en termes de rôles et de groupes. Cependant, le modèle ne fournit aucune méthodologie concernant l'allocation de rôles à chaque agent (voir section 3.4.1, page 39). Dans la plupart des cas [1], les agents et leurs organisations sont construits de manière ad hoc avec une prise de rôle prédéfinie à un moment particulier de l'exécution. Nous pensons que ce type de démarche conduit au développement de systèmes relativement fermés et peu évolutifs (au sens de l'adaptation) et ne convient donc pas au développement de systèmes multi-robots.

Nous considérons qu'un agent doit être autonome non seulement du point de vue de son comportement, mais également du point de vue de sa participation à une organisation. L'agent doit déterminer lui-même les rôles à endosser (ou abandonner) et ainsi rejoindre (ou quitter) un groupe. Cette capacité à changer de rôle fait que le groupe apparaît comme une structure dynamique évoluant dans le temps. Ainsi, sous réserve de respecter les contraintes requises par les spécifications des rôles, un agent est libre de prendre part à n'importe quel groupe et d'endosser n'importe quel rôle.

Cette proposition nécessite que les agents puissent raisonner sur l'organisation dans leur processus de prise de décision quant aux rôles à endosser et aux groupes à rejoindre. Pour ce faire, nous introduisons la notion de description d'organisation permettant aux agents de connaître la structuration souhaitée du

1. et particulièrement dans la plate-forme Madkit qui est l'implémentation de référence du modèle AGR

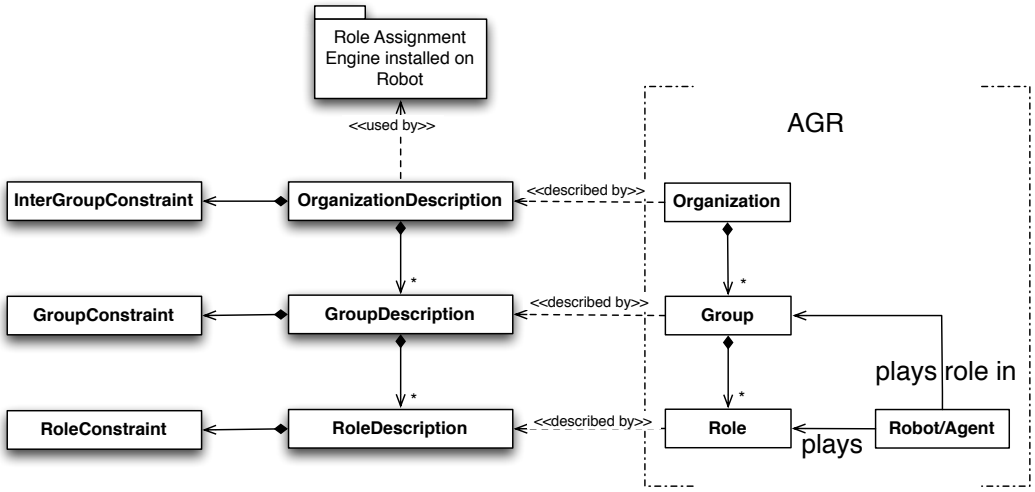


FIGURE 5.1 – *Modification du modèle AGR pour les systèmes multi-robots [Le et al., 2009b]*

système. La figure 5.1 donne une vision de notre proposition d'extension du méta-modèle AGR.

La structuration souhaitée du système passe par la description de l'organisation, des groupes et des rôles :

- **Description de rôle** : *une description d'un rôle est l'ensemble des contraintes qu'un robot doit satisfaire afin d'endosser le rôle.*
- **Description de groupe** : *une description d'un groupe est une spécification constituée d'un ensemble de descriptions de rôles, des contraintes liées au groupe (e.g. le nombre d'instances d'un même type de groupe) et un protocole d'interaction pour la collaboration entre les membres du groupe.*
- **Description d'organisation** : *une description d'une organisation est une spécification qui réunit un ensemble de descriptions de groupes et un ensemble de contraintes inter-groupes.* Un exemple de contrainte inter-groupe est qu'un agent faisant partie d'un groupe G_1 ne doit pas participer à un groupe G_2.

5.1.2 Déploiement d'une organisation AGR sur un système multi-robots

Le déploiement d'une organisation est basé sur la description de celle-ci. Ce processus consiste à allouer dynamiquement les rôles aux différents robots disponibles.

Concepts

- **Voisinage** *: est une fonction de distance permettant de déterminer quels robots sont voisins avec un robot donné.*

Cette fonction peut être construite de manière ad hoc et donc varier d'une application à une autre. Dans sa forme la plus simple, deux robots sont voisins s'ils sont en mesure de communiquer directement (i.e. sans intermédiaire).

- **Gestionnaire de groupe** *: pour chaque groupe, il existe exactement un et un seul membre en charge de la gestion de ce groupe. Ce robot gestionnaire doit connaître tous les robots du groupe. Ces derniers appartiennent nécessairement au voisinage du gestionnaire qui contrôle leur appartenance au groupe.*

L'unicité du gestionnaire au sein de chaque groupe est exprimée de manière explicite. Nous la spécifions sous la forme contrainte dans la description du rôle de gestionnaire.

– **Gestionnaire d'organisation** *: pour initier le processus d'allocation de rôle, un des robots doit détenir la description de l'organisation. Il devient alors l'unique gestionnaire de l'organisation. Notons que le gestionnaire connaît tous les gestionnaires de groupe.*

Le gestionnaire de l'organisation est aussi gestionnaire d'un groupe particulier qui rassemble tous les gestionnaires du groupe. Ce regroupement facilite la coordination entre les gestionnaires de groupe et la prise en compte des contraintes inter-groupes.

Protocole d'allocation de rôles

Le protocole que nous présentons ici est inspiré de l'algorithme DMAC (Distributed and Mobility-Adaptive Clustering) proposé par Basagni [Basagni, 1999]. DMAC a été introduit pour partitionner un réseau ad hoc mobile (MANET) en *clusters* (grappes). Un cluster est un sous-ensemble des nœuds du réseau avec un maître et des membres. Dans notre cas, chaque nœud est un robot. Le maître d'un cluster est le robot qui a le plus grand nombre de voisins « directs ».

L'idée de base de notre approche est d'associer chaque cluster à un groupe. Le maître du cluster jouerait le rôle du gestionnaire du groupe. Cependant, nous donnons aux robots tous les éléments pour qu'ils raisonnent et choisissent eux-mêmes les rôles. Chaque robot analyse et vérifie localement les contraintes des différents rôles et choisit ceux qu'il veut jouer.

De la même manière que dans DMAC, nous faisons un certain nombre d'hypothèses quant au processus d'allocation de rôles. Il faut noter qu'avec notre algorithme du maintien de la connectivité (c.f. chapitre 4, page 55), ces hypothèses sont tout à fait raisonnables.

– Tout message envoyé par un robot sera reçu en un temps fini par tous ses voisins.
– La couche basse fournit les services d'infrastructure permettant aux robots de détecter l'apparition ou la disparition de voisins. Les robots connaissent également le rôle de leurs voisins.
– Nous supposons que chaque robot dispose d'un identifiant unique dans le système.

Juste après leur démarrage ou leur intégration à l'organisation, les robots sont dans leur état initial et cherchent à rejoindre une organisation. Tous les robots sont dans cet état à l'exception du robot qui dispose de la description de l'organisation.

La formation de l'organisation s'appuie sur quatre types de messages échangés par les robots de proche en proche (chaque robot avec ses voisins immédiats) :

– `OrgDescResquest` : est un *broadcast* message envoyé par les robots qui ne sont pas encore membres d'une organisation et qui souhaiteraient en intégrer une. Chacun de ces robots diffuse le message `OrgDescRequest` à tous ses voisins, pour demander une description de l'organisation, de manière à pouvoir ensuite sélectionner un rôle à jouer. Cette diffusion est

répétée à intervalles de temps réguliers jusqu'à ce que le robot reçoive une réponse d'un membre de l'organisation.
- `OrgDescResponse` : Cet *unicast* message contient la description de l'organisation avec une description de tous les groupes et rôles associés. Il est émis par tout membre d'une organisation qui reçoit le message `OrgDescRequest`. Seuls les robots qui ne font partie d'aucune organisation traitent le message `OrgDescResponse`. Chacun de ces robots extrait du message la description de l'organisation. Le robot vérifie ensuite les contraintes de rôles afin de trouver un rôle approprié. En fonction du résultat de cette étape, le robot diffuse soit un message `NewGroup`, soit un message `Join`.
- `NewGroup` : est un message informant de la création d'un nouveau groupe. Ce message est créé par un robot qui décide de prendre en charge la gestion d'un nouveau groupe. Comme ce message indique une modification de l'organisation, les robots qui le reçoivent doivent à nouveau vérifier les contraintes définissant leurs propres rôles. Si l'une des contraintes n'est pas satisfaite, ils doivent ré-exécuter la procédure de sélection de rôle.
- `Join` : est envoyé par un robot au gestionnaire d'un groupe pour l'informer de son intention de prendre un rôle au sein du groupe. Si l'émetteur du message est lui-même gestionnaire de groupe et décide donc de prendre part à un deuxième groupe, il diffuse ce message à tous les membres de son premier groupe. Il s'agit là d'une notification de son aptitude à faire l'intermédiaire pour la communication entre les agents des deux groupes. à la réception de ce message, le gestionnaire de groupe ajoute l'émetteur dans sa liste de membres.

Enfin, lorsqu'un robot disparaît [2], le robot responsable du groupe retire l'identifiant du robot disparu de la liste des membres du groupe. Dans le cas où le responsable du groupe disparaît, les robots membres du groupe doivent ré-exécuter la procédure de sélection de rôles pour éventuellement rejoindre d'autres groupes ou en créer de nouveaux.

Notons que dans ce protocole, il n'y a pas de négociation entre le gestionnaire du groupe avec un autre robot qui veut devenir membre de son groupe. Nous qualifions l'approche présentée *stratégie optimiste*, car les robots sont responsables de la sélection d'un rôle approprié tout en respectant les contraintes imposées. Sous réserve que les contraintes soient correctement exprimées localement, l'intégrité globale est garantie. De plus, comme les messages sont échangés entre les voisins « directs », aucun problème d'obsolescence de l'information ne se pose.

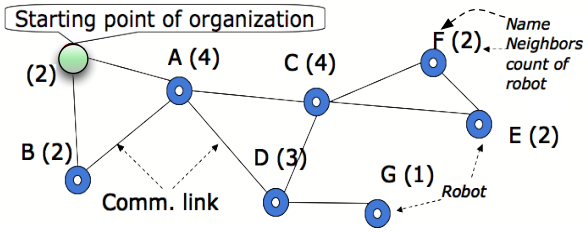
Exemple d'allocation de rôles

Pour illustrer le processus d'allocation de rôles, prenons l'exemple d'une flotte de robots équipés d'interfaces réseau sans fil, qui doivent réaliser une mission de secours sur un terrain inconnu.

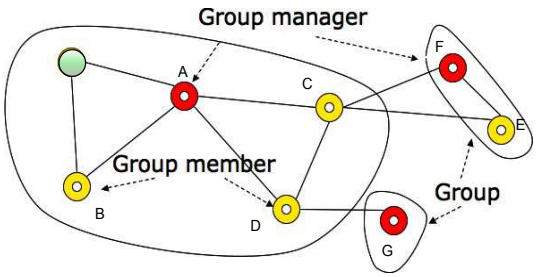
Voisins : Dans cet exemple, deux robots sont dits "voisins" s'ils se trouvent dans l'intersection de leurs zones de couverture radio.

Contraintes : Pour définir les clusters, nous utilisons une fonction de pondération qui consiste à calculer le nombre de robots voisins. Un robot est maître

2. C'est détecté par la couche basse (voir les hypothèses).



(a) *Etat initial de l'organisation*



(b) *Formation complète de l'organisation*

FIGURE 5.2 – *Exemple de déploiement d'une organisation [Le et al., 2009b]*

d'un cluster si (1) aucun de ses voisins n'est déjà à la tête d'un cluster et (2) s'il a le plus grand poids (i.e. nombre de voisins) par rapport à ses voisins. Dans le cas contraire, il rejoindra le cluster ayant pour maître le voisin possédant le plus grand poids. Pour le groupe de gestionnaires, la contrainte est que chaque robot membre soit maître d'un cluster. A noter que pour ce groupe nous excluons la contrainte de voisinage entre les membres et le gestionnaire. En effet, les groupes et donc leurs gestionnaires sont supposés être distants (pas de voisinage direct entre gestionnaires de groupes).

Considérons maintenant la formation de groupes pour un système multi-robots déployé comme montré sur la figure 5.2a. Pour simplifier, nous considérons que les robots sont à l'arrêt. La formation de l'organisation est initiée par un premier robot que nous appelons - robot initiateur (figure 5.2a).

Au pas de temps 1, tous les robots diffusent le message `OrgDescRequest`. à l'issue de cette première étape, seulement deux voisins : les robots A et B reçoivent le message `OrgDescResponse` contenant la description de l'organisation. Ils le traitent en parallèle.

- Comme le robot A a un nombre de voisins plus grand que le robot initiateur, A prend le rôle de gestionnaire du groupe. Le robot A diffuse alors un message `NewGroup`. A la réception de ce message, le robot initiateur réagit en ré-exécutant la procédure de sélection de rôle. Il choisit alors de devenir simple membre et de s'affilier au groupe du robot A.
- Dans le même temps, le robot B décide de rejoindre le groupe du robot initiateur. Mais lorsqu'il reçoit le message `NewGroup` émis par A, le robot B ré-exécute la procédure de sélection de rôle. Il s'adapte ainsi au changement d'organisation et rejoint le groupe du robot A.

L'organisation finale obtenue est celle de la figure 5.2b. Au départ (figure 5.2a), tous les robots sont considérés comme les gestionnaires de leur propre groupe (c'est pour cela qu'ils sont représentés par les mêmes symboles). Après, certains sont affiliés aux autres, ils changent leur représentation (figure 5.2b).

Notez que les robots C, D, E, F, et G relancent une requête `OrgDescRequest` à leurs voisins, mais seuls les robots C et D reçoivent la réponse du robot A. Le robot C a un nombre de voisins égal à celui du robot A. Mais, comme A est à la fois gestionnaire d'un groupe et voisin direct de C, alors C rejoint le groupe de A.

5.1.3 Vers une solution générique

L'approche que nous avons proposée ci-dessus est raisonnable dans le cas où les contraintes peuvent être vérifiées localement par chaque robot. Cependant, le protocole de base présenté dans la section 5.1.2 est insuffisant pour traiter les contraintes globales. Considérons une équipe de football. La formation des groupes doit satisfaire la contrainte globale sur le nombre de groupes, le nombre de membres dans chaque groupe, c'est-à-dire la formation équipe choisie : 3-5-2 ou 4-4-2 par exemple. Dans un tel cas, l'approche optimiste n'est pas vraiment appropriée puisque les contraintes globales sont difficiles à satisfaire. En fait, il n'existe pas de solution unique pour tous les cas applicatifs possibles. Nous proposons donc pour le concepteur une autre stratégie d'allocation de rôles alternative que nous qualifions de *pessimiste*.

– *Stratégie pessimiste* : chaque robot envoie une requête demandant qu'un rôle lui soit assigné. Lorsqu'il s'agit d'un rôle dans un groupe, le destinataire de la requête est le gestionnaire du groupe. Par contre, lorsqu'un robot veut créer un nouveau groupe et donc endosser un rôle de gestionnaire de groupe, il doit envoyer une requête au gestionnaire de l'organisation (i.e. le robot qui détient initialement la description de l'organisation). Le processus d'assignation de chaque rôle se passe donc en deux temps, un robot émet d'abord une requête indiquant son souhait d'endosser un rôle. L'assignation du rôle n'a lieu que si la requête est acceptée par un robot gestionnaire. Prenons l'organisation d'une équipe de foot par exemple, il doit y avoir un gestionnaire de l'équipe qui gère globalement la création d'un nouveau groupe. Dans chaque groupe, le gestionnaire de chaque groupe gère le nombre de membres dans son groupe.

La stratégie optimiste offre l'avantage d'être distribuée et ouverte, mais la sécurité et l'intégrité des contraintes globales restent difficiles à résoudre. Ces deux points sont en revanche plus facilement traitables avec l'approche pessimiste qui présente toutefois le défaut d'être centralisée, alors que le système multi-robots est naturellement distribué.

5.2 Formation dynamique de coalitions

Avant de présenter notre proposition de formation dynamique de coalitions, rappelons que nous faisons l'hypothèse que les robots sont connectés au même réseau. Autrement dit, nous faisons l'hypothèse que notre algorithme de maintien de la connectivité (chapitre 4) est mis en œuvre.

5.2.1 Concepts

Mission, coalitions, tâches, robots, et rôles

Une mission robotisée se compose de tâches différentes qui peuvent être réalisées indépendamment les unes des autres. Nous considérons que 2 tâches sont différentes même si elles sont équivalentes dans le sens où elles peuvent être traitées de la même manière [3]. Pour résoudre une tâche, il faut former une coalition qui peut comporter un ou plusieurs robots. Pour être membre d'une coalition, un robot doit endosser un rôle dans la coalition.

L'identification d'un rôle signifie l'identification des services, des fonctionnalités fournies par le robot qui l'endosse. En plus de cela, nous imposons que les fonctionnalités de rôles différents ne doivent pas se chevaucher et qu'un rôle ne soit pas nécessairement équivalent à un robot. Par conséquent, un robot peut jouer plusieurs rôles (dans une même coalition ou dans plusieurs coalitions). Les avantages sont :

1. *Promouvoir la réutilisation de la solution avec des systèmes de robots différents*. Dans la mission de déplacement de boîtes (voir la section 3.1, page 33 du chapitre 3) par exemple, nous pouvons réutiliser la solution avec n'importe quel système de robots à condition qu'il y ait des robots capables remplir les trois rôles attendus.
2. *Rendre la solution la plus flexible et la plus robuste possible* : Lorsque le Pusher ne reçoit plus d'information de localisation du fait d'une défaillance du Pilote par exemple, la coalition « recrute » simplement un nouveau robot pour le rôle défaillant.

À notre connaissance, ces avantages ne se retrouvent dans aucune des approches basées sur les rôles dans la littérature.

Adéquation rôle-robot

Comme nous l'avons déjà dit, les rôles régissent le comportement des robots. En effet, les robots contribuent à la mission selon les règles comportementales dictées par les rôles. Ainsi, à tout moment, chaque robot, dispose d'*au moins* un rôle *actif*, c'est-à-dire au moins un rôle qui régit l'activité du robot.

Afin d'être capable de jouer un rôle particulier, un robot devrait posséder *a priori* les connaissances, les compétences requises par ce rôle. Autrement dit, avant même de participer à une coalition pour résoudre une tâche, un robot est en mesure d'identifier les rôles qu'il peut potentiellement endosser. Nous appelons de tels rôles les *rôles qualifiés* pour un robot donné.

Par ailleurs, les robots sont supposés pouvoir jouer plusieurs rôles en même temps. Cependant, certains rôles ne peuvent être actifs simultanément, par exemple parce qu'ils introduisent des comportement contradictoires. Nous appelons cette restriction : *contrainte de concurrence*. Elle interdit à un robot d'endosser en même temps certaines combinaisons parmi ses rôles qualifiés. Il faut noter que cette contrainte est dynamique, dans le sens où le robot sélectionne les rôles à endosser en fonction du contexte à un instant donné (l'environnement d'exécution, le niveau des batteries, etc).

3. En fait, les tâches sont distinctes au moins par les endroits et les moments où elles apparaissent.

5.2.2 Formulation abstraite du problème

Nous formulons le problème d'allocation de rôles comme une triplet $(\mathcal{T}, \mathcal{R}, \mathcal{C})$, où $\mathcal{T} = \{T_1, T_2, \ldots, T_m\}$ est l'ensemble ordonné suivant les priorités des m tâches à réaliser par l'ensemble de n robots $\mathcal{R} = \{R_1, R_2, ..., R_n\}$. Une tâche $T_i \in \mathcal{T}$ requiert une coalition $C_i \in \mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ pour être résolue. Chaque coalition se compose de robots jouant des rôles. Donc, afin de faire partie d'une coalition, les robots doivent endosser les rôles.

