Une approche de patrouille multi-agents pour la détection d'évènements

Elie Tagne-Fute

SPIM

Thèse de Doctorat

école doctorale **sciences pour l'ingénieur et microtechniques**

UNIVERSITÉ DE TECHNOLOGIE BELFORT-MONTBÉLIARD

Une approche de patrouille multi-agents pour la détection d'évènements

Elie TAGNE FUTE

SPIM

Thèse de Doctorat

école doctorale **sciences pour l'ingénieur et microtechniques**

UNIVERSITÉ DE TECHNOLOGIE BELFORT-MONTBÉLIARD

N° 1 9 7

THÈSE présentée par

Elie TAGNE FUTE

pour obtenir le

Grade de Docteur de
l'Université de Technologie de Belfort-Montbéliard

Spécialité : **Informatique**

Une approche de patrouille multi-agents pour la détection d'évènements

Soutenue le 5 mars 2013 devant le Jury :

Lhassane IDOUMGHAR	Rapporteur	Maître de Conférences HDR, Université de Haute Alsace
Krzysztof CETNAROWICZ	Rapporteur	Professeur, University of Science and Technology (Pologne)
René MANDIAU	Examinateur	Professeur, Université de Valenciennes et du Hainaut-Cambrésis
Albert DIPANDA	Examinateur	Professeur, Université de Bourgogne
Olivier SIMONIN	Examinateur	Maître de Conférences HDR, Université de Lorraine
Emmanuel TONYE	Directeur	Professeur, Université de Yaoundé 1
Abderrafiâa KOUKAM	Directeur	Professeur, UTBM
Fabrice LAURI	Co-Directeur	Maître de Conférences, UTBM

Dédicaces

Aux membres de ma jeune famille: Christ, Merveille, Divin et Victorine Piaptié

Remerciements

Je remercie très chaleureusement:

Tout d'abord, toute ma reconnaissance à l'Eternel Dieu Tout-Puissant sans qui rien n'aurait été possible.

Professeur René MANDIAU de l'Université de Valenciennes et du Hainaut-Cambrésis pour avoir accepté de présider le jury de soutenance.

Professeur Krzysztof CETNAROWICZ de l'University of Science and Technology (Pologne) et M. Lhassane IDOUMGHAR, Maître de Conférences HDR de l'Université de Haute Alsace pour avoir accepté de rapporter ce travail.

Professeur Albert DIPANDA de l'Université de Bourgogne et M. Olivier SIMONIN, Maître de Conférences HDR de l'Université de Lorraine pour avoir accepté d'examiner ce travail.

Professeur Emmanuel TONYE pour sa simplicité et la disponibilité dont il a fait preuve en acceptant de diriger cette thèse. Sa disponibilité, son dévouement, son dynamisme, ses conseils judicieux et sa rigueur au travail bien fait a constitué pour moi un modèle d'encouragement.

Professeur Abderrafiâa KOUKAM qui a accepté sans condition de diriger cette thèse. J'ai beaucoup apprécié les différents échanges enrichissants et ses multiples conseils. Je lui suis profondément reconnaissant pour son aide, sa disponibilité et sa patience.

M. Fabrice Lauri, Maître de Conférences à l'Université de Technologie de Belfort-Montbéliard qui a accepté de co-diriger cette thèse, pour les différents échanges enrichissants et aussi pour son aide, sa disponibilité et sa patience.

Le SCAC pour avoir financé cette thèse à travers leur programme de mobilité.

Tous mes enseignants des Universités de Dschang et Yaoundé I qui ont contribué énormément par leurs enseignements à l'édification de l'être que je suis devenu.

Pr Clémentin Tayou, Pr Laure Pauline Fotso, Pr Marcellin NKENLI-FACK pour les différents échanges enrichissants et leurs multiples conseils.

Mes parents Fute Pierre, Rose Mamno pour toute la peine qu'ils se sont donnés depuis mes tous premiers jours pour assurer mon éducation.

Me Felix Taptué pour tous ses multiples conseils et soutients multiformes.

Tous mes frères (Bertrand, Jean, Jean Pierre, ...), soeurs (Martine, Valerie, Pélagie, ...), cousins, cousines (Nadine, ...), oncles, tantes, Pascaline,

Flore(Lili), ... pour leurs multiples encouragements.

Mes amis et collaborateurs du laboratoire SeT de l'UTBM : Nicolas Gaud, Ariane Glatigny, Adnen, David Meignan, Stéphane Galland, Vincent Hilaire,... pour leurs soutients multiformes.

Mes amis : Mathurin Mboupda(Moy), Benoît Azanguezet, Michel Mfeze, David Kamgue G.(Yo'o), Alexis Zebaze, Paulin Melataguia, Janvier Fotsing, N. Tsopzé, Landry Ewoussoua, Pierre Tsafack, Mathurin Soh, Maurice Tchoupé, Anicet Kouomou, Guy Atenekeng, Djeumeni T., Guy Nguezet, Mose's F., Alain Bomgni, Anne Chana, A. Tsague, Célestin Lélé, David Dongo, Berge Tsanou, A. Takoudjou, P. Nankep, N. Talla, Bertrand Tamokwé, Isabelle(Bibi), Alain Fotué, ... j'en oublie certainement bien d'autres, pour leurs multiples soutients et encouragements.

A tous ceux qui ont contribué de près ou de loin à la réussite de ce travail, je leur dis MERCI!

Last but surely the first, ma très tendre épouse Victorine Piaptié(Dada) qui a su être là en prodiguant le conseil et le soutient moral qu'il fallait au moment opportun.

Table des matières

Table des figures

Résumé

Pouvoir lutter efficacement contre certains fléaux comme les incendies de forêt, les feux de brousse ou les catastrophes naturelles constitue un enjeu majeur dans plusieurs villes du monde. Avec l'avènement de la technologie de pointe représentée par les réseaux de capteurs, la détection de ces phénomènes devient plus aisée. En effet, des capteurs peuvent être déployés dans des zones difficiles d'accès et s'ils sont suffisamment nombreux pour couvrir la totalité de l'environnement à surveiller, une alerte peut être directement donnée par le capteur ayant détecté un certain type d'évènement (feu, secousse sismique...). Le centre de contrôle ayant reçu l'alerte peut ensuite décider d'intervenir sur la zone en cause.

Nos travaux se situent dans ce cadre de la détection de phénomènes par un réseau de capteurs, en supposant que l'environnement est connu et que les capteurs sont mobiles, sans fil et en nombre insuffisant pour couvrir la totalité de l'environnement à surveiller. Parler de surveillance par un nombre faible d'entités mobiles nécessite de parcourir régulièrement certaines zones critiques de l'environnement, ce qui peut s'apparenter à une tâche de patrouille.

Dans le cadre de cette thèse, nous nous sommes focalisés sur la détermination de stratégies de patrouille multi-capteurs appliquée à la détection d'évènements. Un problème similaire au nôtre est celui de la patrouille multi-agents dans un environnement connu. Ce problème consiste à faire visiter régulièrement les noeuds d'un graphe (représentant l'environnement) par des agents. Les capteurs peuvent être considérés comme des agents ayant des ressources limitées, en terme d'énergie en particulier. Le cadre de la patrouille multi-agents et les techniques proposées pour le résoudre ne peuvent pas être utilisés ici. Après avoir formulé mathématiquement le problème de la patrouille multi-capteurs appliquée à la détection d'évènements, nous proposons une technique de résolution approchée basée sur des colonies de fourmis. Des simulations ont été réalisées en considérant différents scenarii (topologies d'environnement, populations de capteurs, apparitions des événements) afin d'évaluer la pertinence de notre approche. Les résultats expérimentaux montrent que notre approche permet de déterminer des stratégies de patrouille satisfaisantes dans la majorité des scenarii.

Mots clés : Patrouille multi-capteurs, colonie de fourmis, détection d'évènements, réseaux de capteurs.

Chapitre 1

Introduction

1.1 Contexte général

Nous sommes au quotidien très souvent amenés à résoudre des problèmes liés à la surveillance d'environnements. Parmi ces problèmes, on peut citer la détection d'incendies de forêt, de feux de brousse, d'intrusions ou de catastrophes naturelles. L'objectif consiste à surveiller le plus régulièrement possible un environnement et de détecter le plus rapidement possible des évènements particuliers pouvant y survenir. Plus généralement, la surveillance et la détection d'évènements constituent un enjeu majeur dans le maintien de la sécurité d'un territoire. Une surveillance et une détection efficace demande l'utilisation d'un certain nombre de technologies récentes. Les réseaux de capteurs se prêtent particulièrement bien à des applications de remontées d'alarmes où la zone à surveiller peut être difficile d'accès. Ainsi, les réseaux de capteurs sont particulièrement adaptés pour résoudre des tâches de surveillance et de prévention. Parler de surveillance nécessite de parcourir régulièrement certaines régions critiques de l'environnement, ce qui peut s'apparenter à une tâche de patrouille. L'objectif visé ici est de minimiser les délais de passage entre les régions critiques connexes de l'environnement. Le nombre limité de capteurs impose de surveiller non pas la totalité de l'environnement, mais seulement certaines régions dites critiques, c'est-à-dire celles où un événement particulier a une forte probabilité d'apparaître.
L'utilisation des réseaux de capteurs comme agents patrouilleurs soulève des problématiques liées entre autres à leur déploiement. Les principales problématiques que nous retenons sont la minimisation de la consommation globale en énergie du réseau, ainsi que le respect des contraintes communicationnelles imposées par les ressources des capteurs. Les applications auxquelles sont dédiées les réseaux de capteurs imposent des exigences supplémentaires

de fiabilité et surtout d'économie d'énergie [Sam08].

1.2 Objectif de nos travaux

Nos travaux se situent à l'intersection des domaines de l'optimisation multi-objectif sous contraintes, des systèmes multi-agents et des réseaux de capteurs. Ils s'inscrivent dans le champ d'application des réseaux de capteurs pour les problèmes de surveillance d'environnement.

L'objectif principal de cette thèse peut être résumé ainsi :

> Proposer un algorithme capable de déterminer des stratégies de patrouille satisfaisantes afin de permettre à des capteurs mobiles sans fil de détecter le plus rapidement possible des événements dans des environnements connus. Nous supposerons que : (1) il existe un ou plusieurs capteurs particuliers dans le réseau de capteurs. Chacun de ces capteurs, dénommé *Sink*, est capable d'alerter directement le centre de contrôle lorsqu'un événement survient ; (2) une détection efficace d'un événement en particulier a déjà été intégrée dans les capteurs. Une stratégie de patrouille satisfaisante doit optimiser un certain nombre de critères et respecter des contraintes inhérentes aux réseaux de capteurs.

Dans une approche de patrouille par un réseau de capteurs mobiles, les *Sinks* et les autres capteurs doivent se rencontrer le plus souvent, afin de garantir des échanges de données efficaces. Dans notre cas, cela permettra aux Sinks d'envoyer des alertes au centre de contrôle dans le cas où un évènement serait perçu par un autre capteur pendant la patrouille.