Une coalition C_i est spécifiée par $(roles, weights, constraints, utilities)$, où $roles = \{r_1, r_2, ..., r_{l_i}\}$ est l'ensemble des rôles ordonné par priorité [4] dans la coalition, ainsi que leurs poids relatifs $weights = \{w_1, w_2, ..., w_{l_i}\}$, $w_h > w_t$, $\forall h < t$. Le poids d'un rôle est déterminé par le concepteur pour désigner l'importance relative du rôle par rapport aux autres. $constraints = \{c_1, c_2, ..., c_{l_i}\}$ est l'ensemble des contraintes imposées sur $roles$. Chaque contrainte c_k pour un rôle r_k est de la forme $(min_{r_k}, max_{r_k}, concurrents_{r_k})$, où :

- $0 \leq min_{r_k} \leq max_{r_k} \in \mathbb{N}$ indique la contrainte sur le nombre de robots qui peuvent se voir assigne le rôle r_k
- $concurrents_{r_k}$: ensemble des rôles qu'un robot peut jouer simultanément avec le rôle r_k.

L'ensemble $utilities = \{u_k(R) | u_k(R) : \mathcal{R} \mapsto \mathbb{R}^+, 1 \leq k \leq l_i\}$ est un ensemble de fonctions d'utilités. Une fonction d'utilité u_k permet au robot de mesurer le bénéfice qu'il obtiendrait après avoir rempli le rôle r_k dans la coalition. Notez que la fonction d'utilité (cf. section 3.2.4, page 36) est attachée aux rôles, car différents robots peuvent jouer un même rôle.

Comme dans [Gerkey and Matarić, 2004a], l'utilité est un concept abstrait, qui peut être défini de manière extrêmement flexible en pratique pour englober des calculs arbitraires. Avec les fonctions d'utilité pré-définies, les robots sont capables de décider eux-mêmes des rôles à endosser parmi leurs rôles qualifiés respectifs. Grace à cette abstraction, nous pouvons ignorer la complexité du calcul effectué par les robots pour chaque rôle particulier. Cette abstraction sépare bien les préoccupations à l'égard de l'optimisation de notre problème principal : l'allocation de tâches aux robots.

Nous utilisons l'opérateur « . » pour désigner un élément appartenant à une structure quelconque. Par exemple, $C_i.roles$ indique l'ensemble des rôles dans la coalition C. De même, on peut écrire $C_i.r_j$ pour faire référence au rôle r_j dans la coalition C_i.

Nous supposons que chaque robot $R \in \mathcal{R}$ connaît lui-même son ensemble de rôles qualifiés, noté $qualified_R$, avec le poids et les utilités correspondants. Ainsi, nous appelons $roles_R$, l'ensemble des rôles actuellement attribué à R et $robots_r$, l'ensemble des robots actuellement jouant le rôle r. En générale, $qualified_R$ est l'ensemble des compétences « dures » [5] d'un robot car cet ensemble est plutôt statique, tandis que $roles_R$ et $robots_r$ sont dynamiques, c'est-à-dire qu'ils peuvent changer au fil du temps. En effet, un robot endosse un sous-ensemble de ses

4. Comme dans [Gerkey and Matarić, 2004b], certains rôles peuvent être plus importants que d'autres.

5. Contraintes qui doivent être nécessairement satisfaites par la solution. Par opposition à des contraintes « souples » qui expriment seulement une préférence pour certaines solutions, par exemple celles qui ont un faible coût par rapport à des solutions ayant un coût plus élevé.

rôles qualifiés en fonction du contexte et des contraintes de concurrence entre les rôles.

Définition 5.1 (Assignation valide) *Une allocation d'un rôle r à un robot R, noté par $assign(r, R)$, est valide si*

- $r \in qualified_R$, *et*
- $\mid robots_r \mid \leq max_r$, *et*
- $\mid robots_r \mid \geq min_r$, *et*
- $\forall r_i \in roles_R, r_i \in concurrents_r$ [6].

Définition 5.2 (Solution faisable) *Un ensemble d'assignations valides $\mathcal{VA}$ ensemble de couples (rôle, robot) est une solution faisable.*

Définition 5.3 (La récompense d'une solution faisable) *La récompense d'une solution $\mathcal{VA}$ est définie par :*

$$reward_{\mathcal{VA}} = \sum_{assign(r,R) \in \mathcal{VA}} w_r u_r(R) \tag{5.1}$$

Définition 5.4 (Formation optimale des coalitions)

$$\mathcal{VA}_{optimale} = argmax(Reward) \tag{5.2}$$

avec

$$Reward : \mathcal{VA} \rightarrow reward_{\mathcal{VA}}$$

5.2.3 Algorithmes de formation de coalition

Formation optimale instantanée des coalitions avec un DCOP

D'abord, nous considérons le cas où toutes les tâches sont connues. Dans ce cas, nous pouvons former toutes les coalitions en même temps et de manière optimale. C'est la formation optimale instantanée des coalitions.

La formulation abstraite ci-dessus nous suggère intuitivement qu'une formation distribuée et optimale des coalitions peut être trouvée au moyen d'un algorithme DCOP [7] [Modi et al., 2006]. L'emploi des agents pour modéliser ce problème est une idée naturelle du fait des similarités entre agents et robots. On arrive à modéliser la classe de problème d'allocation de tâches *MT-MR-IA* (voir la taxonomie MRTA 3.5.1, page 45 du chapitre 3) comme un DCOP. L'ensemble des agents est l'ensemble $\mathcal{R}$ de robots. Un robot/agent R possède une variable qui correspond à l'ensemble de rôles qualifiés $qualified_R$. En plus de ces variables, les robots connaissent les contraintes « dures » [8] $concurrents_r$ et les fonctions d'utilité $u_r(R)$ qui sont les contraintes « souples » [9]. Toutes ces contraintes sont locales à chaque robot.

On voudrait s'assurer que les contraintes cardinales sont également satisfaites, c'est-à-dire, $\forall r, min_r \leq robots_r \leq max_r$. S'il y a k robots qualifiés pour un rôle r, on devrait avoir besoin d'une nouvelle contrainte k-naire entre ces robots. La contrainte prend k arguments : $x_1, x_2, \ldots, x_k$, avec :

6. Notez que, si $r_i \in concurrents_r$, alors on a également $r \in concurrents_{r_i}$.
7. Distributed Constraint Optimization Problem
8. Contraintes qui doivent être nécessairement satisfaites par la solution.
9. Expriment seulement une préférence pour certaines solutions, par exemple celles qui ont un faible coût par rapport à des solutions ayant un coût plus élevé.

$$x_i = \begin{cases} 1 & \textit{si le robot } R_i \textit{ prend } r \\ 0 & \textit{si non} \end{cases}$$

Avec ce modèle DCOP complet, on peut appliquer un algorithme de type DCOP pour former des coalitions de manière optimale. Cependant, on peut rapidement trouver que l'ensemble de robots qui sont qualifiés pour un rôle doit être connu *a priori* par tous les robots de même type. Ce pré-requis est un inconvénient car une telle exigence n'est pas désirable dans par exemple, le contexte de sauvetage où on est confronté au dynamisme dû aux défaillances des robots. En outre, la plupart des algorithmes pour le DCOP ont un coût élevé en terme de communication. La seule exception est l'algorithme ASOD-POP [Ottens and Faltings, 2009] qui émet un nombre de messages linéaires par rapport à la complexité du graphe des contraintes [10]. Enfin, comme un COP [11] est un problème $\mathcal{NP}$-complet, l'effort pour chercher une solution optimale est trop important pour qu'une solution soit réaliste.

Formation en-ligne de coalitions

Comme montré dans notre étude de l'état de l'art de la formation de coalitions pour la classe de problème *MR-ST-IA* (Multi-Robot task, Single Task robot, Instantaneous Assignment – c.f. la section 3.5.2, page 47), très peu de problèmes MRTA présentent exactement la structure d'allocation instantanée dans une seule fois pour appliquer un algorithme de type DCOP comme mentionné ci-dessus. L'allocation devrait prendre en compte la dynamique de l'environnement et des ressources de l'équipe robotique. Cette dynamique se retrouve par exemple dans la mission d'observation des objets mobiles étudiée dans [Parker, 1999], ou dans les applications où les tâches peuvent changer pendant l'exécution du système [Gerkey and Matarić, 2002]. Deux approches sont possibles pour gérer de tels cas : l'allocation répétitive et l'allocation en-ligne (section 3.5.2).

Dans le cadre d'une mission de sauvetage, nous nous sommes intéressés à l'allocation en-ligne. Au lieu de donner toutes les tâches à l'équipe de robots dès le début de la mission, les tâches sont supposées apparaître dynamiquement une par une. Une coalition est formée en-ligne lors de l'apparition d'une tâche. L'algorithme de base est simple :

– S'il y a une tâche, alors former une coalition pour la résoudre.

Nous examinons dans la suite les algorithmes de formation d'une coalition pour résoudre une tâche.

Allocation de rôles à base d'enchères

Nous nous basons sur le protocole de réseau de contrats CNP [Smith, 1980] pour la formation d'une coalition. Le CNP est choisi en raison du compromis

10. Le graphe des contraintes est un graphe dont les sommets sont les variables et dont chaque arête représente une contrainte entre deux variables.

11. Constraint Optimization Problem

entre le coût de communication et de calcul. La formation d'une coalition a besoin d'un robot *commissaire-priseur*. Ce dernier est choisi aléatoirement. Une alternative est que ce rôle soit endossé par le robot qui détecte la tâche. Afin de gérer les contraintes *concurrents* sur les rôles qualifiés de robots, deux approches sont possibles :

- *Allocation avec pré-filtrage* : Ce sont les robots qui choisissent l'ensemble de rôles qui ne violent pas les contraintes *concurrents* qui leur sont locales. Dans ce cas, chaque robot va essayer d'envoyer l'ensemble de rôles qui maximise sa somme de fonctions d'utilité. Puis, le commissaire-priseur choisi les meilleures enchères sans se soucier des contraintes.
- *Allocation avec filtrage tardif* : Dans cette approche, les robots donnent d'une part, un prix pour chacun de leurs rôles qualifiés, et d'autre part l'ensemble des contraintes de concurrence. Puis c'est le commissaire-priseur qui s'assure que les contraintes *concurrents* sont bien respectées. Cette approche nous confronte à la résolution d'un COP.

Comme les contraintes de concurrence sont vérifiées de façon décentralisée par les participants dans l'allocation avec pré-filtrage, le processus entier est beaucoup plus rapide que celui de l'allocation avec filtrage tardif. Le commissaire-priseur de la deuxième approche doit effectuer plus de traitements que celui de la première approche. Cependant, pour maximiser la somme totale des fonctions d'utilité, l'allocation avec filtrage tardif est meilleure car le robot qui s'occupe d'assurer les contraintes de concurrence possède une vue complète sur l'ensemble de ces contraintes. Reste que dans les deux cas, le commissaire-priseur a la charge de contrôler la contrainte cardinale (max, min) de rôle.

En ce qui concerne la communication, le coût pour l'allocation avec filtrage tardif est plus élevé que pour l'allocation avec pré-filtrage. En effet, avec le filtrage tardif chaque robot envoie toutes les enchères potentielles, ainsi que son propre ensemble de contraintes de concurrence. La taille du message est donc plus grande.

Enfin, l'approche avec pré-filtrage est plus décentralisée que celle avec filtrage tardif. La première approche est donc à préférer pour rester dans notre logique où nous voulons que les robots soient les plus autonomes possible.

Algorithme d'allocation avec pré-filtrage

L'algorithme 16 prend deux entrées. Il s'agit de la spécification de la coalition C et de l'ensemble des enchères $bids_C$ émis par les robots sur les rôles de C.

L'algorithme 16 assigne les rôles aux robots en deux étapes. La première étape (de la ligne 2 à la ligne 7) isole les allocations possibles pour chaque rôle. La deuxième étape est réalisée rôle par rôle (itération qui débute en ligne 9). Pour un rôle donné, l'ensemble des allocations potentielles isolé dans la première étape est trié dans l'ordre décroissant des valeurs d'utilités (ligne 12). Ensuite, nous sélectionnons parmi cet ensemble trié le plus grand nombre possible d'allocations en commençant par celles ayant les plus grandes utilités (ligne 13). Ce nombre ne peut pas excéder le nombre maximal d'assignations admis pour le rôle en question.

Algorithme 7: Allocation avec pré-filtrage

Entrées : C, $bids_C$.
/* C est la spécification d'une coalition. */
/* $bids_C$ est la l'ensemble de toutes les enchères pour tous les rôles dans C. */

Sorties : Ensemble d'assignations valides $\mathcal{VA}$ en cas de succès, $\emptyset$ sinon.

1 **début**
2 **pour chaque** $r \in C.roles$ **faire**
3 $bids[r] \leftarrow \emptyset$;
   /* $bids[r]$ l'ensemble de toutes les allocations potentielles pour le rôle r */
4 **fin**
5 **pour chaque** $bid \in bids_C$ **faire**
6 $bids[bid.role] \leftarrow bids[bid.role] \cup \{bid\}$;
   /* $bid.role$ désigne le rôle qui a pour enchère bid. */
7 **fin**
8 $\mathcal{VA} \leftarrow \emptyset$;
9 **pour chaque** $r \in C.roles$ **faire**
10 **si** $\mid bids[r] \mid < min_r$ **alors**
11 retourne $\emptyset$;
12 *trier les éléments dans* $bids[r]$ *ordre décroissant selon la valeur de l'enchère, i.e. la valeur de la fonction d'utilité*;
13 $\mathcal{VA} \leftarrow \mathcal{VA} \cup \{$*les* $\min(\mid bids[r] \mid, max_r)$ *premiers éléments dans* $bids[r]\}$;
14 **fin**
15 retourne $\mathcal{VA}$;
16 **fin**

Nous analysons maintenant la validité, la complétude et l'optimalité de l'algorithme proposé. La première propriété signifie que l'algorithme génère une solution correcte. La complétude est la garantie de trouver la solution si elle existe. Enfin, l'optimalité désigne que la solution obtenue par l'algorithme est optimale.

Proposition 5.1 (Validité) *La solution générée par l'algorithme 16 est correcte.*

Démonstration. La solution est correcte si elle respecte la définition 5.2 (page 87). Tout d'abord, chaque robot n'émet que des enchères sur ses rôles qualifiés, donc compatibles avec ses capacités. Pour ce qui est des contraintes de concurrence, elles sont vérifiées par la solution à cause du pré-filtrage effectué par les robots. En effet, un robot ne transmet au commissaire-priseur que des rôles qu'il peut endosser sans violer les contraintes de concurrence.

Reste les contraintes de cardinalité. L'algorithme retourne un ensemble vide s'il n'y a pas assez d'offres pour au moins un rôle (lignes 10 et 11). Enfin, la ligne 13 empêche l'attribution d'un rôle à un nombre de robots supérieur au maximum admis. □

Supposons que les enchères des robots sont réparties uniformément entre les rôles. Avec $| C.roles | = l$, $| bids_C | = p$, nous avons la proposition suivante.

Proposition 5.2 (Complétude et optimalité) *L'algorithme 16 garantit de trouver la solution optimale quand elle existe avec une complexité de* $O(l + p + p\log(p))$.

Démonstration. La première étape (ligne 2 – 7) de l'algorithme prend $l + p$ pas pour partitionner les p allocations potentielles dans un ensemble comportant l rôles.

Comme on a fait l'hypothèse que les enchères sont réparties uniformément, alors il y a $t = p/l$ allocations en moyenne pour chaque rôle. Donc, l'opération de tri dans chaque itération de la troisième boucle (ligne 9 – 14) contribue avec une complexité de $O(t\log(t))$. La complexité de toute la boucle est $O(lt\log(t))$ ou $O(p\log(t))$. Au total, la complexité temporelle de l'algorithme est $O(l + p + p\log(t))$.

Le pire cas, c'est celui où l'équipe de robots est homogène et que tous les rôles sont qualifiés pour tous les robots. Alors, on a $p = nl$, et la complexité dans ce cas est en $O(nl + l + nl\log(n))$.

Enfin, comme la recherche est exhaustive, nous avons la garantie que la solution optimale, i.e. si elle existe, sera trouvée. □

Algorithme d'allocation avec filtrage tardif

L'allocation avec filtrage tardif nous amène à résoudre un COP pour lequel nous pouvons utiliser un algorithme COP connu[Dechter, 2003]. Cependant, nous donnons un algorithme approximatif pour les raisons suivantes. Premièrement, le COP est un problème $\mathcal{NP}$-complet, cela veut dire qu'il n'existe pas d'algorithme dont la complexité de temps soit polynomiale. Or, dans le contexte d'une application en temps réel, il faut éviter ce genre algorithme. Deuxièmement, un algorithme pour un COP général, peut ne pas prendre en compte les spécificités d'un problème pour l'optimiser.

Nous proposons l'algorithme 13 [12] basé sur une heuristique. Il ne garantit pas de solution optimale, mais produit une solution satisfaisante dans un temps raisonnable. Cet algorithme est de type *retour sur trace* [13]. Il cherche une solution de façon gloutonne : il avance rôle par rôle en commençant par le rôle dont le poids est le plus grand. Pour chaque rôle, l'enchère avec la valeur la plus grande est prise. Chaque fois il prend une enchère, toutes les enchères qui ne sont pas encore considérées seront filtrées pour éliminer celles en conflit avec la dernière assignation choisie. Cette technique *regarder devant soi* [14] a pour but d'assurer qu'une allocation choisie est sans conflit avec celles déjà sélectionnées. S'il arrive à une situation où aucune solution ne peut être trouvée, il effectue un retour arrière pour annuler la dernière allocation et essayer avec une autre. Les détails de cet algorithme sont donnés dans la fonction récursive assigner (fonction *assigner*, page 93).

12. la définition de la fonction $assigner(i, j, k)$ est donnée dans la page 93
13. backtracking
14. Look-ahead, ou forward checking

Algorithme 8: Heuristique pour l'allocation avec filtrage tardif

Entrées : C, $bids_C$.

Sorties : Ensemble d'assignations valide $\mathcal{VA}$ en cas de succès, $\emptyset$ autrement.

```
1  début
2      pour chaque r ∈ C.roles faire
3          bids[r] ← ∅;
4      fin
5      pour chaque bid ∈ bids_C faire
6          bids[bid.role] ← bids[bid.role] ∪ {bid};
7      fin
8      VA ← ∅;
           /* Commencer par l'enchère ayant la valeur la plus
       grande pour le rôle le plus prioritaire. */
9      si assigner(1, 1, 0) alors
10         retourne VA ;
11     sinon
12         retourne ∅;
13 fin
```

Engagement des robots et ré-affection des rôles

Les robots peuvent jouer plus d'un rôle en même temps dans les différentes coalitions et donc travailler sur plusieurs tâches simultanément. Ainsi, la formation d'une nouvelle coalition pourrait *opportunément* utiliser les robots déjà occupés sur une tâche pour travailler sur une autre tâche. La ré-allocation de rôles a pour but d'améliorer la qualité de la solution par l'allocation dynamique. Nous distinguons deux types d'engagements d'un robot avec les rôles qui lui sont assignés.

Engagement fort et engagement faible Les robots se voient assigner des rôles en fonction d'enchères qu'ils ont placées. Pour un rôle r, les premiers min_r robots ayant les valeurs d'enchères les plus hautes [15] seront désignés avec des allocations requises d'un engagement fort. Les robots qui sont classés à partir de la place $min_r + 1$-ième jusqu'à la max_r-ième seront assignés avec un engagement faible. Un engagement fort exige que le robot participe la réalisation de la tâche jusqu'au bout. Par contre, un engagement faible permet au robot de quitter le rôle dans la coalition pour participer à une autre coalition si cela est bénéfique.

Ré-allocation des rôles Lors d'une nouvelle enchère, un robot déjà membre d'une coalition peut placer des enchères sur des rôles qui ne sont pas en contradiction avec les rôles pour lesquels il est fortement engagé. S'il est sélectionné, il peut le cas échéant se désengager des rôles pour lesquels il est faiblement engagé. Ce désengagement peut être pour respecter des contraintes de concurrence entre les nouveaux rôles et ceux pour lesquels il est déjà engagé, mais faiblement.