Les avantages de disposer d'un Sink sont multiples. Un premier avantage est lié à la réduction du coût de mise en oeuvre et des ressources matérielles. Cela peut s'expliquer par le fait que tous les capteurs n'ont pas nécessairement les mêmes caractéristiques, permettant ainsi de réduire les coûts de mise en oeuvre et par là d'orienter les dépenses vers ce qui est nécessaire. On n'a pas nécessairement besoin par exemple d'équiper tous les capteurs des ressources pour l'aspect communicationnel avec le centre de contrôle car seul le Sink peut assurer cette fonction.

Un deuxième avantage est de disposer d'un nombre réduit de points d'entrée au réseau de capteurs de façon à faciliter l'ajout d'instructions supplémentaires au réseau. Un troisième avantage est aussi de disposer d'un nombre

réduit de points de sortie du réseau, ce qui permet de mieux contrôler les actions issues du groupe.
Disposer d'un Sink n'a pas que des avantages. Un inconvénient de disposer de capteurs Sinks est que la tâche de ces derniers est plus élaborée comparativement à celle des autres et consomme par conséquent plus d'énergie. Ils doivent disposer d'assez de ressources pour pouvoir répertorier et transmettre tous les évènements perçus par les autres capteurs. Un autre inconvénient est qu'un évènement perçu par un capteur n'est pas directement transmis au centre de contrôle et doit transiter par un Sink, à moins que l'évènement ne soit directement perçu par un Sink. Pour pallier cet inconvénient, les stratégies de patrouille doivent prendre en compte la fréquence des communications entre les Sinks et les capteurs.

La démarche que nous adoptons pour atteindre cet objectif est la suivante. Tout d'abord, nous formulons la patrouille multi-capteurs comme un problème d'optimisation multi-objectif. Ensuite, pour résoudre ce problème, nous proposons une approche algorithmique permettant de déterminer différentes stratégies de patrouille, en considérant des environnements dans lesquels les régions critiques sont connus. Enfin, nous procédons à la validation de notre approche, sur la base de l'analyse des résultats issus d'expériences de simulations.

1.2.1 Définition du problème de la patrouille multi-capteurs

Le problème classique de la patrouille multi-agents s'appuie sur un graphe et consiste généralement à déterminer une stratégie qui minimise la valeur de la pire oisiveté du graphe. L'oisiveté d'un graphe est basée sur la notion d'oisiveté d'un noeud du graphe, qui représente le temps durant lequel ce noeud est resté non visité. La pire oisiveté du graphe est le temps maximum durant lequel un des noeuds du graphe est resté non visité.

Dans le cas d'une patrouille classique, dénommée patrouille multi-agent, aucune contrainte n'est prise en compte pour déterminer les stratégies de patrouille qui seront suivies individuellement par les agents. Dans notre cas, la principale difficulté est d'intégrer les contraintes inhérentes aux réseaux de capteurs. Si l'on considère des agents patrouilleurs comme étant des capteurs mobiles sans fil, alors les principales contraintes sont d'une part d'ordre énergétiques, et d'autre part d'ordre communicationnelles. En effet, les capteurs disposent de ressources limitées pour percevoir leur environnement, faire des calculs et communiquer, et leurs communications sont limitées dans l'espace, dans le sens où un capteur ne peut communiquer qu'avec les capteurs proches de lui.

Le problème de la patrouille multi-capteurs peut se définir en tant que problème d'optimisation multi-objectif. Ce problème suppose l'existence d'un graphe. Chaque noeud du graphe représente une des régions critiques de l'environnement. Les arcs représentent les moyens de transport entre les noeuds. Résoudre un problème de patrouille multi-capteurs consiste à déterminer l'ensemble des stratégies de patrouilles individuelles de chaque capteur. Chaque stratégie individuelle est composée de la liste des noeuds de l'environnement qu'un capteur doit visiter. Cet ensemble de stratégies doit respecter certaines contraintes inhérentes aux capteurs et minimiser la pire oisiveté du graphe, la pire oisiveté communicationnelle ainsi que la consommation globale d'énergie du réseau de capteurs. L'oisiveté communicationnelle représente la durée pendant lequel un capteur n'a pas reçu d'informations de la part d'autre capteurs.

1.2.2 Approche de résolution du problème de patrouille appliquée à la détection d'évènements

Plusieurs approches ont été utilisées pour résoudre des problèmes multiobjectifs (*PMO*). L'approche la plus simple consiste à transformer le problème multiobjectif en un problème mono-objectif, c'est-à-dire qu'on ramène un problème avec plusieurs objectifs en un problème à un seul objectif. Une telle transformation implique une contrainte très forte et demanderait que les différents objectifs aient une même échelle de valeur. Ce qui n'est généralement pas le cas.
Une autre façon de résoudre les PMO est d'utiliser les approches non Pareto. Dans de tels cas, on est amené à traiter séparément les différents objectifs. Ainsi, dans de telles situations, on peut privilégier certains objectifs par rapport à d'autres.
La dernière stratégie est basée sur les approches Pareto. Ces approches utilisent directement la notion d'optimalité au sens de Pareto dans leur processus de recherche. Le processus de sélection des solutions générées est basé sur la notion de non-dominance et de solutions représentant des compromis.
Nous nous situons dans le contexte des approches Pareto. Nous avons développé une approche multi-colonies que nous avons baptisée *AMCPMC* (Approche Multi-Colonies pour les problèmes de Patrouille Multi-Capteurs). Comme son nom l'indique, notre stratégie de résolution est basée sur une approche à base de colonies de fourmis. Contrairement aux approches n'utilisant qu'une seule colonie de fourmis, nous considérons une compétition entre plusieurs colonies. Chaque colonie représente une solution au problème, c'est-

à-dire, dans notre cas, un ensemble de stratégies de patrouille individuelles. Cette compétition entre les colonies a un double intérêt. D'une part, elle favorise la diversité des solutions impliquée par la multiplicité des colonies, et d'autre part, elle permet d'accélerer la recherche vers une solution optimale.

1.2.3 Implémentation et simulations

L'approche *AMCPMC* a été implémentée et validée sur la base de résultats expérimentaux essentiellement. Plusieurs simulations, en faisant varier la topologie du graphe, le nombre de capteurs, leurs dispositions ainsi que la localisation et le timing des événements, ont permis d'apprécier la pertinence et l'efficacité de notre approche.

1.3 Plan de la thèse

Ce mémoire est organisé en cinq chapitres. Les différents chapitres qui le composent se résument ainsi :

Chapitre 2 : Le problème de la patrouille multi-agents et les réseaux de capteurs. Ce chapitre présente un état de l'art sur la problématique de la patrouille. Il décrit la formulation la plus employée du problème de la patrouille ainsi que les critères d'évaluation de ces stratégies de patrouille. Puisque les agents patrouilleurs sont des capteurs, nous faisons ensuite un état de l'art des réseaux de capteurs. Nous abordons enfin dans ce chapitre l'utilisation des réseaux de capteurs pour le problème de la patrouille, tout en mettant l'accent sur les contraintes et les difficultés liées à leur exploitation.

Chapitre 3 : Formulation du problème de patrouille appliquée à la détection d'événements en tant que problème d'optimisation multiobjectif. Le principal but de ce chapitre est de formuler le problème de la patrouille multi-capteurs sous la forme d'un problème d'optimisation multiobjectif où les agents patrouilleurs sont des capteurs. Puisque le problème est considéré sous un angle multiobjectif, nous présentons des concepts essentiels dans le domaine de l'optimisation multiobjectif, tout en mettant un accent sur les approches permettant de déterminer des solutions de compromis. Nous présentons également dans ce chapitre la notion d'optimalité au sens de Pareto et le concept de dominance, sans oublier les principales difficultés qui caractérisent les problèmes à objectifs multiples.

Chapitre 4 : Approches de résolution du problème de détection d'évènements. Dans ce chapitre, nous présentons un algorithme permettant de déterminer des stratégies de patrouille multi-capteurs en environnements connus. L'algorithme proposé, nommé **AMCPMC**, est basé sur un algorithme de colonie de fourmis multi-objectifs.

Chapitre 5 : Résultats expérimentaux. Nous présentons dans ce chapitre les résultats expérimentaux de différentes stratégies de patrouille appliquée à la détection d'évènements en environnements connus. Nous réalisons une analyse de ces résultats et abordons aussi les forces et faiblesses de l'approche proposée.

Chapitre 6 : Conclusion générale. Le mémoire se termine par un bilan des travaux de recherche. Nous dressons un ensemble de perspectives, ouvrant ainsi de nouvelles voies et propositions dans le but d'améliorer ce travail.

Chapitre 2

Le problème de la patrouille multi-agents et les réseaux de capteurs

2.1 Introduction

Plusieurs problèmes nécessitent de surveiller un environnement pendant une période de temps donnée. Si nous ne disposons pas assez de ressources perceptives pour contrôler en permanence les régions critiques de notre environnement, parler de vigilance ou de surveillance demande de parcourir très souvent certains lieux de l'environnement. De ce fait, réaliser une telle tâche s'assimile à une patrouille. Il existe plusieurs situations réelles où nous pouvons avoir recourt au problème de la patrouille. Nous pouvons ainsi citer :

- surveiller les feux de forêts
- recenser périodiquement les données pertinentes liées à la pollution (atmosphérique ou marine)
- détecter des intrusions de la part de l'ennemi, dans la surveillance des frontières par exemple
- contrôler les zones accidentées dans le but de détecter les catastrophes (glissements de terrains, les inondations, ...)

En ce qui concerne les feux de forêts, les dégâts résultant des incendies de forêts présentent des conséquences très néfastes sur l'environnement. La détection des feux constitue un des enjeux majeurs dans la lutte contre les incendies de forêt. Le déploiement de patrouilles de surveillance constitue une solution permettant de visiter très souvent certaines régions stratégiques de la forêt, afin de détecter au plus tôt les évènements liés au départ de feu. Par exemple, dans le cadre de la surveillance des frontières, une collabora-

tion France-Suisse s'est établie dans la région de Bâle, où les gardes-frontière suisses ont lancé une opération de surveillance des frontières à l'aide de drones, des petits avions sans pilote téléguidés du sol. D'autres pays également, à l'instar de la Suisse, utilisent des drones pour la surveillance de leurs frontières, remplaçant ainsi les opérations de surveillance aérienne, beaucoup plus onéreuses, effectuées auparavant par des hélicoptères.
Généralement, des patrouilles mixtes surveillent les frontières entre la Suisse et les pays voisins, dans une traque commune contre la criminalité et l'immigration illégale. Submergée par l'afflux de clandestins entrés illégalement sur son territoire, l'Italie multiplie les opérations d'intervention sur ses frontières afin de tenter d'endiguer le phénomène. Il convient donc de surveiller en permanence ces frontières contre les entrées clandestines. Mais ils sont confrontées à un espace particulièrement vaste dans le nord de la Botte, ce qui complique les opérations de contrôle. C'est ainsi que l'Italie et la Suisse se sont entendus pour patrouiller ensemble sur une partie des 770 kilomètres de frontière qu'ils ont en commun.
De ce qui précède, il ressort donc un intérêt évident pour les problèmes de la patrouille.

Dans nos travaux, nous nous intéressons à l'utilisation de la patrouille à l'aide des réseaux de capteurs pour la surveillance et la détection d'évènements dans des environnements connus des capteurs. Nous supposons que nos capteurs sont mobiles et ne peuvent identifier et détecter qu'un type d'évènement.

L'objectif de ce chapitre est de mettre en évidence l'intérêt, les objectifs et la problématique de la patrouille. La section 2.2 présente une définition, ainsi que les objectifs de la patrouille. A la section 2.3, la problématique et un état de l'art du problème de la patrouille en environnements connus sont présentés. La section 2.4 quant à elle présente la problématique et un état de l'art de la patrouille en environnements inconnus. La section 2.5 présente les définitions de capteurs et réseaux de capteurs. Elle présente aussi les problématiques liées aux réseaux de capteurs, ainsi que les contraintes et les difficultés liées à l'utilisation des réseaux de capteurs mobiles (ou MWSN [1]) pour le problème de la patrouille. Nous présentons dans la même section, un état de l'art de la patrouille multi-capteurs. Ce chapitre se termine par une synthèse dans la section 2.7.

1. Mobile Wireless Sensor Network

2.2 Définition et objectifs de la patrouille

Communément, une patrouille peut être considérée comme une marche de surveillance ou encore une exploration réalisée par un détachement d'agents (engins, soldats, agents de sécurité, agents logiciels, ...). D'après [Fer95, BD01], un agent est généralement défini comme étant une entité logicielle (virtuelle) ou physique capable d'effectuer ou d'accomplir des missions de manière autonome et si possible en coopération avec d'autres agents.
La patrouille est encore appelé *«patrolling»* en anglais, et sa définition selon [Aba96] est la suivante : *«the act of walking or travelling around an area, at regular intervals, in order to protect or surpervise it»*. Ainsi, cela reviendrait à déployer un ensemble d'agents, afin que ceux-ci visitent à intervalle de temps réguliers certains lieux stratégiques d'un environnement.

Le problème de la patrouille multi-agent est habituellement formulé ainsi [MRZD02, Che06, LC06] : élaborer une stratégie de parcours multi-agents d'un environnement. Lorsqu'on suppose que l'environnement est connu, il est représenté par un graphe $G = (V, E)$, où V représente les zones stratégiques ou pertinentes et E représente les moyens de transport ou de communication entre ces zones. Un tel environnement est dit discret et connu de tous les capteurs. D'une manière informelle, une stratégie efficace serait celle qui minimise le délai entre deux passages à un même endroit, et ainsi pour tous les endroits [MRZD02]. Une patrouille implique donc une équipe d'agents dont l'objectif est généralement de visiter pendant une période donnée les lieux les plus pertinents aussi souvent que possible. Pour une telle tâche, les agents doivent coordonner leurs actions dans le but d'obtenir des performances optimales.
En environnement inconnu, les agents doivent généralement effectuer deux tâches, à savoir celle d'exploration de l'environnement et ensuite celle de patrouille. C'est dans la phase d'exploration de l'environnement que l'agent construit une représentation de son environnement.

Les missions à assigner à une patrouille peuvent être diverses. Elles peuvent être par exemple, d'empêcher les intrusions de la part de l'ennemi, de prévenir des éventuels troubles, de recenser les données pertinentes. D'après [CEBP99, EG99, MRZD02, Che02], la patrouille a pour objectif le recueil d'informations sur un espace donné et implique la répétition des visites.
Pour [Lau05], une patrouille consiste à visiter à intervalle régulier les lieux stratégiques d'une région, dans le but :

- de collecter des renseignements fiables, régulièrement mis à jour par la patrouille,

– de la surveiller afin de la défendre contre une invasion ennemie,
– ou encore de faire transiter des informations entre plusieurs lieux d'une région.

La patrouille multi-agents vu comme tâche collective, peut se voir assigner les tâches et objectifs de la robotique collective. On peut recenser plusieurs applications dont les plus répandues sont :

– le fourragement, qui consiste à récupérer des objets dispersés dans l'environnement pour les rassembler dans un lieu précis [BHD94, FM98, Sim01].
– le déplacement en formation, qui permet aux robots de former une figure (ligne, triangle...) et de la maintenir au cours de leurs mouvements [YB96, BA98].
– la manipulation et la poussée collectives, qui concernent le transport d'objets qu'un unique robot ne pourrait déplacer seul [GM02, MM03].

Nous allons aborder dans les deux sections qui suivent la problématique de la patrouille en environnements connus d'une part, et en environnements inconnus d'autre part.

2.3 Patrouille en environnements connus

2.3.1 Problématique

En environnement connu, le problème de la patrouille multi-agent est habituellement formulé comme suit [MRZD02, SRa04, LCS05, Che05, Che06, LC06, LK08].
L'environnement à patrouiller est modélisé par un graphe $G = (V, E)$, où V représente les zones stratégiquement pertinentes et E les moyens de transport ou de communication entre eux. Un coût c_{ij}, associé à chaque arc (i, j), mesure le temps nécessaire pour aller d'un noeud i à un noeud j. Considérons r agents ayant pour but de visiter à intervalles réguliers les zones définies dans le graphe G. On suppose que chaque agent se trouve sur un des noeuds de V à l'instant initial.
Ce type de représentation convient pour la patrouille si les lieux d'intérêts ont initialement été recensés. Les agents sont ici dotés d'une pré-connaissance de l'environnement. Les agents mémorisent la carte (ou environnement) car on suppose une planification du parcours optimale, avant l'exécution de leur tâche [GR01, ARS+04, LC06] de patrouille. Cette tâche de planification est faite de façon *offline*, comme c'est le cas dans [GR01, ARS+04, LC06], le but étant de déterminer le parcours optimale associé à chaque agent.

Résoudre le problème de la patrouille consiste alors à élaborer une stratégie (cf. définition 2.3.1) de parcours multi-agent du graphe G. Une telle stratégie doit optimiser un critère de qualité donné basé sur la notion d'oisiveté (cf. définition 2.3.2).

Définition 2.3.1 (Stratégie de patrouille [LCS05, Che06])**.** *On appellera stratégie de patrouille d'un agent, la fonction* $\pi : \mathbb{N} \rightarrow V$ *telle que* $\pi(j)$ *désigne le* j^{me} *noeud visité par l'agent. On désignera par* $\Pi = \{\pi_1, ..., \pi_r\}$ *la stratégie multi-agent, définie comme étant l'ensemble des stratégies* $\pi_1, ..., \pi_r$ *des* r *agents.* $\Pi^r(G)$ *désigne l'ensemble des stratégies multi-agents possibles avec* r *agents sur le graphe* G*.*

Un ensemble de critères ont été développés afin d'évaluer la qualité d'une stratégie de patrouille multi-agents après un certain nombre de pas de temps (ou cycles) de simulation. Globalement, la littérature présente deux critères [Gla11] : la fréquence ou nombre de visite et l'oisiveté.

Le premier critère (le nombre de visites) est moins répandu que l'oisiveté. Il repose sur la maximisation de la fréquence des visites des noeuds du graphe [EAK07]. On distingue globalement deux principaux critères relatifs à la fréquence de visites des noeuds : la fréquence uniforme et la fréquence moyenne. La première est un critère qui consiste à observer la variance entre les fréquences de visite des noeuds de l'environnement, tandis que la deuxième repose simplement sur la moyenne des fréquences de visite des noeuds du graphe.

Le deuxième critère (l'oisiveté ou *idleness*) peut se définir à partir de la définition intuitive d'une stratégie de patrouille pertinente [ARS+04]. En effet, il est communément admis qu'une stratégie de patrouille pertinente est une stratégie qui minimise pour chaque noeud (lieu ou région) le délai entre deux visites. L'évaluation des performances d'une stratégie est basée sur le concept d'*oisiveté instantanée*, définie par :

Définition 2.3.2 (Oisiveté instantanée d'un noeud [Che02])**.** *L'oisiveté instantanée* $I_n(t)$ *d'un noeud* n *au temps* t *est l'intervalle de temps écoulé depuis la dernière visite d'un agent à ce noeud. Par convention, à l'instant* 0*, l'oisiveté de chaque noeud est nulle.*

[MRZD02] présente différents critères basés sur l'oisiveté instantanée afin d'évaluer des stratégies de patrouille. Les auteurs proposent une évaluation de la patrouille basée sur l'oisiveté instantanée d'un noeud, l'expression de la pire oisiveté et du temps d'exploration. La raison principale est que ces

critères sont statistiquement parlant, et parfaitement adéquats pour mesurer la qualité des solutions. Pour ces mêmes raisons, nous utiliserons des critères basés sur l'oisiveté.
A partir de l'oisiveté d'un noeud, on peut construire divers critères pour mesurer l'oisiveté du graphe à un instant donné. Dans notre cas, nous en définissons deux : l'oisiveté moyenne (Average Idleness cf. équation 2.2) et la pire oisiveté (Worst Idleness cf. équation 2.1).

L'oisiveté instantanée du graphe ou de l'environnement (équation 2.2) est la moyenne des oisivetés instantanées de tous les noeuds du graphe à un instant donné.
La pire oisiveté (équation 2.1) est la plus grande valeur obtenue de l'oisiveté instantanée des noeuds pendant toute la durée de simulation.

En environnements connus, plusieurs études ([Che02, MRZD03, ARS+04, LC06, LK08]) se basent sur la pire et/ou la moyenne oisiveté du graphe pour mesurer les performances des stratégies de patrouille. Ainsi, une bonne stratégie de patrouille π sur le graphe G est celle qui minimise la pire oisiveté $WI^{\pi}(G)$ ou la moyenne oisiveté $AI^{\pi}(G)$. Celles-ci se définissent à partir de la pire oisiveté instantanée $WI_t^{\pi}(G)$ du graphe G à l'instant t ou de la moyenne oisiveté instantanée $AI_t^{\pi}(G)$ du graphe G à l'instant t, telles que :

$$WI_t^{\pi}(G) = max_{n \in V} I_n^{\pi}(t) \tag{2.1}$$

$$AI_t^{\pi}(G) = \frac{1}{|V|} \sum_{n \in V} I_n^{\pi}(t) \tag{2.2}$$

Dans les expressions précédentes de WI et AI, V désigne l'ensemble des noeuds du graphe et $|V|$ la cardinalité de cet ensemble.
Une stratégie de patrouille multi-agent π quelconque peut être évaluée sur la base de $T \to \infty$ cycles de simulation en utilisant soit le critère de la *pire oisiveté* notée WI_T (cf. équation 2.3), soit le critère de l'*oisiveté moyenne* notée AI_T (cf. équation 2.4). Comme le montrent les équations 2.4 et 2.3, les moyenne et pire oisivetés globales se définissent en fonction des moyenne et pire oisivetés à l'instant t.

$$WI_T^{\pi}(G) = max_{t=1,2,\ldots,T} WI_t^{\pi}(G) = max_{t=1,2,\ldots,T} \left(max_{n \in V} I_n^{\pi}(t) \right) \tag{2.3}$$

$$AI_T^{\pi}(G) = \frac{1}{T}\sum_{t=1}^{T} AI_t^{\pi}(G) = \frac{1}{T}\sum_{t=1}^{T}\left(\frac{1}{|V|}\sum_{n\in V} I_n^{\pi}(t)\right) \tag{2.4}$$

La stratégie de patrouille optimale π^* sur G est telle que $\pi^* = argmin_{\pi} AI_T^{\pi}(G)$ si le critère AI est utilisé et $\pi^* = argmin_{\pi} WI_T^{\pi}(G)$ si le critère WI est utilisé.