15. Rappel que les robots utilisent la fonction d'utilité pour évaluer leur enchère.

Fonction assigner(i, j, k) : boolean

```
   /* considère la j^ieme enchère placée sur le rôle r_i.      */
   /* k est le nombre d'enchères pris pour rôle r_i, c-a-d,
      le nombre de robots assignés au rôle r_i.                */
 1 début
 2   si i >| C.roles | alors                  /* solution trouvée. */
 3     retourne true;
       /* dépassement de la dernière enchère du rôle r_i       */
 4   si (j >| bids[r_i] |) alors
 5     si k ≥ min_{r_i} alors                 /* passer au rôle suivant. */
 6       retourne assigner(i + 1, 1, 0)
 7     si VA = ∅ alors                        /* pas de solution. */
 8       retourne false;
         /* back-track: annuler la dernière allocation,
            passer à l'enchère suivante du même rôle.          */
 9     revenirDeLaDerniereAllocation();
10     i ← l'index du dernier rôle affecté;
11     j ← l'index de l'enchère considérée sur le rôle r_i;
12     k ← le nombre d'allocations réalisées pour le rôle r_i;
         /* passer à l'enchère suivante, la (j+1)^ieme, sur le
            rôle r_i, avec k robots ayant été assignés au
            rôle r_i.                                          */
13     retourne assigner(i, j + 1, k);
14   fin
15   candidat ← le j^ieme élement dans bids[r_i];
       /* forward checking                                    */
16   pour chaque bids[r_l]|l > i faire
         /* supprimer toutes les enchères en avant, qui
            sont en conflit avec l'enchère choisie             */
17     effacerLesEnchèresEnConfit (candidat, l);
         /* pas de solution sur cette branche, revenir en
            arrière.                                           */
18     si | bids[r_l] |< min_{r_l} alors
19       revenirDuDernierEffacement ();
20       retourne assigner(i, j + 1, k);
21     fin
22   fin
23   VA ← VA ∪ {candidat};                    /* enchère choisie. */
       /* essayer avec l'enchère suivante sur le même rôle,
          ou passer au rôle suivant.                           */
24   si (k + 1 < max_{r_i}) alors
25     retourne assigner(i, j + 1, k + 1);
26   sinon
27     retourne assigner(i + 1, 1, 0);
28 fin
```

Une autre raison est pour quitter des rôles avec faible engagement est l'état du robot en terme de ressource telles que la batterie.

5.3 Discussion

5.3.1 Organisation statique

Dans notre proposition d'extension du modèle AGR, l'organisation est statique. Les rôles ne sont remplacés que lorsque c'est nécessaire (e.g. panne d'un robot). Par conséquent, les robots passent l'essentiel du temps à traiter des tâches. Peu de temps est réservé à la gestion de l'organisation, d'où un meilleur rendement. L'organisation peut être complexe avec des contraintes sur les rôles et des interactions entre groupes. Typiquement un même robot doit jouer des rôles dans des groupes différents afin de permettre la coordination de l'activité des groupes en question. De plus, chaque robot ne récupère/conserve qu'un fragment de la description de l'organisation qui l'intéresse.

A côté de ces avantages, l'approche introduit également certaines limites. D'abord, l'allocation de rôles n'est pas toujours optimale sur la durée de la mission. La répartition des rôles est peut-être optimale au début lors du déploiement, mais en fonction des mouvements des robots, une autre répartition peut s'avérer plus adéquate. Enfin, le concepteur/architecture doit avoir une bonne vision d'ensemble pour concevoir une « bonne » organisation.

5.3.2 Organisation émergente

Dans cette approche, nous proposons de former des coalitions et d'attribuer des rôles aux robots au fur et à mesure de la découverte des tâches. Ainsi, la formation de l'organisation émerge des tâches et de la répartition des ressources robotiques. Les avantages de cette solution sont les suivants :

- **Dynamicité, flexibilité et adaptabilité aux conditions/situations (déplacements/pannes/ressources des robots)**. Les rôles sont redistribués à l'apparition de chaque tâche permettant de prendre en compte l'état et les ressources des robots. De plus, les coalitions sont robustes. Si un robot ne parvient pas à s'acquitter de ses responsabilités, nous avons juste besoin de recruter un nouveau robot de reprendre le rôle. Au niveau individuel, chaque robot réagit rapidement aux changements de son environnement et de son état intérne pour sélectionner le rôle le plus approprié.
- **Distribution et extensibilité**. Parce que la tâche est supposée être détectée de façon indépendante et localement par des robots, il n'y a aucune contrainte d'échelle à la formation de la coalition. Dans une application USAR par exemple, considérons la situation où plusieurs victimes (les tâches) sont détectées en même temps. Si nous supposons qu'il existe suffisamment de robots déployés sur la zone du sinistre, différentes coalitions pour assister les différentes victimes se formeront.
- **Réactivité**. Le système est capable de répondre rapidement aux événements qui apparaissent pendant la mission. Au niveau de la coopération entre les robots dans le système, à condition que la communication soit

fiable, les robots peuvent répondre rapidement aux requêtes, donc participer à la formation d'une nouvelle coalition pour résoudre une tâche. Au niveau individus, un robot peut réagir immédiatement aux défaillances partielles selon les conditions pré-définies dans *Context Manager*.

Cependant, cette approche ne convient pas pour les organisations complexes. Il n'y a notamment pas de support générique pour les interactions entre coalitions. De plus, les robots « gaspillent » une partie de leurs ressources (temps, énergie, calcul) à former les coalitions.

Organisation émergente et écoute flottante

Une approche alternative pour faire émerger l'organisation consiste à recourir au mécanisme de « l'écoute flottante[16] » comme cela a été fait dans [Legras and Tessier, 2003, Legras and Tessier, 2004]. Dans cette proposition, une organisation se compose de plusieurs groupes créés à l'initiative des agents. Un agent peut rejoindre ou quitter un groupe à n'importe quel moment. Ainsi, l'organisation émerge et évolue en fonction des décisions prises par les agents. L'absence de contrôle de l'organisation est compensé par l'écoute flottante. En effet, chaque agent maintient une carte qui représente les croyances de l'agent concernant l'organisation du système multi-agents.

Notons que dans ce travail, il n'y a pas de notion de rôle comme dans notre proposition. L'explicitation des rôles est de notre point de vue importante. En effet, elle permet d'abstraire la description de l'organisation et la rendre plus facilement compatible avec différents types de robots.

Conclusion du chapitre

Nous avons présenté dans ce chapitre nos solutions pour l'allocation des rôles aux robots. Nous avons proposé deux approches. La première consiste à construire des organisations de manière statique. La seconde est dynamique dans la mesure où l'organisation est émergente.

Chacune de ces deux approches a ses avantages et inconvénients. Le choix de l'approche appropriée dépendra de la nature de l'application. Une solution hybride serait de les combiner. Il s'agit d'utiliser l'approche basée sur une organisation statique en début de mission, puis pendant l'exécution, le système s'adapterait lui-même par la formation dynamique de coalitions.

16. overhearing

6 Implémentation et Expérimentations en simulations

Sommaire

Ce chapitre présente des expérimentations qui valident les résultats théoriques des chapitres précédents. Il se compose de 5 parties. La première partie (section 6.1) sert à présenter la vue générale du simulateur Scène, qui a été développé au cours de ce travail de thèse pour tester les solutions proposées. Le simulateur est écrit en Smalltalk/Squeak 3.

Dans les deuxième et troisième parties (section 6.2 et 6.3), nous présentons nos expérimentations pour valider les résultats théoriques discutés dans le chapitre 4 sur la sensibilité à la connectivité. La deuxième partie présente nos expérimentations pour valider la première application de la sensibilité. C'est la

vérification de la robustesse de la connectivité dans un réseau sans fil (voir la section 4.5, page 68). Puis, les résultats des expérimentations avec l'utilisation de la sensibilité pour le maintien de la connectivité dans l'exploration multi-robots sont présentés dans la partie suivante. Ces expérimentations ont pour objectif de montrer l'utilisation de la sensibilité à la connectivité dans le contrôle décentralisé des déplacements de robots de sorte que la connectivité globale du réseau soit préservée.

Les expérimentations en simulation concernant l'allocation des rôles sont présentées dans les deux dernières parties. La section 6.4 traite de l'allocation des rôles par l'adaptation du modèle AGR dont la base théorique a été exposée dans section 5.1, page 79 du chapitre 5. La section 6.5 décrit une expérimentation d'un scénario robotique USAR qui rassemble les préoccupations du maintien de la connectivité et de l'allocation des rôles pour former des coalitions en confrontant des tâches diverses dans une mission de sauvetage robotisée.

6.1 Simulateur Scène

Notre étude porte sur sur les aspects haut niveau d'un système multi-robots. Un hypothèse importante est que les robots doivent être capables échanger les informations. Nous avons envisagé de recourir au simulateur Player/Stage [1]. Mais, nous l'avons écarté car il n'offre pas de support pour les communications explicites entre robots. En effet, Player/Stage traite avantage des aspects physiques de bas niveau.

Nous avons également envisagé le recours à NetLogo [2]. Ce simulateur est largement utilisé dans la communauté multi-agents. Cependant, les agents dans NetLogo sont également incapables de directement échanger des données.

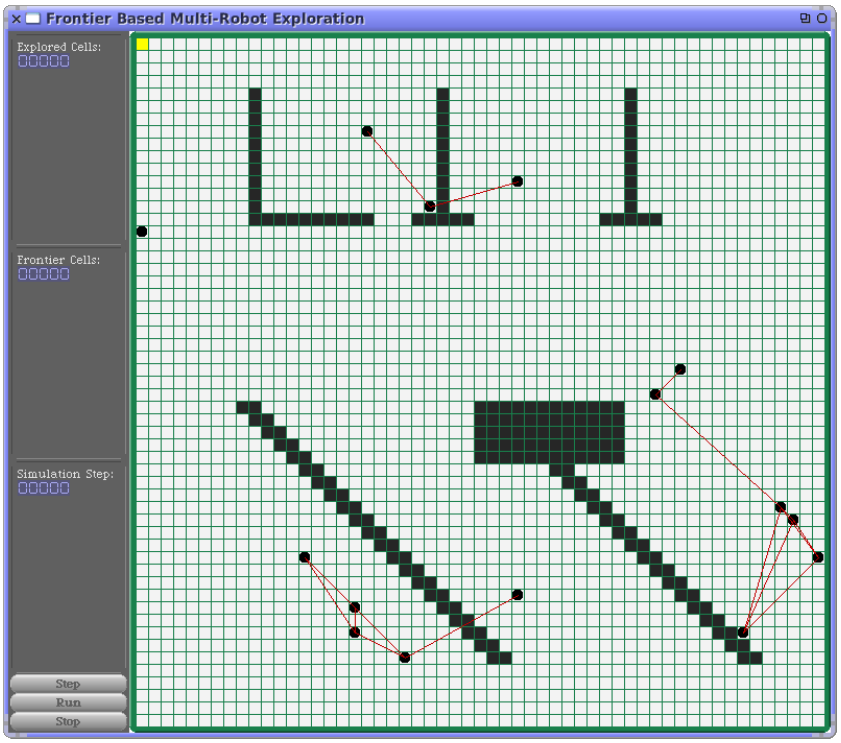


Figure 6.1 – *Capture d'écran du simulateur* Scène

Face à ces limitations, nous avons choisi de développer notre propre simulateur au lieu d'utiliser un de ceux existants. Ainsi, nous avons réalisé Scène qui nous sert de plate-forme d'expérimentation pour tester les solutions que nous proposons. La figure 6.1 donne l'interface d'utilisateur pour des simulations multi-robots avec Scène. Le développement a été réalisé avec le langage Smalltalk [Goldberg and Robson, 1983, Nierstrasz et al., 2009]. Ainsi, en plus de la puissance d'expression de l'approche objet, nous disposons de la flexibilité des langages de script du fait de la dynamicité de Smalltalk (réflexivité, recompilation à chaud, typage dynamique, ...)

1. `http://playerstage.sourceforge.net/`
2. `http://ccl.northwestern.edu/netlogo/`

6.1.1 Monde simulé dans Scène

La plate-forme Scène simule un monde 2D, dans lequel les robots communicants sont déployés pour réaliser différentes tâches. Un monde simulé dans Scène est abstrait et représenté simplement par une grille à 2 dimensions composée de cellules carrées identiques.

La grille contient les agents situés dans le monde, les tâches à réaliser par les robots, et les obstacles. Pour simplifier, la taille d'une cellule et d'un robot sont modifiables de telle sorte qu'un robot puisse être contenu dans une seule cellule, de même pour un obstacle ou une tâche. La taille d'une cellule est utilisée également dans la mesure de distance. Autrement dit, une cellule est une unité de distance dans les simulations.

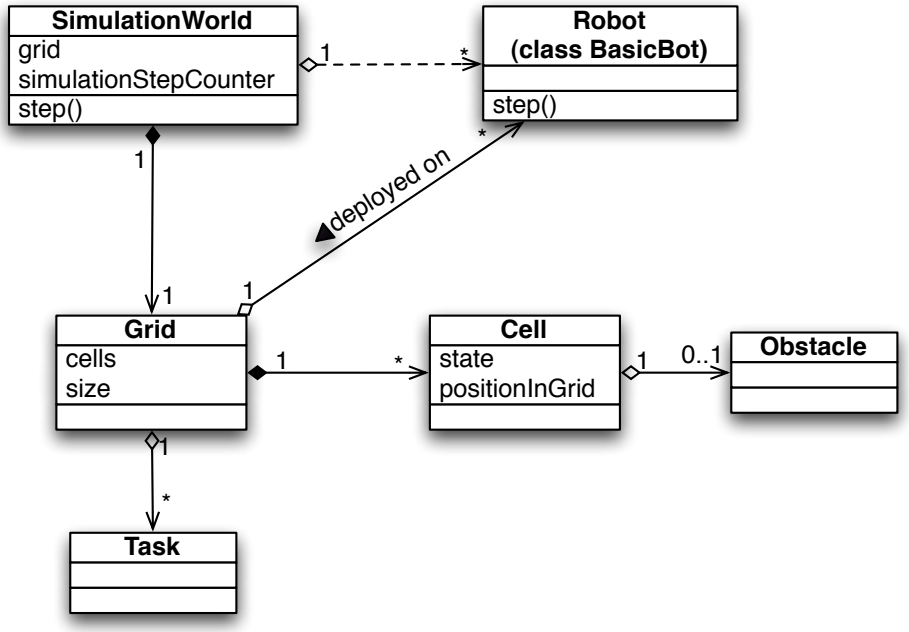


Figure 6.2 – *Classes principales de* Scène

Le figure 6.2 montre les relations entre les classes principales du simulateur. Un robot est représenté par la classe BasicBot qui sert de classe mère pour l'ensemble des robots. En général, on dérive de nouvelles sous-classes à partir de BasicBot pour construire de nouvelles sortes de robots. Un robot est défini par un corps et une tête (voir détail dans la section 6.1.2). Mais, dans un souci de lisibilité, nous n'avons pas fait figurer les classes correspondantes sur la figure 6.2. Enfin, les tâches à réaliser par les robots sont définies et déployées sur la grille. C'est la signification de la flèche représentant la relation entre la classe tâche et la grille.

Fonctionnement des simulations

Scène est un simulateur discret : le temps est subdivisé en « tic » temporels [3]. Un tic est défini comme une période de temps suffisante pour qu'un robot

3. Cette hypothèse sur l'espace de planification discret a été introduite dans la section 4.1.1, page 57 du chapitre 4.

puisse réaliser certaines tâches comme par exemple mettre à jour son tableau de connectivité, se déplacer d'une cellule à une autre cellule voisine, compléter les échanges de données collectées pendant le tic précédent avec ses voisins, etc. Quand on programme un agent (cf. section 6.1.2), ces actions doivent être mises dans la méthode `step` de la classe correspondantes. Autrement dit, la méthode `step` contient les actions qui doivent être exécutées à chaque « tic ».

Il y a un seul thread (fil d'exécution) dans la simulation pour gérer l'activité de l'ensemble des robots. Ce thread principal exécute le méthode `step` d'un objet `SimulationWorld` dans une boucle principale. A son tour, cette méthode `step` va appeler la méthode `step` de chaque agent.

Communication entre agents

Les agents déployés sur la grille communiquent et découvrent leurs voisins au moyen des services de base fournis par le monde simulé (les méthodes `sendMessage(message, receivers)` et `neighboringRobotsOf(robot)` dans la classe `SimulationWorld` sur la figure 6.3).

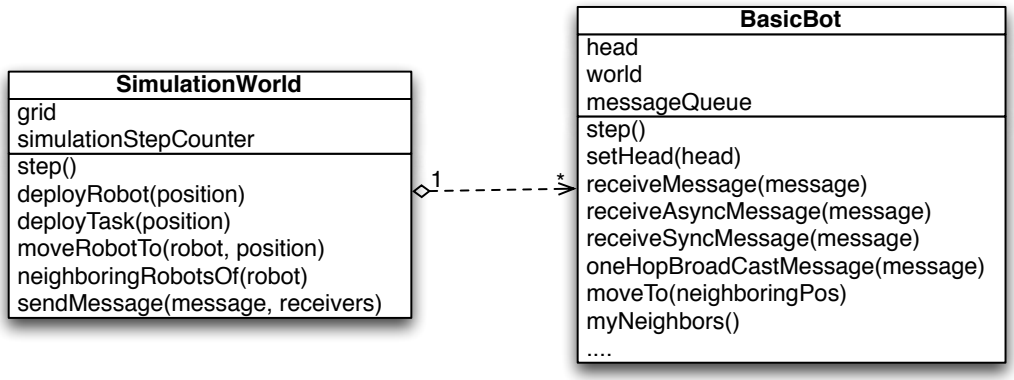


FIGURE 6.3 – *Interaction du corps du robot avec l'environnement en utilisant les services de base fournis par le monde simulé*

Le modèle de communication implémenté est celui du graphe unité (voir la section 2.3.1 du chapitre 2, page 24). Pour simplifier, on suppose que la transmission des messages est fiable, sans perte ou sans retard. Au niveau agent (robot), les agents sont capables de recevoir deux types de messages : des messages synchrones et des messages asynchrones. Les messages asynchrones reçus sont stockés dans la boîte aux lettres des agents. Ils sont traités sur la base du premier arrivé, premier servi dans le tic suivant. Les messages synchrones nécessitent un traitement et retour du traitement immédiat dans un tic pour simuler les échanges de données qui ont lieu assez rapidement dans un tic.

6.1.2 Programmation des agents

Un agent (robot) se compose d'une tête et d'un corps. Les services liés à la perception et à l'action dans le monde appartiennent au corps de l'agent. Pour contrôler l'agent et exploiter ses capacités, il faut programmer la tête. La séparation du corps de la tête a pour objet de séparer les préoccupations concernant les

capacités physiques des robots (le corps), et comment on emploie ces capacités dans des individus (par programmation de la tête) indépendamment du robot.

Création des agents

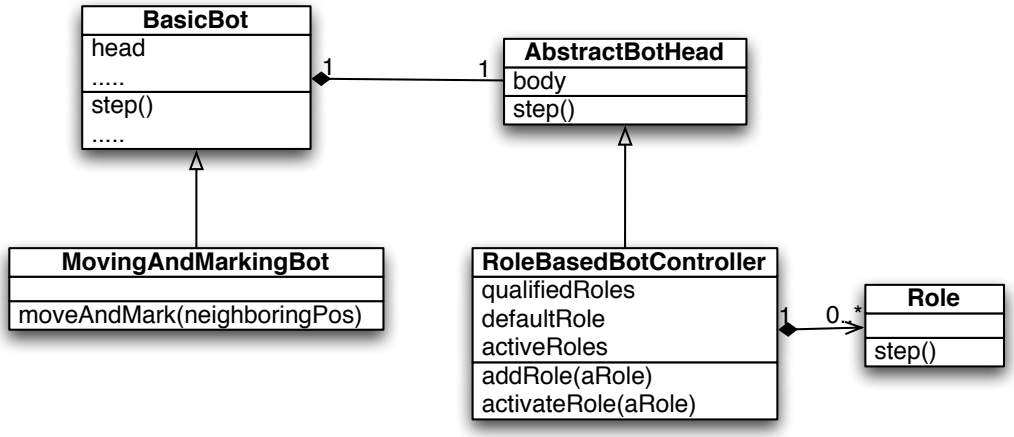


FIGURE 6.4 – *Classes nécessaires à la création d'un robot*

Pour créer un robot simulé, nous devons construire le corps par une nouvelle classe dérivée de la classe abstraite `BasicBot` (cf. figure 6.4). Puis, nous dérivons une nouvelle sous-classe de la classe abstraite `AbstractBotHead`. À ce moment là, un nouveau robot peut être créé en attachant la tête au corps. Le code source de création d'un robot est le suivant :

```
bot11 := MovingAndMarkingBot new.
bot11 head: RoleBasedBotController new.
```

FIGURE 6.5 – *Création d'un robot simulé.*

Ajout des rôles aux robots

Un robot est contrôlé par les rôles qu'il prend. Lors de la création d'un robot, nous pouvons ajouter dynamiquement des rôles. Les rôles sont ajoutés à la tête du robot comme ci-dessous :

```
bot11 head addRole: MaskProvider new.
bot11 head addRole: Pusher new.
```

FIGURE 6.6 – *Addition des rôles qualifiés au robot. Les classes* MaskProvider *et* Pusher *sont dérivées de la classe* Role.

Activation et exécution des rôles

Les rôles formant une coalition sont assignés aux robots selon l'un des algorithmes 16 ou 13 décrits dans la section 5.2.3, page 88. La tête du robot peut

alors activer le rôle en l'ajoutant à l'ensemble des rôles actifs du robot, comme montré dans la figure 6.7. L'opération inverse qui consiste à désactiver un rôle, se traduit donc par la suppression du rôle à désactive de l'ensemble des rôles actifs du robot.

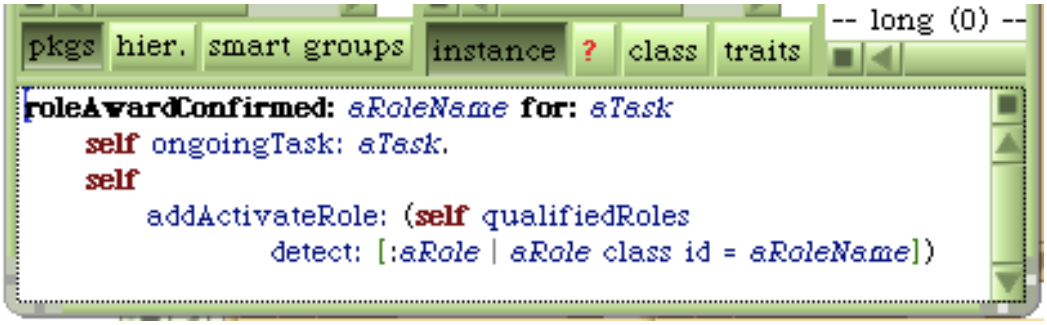


FIGURE 6.7 – *Activation d'un rôle.*

Lorsqu'un rôle est activé, il participe au contrôle du robot. Ainsi, à chaque pas de simulation, la méthode `step` du robot est appelé par le thread principal de l'application. Puis, à son tour, elle appelle les méthodes `step` de tous les rôles actifs du robot.

6.1.3 Réalisation d'une simulation et récupération des résultats

Le premier pas pour réaliser une simulation dans SCÈNE consiste à définir le monde comme montré sur la figure 6.8. La définition d'un monde simulé dans SCÈNE consiste à définir la taille de la grille. Cette taille est déterminée par une longueur et une largeur en nombre de cellules.

```
world := SimulationWorld withGrid: (Grid withSize: 60 @ 45).
robots := RobotFactory create: 50 of: MovingAndMarkingBot.
robots do: [:r | r head: RoleBasedBotController new].
world randomlyDeployRobots: robots.
```

FIGURE 6.8 – *Création d'un monde simulé avec des robots homogènes*

Après la création du monde simulé vient la création des robots. Afin de réaliser une création en masse de robots homogènes, nous disposons d'une classe de type fabrique, nommée `RobotFactory`. Enfin, les robots sont déployés sur la grille. Dans l'exemple donné par la figure 6.8, les robots sont placés de manière aléatoire sur la grille du monde simulé. Cependant, un robot peut être placé dans une case précise comme indiqué dans la figure 6.9.

```
r := MovingAndMarkingBot new.
r head: RoleBasedBotController new.
world
    deployRobot: r
    at: 50 @ 15.
```

FIGURE 6.9 – *Déploiement des robots*

En ce qui concerne les tâches, elles peuvent être déployées de la même manière que les robots.

Dans les sections suivantes, nous présentons les simulations réalisées au moyen du simulateur Scène, afin de tester les différentes solutions proposées dans ce travail de thèse. A cet effet, nous exploitons un mécanisme de journalisation des données (« log ») fourni par Scène pour recueillir les résultats.

6.2 Vérification de la complétude de la liste d'accesseurs et de la robustesse de la connectivité

Pour maintenir la connectivité, un robot doit être conscient de la connectivité globale du réseau afin d'être capable d'effectuer un déplacement tout en restant connecté. L'exigence est que :

1. le calcul doit être décentralisé, c'est-à-dire que le robot prend les décisions de manière autonome (notamment pour ce qui concerne ses déplacements), et que
2. le nombre de messages doit être minimisé.

L'approche à base de sensibilité proposée dans le chapitre 4 répond bien au point (1) ci-dessus. Avec l'approche originale basée sur un nœud de référence, nous pouvons arriver à déduire la connectivité globale à partir des connaissances locales sur la topologie du réseau. En outre, la base théorique que nous avons élaborée montre que la sensibilité permet d'assurer la robustesse de la connectivité.

Pour construire la sensibilité, les robots doivent échanger leurs connaissances locales sur la topologie du réseau par envoi de messages. La version triviale de notre algorithme est complète. Malheureusement, elle émet un grand nombre de messages dans le pire cas. C'est-à-dire que l'exigence du point (2) mentionné ci-dessus n'est pas satisfaite. La version avec filtrage que nous avons proposée n'est pas théoriquement complète. Les expérimentations conduites dans cette section ont pour l'objectif d'évaluer le nombre de messages émis par les algorithmes 7, et 13. De plus, une application directe de la sensibilité à la vérification distribuée de la robustesse est également présentée.

Ces simulations sont réalisées avec des réseaux correspondants à différentes configurations. Elles ont été générées aléatoirement en faisant varier le nombre de nœuds (donc la taille des réseaux) et leur densité : un nœud peut avoir un nombre de voisins allant de 3 à 18. À noter qu'il n'est pas nécessaire de varier la portée des interfaces de communication sans fil, car cela est équivalent au changement de la densité du réseau. Pour chaque paramètre, 50 configurations ont été générées pour les tester.

6.2.1 Complétude de la liste d'accesseurs

L'algorithme 7 (page 62) assure théoriquement que la liste d'accès générée est complète pour tous les robots autres que le Robot de Référence dans le réseau. Cependant, nous avons constaté que cet algorithme n'est applicable que pour un réseau de très petite taille car le nombre de messages émis lorsque le graphe est complet, est important (voir nos discussions dans la section 4.5.2, page 70). Par

exemple, dans une simulation avec un réseau composé de 20 robots, le nombre moyen de voisins robot est de 4, cet algorithme a émis 1.072.254 messages, et cela signifie que chaque robot doit traiter environ 53.000 messages en moyenne.

La table 6.1 présente les résultats de nos simulations pour calculer le nombre de nœud ayant la liste d'accès incomplète avec l'algorithme 13 (page 64).

TABLE 6.1 – *Statistique sur le nombre de robots dont la liste d'accès est incomplète.*

Nombre de robots avec liste d'accès incomplètes	Taille du réseau					
	11	15	20	50	100	500
max	0	5	6	11	17	40
moyen	0	0	0	0	2	10

Comme on peut le constater dans la table 6.1, la liste d'accès des robots, exprimée sous forme d'un pourcentage, est incomplète dans certains cas. Mais, le nombre de robots concerné est petit, même pour grand réseau (plusieurs centaines de nœuds). En outre, la plupart des listes incomplètes n'ont qu'un seul robot d'accès qui manque. Par conséquent, pour plusieurs buts comme par exemple l'algorithme d'exploration multi-robots présenté dans [Le et al., 2009a], la liste construite par l'algorithme 13 est suffisante. Pour un réseau de taille inférieure à 10, l'algorithme fonctionne de manière satisfaisante, i.e. tous les robots possèdent une liste d'accesseurs complète.

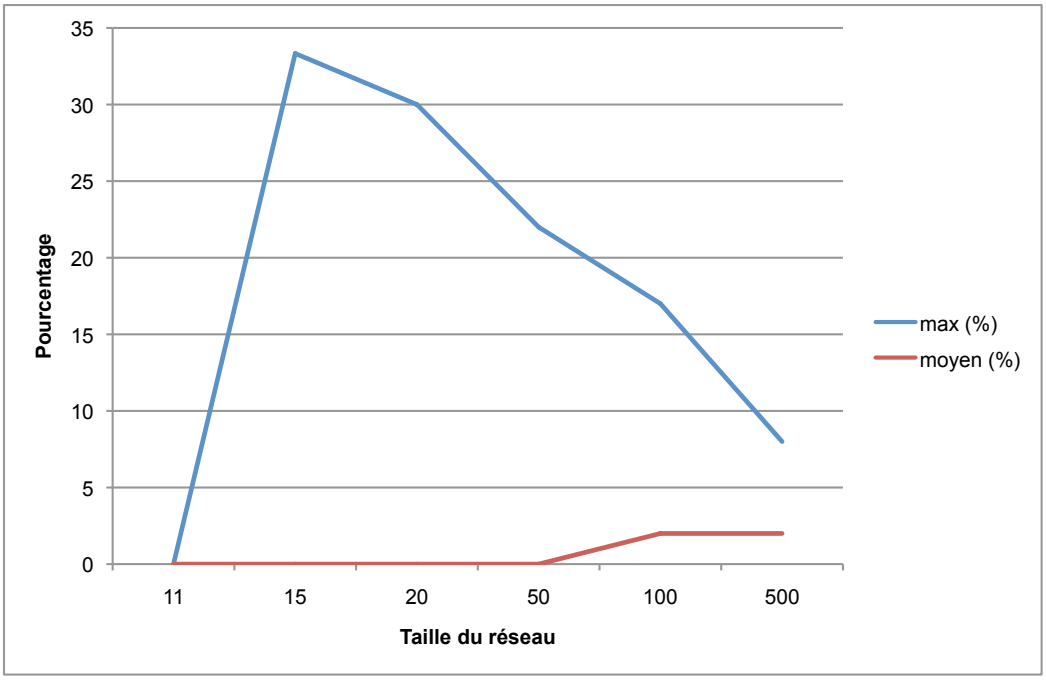


FIGURE 6.10 – *Pourcentages max et moyen des robots ayant leur liste d'accesseurs incomplète*

En terme de pourcentage de nœuds du réseau avec une liste incomplète, le graphique de la figure 6.10 montre que ce ratio moyen est inférieur à 2% (calculée par la fonction statistique médiane) pour les réseaux de taille de moins de 500 nœuds. Autre résultat intéressant : l'écart entre le ratio moyen et le ratio maximal tend à diminuer quand la taille du réseau augmente. Cela signifie que *ce ratio est plus stable quand la taille du réseau est assez grande*.

6.2.2 Vérification de la robustesse

Selon les éléments théoriques que nous avons présentés dans la section 4.5, page 68, les robots peuvent déterminer eux-mêmes de manière décentralisée s'ils sont critiques en terme de connectivité réseau à condition qu'ils possèdent une liste correcte de robots d'accès. La détermination des nœuds critiques est basée sur le théorème 4.4 (page 68), et 4.5 (page 69) pour les nœuds « ordinaires » et le corollaire 4.2 (page 69) pour le Nœud de Référence. La détection des nœuds critiques dans le réseau peut être effectuée simplement en attendant pendant une certaine période de temps afin que l'état de la sensibilité se stabilise, en appliquant le théorème et le corollaire précédents pour identifier les nœuds critiques. Dans le cas du Robot de Référence, selon le corollaire 4.2, si un de ses voisins trouve que le Nœud de Référence est son robot d'accès unique, alors le voisin annoncera au Nœud de Référence qu'il est critique [4].

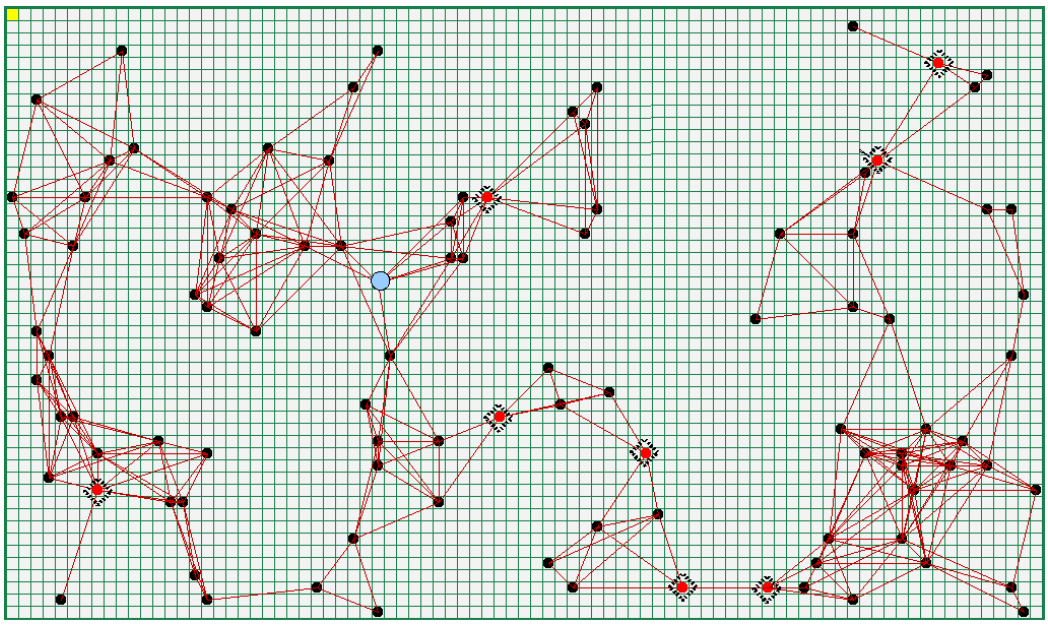


FIGURE 6.11 – *Capture d'écran de la simulation de détection de nœuds critiques. Le Robot de Référence est représenté avec un grand disque. Les nœuds critiques sont mis en évidence par un diamant. Il y a 100 nœuds dans le réseau. Les 8 nœuds critiques sont détectées avec succès.*

Pour la détection des nœuds critiques, il faut compléter d'abord la liste d'accesseurs. Nous utilisons pour cela la procédure présentée dans la section 4.5.2, page 70. Nous avons refait les simulations avec l'algorithme 13 (page 64). Notre algorithme détecte avec succès tous les nœuds critiques. Une capture d'écran des résultats obtenus avec le simulateur SCÈNE est donnée à titre d'illustration sur la figure 6.11.

La métrique utilisée pour évaluer les performances des algorithmes est la complexité en termes de communication, à savoir le nombre total de messages. Comme le montre le tableau 6.2, le nombre maximum de messages envoyés par un robot est d'environ 10. De plus, nous pouvons voir que le nombre moyen de messages envoyés par un robot est très faible (entre 3 et 4). Le graphique de la

4. Le Robot de Référence est le seul robot unique qui ne peut pas lui-même déterminer s'il est critique.

TABLE 6.2 – *Nombre total de messages moyens traités par chaque robot pour compléter sa liste d'accès.*

Messages par robot	Taille du réseau				
	15	20	50	100	500
max	6	7	9	9	9
moyen	3	3	4	4	4
min	2	2	3	3	3

figure 6.12 montre que le nombre de messages émis par un nœud du réseau est indépendant de la taille du réseau.

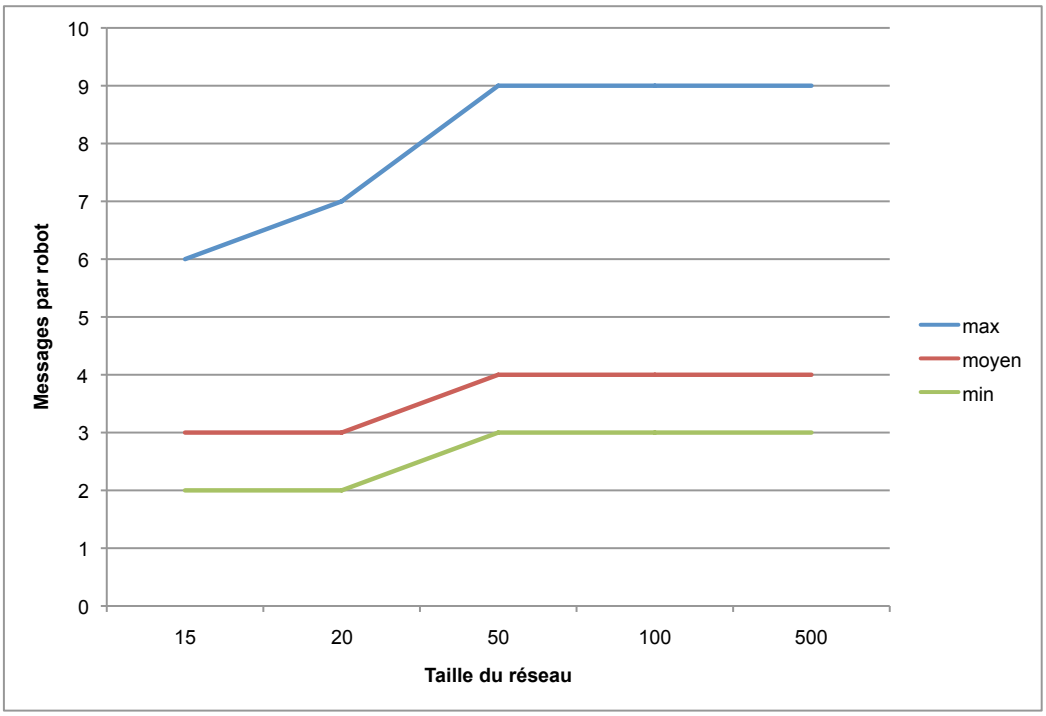


FIGURE 6.12 – *Pourcentage de robots ayant la liste d'accesseurs incomplète*

À partir de ces simulations, on peut donc établir une complexité empirique pour construire la liste d'accès complète de $O(kn)$ avec $3 \leq k \leq 9$. On peut comparer ce résultat avec celui de l'algorithme de [Ahmadi and Stone, 2006a, Ahmadi and Stone, 2006b], où la complexité de messages est de $O(n!)$ (voir la section 4.5.3, page 72 pour une comparaison plus complète).

6.3 Maintien de la connectivité dans l'exploration multi-robots

L'exploration est une application importante en robotique. L'approche la plus répandue est dérivée des travaux de [Yamauchi, 1997, Yamauchi, 1998]. L'idée de base est simple : *afin d'acquérir un maximum de nouvelles informations sur un environnement inconnu, les robots de l'équipe ont besoin de se déplacer jusqu'à la limite entre la zone connue et le territoire qu'ils n'ont pas encore exploré*. La limite entre les zones

explorées et non-explorées définit une « frontière », d'où le nom *d'exploration basée sur la frontière* pour l'ensemble des travaux dérivés de cette approche.

De nombreux travaux se sont intéressés à accélérer le processus d'exploration dans le contexte multi-robots et de diminuer l'incertitude de l'information acquise [Burgard et al., 2005, Sheng et al., 2006b, Rooker and Birk, 2007]. Ces travaux portent principalement sur la proposition de mécanismes de collaboration efficaces minimisant les chevauchements des zones couvertes par les robots. Nous avons choisi de modifier l'algorithme proposé par [Rooker and Birk, 2007] pour illustrer l'utilisation de la sensibilité à la connectivité dans le contrôle de déplacement de robots.