Pour mesurer la robustesse des stratégies, [Pou11] utilise un critère de performance basé sur la notion d'*intervalle quadratique moyen* (*Mean Square Interval* : *MSI*). Il introduit d'autres métriques telles que le *temps de stabilisation* et les *courbes de stabilisation*. Le *temps de stabilisation* permet de mesurer le temps au bout duquel une simulation trouve un état stable, tandis que les *courbes de stabilisation* permettent de représenter la répartition des variations dans le temps pour que les simulations se stabilisent.
La notion d'intervalle est aussi utilisé par [SRT10] comme critère d'évaluation pour le problème de la patrouille multi-agents. Les variantes suivantes : intervalle moyen entre les visites, la variance des intervalles et l'intervalle maximum sont utilisées dans le but de comparer les différentes stratégies.

2.3.2 Etat de l'art

[GR01] propose une approche de patrouille basée sur une heuristique reposant sur l'utilisation de *spanning trees* et permettant de construire des cycles hamiltoniens. En théorie des graphes, les spanning trees permettent de représenter un graphe par un arbre reliant les différents noeuds du graphe. Dans la méthode proposée par [GR01], l'environnement est considéré comme une grille avec des cellules de côté de longueur n, les auteurs lui superposent une seconde grille, de granularité plus importante, avec des cellules de côté de longueur 2n (cf. figure 2.2). Ainsi, à chaque cellule de l'environnement de grande granularité sont associées 4 cellules de l'environnement de petite granularité (environnement original). Un algorithme de construction de spanning tree est ensuite exécuté sur le graphe ou environnement de grande granularité. L'agent n'a alors plus qu'à suivre le spanning tree (chemin fléché conformément au tracé de la figure 2.2) pour effectuer une patrouille optimale de l'environnement.

Si une patrouille optimale est rapidement planifiée avec cette méthode (avec une complexité en temps linéaire $O(N)$, où N représente le nombre de cellules de l'environnement), elle reste cependant assez limitée car elle pose certaines contraintes fortes sur l'environnement. D'une part, le graphe où on effectue la patrouille doit être une grille à voisinage de Von Neumann (chaque

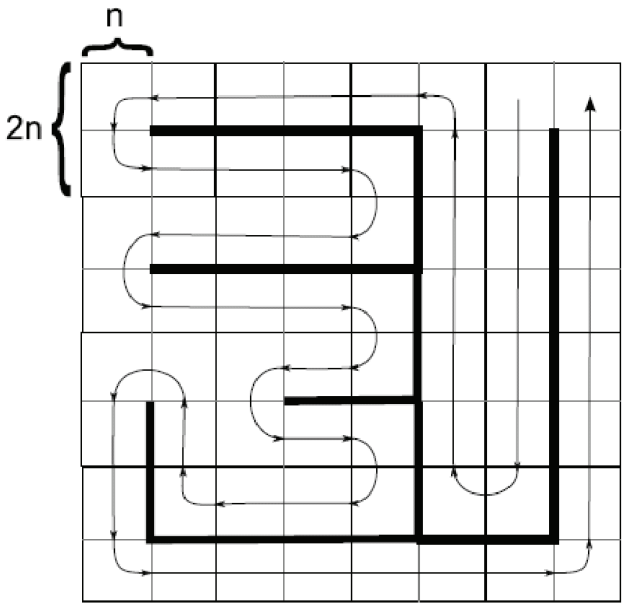


FIGURE 2.1 – Un exemple de cycle hamiltonien réalisé à partir d'un spanning tree sur un environnement grille de taille 8x8 noeuds. Extrait de [Gla11]

cellule (noeud) à au plus 4 voisins) dont les cotés sont de longueurs paires. Et d'autre part, les obstacles qu'on rencontre dans l'environnement peuvent rendre cette méthode inopérante si leur densité est importante.

[LCS05] propose une formulation du problème de la patrouille multi-agent ($PPMA$) à l'aide d'un *processus décisionnel de Markov* (PDM), et de ce fait, trouver une politique optimale de patrouille se réduit alors à résoudre ce PDM. Pour les auteurs, le problème de la patrouille multi-agent sur un *graphe unitaire*[2] peut être caractérisé de manière globale par un processus décisionnel de Markov $\theta = (S, A, T, R)$, tel que :

- $S = |V|^r \times \mathbb{N}_+^{|V|} \times \mathbb{N}$ est l'espace d'états incorporant la position actuelle de chacun des r agents, l'oisiveté de chacun des noeuds du graphe, ainsi que la pire oisiveté rencontrée jusqu'à présent.
- $A \subseteq |V|^r$ représente l'ensemble des actions jointes possibles des agents.
- $S \times A \times S \to \{0, 1\}$ est la fonction de transition déterministe entre les états.

2. Graphe dont chaque arc est de poids 1

$$R = \begin{cases} 0 \text{ si l'état courant possède une pire oisiveté inférieure ou égale à} \\ \text{celle de l'état précédemment atteint} \\ \text{-1 sinon.} \end{cases}$$

La fonction de transition peut être dérivée à partir de la structure du graphe. Considérons : un état $s = \left\langle (p_1, \ldots, p_r), \left(I_1, \ldots, I_{|V|}\right), WI \right\rangle$, un autre état $s^{'} = \left\langle \left(p_1^{'}, \ldots, p_r^{'}\right), \left(I_1^{'}, \ldots, I_{|V|}^{'}\right), WI^{'} \right\rangle$, et une action $a = (a_1, \ldots, a_r)$, alors $T(s, a, s^{'}) = 1$ ssi :

$$(\forall k) : \begin{cases} a_k = p_k^{'} \\ (p_k, a_k) \in E \\ \begin{cases} I_k^{'} = 0, & \text{si } (\exists m) \text{ tel que } a_m = k \\ I_k^{'} = I_k + 1, & \text{sinon} \end{cases} \end{cases}$$

$$\text{et } WI^{'} = max\left\{WI, I_1^{'}, I_1^{'}, \ldots, I_{|V|}^{'}\right\}$$

D'une part, pour une stratégie de patrouille optimale π, la valeur de l'oisiveté de chaque noeud est bornée. D'autre part, le nombre d'états effectivement visités par une stratégie de patrouille optimale π est fini car le nombre des positions jointes possibles est fini puisque le graphe a une taille finie, et que les valeurs des oisivetés pour chaque noeud sont finies.
Ainsi, la résolution du $PPMA$ peut se réduire à trouver des stratégies finies ayant une structure particulière. Donc pour n'importe quel $PPMA$, il existe une stratégie de patrouille à horizon infinie π qui est cyclique, et qui peut être représentée par $\pi = \pi_{init}.(\pi_{cyc})^*$, où π_{cyc} désigne une sous-stratégie cyclique et π_{init} désigne la sous-stratégie qui précède π_{cyc} dans π. Cette formalisation du problème de la patrouille multi-agent par un PDM est intéressante d'un point de vue théorique mais le PDM engendré devient vite intractable pour lorsque le graphe et la population d'agents patrouilleurs sont de grande taille.

[Che06] effectue une étude théorique de la patrouille et aborde le problème sous l'angle de l'optimisation combinatoire. Certains problèmes d'optimisation combinatoire ont des points communs avec le problème de la patrouille. En particulier, l'auteur a pu mettre en évidence la similarité entre le problème du voyageur de commerce (*TSP*) et celui de la patrouille mono-agent. Il montre que pour le cas particulier d'un seul agent, le problème de la patrouille est équivalent à une certaine variante du TSP. Ainsi, partant d'une analogie entre la patrouille et le TSP, il cherche à définir l'optimalité de

quelques types de stratégies de patrouille (*à cycle unique*, *régionalisée* et *mixte*) en terme de pire oisiveté du graphe.
La première stratégie, dite *à cycle unique*, consiste à faire réitérer par les agents le parcours TSP du graphe, sachant que les agents sont espacés régulièrement sur le cycle. L'auteur démontre que dans le cas multi-agents, l'oisiveté maximum obtenue est alors inférieure à 3 fois l'oisiveté maximum optimale + 4 fois la longueur du plus long arc, c'est-à-dire par $3 \times opt + 4 \times max_{ij} \{c_{ij}\}$. La valeur de *opt* désigne ici la valeur optimale du critère de pire d'oisiveté sur les classes des stratégies à cycle unique. Lorsque les arcs possèdent des longueurs très variées, le cycle unique est inadéquat.
La deuxième stratégie considérée, la stratégie régionnalisée, est une stratégie de parcours du graphe G où chaque agent i ne visite les noeuds que d'une région P_j. Si le nombre d'agents est égal au nombre de régions, alors on dit que la stratégie est individuellement régionalisée. Ainsi, avec les stratégies individuellement régionalisées, chaque noeud n'est visité que par un seul agent (cf. figure 2.2).
La classe des stratégies mixtes combine les stratégies à cycle unique et les stratégies régionalisées. La stratégie mixte permet d'améliorer les résultats obtenus avec des stratégies à cycle unique et éviter aux agents la traversée des arcs les plus long en partitionnant le graphe. Chaque agent se voit alors attribuer un territoire sur lequel la stratégie à cycle unique est appliquée. Cette approche permet de montrer que le problème de la patrouille est 15-approximable sans restriction quant au type de graphe c'est-à-dire qu'il existe un algorithme polynomial qui détermine une stratégie de patrouille avec une complexité au plus 15 fois plus grande que celle de l'algorithme optimum. Cela voudrait dire qu'on peut déterminer une stratégie mixte avec une complexité inférieure à 15 fois celle de complexité optimale. Remarquons que la stratégie mixte déplace simplement le problème car la question du partitionnement reste entière. La figure 2.2 extraite de [Che06] présente une stratégie à cycle unique et une autre régionalisée.

Tel que nous le verrons plus loin dans ce chapitre et le chapitre suivant, si l'on se place dans un contexte des réseaux de capteurs, nous pourrions anticiper sur les difficultés liées aux contraintes communicationnelles. Le cycle unique sera applicable, à condition que les capteurs soient en communications directes ou indirectes avec un des *Sinks* [3], ce qui n'est pas toujours le cas. De même, une stratégie régionalisée devra aussi prendre en compte cet aspect communicationnel.

3. Un Sink est un capteur particulier qui est capable de communiquer directement avec le centre de contrôle. Nous reviendrons sur son rôle ultérieurement.

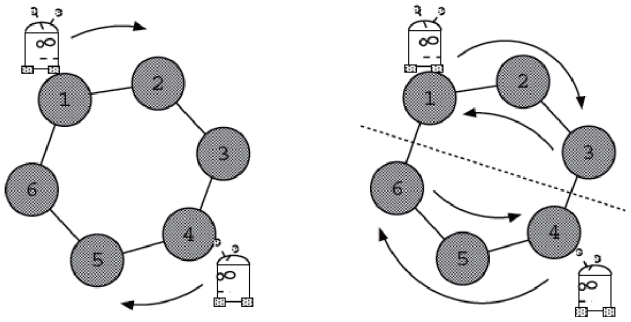


FIGURE 2.2 – Exemples de stratégies à *cycle unique* (à gauche) et de stratégies *régionalisées* (à droite) [Che06]. La stratégie à cycle unique Π_{cyc} est formée de deux cycles $\{\pi_{cyc1}, \pi_{cyc2}\}$ avec π_{cyc1}= 1,2,3,4,5,6,1,2,3,... et π_{cyc2}=4,5,6,1,2,3,4,5,6,1,.... La stratégie régionalisée Π_{Reg}= $\{\pi_{reg1}, \pi_{reg2}\}$, avec π_{reg1}=1,2,3,2,1,2,... et π_{reg2}=4,5,6,5,4,5,....

[LC06, LK08] propose une stratégie de patrouille déterminée à partir d'un algorithme utilisant des colonies de fourmis [CDM91, CDM92, DMC96, SD02, DB05]. Il s'agit de déterminer *offline*, par simulation, les stratégies à appliquer aux différents agents patrouilleurs. Une stratégie est définie par une liste des noeuds du graphe que va visiter l'agent associé. L'approche par colonies de fourmis consiste à déployer, à chaque itération, différentes colonies de fourmis sur un graphe, comme le montre la figure 2.3. La meilleure colonie, c'est-à-dire celle qui minimise la pire oisiveté du graphe, est sélectionnée à l'issue du déploiement. Les fourmis de la meilleure colonie déposent des phéromones sur les noeuds du graphe et une nouvelle itération de l'algorithme commence. Ce nouveau dépôt de phéromones incite toutes les fourmis à suivre avec une plus forte probabilité la solution de la meilleure colonie.

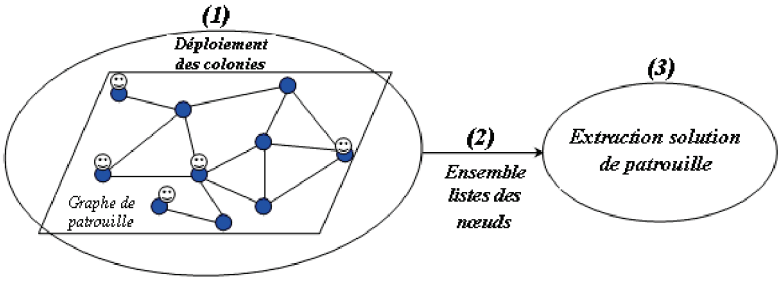


FIGURE 2.3 – Structure globale de l'Approche ACO sur un graphe

La figure 2.3 présente la structure globale de l'approche OCF sur un graphe. Le (1) matérialise le processus de déploiement et de la mise en compé-

tition des différentes colonies. A l'issue du processus (1), on a comme résultat (représenté par (2)) un ensemble de listes de noeuds représentant l'ensemble des chemins obtenus par les fourmis des colonies. Sur la base de l'ensemble de chemins découlant de (1), le processus (3) permet l'extraction de la solution de patrouille à fournir aux agents qui à leur tour vont procéder à la patrouille proprement dite. Deux approches sont étudiées relativement aux noeuds de départ des agents. Dans [LC06], les agents partent d'un même noeud initial, tandis que dans [LK08], une étude comparative est faite entre le départ d'un même noeud initial et celui à partir des différents noeuds du graphe (dispersion des agents à travers le graphe). La stratégie par dispersion initiale des agents permet d'obtenir les meilleurs résultats expérimentaux de patrouille.

Au sein d'une colonie, les fourmis se déplacent sur les différents noeuds du graphe en allant préférentiellement vers les noeuds adjacents les moins visités. Chaque fourmi k d'une colonie l enregistre dans une liste tabou associée à la colonie la liste des noeuds déjà visités. Une fourmi choisit le prochain noeud à visiter selon une probabilité $p_{ij}^{k,l}$. Ce prochain noeud j sera celui ayant la plus grande probabilité parmi tous les noeuds adjacents. La probabilité de sélection d'un noeud par une fourmi k d'une colonie l est donnée par l'expression $p_{ij}^{k,l}$ ci-dessous :

$$p_{ij}^{k,l}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{u \in autorisé_l}[\tau_{iu}(t)]^{\alpha}[\eta_{iu}]^{\beta}} & \text{si } j \in autorisé_l \\ 0 & \text{sinon} \end{cases}$$

où
$autorisé_l$ représente l'ensemble des noeuds non visités par les fourmis de la colonie l
$\tau_{ij}(t)$ représente l'intensité des phéromones sur l'arête (i, j) au cycle t
$\eta_{ij} = 1/c_{ij}$ indique la visibilité du noeud j par rapport au noeud i
α et β sont considérés comme des paramètres pour le contrôle respectivement, de l'intensité des phéromones et de la visibilité.

En ce qui concerne la mise à jour des phéromones, elle se fait selon l'équation 2.5, et est inspirée de celle adoptée par Colorni *et al.* [CDM91, CDM92, DMC96]. Ainsi, à chaque cycle, l'intensité de phéromone sur chaque arête du graphe diminue par évaporation. Lors de leur tournée, chaque fourmi dépose une quantité de phéromone $\Delta_{\tau_{ij}}$ sur l'arête (i, j).

$$\tau_{ij}(t+1) = \rho\, \tau_{ij}(t) + \Delta_{\tau_{ij}} \tag{2.5}$$

où
$(1 - \rho)$ est le coefficient d'évaporation.

$\tau_{ij}(t+1)$ et $\tau_{ij}(t)$ représentent les intensités des phéromones sur l'arête (i,j) au temps de cycle t+1 et t.
$\Delta_{\tau_{ij}}$ est la quantité de phéromone déposée sur l'arête (i,j) par toutes les fourmis de la colonie à ce cycle.

[Pou11] étudie le problème de la patrouille dans un système ouvert, dans lequel les agents peuvent rejoindre ou quitter la tâche quand ils le veulent. L'auteur a choisi comme domaine d'application une catastrophe naturelle (tremblement de terre, tsunami...), où des victimes doivent être détectées et sauvées. Pour cela, une équipe d'agents doit être déployée sur la zone sinistrée pour chercher les victimes dans les décombres. Le problème est étudié sous un angle de la patrouille dans un système ouvert, dans lequel les agents peuvent rejoindre ou quitter la tâche quand ils le veulent. Dans ce nouveau problème, l'auteur s'attelle à comprendre comment les différentes stratégies peuvent être adaptées pour être capable de gérer un changement qui survient dans le système. Il s'intéresse à l'évaluation des performances de l'approche de résolution qu'il propose. C'est ainsi que pour mesurer la robustesse des stratégies, il utilise un critère de performance basé sur la notion d'*intervalle quadratique moyen* (*Mean Square Interval* : *MSI*). Afin d'étudier la durée des phases de transition et la vitesse à laquelle les stratégies retrouvent la stabilité, les métriques pour l'évaluation sont respectivement le *temps de stabilisation* et les *courbes de stabilisation*. Le *temps de stabilisation* est le temps en terme de nombre de cycles au bout duquel la simulation retrouve un état stable. Les *courbes de stabilisation* représente la répartition des variations dans le temps. Ces variations s'appuient sur une comparaison des vitesses auxquelles les simulations atteignent un certain seuil de la valeur stabilisée.
Cette notion d'intervalle a aussi été utilisé par [SRT10] comme critère d'évaluation pour le problème de la patrouille multi-agents. Les auteurs exploitent l'intervalle moyen entre les visites, la variance des intervalles et l'intervalle maximum pour comparer les différentes stratégies.
L'intervalle moyen et le MSI sont complémentaires car le premier renseigne sur la fréquence globale des visites, tandis que le second renseigne sur la régularité de celles-ci.

C. Poulet a étudié différentes stratégies, à savoir :

- Random Reactive Agent (RR) où les agents sont réactifs et non communicant. Un agent choisit le prochain noeud (considéré comme but) au hasard parmi les noeuds connectés à celui sur lequel il se trouve.
- Conscientious Reactive Agent (CR) où les agents sont réactifs et non

communicant. Un agent choisit comme prochain noeud (but), le noeud connecté à celui sur lequel il se trouve qui présente la plus haute oisiveté instantanée.
- Cognitive Coordinated Agent (CC) où les agents sont cognitifs et communicant avec un coordinateur global qui choisit comme prochain noeud, le noeud du graphe ayant la plus grande oisiveté instantanée.
- Heuristic Pathfinding Cognitive Coordinated Agent (HPCC) où les agents sont cognitifs et communicant. L'agent coordinateur utilise une heuristique qui combine oisiveté et distance pour définir les tâches de patrouille.
- Single Cycle (SC) où les agents utilisent le plus court cycle passant par tous les noeuds du graphe. Un coordinateur calcule le cycle minimal dans le graphe, puis distribue uniformément les agents autour de ce cycle.

Une étude comparative des différentes stratégies permet de dire que si l'on recherche une transition lisse et courte, au prix de performances globales légèrement moindres, alors *HPCC* est la stratégie la plus efficace. Par contre, si une perte de performance est permise pendant la transition, alors *SC* est le meilleur choix. Globalement, ces deux stratégies présentent de meilleures performances que les autres stratégies.
C. Poulet a envisagé un scénario simple, avec une implémentation sur un seul graphe. Les stratégies qu'il a implémentées ont été choisies pour leur intérêt historique et pour leurs performances.