Basé sur l'approche de Yamauchi [Yamauchi, 1998] et étendu avec les contraintes d'un réseau sans fil, l'algorithme de Birk et Rooker assure que pendant l'exploration, aucun robot ne va être déconnecté du reste de l'équipe. Il s'agit en fait d'une approche totalement centralisée. Pour atteindre cet objectif, une entité centrale recueille les positions actuelles de tous les robots et génère un ensemble de configurations pour la flotte (ensemble de positions futures possibles des robots). En raison du nombre élevé de combinaisons disponibles, toutes les configurations ne peuvent être considérées, mais seulement un nombre limité d'entre elles. Parmi ce nombre de configurations générées, l'entité centrale choisit la meilleure selon une fonction d'utilité. Cette fonction attribue des pénalités aux positions occupées par des obstacles ou qui mettraient les robots hors de la portée radio les uns des autres.

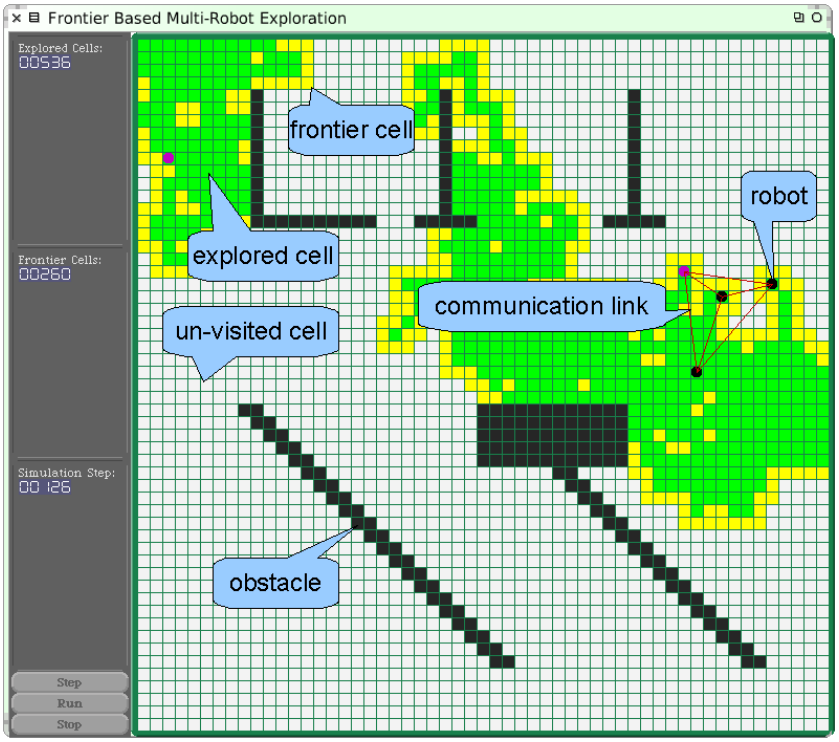
Par rapport à leur approche, le même résultat peut être obtenu de manière distribuée en utilisant notre solution pour la préservation de la connectivité, ainsi que pour le contrôle des déplacements des robots. La figure 6.13 (page 109) montre une capture d'écran de notre simulateur lors d'une phase d'exploration.

6.3.1 Mise en œuvre de l'algorithme d'exploration

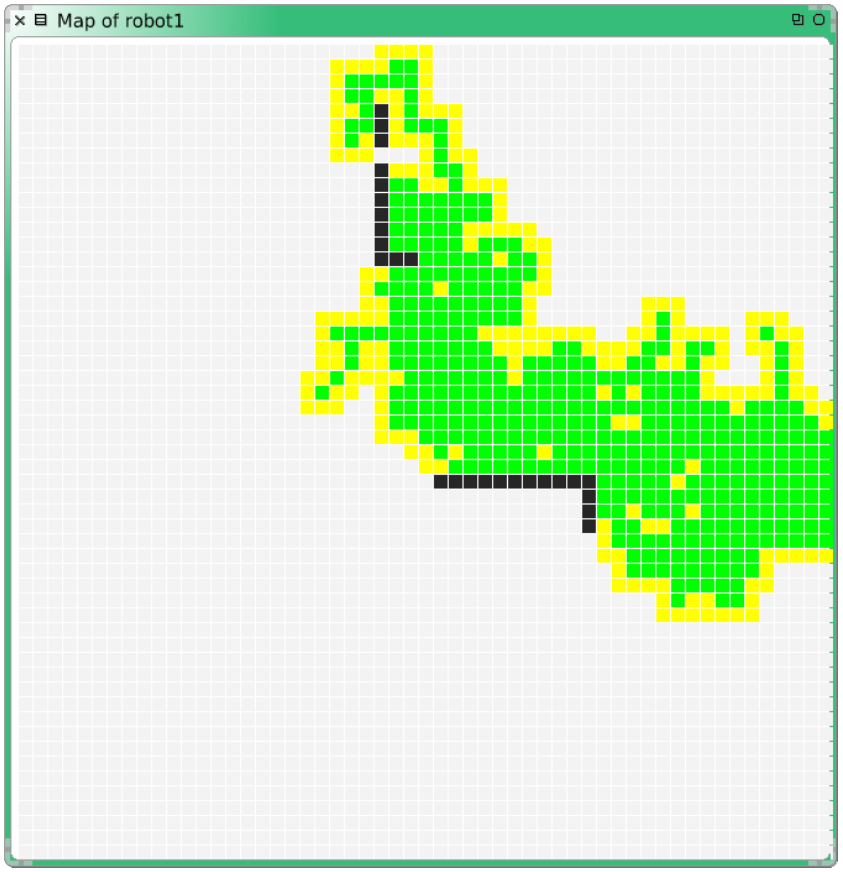
Dans le simulateur Scène, le terrain à explorer est modélisé par une grille de 45×45 cellules, appelée grille d'occupation dans le contexte de l'application d'exploration multi-robot. Chaque cellule de la grille possède une valeur parmi quatre possibles : *inconnue*, *visitée*, *frontière* et *obstacle*. Une cellule inconnue est celle qui n'a pas encore été visitée par un robot. Dès qu'un robot se positionne sur une cellule dépourvue d'obstacle, il marque cette cellule comme ayant été visitée. Il marque aussi les cellules voisines dans l'état *inconnues*, comme *obstacles* ou comme *frontières* selon qu'elles sont respectivement occupées par un obstacle ou non.

Les robots sont déployés aléatoirement sur le terrain. Pendant la mission, chaque robot construit une carte locale. Nous supposons que chaque robot est capable de se localiser seul en utilisant sa propre carte. Lorsque les robots perçoivent et mettent à jour l'état de cellules qui étaient jusque là inconnues, ils mettent à jour cette information dans leur propre carte, puis diffusent la mise à jour à leurs équipiers.

Un pas (tic) de simulation (voir la section 6.1.1 ci-dessus) est défini comme la période de temps dont un robot a besoin pour calculer sa position suivante et accomplir le déplacement correspondant. Nous supposons également que toutes les mises à jour liées au changement de la topologie du réseau de robots et à



(a) *L'écran principal de la simulation*



(b) *Carte locale d'un robot*

FIGURE 6.13 – *Capture d'écran d'une exploration multi-robots [Le et al., 2009a]*

la sensibilité à la connectivité sont également accomplies durant un seul pas de simulation. À chaque tic, la nouvelle position est déterminée comme suit : le robot calcule d'abord la frontière la plus proche par rapport à sa position actuelle. Puis, le robot se déplace vers une cellule libre parmi les cellules voisines de sa position initiale si ce déplacement ne le met pas hors de la zone de communication avec son *dernier* robot d'accès (voir l'algorithme 6, page 74 pour le calcul d'un déplacement en préservant la connectivité).

A part pour le maintien de la connectivité, tous les robots ont le même rôle à jouer : le rôle d'explorateur. Le squelette de l'algorithme de contrôle à exécuter sur chaque robot est fourni ci-dessous.

Algorithme 9: Algorithme d'exploration avec maintien de la connectivité

Entrées : L'ensemble de cellules frontières.
Sorties : La carte locale des individus est mise à jour.

1 **début**

`// un pas de simulation de l'exploration`

2 **tant que** *il y a encore des cellules frontières* **faire**

- mettre à jour le tableau de la connectivité;
- trouver la cellule frontière la plus proche en prenant en compte le le tableau de la connectivité;
- calculer le déplacement pour s'approcher vers la cellule choisie;
- réaliser le déplacement;
- diffuser les mises à jour aux voisins;

3 **fin**

4 **fin**

6.3.2 Résultats des simulations

Tout d'abord, nous évaluons la performance de l'algorithme d'exploration. Le résultat de cette évaluation est donnée dans le diagramme de la figure 6.14. Le nombre de robots dans les simulations varie de 5 à 10. Bien qu'il existe différentes pistes d'améliorations qui pourraient être ajoutées à notre implémentation, l'algorithme est déjà très intéressant avec un passage à l'échelle linéaire. En effet, plus il y a de robots, plus le temps d'exploration est court. Nous avons évalué le fonctionnement de l'algorithme dans deux cas : avec et sans limite de portée de communication. Afin de limiter la portée de communication, nous avons mis la portée de communication effective et la portée de communication avec marge de sécurité sécurité[5] à $R = 15$ et $r = 12$, respectivement (l'unité de distance étant une cellule de la grille).

Bien que la sensibilité à la connectivité permet aux robots de rester proches les uns des autres, il est difficile de maintenir le réseau multi-robots connecté dans un environnement avec des obstacles. Afin d'évaluer le maintien de la connectivité seul, nous avons opéré une série de simulations sans obstacles. Nous avons

5. voir la section 4.6.1, page 73.

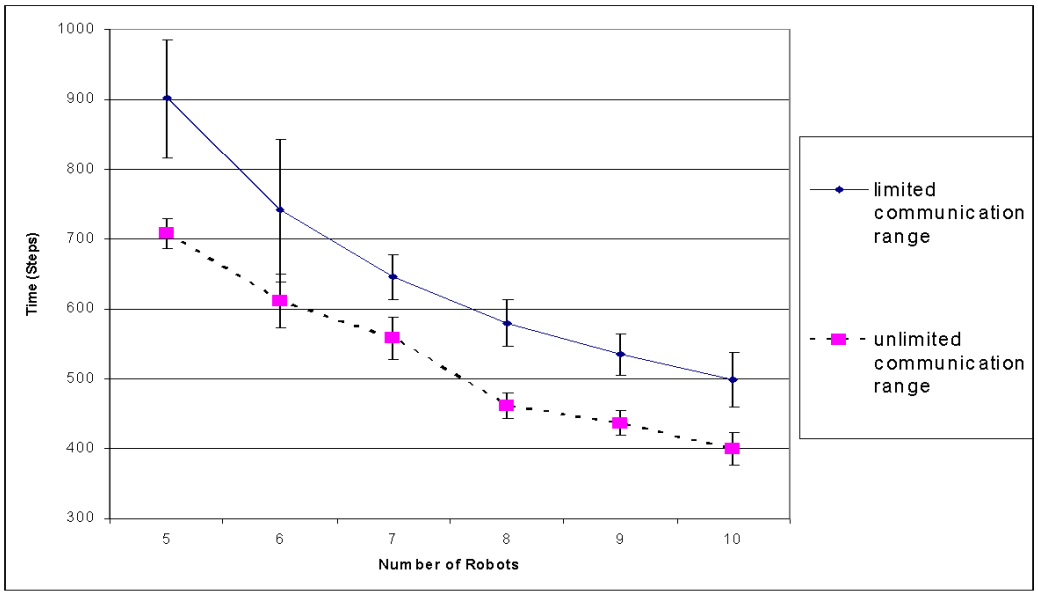


FIGURE 6.14 – *Performance de l'algorithme d'exploration avec et sans limite de portée de communication [Le et al., 2009a]*

placé les robots à proximité les uns aux autres au début de chaque mission de manière à former un réseau. Ensuite, nous avons mesuré le nombre de pas de simulation avant que la première partition du réseau ait lieu. Nous avons fait des mesures pour des groupes de 5 et 10 robots, avec une portée de communication variant de 7 à 14 unités de distance. Chaque configuration est simulée 20 fois. Les valeurs médianes de ces durées sont indiquées sur le diagramme de la figure 6.15.

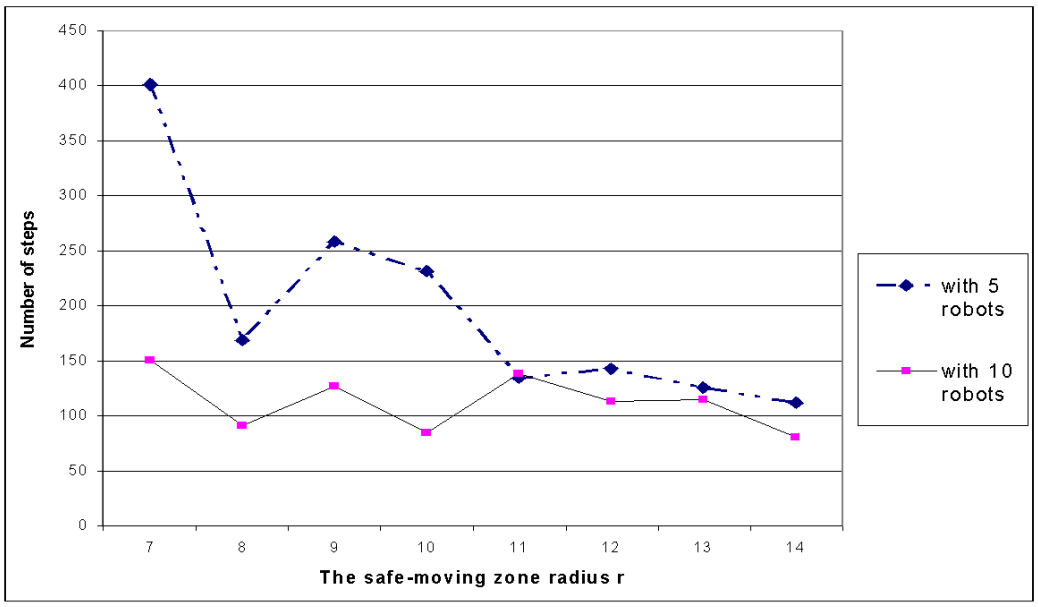


FIGURE 6.15 – *Durée de cohésion du réseau avant la première partition [Le et al., 2009a]*

Nous pouvons en conclure, d'après ce schéma que le paramètre de portée de communication a plus d'impact sur les réseaux de petite taille.

6.4 Allocation de rôles pour des organisations statiques

Dans cette section, nous présentons les résultats de différentes simulations que nous avons pu réaliser afin de valider notre proposition présentée dans la section 5.1 (page 79) du chapitre 5, puis nous en analysons le fonctionnement.

6.4.1 Résultats de simulation

Avec le simulateur Scène, nous avons réalisé une série de 1000 simulations avec 1000 robots, répartis aléatoirement sur un terrain de 1300 cellules $\times$ 900 cellules et avec une portée radio simulée de 100 cellules. Les robots sont placés initialement au hasard sur le terrain. L'infrastructure de simulation offre aux robots un service leur permettant de détecter leurs voisins. Au début de chaque simulation, nous choisissons au hasard un robot et nous l'initialisons comme le premier membre de l'organisation. Le robot devient alors le gestionnaire de l'organisation et dispose donc de la description de l'organisation.

Table 6.3 – *Nombre de pas pour allouer les rôles d'une organisation statique*

Nombre de pas	10	11	12	13	14	15	16	17	18
Pourcentage de simulations (%)	3,3	17	11,1	18	16,3	15,3	12,2	6,5	0,3

Nombre de pas : Nous considérons qu'un pas (un tic) dans la simulation correspond à l'intégration dans l'organisation de tous les voisins des robots faisant initialement partie de ladite organisation. Comme le montre la table 6.3, le nombre minimum de pas pour déployer une organisation est de 10 (dans 3,3% des simulations). Le nombre maximum est de 18 pas, mais pour 0,3% des simulations seulement. Dans la majorité des cas, les simulations ont nécessité entre 11 et 17 pas.

Changement de rôles : Les robots ne se déplacent pas car nous ne les avons doté d'aucune stratégie d'exploration. Cependant, à chaque pas de simulation, de nouveaux membres rejoignent l'organisation et provoquent des changements de rôles des anciens membres. Autrement dit, une réorganisation a implicitement lieu à chaque pas. Ces changements sont analogues à ce qui se passe dans le cas de déplacements ou de pannes des robots. Le graphique de la figure 6.16 montre la relation entre le nombre de nouveaux membres et le nombre correspondant à de changements dus à l'arrivée de ces membres au sein de l'organisation. En moyenne, sur cet exemple avec 1000 simulations, chaque robot change de rôle entre 2 et 3 fois.

Nombre de messages émis : Le nombre de messages émis est raisonnable. En effet, les 1000 robots émettent en cumulé entre 15000 et 20000 messages. La figure 6.16 montre que la courbe du nombre de changements de rôles et celle du nombre de messages émis ont des allures proches. Cela peut être expliqué par le fait que chaque robot émet le message `OrgDescRequest`, avant de rejoindre l'organisation.

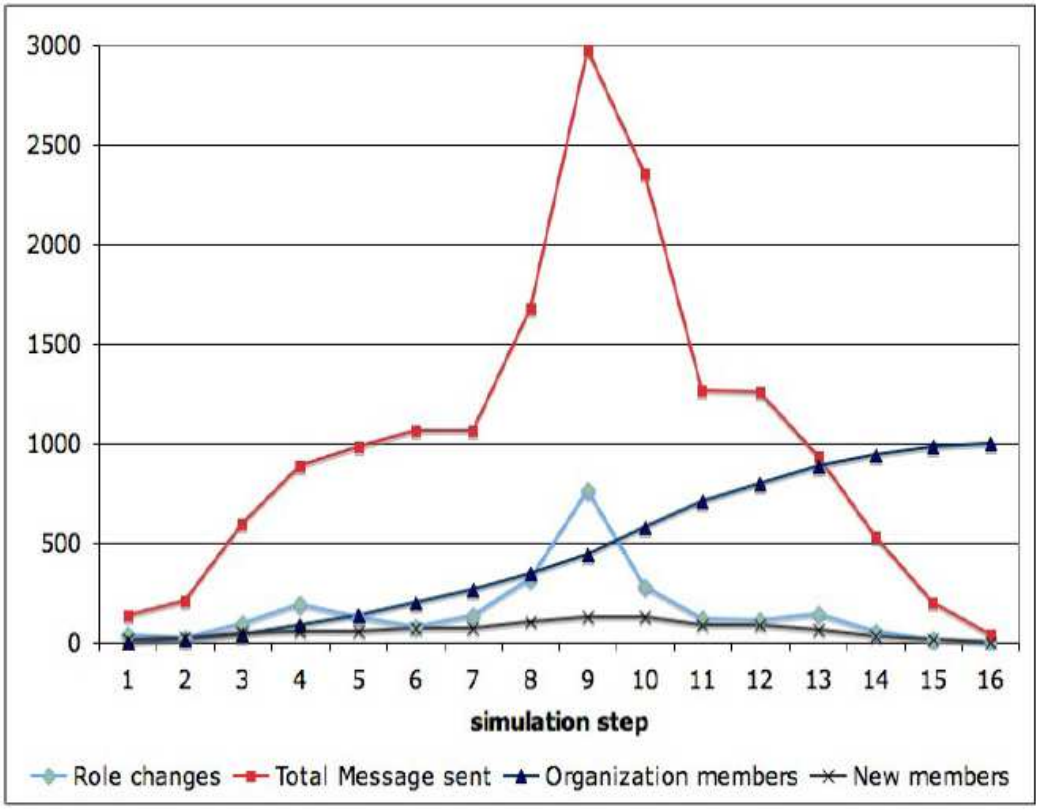


FIGURE 6.16 – *Nombre de membres, nouveaux membres, changements de rôles et de messages envoyés à chaque pas de simulation*

6.4.2 Optimisation de la diffusion de la description d'organisation

La propagation de la description de l'organisation que nous avons présentée plus haut est basée sur une logique de requête (*pull*). Un robot qui ne fait pas partie d'aucune organisation cherche à en rejoindre une, en diffusant le message `OrgDescRequest`. Nous pouvons aisément constater avec l'aide des simulations que cette stratégie inonde le réseau avec des messages inutiles quand les robots faisant partie de l'organisation sont trop éloignés des robots émetteurs.

Une stratégie alternative consiste à inverser le processus (approche *push*) de manière à ce que les membres de l'organisation diffusent la description de l'organisation à leurs voisins libres (qui ne font partie d'aucune organisation). Ce choix réduit le nombre de messages émis en moyenne d'un facteur 6 comparé à la stratégie précédente de requête systématique de la description de l'organisation.

6.4.3 Impact de la densité de robots

La densité des robots a un impact significatif sur le processus de formation de l'organisation. Nous avons simulé la modification de la densité des robots en faisant varier la portée de leur interface radio pour des robots placés aux mêmes positions sur le terrain. À chaque fois, nous avons utilisé le même robot comme point de départ pour la diffusion de la description de l'organisation. Nous avons fait 1000 simulations pour chacune des 5 portées radio suivantes : 70, 170, 270, 400 et 500 cellules (soit 5000 simulations au total toutes portées confondues).

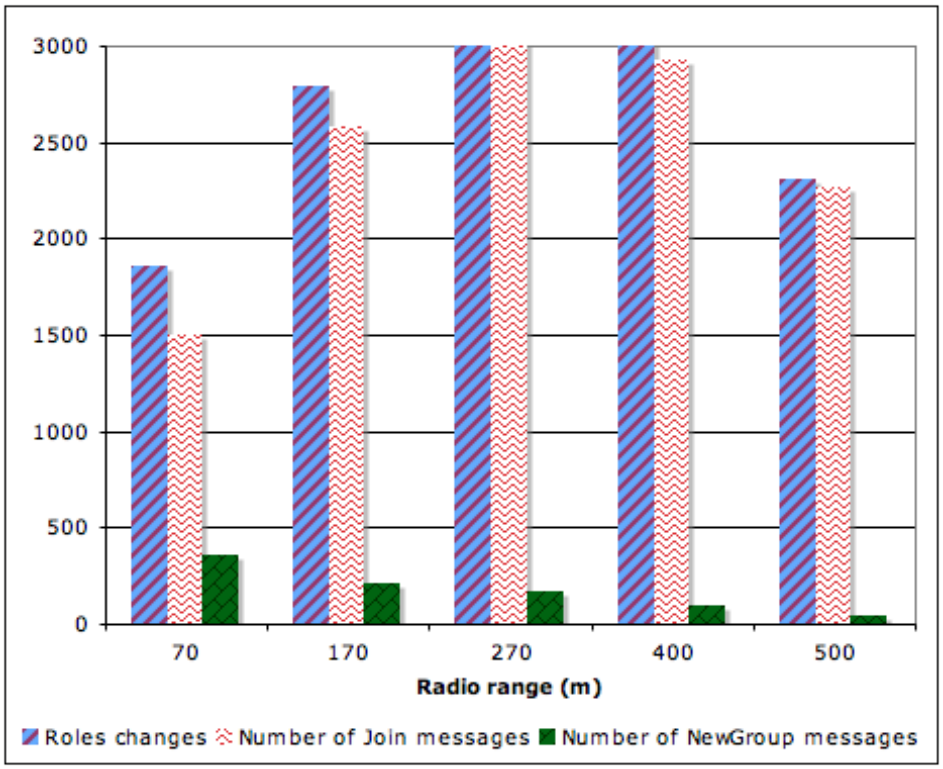


FIGURE 6.17 – *Impact de la portée radio sur le nombre de messages émis et les changements de rôles*

Chaque simulation a fait intervenir 1000 robots sur un terrain de 1300 cellules × 900 . La stratégie de diffusion de l'organisation optimisée telle que décrite dans la section 6.4.2 a été utilisée. La figure 6.17 montre l'impact de la variation des portées radio et donc de la variation de la densité.

Messages émis : Les nombres de messages envoyés de type `OrgDescRequest` et `OrgDescResponse` sont similaires dans tous les cas, puisque chaque robot émet exactement une fois chacun de ces messages. En revanche, le nombre de messages `NewGroup` et `Join` varient d'un cas à l'autre.

Changement de rôle : Le nombre de messages émis et les changements de rôles sont proches. Le pic de changements correspond à une portée radio moyenne (270 cellules).

Nombre de pas : Plus la zone de couverture radio – et donc la densité – est grande et plus le nombre de pas de simulation est petit. Ainsi, pour une portée de 70 cellules, 20 pas de simulation sont nécessaires pour diffuser l'organisation à tous les robots. En revanche, seulement 2 pas sont nécessaires pour une portée de 500 cellules. En effet, comme le nombre de robots est fixe, l'augmentation de la densité se traduit par une augmentation du nombre de voisins pour le robot disposant initialement de l'organisation. Ils peuvent donc rejoindre l'organisation plus vite.

Nous pouvons conclure qu'une grande portée radio peut significativement améliorer le processus de déploiement d'une organisation.

6.5 Une mission robotique de type USAR

Cette section développe un scénario complet permettant ainsi de rassembler les différents éléments présentés dans la thèse dans un contexte de sauvetage robotique en milieu urbain (application décrite dans l'introduction de la thèse).

Dans une mission robotique USAR [6], de nouvelles tâches imprévues peuvent apparaître lors de l'exécution comme dans le scénario suivant. Après une catastrophe naturelle, une équipe de robots est déployée sur le terrain pour explorer et pour construire la carte de la zone sinistrée. En explorant le terrain, les robots peuvent détecter des victimes. Les robots les plus proches doivent alors s'organiser automatiquement en coalitions pour aider la victime.

Cette expérimentation traite d'un scénario un peu plus élaboré pour montrer que le mécanisme dynamique de formation des coalitions opère de manière satisfaisante. Pour cela, nous fixons le nombre de tâches et leurs positions dans le terrain. La portée des interfaces Wi-Fi est fixe, le nombre de robots peut varier ainsi que leurs capacités.

6.5.1 Implémentation de l'algorithme de formation des coalitions

Pour implémenter les algorithmes proposés dans la section 5.2.3 (page 87 du chapitre 5), nous avons besoin d'un certain nombre de notions supplémentaires.

Specification de rôle

Une mission est décomposée en coalitions. Une coalition est une ensemble de ces rôles, avec les relations qu'ils entretiennent les uns avec les autres. Une spécification de rôle contient les informations suivantes :

- *RoleID* : l'identificateur unique du rôle, non seulement au niveau de la coalition, mais aussi au niveau du système entier.
- *TaskType* : indique le type de tâches auxquelles le robot peut contribuer en prenant ce rôle.
- *Concurrents* : la liste des rôles concurrents que le robot peut prendre en même temps avec ce rôle. Cet ensemble sera utilisé dans le cas où l'algorithme de formation de coalitions avec pré-filtrage est employé.
- *Utility* : la définition de la fonction d'utilité du rôle. Le robot calcule son aptitude en fonction de son état actuel au moyen de cette fonction. Cette valeur est ensuite utilisée pour déterminer son enchère pour obtenir le rôle identifié avec *RoleID*.
- *RoleImplementation* : est une référence au module logiciel qui pilotera le robot avec les règles imposés par le rôle.

Les spécifications des rôles sont dès le départ stockés sur les robots. En effet, chaque robot doit disposer des spécifications des rôles pour lesquels il est éligible.

6. Urban Search And Rescue

Spécification de coalition

Une spécification de coalition représente la spécification formelle (c.f. section 5.2.2, page 86) de la coalition correspondant à une tâche prédéfinie. La spécification de coalition est aussi stockée sur robot qui va prendre le rôle de commissaire-priseur dans le processus de formation de la coalition.

Notez que dans cette phase de mise en œuvre, ni la spécification de rôle, ni la spécification de coalition ne contiennent les conditions pour qu'un robot puisse jouer un rôle. L'idée est que, si une spécification d'un rôle quelconque est stockée dans un robot, alors le robot est « théoriquement » éligible au rôle, et au contraire, les spécifications de rôles incompatibles avec le robot ne sont pas stockées sur le robot. Lorsqu'il y a une tâche exigeant une coalition avec ce rôle, le robot va calculer leur utilité pour le rôle en prenant en compte les conditions présentes afin de participer à l'enchère selon l'algorithme présenté dans la section 5.2.3 (page 87).

Rôles extra-fonctionnels

Comme indiqué précédemment, les robots dans le système sont totalement autonomes. Afin d'assurer ce contrôle totalement décentralisé, nous employons des rôles *extra-fonctionnels*.

- *Task Announcer (TA)*. Un robot qui joue ce rôle déclenche et coordonne alors la formation d'une coalition (voir la section 5.2.3, page 87) lorsqu'il détecte une tâche. Le *TA* émet un message *New Task Announcement (NTA)* et le diffuse à d'autres robots pour appeler à la participation à la nouvelle coalition. Ce message contient des informations liées à la coalition comme l'emplacement de la tâche, le délai d'attente, etc. Le plus important est que ce message doit mentionner le type de tâche pour que les robots qui le reçoivent puissent déterminer s'ils participent au processus de formation de la nouvelle coalition ou non (cf. les champs *TaskType* dans la spécification de rôle ci-dessus).
- *Task Monitor (TM)*. Une fois que la coalition a été formée, le rôle de *TM* est de faire un suivi dans la réalisation des tâches et le rapporter au niveau supérieur si nécessaire. Il peut aussi coordonner, synchroniser les actions des membres de la coalition. De plus, en cas d'échec de robot, *TM* se charge de recruter un robot qui prendra en charge le rôle devenu disponible. Notons que le rôle *TM* fait partie de chaque coalition. Aussi, son allocation est semblable à l'allocation de n'importe quel autre rôle.

Utilisation des rôles extra-fonctionnels

En ce qui concerne la question de quand et comment ces rôles extra-fonctionnels sont attribués, nous ne précisons pas ici une méthode spécifique car une stratégie optimale pourra varier en fonction des applications. Par exemple dans une mission USAR, un robot qui, détecte une tâche urgente, peut prendre temporairement le rôle *TA* pour lancer le processus de formation d'une coalition. Cela diffère des applications où les tâches sont fournies par un opérateur humain et qui vont alors nécessiter de désigner un robot pour ce rôle. Dans le cas d'une

solution générique pour le cas des robots homogènes, un robot peut jouer le TA quand il trouve un tâche, puis il passe à TM dans la coalition.

Considérons la situation où il y a deux robots qui détectent la même tâche, il se peut qu'il y ait un « conflit » dans leur annonce de la nouvelle tâche. Ces deux robots doivent être dans le même (sous-)réseau. Cette hypothèse est assurée d'une part par notre travail sur le maintien de la connectivité, et d'autre part, par le fait que les deux robots sont assez proches l'un de l'autre. En effet, la zone de perception des tâches est généralement beaucoup moins large que la zone de communication. Ainsi, deux robots qui détectent la même tâche doivent être assez proches l'un de l'autre. Nous supposons que les informations de références (type de tâche, la localisation, la situation de tâche, etc.) détectés par les robots peuvent être déterminée de manière précise (i.e sans incertitudes). Avec cette hypothèse, nous pouvons nous attendre à ce que les conflits soient détectés et résolus par négociation entre les TA concurrents.

Un autre problème possible est celui de la participation d'un même robot à différentes coalitions avec des rôles concurrents. Examinons l'exemple suivant. Nous avons une équipe de 3 robots : R_1, R_2, et R_3 et deux tâches « cachées » T_1, et T_2 à découvrir.

Supposons que la structure de coalition possible pour ces deux tâches soient $C_1 = \{R_1, R_3\}$ et $C_2 = \{R_2, R_3\}$ pour la tâche T_1, et T_2 respectivement. Puis, supposons que la tâche T_1 est détectée d'abord par R_1. Ce robot diffuse alors un message NTA aux autres robots. Parce que le robot R_3 est admissible pour cette tâche, il place une enchère sur un rôle dans C_1. Et en attendant la confirmation de R_1 pour cette proposition, R_3 reçoit une autre offre de R_2 qui vient de détecter T_2. Le robot R_3 envoie une enchère pour intervenir dans C_2. Supposons que R_3 reçoit en même temps les confirmations de R_1 et de R_2, pour rejoindre respectivement les coalitions C_1 et C_2. R_1 et R_2 attendent que le robot R_3 participe à leurs coalitions respectives. Mais, comme R_3 ne peut assumer ces deux rôles simultanément, il décide de rejoindre la coalition C_2 gérée par R_2. Que doit faire le robot R_1 ? Doit-il attendre R_3 et suspendre la tâche T_1 ? Pour traiter une telle situation, nous exigeons que si un robot se voit attribué un rôle, il doit confirmer qu'il accepte de l'endosser. Si un TA ne reçoit pas de confirmation au bout d'une période de temps prédéfinie, il suppose que ce robot n'est plus disponible pour ce rôle et ouvre à nouveau les enchères pour le rôle en question.

6.5.2 Tâches dans le scénario USAR et décompositions en rôles

Dans notre scénario, nous considérons les quatre tâches suivantes placées uniformément sur le terrain au début de la mission :

- 1 tâche qui consiste à fournir un masque de secours à une victime lorsqu'elle est localisée. La tâche est alors complétée. Cette tâche a besoin seulement d'un seul robot fournisseur de masque (rôle Mask Provider).
- 1 tâche d'extraction d'une victime qui est bloquée dans une zone sinistrée. Il s'agit de déplacer les obstacles empêchant la victime de sortir de cet endroit.
- 2 tâches de déplacement d'une victime gravement blessée et donc incapable de se déplacer seule. Les robots participant à cette tâche doivent pousser les obstacles et déplacer la victime à un endroit sécurisé.

Un robot qui prend un rôle doit être capable de réaliser les comportements exigés par le rôle. Ces comportements sont donnés dans la table 6.4.

TABLE 6.4 – *Les comportements des rôles dans notre scénario USAR*

Rôle	Comportements/charges du rôle
Mask Provider	fournir un masque à une victime.
Path Clearer	déplacer les obstacles qui empêchent d'extraire une victime et déplacer les obstacles quand les autres robots déplacent une victime vers un endroit sûr.
Pilot	diriger les autres robots dans le groupe.
Pusher	transporter une victime.

À partir de l'analyse des comportements/charges de chaque rôle, nous pouvons déterminer les capacités physiques requises d'un robot pour jouer le rôle. Ces pré-requis doivent être interprétés en termes de rôles pré-installés sur le robot avant la mission (voir la section 5.2.2, page 86). Pour notre exemple, nous considérons les contraintes concurrentielles indiquées dans la table 6.5.

TABLE 6.5 – *Contraintes Concurrentielles*

Rôle	Les rôles qu'un robot peut jouer simultanément
r_1	r_2, r_3, r_4
r_2	r_1, r_3
r_3	r_1, r_2
r_4	r_1

6.5.3 Traitement des tâches dans les simulations

Nous n'allons pas détailler ici de manière précise ces tâches, puisque seules les tâches d'exploration et de formation de coalitions nous intéressent. Au moyen du simulateur SCÈNE, nous avons abstrait le traitement des tâches par la spécification des divers paramètres tels que : les rôles exigés pour certaines tâches, la période de temps nécessaire pour résoudre la tâche avec une coalition.

TABLE 6.6 – *Décomposition des tâches en rôles dans le scénario USAR*

Tâches (nombre)	Durée	Mask Provider (r_1)	Path Clearer (r_2)	Pilot (r_3)	Pusher (r_4)
Fournir un masque de secours (1)	2	1	0	0	0
Extraire une victime (1)	4	0	0	1	1
Déplacer une victime blessée (2)	8	0	1	1	2

Au début de la mission, les robots qui sont connectés via un réseau ad hoc, sont déployés sur le terrain et commencent l'exploration en utilisant l'algorithme d'exploration présenté dans la section 6.3 (et également dans [Le et al., 2009a]). Ainsi, les robots construisent la carte et localisent les victimes, qui sont initialement « cachées » aux robots. Lors de la détection d'une victime, les robots

doivent d'abord déterminer quel type d'assistance (une tâche) la victime pourrait avoir besoin afin de lancer la formation d'une nouvelle coalition. Afin que les quatre tâches soient résolues (en plus la tâche d'exploration par défaut), chacune d'entre elles a une période de temps (la colonne durée dans le tableau 6.6) mesurée en tics de la simulation pour que la tâche puisse être achevée à partir du moment où la coalition est formée. La mission est terminée lorsque le terrain est complètement exploré et que toutes les tâches sont traitées.

Afin de faciliter l'identification des victimes, nous implémentons les tâches sous forme d'agents. Autrement dit, les tâches sont *agentifiées* dans le simulateur. Ces agents immobiles peuvent fournir les informations sur leur état aux robots qui sont assez proches d'eux. Cette agentification a pour objectif d'abstraire le traitement, ainsi que l'utilisation de capteurs sur les robots.

Enfin, lorsqu'un robot détecte une tâche, il endosse le rôle *TA* (Task Announcer, voir la section 6.5.1, page 116) pour initialiser le processus de formation d'une nouvelle coalition. Dans le cas où il n'y pas assez de robots pour former la coalition après un période de temps, les robots participant à l'enchère continuent sur la tâche d'exploration et reviennent à la victime plus tard.

6.5.4 Résultat des simulations avec différentes configurations de robots

Dans les expérimentations, nous utilisons trois types de robots avec les ensembles de rôle qualifiés illustrés sur la table 6.7.

TABLE 6.7 – *Robots et rôles qualifiés*

Robots \ rôles	r_1	r_2	r_3	r_4
R_1	x			x
R_2	x		x	
R_3		x		x

TABLE 6.8 – *Résultats des expérimentations*

Ensemble de Robots (type, nombre)	Temps pour terminer toutes les tâches
$\{(R_1,2);(R_2,1);(R_3,1)\}$	465
$\{(R_1,2);(R_2,2);(R_3,2)\}$	250
$\{(R_1,4);(R_2,4);(R_3,4)\}$	131

Nous avons réalisé les expérimentations avec trois ensembles de robots différents, en faisant varier le nombre de robots de chaque type (écrit sous la forme *(type de robot, nombre)* sur le tableau 6.8). Chaque configuration est exécutée pour 20 fois. Le temps moyen nécessaire pour achever la mission pour chaque cas est indiqué sur la deuxième colonne du tableau. Le premier ensemble correspond à la configuration minimale de robots afin que toutes les tâches soient résolues. Il est utilisé comme référence pour évaluer les deux autres configurations. Les résultats des expériences montrent que la formation de la coalition fonctionne de manière satisfaisante. On observe notamment que l'ajout de robots réduit le temps nécessaire à l'accomplissement de la mission.

Conclusion du chapitre

Dans ce chapitre, nous avons présenté le simulateur Scène que nous avons réalisé pour valider les concepts introduits dans les chapitres 4 et 5. Ainsi, nous avons décrit différentes simulations que nous avons réalisées pour valider les contributions de cette thèse, tant au niveau maintien de la connectivité qu'au niveau formation de coalitions et allocations de rôles. Ces résultats confirment notre étude théorique et valident ainsi nos propositions.

Troisième partie

Conclusion

Bilan et perspectives

7

Sommaire

L'objectif principal de cette thèse a été de proposer une solution au problème de coopération dans les systèmes multi-robots. Un tel système doit se configurer de manière autonome en équipes et sous-équipes en fonction des objectifs actuels de la mission, des robots disponibles et de leurs ressources.

Nous nous plaçons dans le cadre applicatif du sauvetage robotisé en milieu urbain. Les robots se trouvent alors situés dans des environnements hautement dynamiques où les tâches peuvent apparaître de manière imprévue. De plus, la configuration du système peut fluctuer à cause des pannes de robots par exemple ou du fait de l'arrivée de nouveaux robots. Les solutions à notre problème de coopération doivent donc satisfaire les exigences suivantes :

- *Auto-adaptation et Réactivité* [1] : cela fait référence à la capacité du système de répondre et à s'adapter lui-même aux changements externes (de l'environnement, comme lors de la détection d'une victime) ou propre aux systèmes multi-robots (quand on ajoute ou enlève des robots ou lors de pannes matérielles). En général, le système doit répondre rapidement aux nouveaux événements qui apparaissent pendant la mission.
- *Extensibilité* : nous pouvons ajouter à la volée des nouveaux robots dans le système. Ces nouveaux robots doivent automatiquement participer au système.
- *Robustesse* : dans le cas d'une panne complète ou partielle d'un robot, le système doit être capable de reprendre pour continuer à partir de l'état actuel.
- *Hétérogénéité* : les robots dans le système peuvent varier en terme de capacité physique.