2.3.3 Discussion

Dans cette section, un état de l'art du domaine de la patrouille en environnements connus a été présenté. L'environnement de patrouille est modélisé dans ce cas par un graphe. Lors de la tâche de patrouille, les agents sont dotés d'une pré-connaissance de l'environnement. Dans la formulation du problème de la patrouille multi-agent à l'aide d'un processus décisionnel de Markov (*PDM*), résoudre le problème de patrouille revient à trouver une politique optimale de patrouille par le biais de la résolution du *PDM* obtenu. L'approche est présentée ici sous l'angle d'une analyse formelle du problème multi-agent de la patrouille à l'aide des processus décisionnels de Markov. Dans le même contexte, Y. Chevaleyre présente une étude théorique de la patrouille et aborde le problème sous l'angle de l'optimisation combinatoire. Comparativement aux études théoriques liées au problème de patrouille, d'autres sont beaucoup plus pratiques. Ces dernières consistent à déterminer les stratégies de patrouille par le biais d'un algorithme utilisant des colonies de fourmis. Cela consiste à déterminer par simulation les

stratégies à appliquer aux différents agents patrouilleurs. Ces stratégies découlent du déplacement des différentes fourmis sur les différents noeuds du graphe. Leurs déplacements utilisent le processus de dépôt et évaporation des phéromones qui permet aux fourmis des différentes colonies de marquer les chemins suivis amenant les autres fourmis à suivre probablement la solution de la meilleure colonie. Cette étude *offline* permet d'envisager une possibilité d'adaptation de l'approche dans un contexte où le groupe d'agents qui patrouillent est un réseau de capteurs.

2.4 Patrouille en environnements inconnus

2.4.1 Problématique

En environnement inconnu, le problème de la patrouille multi-agent peut être définie comme suit : On se donne un environnement G, un nombre de N agents avec une perception limitée, avec pour objectif de visiter régulièrement l'ensemble des lieux ou régions accessibles de G. Le principal but est de parvenir à minimiser le délai entre deux visites d'un même lieu ou d'une même région.
Dans [CGS+07], l'environnement de patrouille est inconnu et on suppose une impossibilité pour les agents de disposer du graphe le représentant. L'espace exploré par les agents est alors représenté par une matrice de cellules dont chaque cellule peut être : *libre, occupée par un agent, ou inaccessible (un obstacle)*. Dans le modèle $CLInG$ de [Sem04], il s'appuie sur la propagation de l'information liée aux visites dans une représentation de l'environnement partagée par les robots. Ces derniers se déplacent sur la base des signaux propagés et perçus localement. Ce modèle privilégie la répartition spatiale des robots sur l'attribution d'une tâche à un individu.

De façon générale, la tâche de patrouille est exécutée sans connaissance de l'environnement par les agents. De ce fait, les agents doivent alors effectuer deux tâches : l'exploration et la patrouille. L'utilisation soit d'une équipe d'agents réactifs soit d'une équipe d'agents doués de coordination et d'auto-organisation est envisageable. Dans la première approche, les agents réactifs doivent réaliser un apprentissage ou encore faire usage des techniques basées sur le marquage de l'environnement pour réaliser les tâches d'exploration et de patrouille. Dans la deuxième approche, les actions sont coordonnées et organisées au sein d'une formation ou d'un groupe. Ainsi, le groupe d'agents est capable d'établir et de maintenir une formation, c'est-à-dire de se rendre à une position donnée de l'environnement, tout en évitant les obstacles, les

collisions entre agents, et en établissant et en maintenant une position relative par rapport à un leader ou à un ou plusieurs de ses voisins.

Plusieurs études [MRZD02, CGS+07, FTKL09b] se basent sur la pire et/ou moyenne oisiveté de l'environnement pour mesurer les performances des stratégies de patrouille. Dans [CGS+07, FTKL09b, FTKL09a], les critères utilisés sont basés sur l'oisiveté instantatée d'une cellule de l'environnement, l'oisiveté instantanée de l'environnement, l'oisiveté moyenne de l'environnement et la pire oisiveté de l'environnement.
Une borne d'oisiveté optimale peut être évaluée en faisant une hypothèse sur l'existence d'un cycle hamiltonien dans l'environnement qui correspond donc à une patrouille optimale. C'est ainsi que [Gla11, CGS+07, Chu07] évaluent les valeurs théoriques optimales d'oisiveté en supposant un environnement constitué de n cellules accessibles. Ainsi, un seul agent pourra visiter l'ensemble des cellules de l'environnement en n itérations. A la fin du parcours, l'oisiveté du noeud le plus ancien sera de $n-1$. Si nous considérons k agents, on obtient une oisiveté maximale égale à $\frac{n}{k}-1$. Puisque la moyenne d'une suite linéaire est sa valeur maximale divisée par deux, l'oisiveté moyenne instantanée optimale est alors évaluée ici par l'expression $\frac{\frac{n}{k}-1}{2}$. Pour ce qui est de l'oisiveté maximale instantanée optimale, elle est évaluée par $\frac{n}{k}-1$.

[Sem04] définit comme critères de comparaison des stratégies de patrouille l'oisiveté instantanée, l'oisiveté normalisée et l'oisiveté d'activité. Dans un premier temps, l'environnement calcule à chaque pas de temps l'oisiveté de chaque cellule accessible en incrémentant sa valeur d'une unité, et dans un second temps, l'environnement effectue la propagation des oisivetés maximales des cellules. L'oisiveté normalisée représente la moyenne des résultats expérimentaux obtenus avec un nombre d'agents et de régions variables. Ainsi, l'oisiveté normalisée se définit comme suit : $oisivete_{normalisee} = oisivete \times \frac{nombre_{agents}}{nombre_{regions}}$. L'oisiveté normalisée représente donc un moyen pratique d'uniformiser les résultats provenant des expérimentations.
L'oisiveté d'activité représente l'oisiveté relative au taux d'activité et se définit comme suit : $oisivete_{activite} = oisivete \times Taux_{activite}$. Dans cette expression, le taux d'activité se définit par $Taux_{activite} = \frac{1}{nombre_{agents}} \sum_{i=1,2,\cdots,nombre_{agents}} Taux^i_{activite}$, où $Taux^i_{activite}$ représente le taux d'activité de l'agent i indiquant aussi la proportion de temps consacré réellement à la patrouille.

Un autre critère d'évaluation des stratégies de patrouille est celui de la fréquence de visite des cellules de l'environnement où il s'agit de maximiser la fréquence de visite des cellules de l'environnement. D'après [Gla11], on

obtient une patrouille optimale lorsque tous les noeuds sont visités à une fréquence f telle que $f = \frac{k}{n}$, où k représente le nombre d'agents patrouilleurs et n le nombre de noeuds ou cellules de l'environnement.

2.4.2 Etat de l'art

La patrouille a pour objectif le recueil d'informations, et peut être vu comme une extension du problème de couverture [Sem04]. La couverture implique que la totalité de l'environnement soit visitée au moins une fois par un robot, ou alors que les capteurs de ces derniers couvrent une surface maximale [WLB96, BFM+00, ZSDT02]. L'objectif peut être la cartographie d'un lieu inconnu, le nettoyage automatique ou la surveillance d'un local. [MRZD02] est un des pionniers de l'introduction de la patrouille dans le cadre des systèmes multi-agents. Il a abordé le problème en utilisant le paradigme multi-agents classique. Une première de ses solutions a consisté à développer des agents réactifs, ne possédant aucune capacité de planification ou de raisonnement, chaque agent se déplaçant en direction de la zone de son voisinage immédiat n'ayant pas été récemment visitée. Dans une deuxième approche basée sur des agents cognitifs, chaque agent se dirige vers une partie du territoire n'ayant pas reçu de visite d'agent depuis longtemps, en utilisant un algorithme du plus court chemin. Un mécanisme de communication entre agents leur permet de se coordonner en se partageant les régions à visiter, afin que deux agents ne se dirigent pas vers le même lieu.

[Sem04] étudie l'influence des perturbations sur l'activité d'un groupe de robots en prenant la patrouille comme exemple de tâche collective. Dans le cadre de sa tâche de patrouille, il propose le modèle CLInG [4] qui s'appuie sur la propagation de l'information liée aux tâches élémentaires (les visites) dans une représentation de l'environnement partagée par les robots. La stratégie de propagation consiste à mettre à disposition une information locale qui synthétise l'état global de l'environnement à travers la propagation de l'oisiveté d'une région à l'autre comme le montre la figure 2.4. Ainsi, une région i portera une oisiveté propagée (OP_i) en plus de son oisiveté individuelle (O_i). Le gradient formé par l'oisiveté propagée est commun à toute la collectivité. Le comportement de visite est déclenché lorsqu'un robot se trouve à un sommet dont l'OP est supérieure ou égale aux OP voisines. Ainsi, les robots perçoivent les valeurs locales du signal propagé, valeurs qui déterminent leur prochain déplacement. Plus formellement :

4. Choix Local fondé sur une Information Globale

$OP_i = max(O_i, max(f(i,j)))$, $\forall j \in V(i)$ l'ensemble des régions voisines de i.
f est une fonction de propagation telle que :

$$\begin{cases} \text{f(i,j)} = OP_j - \alpha.D(i,j) - \beta.I(j) & \text{si } OP_j - \alpha.D(i,j) - \beta.I(j) \geq OP_{min} \\ \text{f(i,j)} = OP_j - 1 & \text{si } OP_j \leq OP_{min} \\ \text{f(i,j)} = OP_{min} & \text{sinon} \end{cases}$$

$D(i,j)$ est le temps estimé pour aller du centre de la cellule i à celui de la cellule j. Il dépend de la distance entre les centres, et de la vitesse moyenne des robots.
α est le coefficient de propagation.
$I(j)$ est la fonction d'interception. Elle dépend du nombre de robots présents dans la région j et de leur état.
β est le coefficient d'interception.
OP_{min} sert de seuil afin que l'oisiveté propagée reste positive et produise toujours un *gradient de l'oisiveté*[5]. OP_{min} est au moins égal au nombre de régions.
L'algorithme de propagation se fait en deux étapes :

1. Pour toutes les régions i : Appliquer la fonction de propagation $f(i,j)$, $\forall j \in V(i)$ l'ensemble des régions voisines de i.
2. Pour toutes les régions i : Mettre à jour la valeur de l'oisiveté propagée à $OP_i = max(O_i, max(f(i,j)))$

Nous présentons à la figure 2.4 un exemple de propagation dans un couloir composé de 5 régions. Par souci de clarté, la fonction de propagation est simplifiée : $f(i,j) = OP_j - 10$. Cette figure présente aussi l'état des régions après 4 passes. La première étape (*Etape 0*) représente l'état initial où dans chaque région, l'oisiveté courante coïncide avec l'oisiveté propagée. A la deuxième étape (*Etape 1*), chaque région envoie aux régions adjacentes son oisiveté propagée diminuée de 10 (c'est-à-dire *oisiveté propagée -10*) conformément à la fonction de propagation $f(i,j)$. A la troisième étape (*Etape 2*), chaque région effectue une mise à jour de l'oisiveté propagée en considérant le maximum entre l'oisiveté courante et les oisivetés propagées reçues conformément à l'équation $OP_i = max(O_i, max(f(i,j)))$.
L'ensemble de ces trois étapes constitue une *passe*. A partir de la troisième étape (*Etape 2*), une première passe permet d'obtenir les valeurs *{(120,120), (110,70), (60,50), (40,40), (30,20)}*. Si on effectue une deuxième, troisième

5. Car le robot effectue le choix de la prochaine action en fonction de l'état des régions voisines. Il suit le gradient de l'oisiveté la plus forte qu'il construit lui même par ses actions.

et quatrième passe, on obtient le résultat présenté à la figure 2.4, avec les valeurs *{(120,120), (110,70), (100,50), (90,40), (80,20)}* où la première composante d'un couple représente l'oisiveté propagée et la seconde composante représente l'oisiveté courante d'une région.