1. responsiveness en anglais

- *Calcul distribué* : les calculs sont répartis entre les robots.
- *Passage à l'échelle* : la solution peut être utilisée avec un grand nombre de robots sans détérioration notable des performances.

Nous avons identifié les deux sous-problèmes suivants du problème de coopération dans les systèmes multi-robots :

1. Maintien de la connectivité dans le réseau constitué par les robots.
2. Formation dynamique de coalitions de robots en fonction des tâches à réaliser (aider les victimes détectées par exemple),

Le premier sous-problème est un prérequis pour le second. En effet, lors de la phase de négociation pour former une coalition, les robots engagés doivent être connectés les uns avec les autres.

Dans ce chapitre final, nous récapitulons les solutions que nous avons proposées en réponse à chacun de ces deux sous-problèmes. Nous en résumons les avantages, ainsi que les limites. Enfin, nous discutons quelques perspectives de ce travail.

7.1 Bilan

Afin de répondre aux deux sous-problèmes évoqués ci-dessus, nous avons apporté deux contributions dans cette thèse : le maintien distribué de la connectivité et la formation dynamique de coalitions croisées.

7.1.1 Contribution sur le maintien de la connectivité

Concernant le maintien de la connectivité, notre solution est basée sur le concept original de la «sensibilité de la connectivité». La sensibilité permet aux robots dans le réseau d'identifier les voisins avec lesquels ils doivent garder le lien. Ces derniers sont appelés robots d'accès ou accesseurs. Cette information est déterminée une fois que le Robot de Référence a été choisi. Afin de garder la connexion globale du réseau, tous les robots essaient de maintenir au moins un chemin d'accès entre eux et le Robot de Référence en utilisant des robots intermédiaires. Les chemins d'accès sont mis à jour dynamiquement pour être toujours en cohérence avec la configuration actuelle du réseau.

La solution proposée répond bien aux besoins d'une application de sauvetage. L'analyse théorique de la solution ainsi que les simulations réalisées montrent que les algorithmes génèrent un faible nombre de messages.

La solution proposée peut être décomposée en deux étapes principales : (i) acquérir les informations nécessaires pour construire sur chaque robot une table avec les informations locales de connectivité, et (ii) exploiter ces connaissances afin de préserver la connectivité réseau lors de l'exécution d'autres tâches. L'avantage de cette séparation, c'est que la première étape peut être considérée comme une abstraction indépendante des applications, et peut être incorporée dans différentes stratégies de maintien de la connectivité. Dans [Le et al., 2009a], nous avons illustré l'utilisation de la connectivité de sensibilisation pour l'exploration multi-robots d'une zone inconnue et la détection efficace de nœuds critiques dans un réseau sans fil. Nous suggérons que la sensibilité de la connectivité réseau devrait être fournie aux robots comme un service réseau de base comme le routage dans les MANET par exemple.

7.1.2 Contribution sur l'allocation des rôles

Une fois que la connectivité du réseau est assurée, nous pouvons nous intéresser à une stratégie permettant aux robots de se configurer automatiquement en prenant en compte différents facteurs. Le point de vue qui considère un robot comme une entité dans une structure sociale nous a amené à proposer des solutions à base du concept de *rôle*, pour structurer le système multi-robots comme une organisation. Cela permet aux concepteurs de rester à un haut niveau d'abstraction lors de la conception de la solution – le processus pourrait se faire d'ailleurs en ignorant la nature des robots réels.

Dans notre approche, nous considérons qu'il existe deux type d'organisation : statique et émergente. Une organisation statique est celle dont la structure est définie avant son déploiement sur les robots. Une organisation émergente n'a pas de structure a priori. La structure d'une telle organisation émerge pendant l'exécution d'une mission et s'adapte aux changements de l'environnement

et du système mutli-robots. Chacune de ces deux organisations a ses propres avantages et inconvénients. L'organisation statique peut être utilisée au début de la mission pour structurer le système. Pendant l'exécution, notre protocole de déploiement permet de changer l'organisation entière de manière efficace. Cependant, la structure reste inchangée. Cette limite de l'organisation statique est levée par l'organisation émergente.

Nous avons développé dans cette thèse des solutions pour l'allocation de rôles à des robots hétérogènes capables d'endosser plusieurs rôles simultanément. Pour l'organisation statique, nous avons proposé un protocole d'attribution de rôles qui prend en compte les descriptions des groupes qui forment l'organisation et les contraintes inter- et intra-groupe. Pour l'organisation émergente, nous avons proposé un algorithme de formation de coalitions croisées [2] au fur et à mesure de la découverte des tâches. Ces propositions ont été validées par les analyses formelles, ainsi qu'au travers de simulations.

Enfin, rappellons que nous sommes parmi les premiers à formaliser le problème de formation des coalitions croisées avec l'hypothèse que les robots peuvent jouer plus d'un rôle en même temps, dans une ou plusieurs coalitions. Nous avons ensuite proposé un algorithme efficace pour un problème appartenant à la classe $\mathcal{NP}$-difficile.

7.1.3 Limitations

Il existe plusieurs limitations à nos contributions, que nous allons maintenant détailler.

Maintien de la connectivité et mobilité

Lors de la sélection simple d'un déplacement à effectuer (voir la section 4.6.1, page 73), les robots n'emploient que des connaissances liées à la sensibilité comme contraintes locales pour maintenir la connectivité avec ses accesseurs. Le Robot de Référence ne dépend d'aucun autre robot pour planifier un déplacement. Par conséquente, les autres robots du réseau ont tendance à suivre le Robot de Référence, et cela n'est pas toujours ce qu'on désire dans certaines applications.

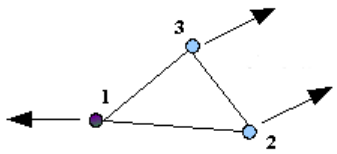


FIGURE 7.1 – *Déplacement s'effectuant de manière concurrente, qui provoque la partition du réseau.*

En outre, considérons la situation montrée dans la figure 7.1 où le robot 2 et le robot 3 sont des robots d'accès l'un pour l'autre. Puisque les calculs sont réalisés en parallèle sur chaque robot, le robot 1 peut choisir de maintenir la connexion avec le robot 2. Simultanément, le robot 2 choisit également le robot 1 pour le

2. un robot peut participer à plusieurs coalitions.

maintien de la connectivité. Le résulat est que simultanément, ils se déplacent hors de la portée de communication du Robot de Référence (le robot 1). Une telle situation montre que la contrainte sur la distance ne suffit pas pour assurer une connectivité permanente. Donc, pour les applications où une connexion permanente est requise, une coordination plus élaborée doit être définie afin de faire face à ce genre de situations.

Nous présenterons une direction à suivre pour pallier ces limitations dans la section 7.2.1.

Allocation de rôle et qualité de la solution : une question ouverte

Concernant le problème d'allocation de rôles aux robots, nous nous sommes placés dans le cas le plus difficile car il s'agit d'un problème $\mathcal{NP}$-difficile. Dans ce contexte, notre algorithme cherche une solution possible de manière gloutonne en utilisant un protocole de type réseau de contrat. Par conséquente, la qualité de la solution globale trouvée n'a pas pu être théoriquement estimée de manière exacte.

[Chaimowicz et al., 2004] traite un cas particulier de ce problème général où chaque robot endosse exactement un et un seul rôle (voir la description dans section 3.5.3, page 49). Dans ce cas, la qualité de la solution est *au moins 3-compétitive* comme estimée dans (c.f. la table 3.1, page 47 du chapitre 3) [Gerkey and Matarić, 2004a].

Hormis la qualité de la solution, la complexité de la communication est exactement la même entre l'algorithme que nous avons proposé et l'algorithme d'affectation des rôles en-ligne de [Chaimowicz et al., 2004]. La complexité de la communication pour former une coalition lorsqu'une tâche est trouvée est de $O(n)$.

7.2 Perspectives

Cette section présente quelques pistes à suivre pour remédier aux limitations que nous avons indiquées dans la section 7.1.3 (page 126), ainsi que des extensions possibles de notre travail.

7.2.1 Utilisation de CSP pour les déplacements avec maintien de la connectivité du réseau

L'exigence de surmonter des limitations dans le choix simple d'un déplacement d'un robot, sous la réserve du maintien de la connectivité (voir la section 7.1.3) nous amène à proposer l'utilisation d'une solution basée sur la «satisfaction de contraintes distribuées» (DisCSP [3] [Yokoo, 2001]).En modélisant le problème de la sélection d'un déplacement comme un CSP [4], nous pouvons éliminer la dépendance asymétrique en autorisant les robots à effectuer une négociation entre eux, pour arriver à un consensus dans la recherche d'une solution. De cette manière, le Robot de Référence choisit sa sélection d'un déplacement planifié en

3. Distributed Constraint Satisfaction Problem
4. Constraint Satisfaction Problem

prenant en compte la contrainte de la connectivité avec ses voisins. Le Robot de Référence lui-même doit satisfaire la contrainte mutuelle du maintien de la connectivité globale.

De même, avec la sensibilité à la connectivité, la formulation du problème du maintien de la connectivité comme un DisCSP est un choix naturel. La projection de notre problème en un DisCSP peut s'effectuer en correspondante parfaite : un robot est un agent, le réseau de contraintes est un sous-réseau du réseau de communication physique. Autrement dit, les agents qui sont voisins dans le terme d'un DisCSP sont aussi des voisins dans le contexte de la sensibilité à la connectivité ou celui d'un réseau MANET [5]. Enfin et surtout, le problème DisCSP pour le maintien de la connectivité est sous-contraint, cela signifie que la solution peut être trouvée très rapidement avec un faible nombre de messages émis. Cette conclusion est partiellement prouvée par des expérimentations que nous avons menées dans [Doniec et al., 2009].

7.2.2 Coopération à base de rôle

Il y a plusieurs aspects qui pourraient être approfondis dans notre approche concernant la coopération entre les robots à base de rôles :

- *évaluer la qualité de solution*. Comme nous l'avons identifié dans la section 7.1.3 précédente, l'évaluation exacte de la qualité des solutions trouvées par l'algorithme de formation des coalitions reste encore une question ouverte. Nous croyons que ce problème est un challenge et mérite une étude approfondie. Une piste serait de s'appuyer sur les critères définis par [Bonnet and Tessier, 2009].
- *Calculer dynamiquement l'ensemble des contraintes concurrentielles*. Dans la formalisation du problème de formation des coalitions (section 5.2.2, page 86), nous avons fait l'hypothèse que l'ensemble des contraintes concurrentielles est statique. Bien que cette hypothèse n'empêche pas la formalisation du problème et donc que sa solution soit générique, l'hypothèse devrait être relâchée dans certains cas. Avec ce relâchement, il s'agit de déterminer l'ensemble de contraintes concurrentielles dynamiquement en prenant en compte les conditions actuelles de l'environnement et du robot lui-même.
- *Implémenter et expérimentation avec de vrais robots*. Les solutions proposées ne sont testées qu'en simulation. Bien que les résultats obtenus soient intéressants, il existe quand même une distance significative entre la réalité et les simulations. Il est sûr que les solutions proposées devraient être ajustées pour opérer de manière satisfaisante sur de véritables robots.

5. Mobile Ad hoc NETwork.

Bibliographie

[Abolhasan et al., 2003] Abolhasan, M., Wysocki, T., and Dutkiewicz, E. (2003). A review of routing protocols for mobile ad hoc networks. *Ad Hoc Networks 2*. (Cité pages 9 et 18.)

[Agüero et al., 2006] Agüero, C. E., Matellán, V., Canas, J. M., and Gómez, V. (2006). Switch ! dynamic roles exchange among cooperative robots. In *Proceedings of the 2nd International Workshop on Multi-Agent Robotic Systems - MARS 2006. INSTICC*, pages 99–105. Press. (Cité page 38.)

[Ahmadi and Stone, 2006a] Ahmadi, M. and Stone, P. (2006a). A distributed biconnectivity check. In *proceeding of the 8th International Symposium on Distributed Autonomous Robotic Systems (DARS'06)*. (Cité pages 72 et 107.)

[Ahmadi and Stone, 2006b] Ahmadi, M. and Stone, P. (2006b). Keeping in touch : Maintaining biconnected structure by homogeneous robots. In *proceeding of the 21st National Conference on Artificial Intelligence (AAAI)*. (Cité pages 24 et 107.)

[Almeida et al., 2008] Almeida, H. L., Carramate, L., Wang, F. Z., and Sun, Y. (2008). Connectivity-aware motion control among autonomous mobile units. In *International Symposium on Industrial Embedded Systems, 2008. SIES 2008*. (Cité page 24.)

[Arkin, 1987] Arkin, R. C. (1987). Motor schema based navigation for a mobile robot : An approach to programming by behavior. In *Proceeding of IEEE International Conference on Robotics and Automation, 1987.*, pages 264–271. (Cité page 50.)

[Arrichiello, 2006] Arrichiello, F. (2006). *Coordination Control of Multiple Mobile Robots*. PhD thesis, Università degli Studi di Cassino. (Cité pages 4 et 34.)

[Awerbuch and Peleg, 1992] Awerbuch, B. and Peleg, D. (1992). Routing with polynomial communication-space trade-off. *SIAM Journal on Discrete Mathematics*, 5(2). (Cité page 20.)

[Báez-Barranco et al., 2007] Báez-Barranco, J.-A., Stratulat, T., and Ferber, J. (2007). A unified model for physical and social environments. In *Environments for Multi-Agent Systems III*, volume 4389, pages 41–50. LNCS. (Cité page 39.)

[Basagni, 1999] Basagni, S. (1999). Distributed clustering for ad hoc networks. In *Proceeding of the 1999 International Symposium on Parallel Architectures, Algorithms and Networks (I-SPAN'99)*. (Cité page 81.)

[Bin and Choon, 2006] Bin, C. B. and Choon, C. M. (2006). Resource management in heterogeneous wireless networks with overlapping coverage. In *Proceedings of COMSWARE'06*. (Cité pages xiii et 18.)

[Bonnet and Tessier, 2009] Bonnet, G. and Tessier, C. (2009). Evaluation d'un système multirobot. cas d'une constellation de satellites. *Revue d'Intelligence Artificielle*, 23(5–6) :565–592. (Cité page 128.)

[Boucher et al., 2009] Boucher, A., Canal, R., Chu, T.-Q., Drogoul, A., Gaudou, B., Le, V. T., Moraru, V., Nguyen, N. V., Vu, Q. A. N., Taillandier, P., Sempé, F., and Stinckwich, S. (2009). The AROUND project : Adapting robotic disaster response to developing countries. In *Proceedings of 2009 IEEE International Workshop on Safety, Security, and Rescue Robotics*, page to appear. IEEE Computer Society. (Cité pages xiii, 4 et 5.)

[Brunet et al., 2008] Brunet, L., Choi, H.-L., and How, J. P. (2008). Consensus-based auction approaches for decentralized task assignment. In *Proceeding of AIAA Guidance, Navigation and Control Conference and Exhibit*. (Cité page 35.)

[Burgard et al., 2005] Burgard, W., Moors, M., Stachniss, C., and Schneider, F. (2005). Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3) :376–386. (Cité pages 17, 57 et 108.)

[Campbell and Wu, 2007] Campbell, A. and Wu, A. S. (2007). Task and role allocation within multi-agent and robotics research. Technical Report CS-TR-07-05, University of Central Florida, EECS. (Cité page 37.)

[Candea et al., 2001] Candea, C., Hu, H., Iocchi, L., Nardi, D., and Piaggio, M. (2001). Coordination in multi-agent RoboCup teams. *Robotics and Autonomous Systems*, 36 :67–86. (Cité pages 38 et 44.)

[Certo et al., 2007] Certo, J., Lau, N., and Reis, L. P. (2007). A generic multi-robot coordination strategic layer. In *Proceedings of the 1st international conference on Robot communication and coordination*. (Cité pages xiii, 38, 39, 43 et 44.)

[Chaimowicz et al., 2004] Chaimowicz, L., Kumar, V., and Campos, M. F. M. (2004). A paradigm for dynamic coordination of multiple robots. *Autonomous Robots*, 17(1). (Cité pages 38, 46, 47, 49 et 127.)

[Choi et al., 2009] Choi, H.-L., Brunet, L., and How, J. P. (2009). Consensus-based decentralized auctions for robust task allocation. *IEEE Transactions on Robotics*, 25(4) :912—926. (Cité page 35.)

[Clark and Colbourn, 1990] Clark, B. N. and Colbourn, C. J. (1990). Unit disk graphs. *Discrete Mathematics*, 86(1–3) :165–177. (Cité page 24.)

[Colman and Han, 2007] Colman, A. and Han, J. (2007). Roles, players and adaptable organizations. *Applied Ontology, IOS Press*, 2 :105—126. (Cité page 40.)

[Cormen et al., 1990] Cormen, T., Leiserson, C., and Rivest, R. L. (1990). *Introduction to Algorithms*. Cambridge : The MIT Press. (Cité page 28.)

[Das et al., 2009] Das, S., Liu, H., Nayak, A., and Stojmenovic, I. (2009). A localized algorithms for bi-connectivity of connected mobile robots. *Telecommunication Systems*, 40(3–4) :129–140. (Cité pages 24 et 72.)

[Dastani et al., 2003] Dastani, M., Dignum, V., and Dignum, F. (2003). Role-assignment in open agent societies. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 489–496. (Cité page 40.)

[Dechter, 2003] Dechter, R. (2003). *Constraint Processing*. Morgan Kaufmann. (Cité page 91.)

[Dias, 2004] Dias, M. B. (2004). *TraderBots : A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments*. PhD thesis, Robotics Institute, Carnegie Mellon University. (Cité pages 3, 35, 37 et 47.)

[Dias et al., 2006] Dias, M. B., Zlot, R., Kalra, N., and Stentz, A. (2006). Market-based multirobot coordination : A survey and analysis. *Proceedings of the IEEE*, 94(7) :1257–127. (Cité page 35.)

[Diosdado, 2006] Diosdado, J. M. V. (2006). *Behaviour Based Simulated Low-Cost Multi-Robot Exploration*. PhD thesis, Institute of Perception, Action and Behaviour School of Informatics University of Edinburgh. (Cité pages 17, 26, 28 et 75.)

[Doniec et al., 2009] Doniec, A., Bouraqadi, N., Defoort, M., Le, V. T., and Stinckwich, S. (2009). Distributed constraint reasoning applied to multi-robot exploration. In *Proceedings of 21st IEEE International Conference on Tools with Artificial Intellige (ICTAI)*, pages 159–166. IEEE Computer Society. (Cité page 128.)

[Duque-Anton et al., 2000] Duque-Anton, M., Bruyaux, F., and Semal, P. (2000). Measuring the survivability of a network : connectivity and rest-connectivity. *Transaction of Telecommunications*, 11(2) :149–159. (Cité page 72.)

[Durrant-Whyte and Bailey, 2006] Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping :. *IEEE Robotics & Automation Magazine*, pages 99–117. (Cité page 24.)

[Emery et al., 2002] Emery, R., Sikorski, K., and Balch, T. (2002). Protocols for collaboration, coordination and dynamic role assignment in a robot team. In *Proceeding of the 2002 IEEE International Conference on Robotics and Automation*, pages 3008–3015. (Cité page 38.)

[Ferber et al., 2005] Ferber, J., Michel, F., and Baez, J. (2005). AGRE : Integrating environments with organization. In *Environments for Multi-Agent Systems*, volume 3374 of *LNCS*. Springer. Proposal for extending the model AGR to AGRE. (Cité page 39.)

[Gale, 1960] Gale, D. (1960). *The Theory of Linear Economic Models*. mcgraw-Hill Book Company, Inc. (Cité page 46.)