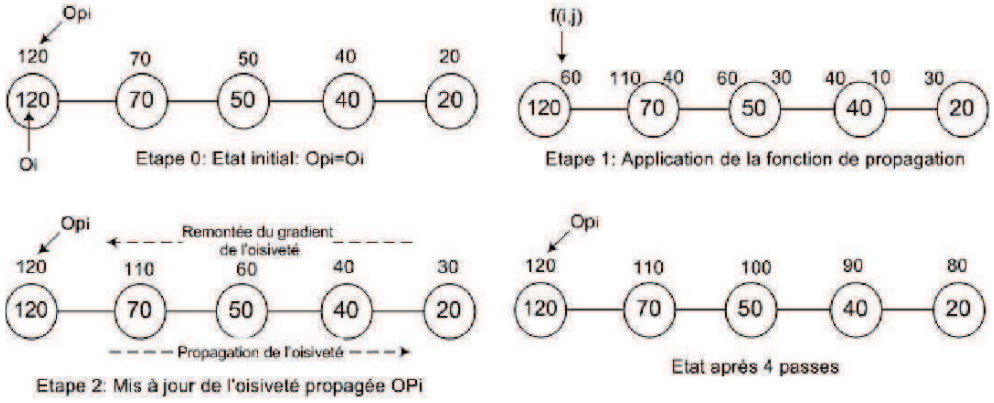


FIGURE 2.4 – Exemple de propagation dans un environnement composé de 5 régions

[Gla11, CGS+07, FTKL09a, FTKL09b] proposent un modèle EVAP [6] pour le problème de la patrouille en environnement inconnu. Ce modèle est basé sur le dépôt et l'évaporation de phéromones. La particularité de ce modèle est d'exploiter uniquement le processus d'évaporation évalué sur la base de l'équation 2.6. L'idée étant de marquer les cellules visitées par une quantité maximale de phéromone q_0 et d'exploiter la quantité restante comme un indicateur du temps écoulé depuis la dernière visite. Ainsi, le comportement d'un agent est défini par une descente du gradient de cette phéromone, c'est-à-dire un comportement menant l'agent à se déplacer vers les cellules contenant le moins de phéromone. Le modèle de déplacement présent dans [CGS+07] a été amélioré dans [FTKL09a, FTKL09b] par un modèle de déplacement à base de force. L'objectif principal des forces est d'une part de tracter les agents vers leurs objectifs locaux et d'autre part de leur permettre d'éviter des obstacles pendant leurs différents déplacements.

$$q_{n+1} = q_n \times (1 - coefEvap) \tag{2.6}$$

$$\text{avec coefEvap} \in [0, 1] \text{ et } q_0 > 0$$

Le comportement de descente de gradient permet aux agents d'explorer les zones les plus anciennement visitées (ou jamais visitées). La perception

6. modèle basé sur l'évaporation des phéromones

de chaque agent se limite aux quatre cellules voisines de sa position courante, et l'agent peut par conséquent lire la quantité de phéromone présente. Il se déplace vers la cellule contenant la valeur minimale parmi les quatre. S'il y a plusieurs cellules voisines contenant la même quantité minimale de phéromone, l'agent choisit aléatoirement où aller parmi ces dernières.

[Gla11, GBSC09] proposent une étude sur le comportement de convergence d'un algorithme fourmi typique à l'instar de VAW[7] ([WLB99]) et $EVAW$ ([GSBC08]) pour le problème de la couverture et de la patrouille. Wagner et al. ([WLB99]) ont introduit les algorithmes fourmis pour la patrouille, dans lesquels chaque agent peut seulement marquer son environnement et s'y déplacer en fonction de ses perceptions locales.
Dans une approche fourmi typique, les agents fourmis ne connaissent pas leur environnement, ne possèdent aucune mémoire, et chaque cellule c_j est dotée d'un marquage $m(c_j)$ appartenant à l'ensemble M des marquages possibles. Les agents sont capables de se déplacer vers une cellule voisine, de marquer leur cellule courante et de percevoir le marquage des cellules voisines. Ce marquage de l'environnement est inspiré des phéromones de fourmis réelles. Le mécanisme de marquage de $VAW/EVAW$ est caractérisé par :

- $M = \mathbb{N}$
- un agent a_i marque sa cellule courante en fixant $m(s(a_i)) \leftarrow t$ (l'instant courant).

Comme illustré à la figure 2.5, et sur la base de l'algorithme 1, à chaque pas de temps un agent EVAW est capable de [GBSC09] :

- se déplacer vers la cellule voisine de plus petit marquage,
- marquer cette cellule avec une valeur correspondant à l'instant courant t.

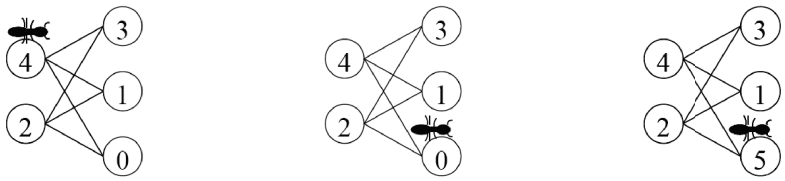


FIGURE 2.5 – Itération de l'algorithme EVAW : a)état courant b)descente (ligne 3) c)marquage (ligne 4), d'après [GBSC09]

Dans l'algorithme EVAW exécuté par un agent, on ne spécifie pas l'implémentation lorsque plusieurs agents l'exécutent en parallèle. L'hypothèse ici est qu'un agent observe son voisinage et se déplace sans qu'aucun autre agent

7. Vertex-Ant-Walk

Algorithme 1 : L'algorithme EVAW exécuté par un agent

```
c ← cellule au hasard dans C ; /* initialisation */
Pour t = 1, ..., ∞ Faire
  c ← arg min_{d∈N(c)} m(d) ; /* descente */
  m(c) ← t ; /* marquage */
FinPour
```

n'agisse en même temps, ce qui est plutôt réaliste en comparaison avec une implémentation sur des robots (qui empêche que deux agents se retrouvent sur la même cellule). Pour permettre à plusieurs agents de s'exécuter quasi-simultanément, une version séquentielle de l'algorithme multi-agent a été développée. On peut alors observer les deux points suivants qui impliquent des choix non-déterministes :

- Conflits : un conflit a lieu quand plusieurs agents *souhaitent* se déplacer vers la même cellule mais que le résultat dépend de celui qui bougera le premier.
- Hésitations : un agent hésite quand il rencontre un choix entre au moins deux cellules dont les dernières visites étaient simultanées (même temps).

Dans le but de résoudre les problèmes posés par l'approche précédente, des améliorations ont été apportées. Dans une simulation, une autre façon de gérer les interactions entre agents est 1) de déplacer tous les agents d'un pas, puis 2) de les laisser tous marquer leur cellule. Cette version *synchrone* d'*EVAW* (tous-les-agents-bougent puis tous-les-agents-marquent) est sans conflits, mais ne modélise pas correctement ce qui arriverait avec des robots puisqu'il est vraisemblable que deux agents se retrouvent ainsi sur la même cellule. Dans un tel cas, les deux agents vont continuer à effectuer les mêmes choix pendant un certain temps (jusqu'à la prochaine hésitation au moins), suivant le même chemin, ce qui produit un mauvais comportement exploratoire.

D'après [GBSC09], pour le cas mono-agent, $EVAW$ converge vers un cycle en temps fini car le nombre d'états accessibles du système est majoré et le comportement de l'agent devient déterministe en temps fini. Pour ce qui est du cas multi-agent, deux comportements sont à examiner : déterministe et non déterministe. Si le système est rendu déterministe, il converge vers un cycle en temps fini. Si le système n'est pas rendu déterministe, son comportement peut être modélisé par une chaîne de Markov sur un espace d'état fini.

[Gla11] s'est intéressé au comportement d'$EVAW$ en environnements irréguliers (Island, Crossword). D'après cette étude, ces environnements n'ont pas

permis à $EVAW$ de converger vers un cycle pendant le temps de simulation. Il n'a pas pu observer de cycle pour un nombre d'agents supérieur à 3 pour le cas de l'environnement Island qui ne comportait pourtant que 50 noeuds. Ainsi, l'efficacité en termes de temps de convergence vers des cycles de l'algorithme $EVAW$ dépend donc fortement de la topologie de l'environnement.

2.4.3 Discussion

Dans cette section, nous avons présenté un état de l'art du domaine de la patrouille en environnements inconnus. Ici, la tâche de patrouille est exécutée sans connaissance de l'environnement par les agents. L'environnement de patrouille est modélisé par une grille et on suppose une impossibilité pour les agents de disposer du graphe représentant l'environnement. Globalement, les approches utilisées sont d'une part basées sur des agents réactifs, ou d'autre part basées sur des agents capables de se coordonner et de s'auto-organiser. Dans les approches à base d'agents réactifs, les différentes stratégies développées ne semblent pas très faciles à utiliser pour des applications robotiques. Que ce soit par l'approche à base de propagation de l'oisiveté ou celle à base de dépôt et évaporation des phéromones, il serait en effet difficile de mettre cela en oeuvre dans le cadre des réseaux de capteurs. D'une part, une propagation de l'oisivité ou des phéromones est un mécanisme difficilement implémentable et une éventuelle possibilité de mise en oeuvre telle que décrite dans ces stratégies n'est pas encore réalisable. En effet, il reste difficile de trouver une substance qui permettrait la mise en oeuvre du processus de propagation de l'oisiveté ou de la phéromone dans un environnement sans créer des problèmes secondaires comme par exemple la pollution. D'autre part, il faudrait équiper les capteurs des dispositifs leur permettant de percevoir ces phéromones, ce qui aurait forcément un impact négatif sur les prix actuellement très compétitifs de ce type de capteurs.
Pour ce qui est des approches à base de coordination et d'auto-organisation, le groupe s'organise autour d'un leader qui se charge d'attribuer et de coordonner l'ensemble des actions du groupe. L'environnement étant inconnu, ces approches demandent un certain temps nécessaire à l'exploration de l'environnement. Selon la complexité de l'environnement, cette tâche d'exploration peut être trop coûteuse voire prohibitive en terme de temps et de ressources.

2.5 Réseaux de capteurs et détection d'évènements

2.5.1 Capteurs

De nombreuses avancées techniques et technologiques dans les domaines de la micro-électronique, de la micro-mécanique, et des technologies de communication sans fil permettent de créer à moindre coût des composants communicants de petites tailles équipés de capteurs. Ces composants appelés capteurs sont équipés d'une ou de plusieurs unité(s) de mesure, d'une unité de calcul, de mémoires et de moyens de communication.. L'alimentation est assurée par une pile, une batterie ou un système de récupération d'énergie dans l'environnement. Cette technologie rend donc possible le déploiement de réseaux de capteurs sans fil.