[Gennaro and Jadbabaie, 2006] Gennaro, M. C. D. and Jadbabaie, A. (2006). Decentralized control of connectivity for multi-agent systems. In *Proceedings of the 45th IEEE Conference on Decision and Control*. (Cité pages 26 et 75.)

[Gerkey and Matarić, 2002] Gerkey, B. P. and Matarić, M. J. (2002). Sold ! : Auction methods for multirobot coordiantion. *IEEE Transactions on Robotics And Automation*, 18(5) :758–768. (Cité pages 4, 34, 35, 47, 48, 49 et 88.)

[Gerkey and Matarić, 2004a] Gerkey, B. P. and Matarić, M. J. (2004a). A formal analysis and taxonomy of task allocation in multi-robot systems. *Robotics Research*, 23(9). (Cité pages xv, 9, 34, 36, 44, 45, 47, 86 et 127.)

[Gerkey and Matarić, 2004b] Gerkey, B. P. and Matarić, M. J. (2004b). *RoboCup 2003 : Robot Soccer World Cup VII*, volume 3020/2004, chapter On role allocation in robocup, pages 43–53. Springer Berlin / Heidelberg. (Cité pages 11, 31, 36, 37, 45, 49 et 86.)

[Godsil and Royle, 2001] Godsil, C. D. and Royle, G. F. (2001). *Algebraic Graph Theory*, volume 207 of Graduate Texts in Mathematics. Springer. (Cité pages 22 et 23.)

[Goldberg and Robson, 1983] Goldberg, A. and Robson, D. (1983). *Smalltalk 80*, volume 1 – The Language and its implementation. Addison-Wesley. (Cité page 99.)

[Graham and Hell, 1985] Graham, R. L. and Hell, P. (1985). On the history of the minimum spanning tree problem. *IEEE Annals of the History of Computing*, 7(1) :43–57. (Cité page 22.)

[Gutknecht and Ferber, 2000] Gutknecht, O. and Ferber, J. (2000). Madkit : a generic multi-agent platform (short paper). In *Autonomous Agents (AGENTS 2000)*, pages 39–56. (Cité pages xiii, 40 et 41.)

[Gutnecht, 2001] Gutnecht, O. (2001). *Proposition d'un modèle organisationnel générique de systèmes multi-agents : Examen de ses conséquences formelles, implémentatoires et méthodologiques*. PhD thesis, Université Montplellier II. (Cité pages xiii, 38, 39 et 40.)

[Hannoun et al., 2000] Hannoun, M., Boissier, O., Sichman, J. S., and Sayettat, C. (2000). Moise : An organizational model for multi-agent systems. In *Proceedings of the International Joint Conference, 7th Ibero-American Conference on AI, 15th Brazilian Symposium on AI (IBERAMIA/SBIA'2000), Atibaia, SP, Brazil, November 2000*, pages 152—161. Springer, Berlin. (Cité pages 38 et 39.)

[Horling and Lesser, 2005] Horling, B. and Lesser, V. (2005). A survey of multi-agent organizational paradigms. *The Knowledge Engineering Review*, 19(4) :281–316. (Cité page 9.)

[Howard et al., 2006] Howard, A., Parker, L. E., and Sukhatme, G. S. (2006). Experiments with a large heterogeneous mobile robot team : Exploration, mapping, deployment, and detection. *International Journal of Robotics Research*, 25 :431—447. (Cité page 38.)

[Hsieh et al., 2008] Hsieh, M. A., Cowley, A., Kumar, R. V., and Taylor, C. J. (2008). Maintaining network connectivity and performance in robot teams. *Journal of Field Robotics*, 25 :111–131. (Cité page 24.)

[Hübner et al., 2005] Hübner, J., Sichman, J., and Boissier, O. (2005). S-moise+ : A middleware for developing organized multi-agent systems. In Boissier, O., Dignum, V., Matson, E., and Sichman, J., editors, *International Workshop on Organizations in Multi-Agent Systems : From Organizations to Organization Oriented Programming (OOOP 2005)*, pages 107—120. (Cité pages 38 et 39.)

[Johnson et al., 2001] Johnson, D. B., Maltz, D. A., and Broch, J. (2001). *DSR : The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks*, chapter 5, pages 139–172. Addison-Wesley Professional. (Cité page 70.)

[Jorgic et al., 2004] Jorgic, M., Stojmenovic, I., Hauspie, M., and Simplot-Ryl, D. (2004). Localized algorithms for detection of critical nodes and links for connectivity in ad hoc networks. In *Proceeding of The Third Annual Mediterranean Ad Hoc Networking Workshop*. (Cité page 72.)

[Kalra et al., 2004] Kalra, N., Stentz, T., and Ferguson, D. (2004). Hoplites : A market framework for complex tight coordination in multi-agent teams. Technical Report CMU-RI-TR-04-41, Robotics Institute Carnegie Mellon University. (Cité page 35.)

[Katz and Dao., 2001] Katz, S. A. S. K. R. and Dao., S. (2001). Distributed power control in ad-hoc wireless networks. In *Proceedings of PIMRC*. (Cité page 20.)

[Kitano and Tadokoro, 2001] Kitano, H. and Tadokoro, S. (2001). Robocup rescue : A grand challenge for multiagent and intelligent systems. *AI Magazine*, 22(1) :39–52. (Cité pages 42 et 44.)

[Le et al., 2009a] Le, V. T., Bouraqadi, N., Moraru, V., Stinckwich, S., and Doniec, A. (2009a). Making networked robot connectivity-aware. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3502–3507. (Cité pages xiv, 73, 105, 109, 111, 118 et 125.)

[Le et al., 2009b] Le, V. T., Stinckwich, S., Bouraqadi, N., and Doniec, A. (2009b). Dynamic role assignment for large-scale multi-agent robotic systems. In *Studies in Computational Intelligence book series*, page to appear. Springer. (Cité pages xiv, 80 et 83.)

[Legras and Tessier, 2003] Legras, F. and Tessier, C. (2003). Lotto : group formation by overhearing in large teams. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems (AAMAS'03)*, pages 425–432. (Cité page 95.)

[Legras and Tessier, 2004] Legras, F. and Tessier, C. (2004). Lotto : group formation by overhearing in large teams. In *Advances in agent communication - Lecture Notes in Artificial Intelligence 2922*, pages 254–270. Springer. (Cité page 95.)

[Liu and Wu, 2001] Liu, J. and Wu, J. (2001). *Multiagent Robotic Systems*. CRC Press LLC. (Cité page 3.)

[Lyons and Arbib, 1989] Lyons, D. M. and Arbib, M. A. (1989). A formal model of computation for sensory-based robotics. *IEEE Transactions on Robotics And Automation*, 5(3) :280–293. (Cité page 50.)

[Mansour, 2007] Mansour, S. (2007). *Un modèle de gestion distribuée de groupes ouverts et dynamiques d'agents mobiles*. PhD thesis, Université de Pau et des Pays de l'Adour. (Cité page 39.)

[McMillen and Veloso, 2007] McMillen, C. and Veloso, M. (2007). Distributed, play-based role assignment for robot teams in dynamic environments. In Alami, R., Chatila, R., and Asama, H., editors, *Proceedings of Distributed Autonomous Robotic Systems 6*, pages 145–154. Springer. (Cité pages 38 et 44.)

[Micheal et al., 2009] Micheal, N., Zavlanos, M. M., Kumar, V., and Pappas, G. J. (2009). Maintaining connectivity in mobile robot networks. In *Experimental Robotics, Springer Tracts in Advanced Robotics Book Series*, volume 54, pages 117–126. Springer Berlin/Heidelberg. (Cité pages 26, 27, 28 et 75.)

[Modi et al., 2006] Modi, P. J., Shen, W.-M., Tambe, M., and Yokoo, M. (2006). ADOPT : Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence Journal (AIJ)*, 161 :149–180. (Cité pages 11 et 87.)

[Nguyen et al., 2004a] Nguyen, H. G., Farrington, N., and Pezeshkian, N. (2004a). Maintaining communication link for tactical ground robots. AUVSI Unmanned Systems North America. (Cité page 75.)

[Nguyen et al., 2004b] Nguyen, H. G., Pezeshkian, N., Gupta, A., and Farrington, N. (2004b). Maintaining communication link for a robot operating in a hazardous environment. *ANS 10th Int. Conf. on Robotics and Remote Systems for Hazardous Environments*. (Cité page 75.)

[Nierstrasz et al., 2009] Nierstrasz, O., Ducasse, S., Pollet, D., and Black, A. P. (2009). *Squeak by Example*. Square Bracket Associates. (Cité page 99.)

[Notarstefano et al., 2006] Notarstefano, G., Savla, K., Bullo, F., and Jadbabaie, A. (2006). Maintaining limited-range connectivity among second-order agents. In *Proceedings of the 2006 American Control Conference*. (Cité pages 26, 27 et 29.)

[Odell et al., 2003a] Odell, J., Parunak, H. V. D., Brueckner, S., and Sauter, J. (2003a). Changing roles : Dynamic role assignment. *Journal of Object Technology*, 2(5) :77–86. (Cité page 40.)

[Odell et al., 2003b] Odell, J. J., Parunak, H. V. D., and Fleischer, M. (2003b). The role of roles in designing effective agent organizations. In *Software Engineering for Large-Scale Multi-Agent Systems*, volume 2603, pages 27–38. Springer-Verlag. (Cité pages 37 et 39.)

[Ottens and Faltings, 2009] Ottens, B. and Faltings, B. (2009). Coordinating agent plans through distributed constraint optimization. In *Proceedings of the IJCAI'09 Distributed Constraint Reasoning Workshop (DCR'09)*. (Cité pages 11 et 88.)

[Parker and Tang, 2006] Parker, B. L. E. and Tang, F. (2006). Building multirobot coalitions through automated task solution synthesis. *Proceedings of the IEEE*, 94(7) :1289–1305. (Cité pages xiii, 9, 46, 50 et 51.)

[Parker, 1999] Parker, L. E. (1999). Cooperative robotics for multi-target observation. *Intelligent Automation and Soft Computing*, 5(1) :5–19. (Cité pages 48 et 88.)

[Parker, 2003] Parker, L. E. (2003). *Multi-Robot Systems From Swarms to Intelligent Automata : Volume II*, volume II, chapter The effect of heterogeneity in teams of 100+ mobile robots, pages 205–215. Kluwer 2003. (Cité pages 3, 4, 38, 39, 44 et 77.)

[Parker, 2008a] Parker, L. E. (2008a). Distributed intelligence : Overview of the field and its application in multi-robot systems. *Journal of Physical Agents*, 2(1) :5–14. (Cité page 38.)

[Parker, 2008b] Parker, L. E. (2008b). *Springer Handbook of Robotics*, chapter Multiple Mobile Robot Systems, pages 921–941. Springer. (Cité pages 34, 45 et 47.)

[Parker et al., 2005] Parker, L. E., Chandra, M., and Tang, F. (2005). Enabling autonomous sensor-sharing for tightly-coupled cooperative tasks. In *Multi-Robot Systems From Swarms to Intelligent Automata*, volume Volume III, pages 119–230. Kluwer. (Cité page 50.)

[Parkes and Ungar, 2000] Parkes, D. C. and Ungar, L. H. (2000). Iterative combinatorial auctions : Theory and practice. In *Proceedings of 17th National Conference on Artificial Intelligence (AAAI-00)*, pages 74–81. (Cité page 35.)

[Perkins, 2001] Perkins, C. (2001). *Ad Hoc Networking*. Addison-Wesley. (Cité pages 9, 17 et 62.)

[Petcu and Faltings, 2005] Petcu, A. and Faltings, B. (2005). DPOP : A scalable method for multiagent constraint optimization. In *IJCAI 05*, pages 266–271. (Cité page 11.)

[Petcu and Faltings, 2006] Petcu, A. and Faltings, B. (2006). O-DPOP : An algorithm for open/distributed constraint optimization. In *Proceedings of the National Conference on Artificial Intelligence, AAAI-06*, pages 703–708. (Cité page 11.)

[Rajaraman, 2002] Rajaraman, R. (2002). Topology control and routing in ad hoc networks : a survey. *ACM SIGACT News*, 33(2) :60–73. (Cité page 20.)

[Reis and Lau, 2001] Reis, L. P. and Lau, N. (2001). FC portugal team description : Robocup 2000 simulation league champion. In *RoboCup 2000 : Robot Soccer World Cup IV, Lecture Notes in Computer Science*, volume 2019, pages 29–40. Springer Berlin / Heidelberg. (Cité page 44.)

[Rooker and Birk, 2005] Rooker, M. N. and Birk, A. (2005). Combining exploration and ad-hoc networking in robocup rescue. School of Engineering and Science, International University Bremen, Germany. (Cité page 17.)

[Rooker and Birk, 2007] Rooker, M. N. and Birk, A. (2007). Multi-robot exploration under the constraints of wireless networking. *Control Engineering Practice*, 15(4) :435–445. (Cité pages 17, 26, 27, 57 et 108.)

[Royer and Toh., 1999] Royer, E. M. and Toh., C.-K. (1999). A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*. (Cité pages 9 et 18.)

[Schuresko, 2009] Schuresko, M. D. (2009). *Controlling Global Network Connectivity of Robot Swarms with Local Interactions*. PhD thesis, University of California. (Cité pages 24, 26, 27, 28, 29, 57 et 75.)

[Schuresko and Cortés, 2009] Schuresko, M. D. and Cortés, J. (2009). Distributed motion constraints for algebraic connectivity of robotic network. *Journal of Intelligent and Robotic Systems*, 56(1–2) :99–126. (Cité pages 24, 28, 29, 57 et 75.)

[Shehory and Kraus, 1998] Shehory, O. and Kraus, S. (1998). Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1–2) :165–200. (Cité pages 9, 47 et 50.)

[Sheng et al., 2006a] Sheng, M., Li, J., and Shi, Y. (2006a). Critical nodes detection in mobile ad hoc network. In *20th International Conference on Advanced Information Networking and Applications (AINA'06)*, volume 2, pages 336–340. (Cité page 72.)

[Sheng et al., 2006b] Sheng, W., Yang, Q., Tan, J., and Xi, N. (2006b). Distributed multi-robot coordination in area exploration. *Robotics and Autonomous Systems*, 54 :945–955. (Cité pages 26, 27, 57 et 108.)

[Siciliano and Khatib, 2008] Siciliano, B. and Khatib, O., editors (2008). *Springer Handbook of Robotics*. Springer-Verlag Berlin Heidelberg. (Cité page 3.)

[Simmons et al., 2000] Simmons, R., Apfelbaum, D., Burgard, W., Fox, D., Moors, M., Thrun, S., and Younes, H. (2000). Coordination for multi-robot exploration and mapping. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*. (Cité page 17.)

[Smith, 1980] Smith, R. G. (1980). The contract net protocol : High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12) :1104–1113. (Cité pages 11, 35 et 88.)

[Spanos and Murray, 2004] Spanos, D. P. and Murray, R. M. (2004). Robust connectivity of networked vehicles. In *Proceeding of the 43rd IEEE Conference on Decision and Control*. (Cité page 24.)

[Srivastava and Spong, 2008] Srivastava, K. and Spong, M. W. (2008). Multi-agent coordination under connectivity constraints. In *Proceedings of the 2008 American Control Conferences*, pages 2648–2653. (Cité page 26.)

[Stone and Veloso, 1999] Stone, P. and Veloso, M. (1999). Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110(2) :241–273. (Cité pages xiii, 38, 39, 40, 41 et 44.)

[Stump et al., 2008] Stump, E., Jadbabaie, A., and Kumar, V. (2008). Connectivity management in mobile robot teams. In *Proceedings of the 2008 IEEE International*

Conference on Robotics and Automation, pages 1525–1530. (Cité pages 24, 26 et 27.)

[Tadokoro, 2005] Tadokoro, S. (2005). Special project on development of advanced robots for disaster response (ddt project). In *Proceedings of the IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO'05)*. (Cité page 5.)

[Tadokoro et al., 2000] Tadokoro, S., Takamori, T., Osuka, K., and Tsurutani, S. (2000). Investigation report of the rescue problem at Hanshi-Awaji earthquake in kobe. In *In Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1880–1885. (Cité page 4.)

[Takayama et al., 2008] Takayama, L., Ju, W., and Nass, C. (2008). Beyond dirty, dangerous and dull : What everyday people think robots should do. In *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, pages 25–32. (Cité page 1.)

[Tang and Parker, 2005a] Tang, F. and Parker, L. E. (2005a). ASyMTRe : Automated synthesis of multi-robot task solutions through software reconfiguration. In *Proceedings of the IEEE International Conference on Robotics and Automation, 2005. ICRA 2005.*, pages 1501–1508. (Cité pages 46 et 50.)

[Tang and Parker, 2005b] Tang, F. and Parker, L. E. (2005b). Coalescent multi-robot teaming through ASyMTRe : A formal analysis. In *Proceedings of 12th International Conference on Advanced Robotics*, pages 817–824. (Cité page 46.)

[Tang and Parker, 2007] Tang, F. and Parker, L. E. (2007). A complete methodology for generating multi-robot task solutions using ASyMTRe-D and market-based task allocation. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3351–3358. (Cité pages 46, 48 et 51.)

[Thomas and Williams, 2005] Thomas, G. and Williams, A. B. (2005). Roles in the context of multiagent task relationships. In *Proceeding of AAAI Fall Symposium "Roles, an Interdisciplinary Perspective : Ontologies, Programming Languages, and Multiagent Systems,"*, TR FS-05-08, ISBN 978-1-57735-254-9. (Cité page 37.)

[Ulam and Arkin, 2004] Ulam, P. and Arkin, R. C. (2004). When good comms go bad : Communications recovery for multi-robot teams. In *Proceeding of the 2004 IEEE International Conference on Robotics and Automation*. (Cité page 75.)

[Vazquez and Malcolm, 2004] Vazquez, J. and Malcolm, C. (2004). Distributed multirobot exploration maintaining a mobile network. In *Proceedings of 2nd International IEEE Conference Intelligent Systems, 2004.*, volume 3, pages 113–118. (Cité pages 26 et 27.)

[Vig and Adams, 2006] Vig, L. and Adams, J. A. (2006). Multi-robot coalition formation. *IEEE Transactions on Robotics*, 22(4) :637–649. (Cité page 46.)

[Vig and Adams, 2007] Vig, L. and Adams, J. A. (2007). Coalition formation : From software agents to robots. *Journal of Intelligent Robot System*, 50(1) :85–118. (Cité pages xiii, 9, 46, 48 et 50.)

[Werger and Matarić, 2001] Werger, B. B. and Matarić, M. J. (2001). *Broadcast of Local Eligibility for Multi-Target Observation*, pages 347–356. Springer-Verlag New York. (Cité page 48.)

[Xu et al., 2001] Xu, Y., Heidemann, J., and Estrin, D. (2001). Geography-informed energy conservation for ad hoc routing. In *Proceedings of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'01)*, Rome, Italy. (Cité page 20.)

[Yamauchi, 1997] Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. In *proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*. (Cité pages 27 et 107.)

[Yamauchi, 1998] Yamauchi, B. (1998). Frontier-based exploration using multiple robots. In *Proceeding of the second International Conference on Autonomous Agents (Agent'98)*. (Cité pages 107 et 108.)

[Yokoo, 2001] Yokoo, M. (2001). *Distributed Constraint Satisfaction : Foundations of Cooperation in Multi-agent Systems*. Springer. (Cité page 127.)

[Zavlanos and Pappas, 2005] Zavlanos, M. M. and Pappas, G. J. (2005). Controlling connectivity of dynamic graphs. In *Proceedings of 44th IEEE Conference on Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05.*, pages 6388–6393. (Cité pages 26 et 27.)

[Zavlanos and Pappas, 2007] Zavlanos, M. M. and Pappas, G. J. (2007). Potential fields for maintaining connectivity of mobile networks. *IEEE Transactions on Robotics*, 23(4) :812–816. (Cité pages 24, 26, 27, 28 et 75.)

[Zavlanos and Pappas, 2008] Zavlanos, M. M. and Pappas, G. J. (2008). Distributed connectivity control of mobile networks. *IEEE Transactions on Robotics*, 24(6) :1416–1428. (Cité pages 24, 26, 27, 28 et 75.)