Qu'est-ce qu'un capteur ?
Un capteur (que nous représentons à la figure 2.6) peut être défini comme un équipement qui à la fois mesure une ou plusieurs donnée(s) et les transmet à un autre capteur pouvant y effectuer des traitements [Gui06, FLKT09]. D'après [Glo07], c'est un équipement qui reçoit une grandeur physique et la transforme en un signal mesurable. Puisque nos capteurs sont mobiles, en plus des capacités de perception, de transmission et de communication, ils doivent être en mesure de se déplacer dans leur environnement. La figure 2.6 illustre un agent capteur avec une représentation de :

- sa zone de communication (r_{com} représente le rayon communicationnel)
- sa zone de perception (r_{per} représente le rayon sensoriel)
- sa surface d'occupation au sol (de rayon r_{occ})

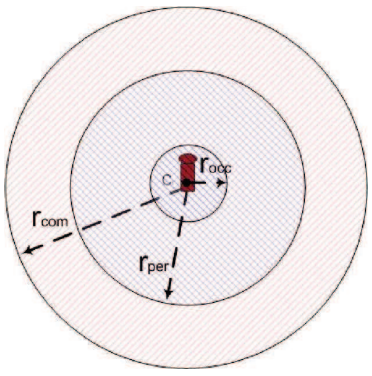


FIGURE 2.6 – Représentation d'un capteur

Un capteur est capable de percevoir et de délivrer des données issues de l'environnement où il est implanté ou du phénomène qui a motivé sa mise en oeuvre. Il en existe plusieurs types, constitués des équipements pouvant permettre une perception vidéo, audio, images fixes, sans oublier les données scalaires (température, humidité, pression, force, position,...) issues de l'environnement. Pour permettre la détection d'évènements (foyers d'incendies, intrusions, pics de pollution, ...), des capteurs à infrarouge ou optique de flamme, des capteurs d'humidité, des capteurs de température ou des détecteurs de fumée peuvent être utilisés. Pour rendre les capteurs mobiles, on peut les intégrer à un véhicule, à un appareil volant de petite taille (drone) que l'on peut piloter d'un centre de surveillance, ou à un appareil de plus grande envergure, comme un avion, un hélicoptère ou un aéronef.

2.5.2 Réseaux de capteurs

L'ensemble des capteurs sont intégrés au sein d'un groupe appelé *réseau de capteurs* (RC). [Aa02] définit un RC comme étant composé d'un ensemble de capteurs, déployés pour l'observation d'un phénomène. Pour être capable de réaliser une observation, un capteur d'un RC doit être en mesure de percevoir tout évènement nouveau qui survient dans son champs de perception et communiquer avec tout capteur situé dans son champs communicationnel. Le déploiement d'un réseau de capteurs nécessite la prise en compte de contraintes liées aux ressources limitées et à la consommation en énergie des capteurs. Dans un réseau de capteurs, un capteur peut être de type *scalaire*[8] ou *multimédia*[9]. Lorsque le RC intègre des capteurs multimédia, on parlera de réseau de capteurs multimédias (RCM). Ainsi, un RCM consiste en une interconnexion d'équipements capables de percevoir de la vidéo, de l'audio, des images fixes et des données scalaires issues du phénomène observé.

Les RCM permettent d'augmenter les domaines d'application des réseaux de capteurs classiques. Les nouveaux domaines d'application incluent [ATC07] :

- surveillance audio-visuelle,
- capacité de stockage des activités pertinentes (violation de trafic, accidents, vols,...) ceci en vue des requêtes futures,
- surveillance et contrôle de trafic,
- télé-médecine : délivrance des soins médicaux avec utilisation d'un ré-

8. Il permet de mesurer des signaux ou données scalaires
9. Il permet d'effectuer des prises audio, vidéo

seau multimédia de type 3G,
- assistance automatique aux personnes âgées,
- surveillance et contrôle environnementale,
- détection et repérage des intrusions,
- contrôle de processus industriels.

Les RCM viennent ajouter au contrôle et surveillance classique de système :
- une vue plus large à travers une répartition distribuée des capteurs audio et visuel,
- une augmentation de la vision à travers différentes vues du même espace ou de la même cible,
- une capacité d'avoir des vues multi-résolutions à travers l'hétérogénéité des supports média qui permet ainsi d'avoir différentes granularités pour un même angle de vue.

L'utilisation d'un réseau de capteurs pour le problème de la patrouille demande le respect d'un certain nombre de contraintes parmi lesquelles les contraintes en énergie, en ressources de calcul et de stockage, et en communication. Dans un réseau de capteurs, un *Sink* possède les ressources suffisantes pour communiquer directement avec le centre de contrôle. Pour que le centre de contrôle soit informé dans les plus brefs délais de l'apparition d'un évènement, les Sinks et les autres capteurs doivent donc se rencontrer le plus souvent possible. Les stratégies de patrouille à déterminer doivent favoriser un échange de données de façon efficace et économique entre les autres capteurs et les Sinks.
Les réseaux de capteurs posent donc un certain nombre de défis sur l'aspect conceptuel, ainsi que des problèmes d'optimisation quant à la localisation, le déploiement, la couverture, le suivi d'un phénomène, et la consommation en énergie.

2.5.3 Problématique dans les réseaux de capteurs

Les réseaux de capteurs ont fait naître de nombreuses problématiques de recherche d'une part, par les applications qu'ils permettent d'envisager, et d'autre part, par les nombreuses contraintes qu'elles imposent. Ils posent un certain nombre de défis sur l'aspect conceptuel. Les *RC* sont utilisés pour résoudre un certain nombre de problèmes parmi lesquel les problèmes d'optimisation sur la localisation et le déploiement [BEGH01, DPG01], la couverture [HT03, FLKT09], la surveillance et le suivi des phénomènes [Fok04, YWM05, Gui06, BsHJG06], la communication [BE02, AKK04] et la consommation en énergie [YF04, YCX06, WCB00].
Un des problèmes fondamentaux dans les réseaux de capteurs est celui du calcul de la couverture [LPJF03, HT03, FLKT09]. D'après [MKQP01], un

des principaux objectifs que l'on voudrait atteindre dans les réseaux de capteurs en matière de couverture, c'est d'allier la localisation ou repérage, le pistage et le déploiement. Il présente le problème de couverture sous deux aspects : déterministe et stochastique. Dans la couverture déterministe, les localisations des capteurs sont prédéfinies et peuvent être uniformes en différents points d'une région. C'est le cas par exemple d'une grille à base de capteurs où les noeuds sont localisés aux points d'intersection de la grille. Dans ce cas, le problème de couverture des capteurs est réduit au problème de couverture d'une cellule et de ses voisins. Dans plusieurs situations, le déploiement déterministe n'est pas applicable. On a alors recours au déploiement stochastique, qui consiste à distribuer de façon aléatoire les capteurs dans un environnement. La distribution aléatoire stochastique peut utiliser un schéma uniforme, Gaussien, Poisson ou toute autre distribution de probabilité la plus adaptée à l'application. [MKQP01] ont supposé un contrôle centralisé au niveau d'un serveur, où les noeuds accèdent via une passerelle. [HT03] formule cela comme un problème décisionnel, le but étant de déterminer si tout point pris à l'intérieur d'une région couverte par un réseau de capteurs est couvert par au moins k capteurs, pour une valeur de k prédéfinie.
[YDCW05] aborde le problème dans le sens de Meguerdichian et al. Il s'intéresse au problème de la meilleure couverture. Ils font usage de la triangulation de Delaunay et de la théorie des graphes dans la recherche de chemin reliant deux points, lequel maximise les plus petites observabilités en tous les points du chemin.

Ces réseaux se trouvent confrontés aux mêmes problèmes que les systèmes distribués et les systèmes embarqués. De part la décomposition, l'allocation, la distribution et la synchronisation des tâches, les systèmes distribués sont difficiles à mettre en oeuvre et pour ce qui est des systèmes embarqués, il faut prendre en compte les interactions fortes entre le logiciel et le matériel. De nombreux systèmes embarqués (téléphones, appareils photo,...) sont contraints en énergie. Cette contrainte est beaucoup plus forte dans les réseaux de capteurs. En plus des contraintes cumulées des systèmes distribués et embarqués, les applications auxquelles sont dédiées les réseaux de capteurs imposent des exigences supplémentaires de fiabilité et surtout d'économie d'énergie [Sam08]. Ainsi, un point important que doit prendre en compte la conception et la mise en oeuvre des RC est la consommation d'énergie [Sam08].

Malgré les contraintes au niveau ressources et énergétiques liées aux RC et au vu des potentialités qu'offrent les réseaux de capteurs, comment peut-on

les exploiter afin de résoudre certains problèmes réels comme par exemple la surveillance d'environnement, la détection des intrusions, des innondations, des feux de brousses ou de forêts ? Afin de résoudre de tels problèmes, nous nous intéressons au problème de la patrouille, où les agents chargés d'effectuer la dite tâche sont des capteurs mobiles.

2.6 Etat de l'art de la patrouille multi-capteurs

La patrouille a pour objectif principal le recueil d'informations et peut être vu comme une extension du problème de couverture [Sem04]. A cause de la proximité de ces deux activités et de la rareté des expérimentations de patrouille non simulées, [Sem04] y fait référence afin d'élargir les perspectives dans le cadre de la coordination en robotique.
Un certain nombre de travaux ont été développés pour l'usage d'approches multi-agents appliquées à la robotique ou avec comme agents patrouilleurs des capteurs, que ce soit dans le cadre de systèmes robotiques réactifs pour le problème d'exploration [Dro93, BHD94, FM98, Sim01, Sem04] ou dans le cadre plus général de robots capables d'opérer de façon autonome.

Dans [PF02], le problème de la patrouille multi-agents est abordé sous l'angle de la coopération et de la coordination. Les auteurs présentent une approche multi-agent résolvant le problème de *poursuite-évasion* pour un ou plusieurs robots mobiles coopératifs possédant une vision omnidirectionnelle. Le principe fondamental est de calculer un *chemin* pour un ou plusieurs robots (agents poursuivants), garantissant qu'un *intrus* ou agent évadé soit nécessairement débusqué ou délogé quels que soient ses déplacements. Le chemin du poursuivant doit à chaque instant, tenir compte des déplacements possibles de l'évadé. Le but des poursuivants est de capturer l'agent évadé en l'encerclant en un minimum de temps. Un agent évadé peut se déplacer dans une zone de l'environnement précédemment explorée par les poursuivants. Les agents *poursuivant* et *évadé* évoluent dans un environnement labyrinthique 2D tel que représenté à la figure 2.7. Comme présenté sur la figure 2.7, les agents *poursuivant* et *évadé* sont modélisés par des points dans un espace euclidien 2D. On suppose par ailleurs que l'agent *poursuivant* connaît l'ensemble de l'environnement.

Les auteurs ont développé une approche basée sur une coopération réelle entre les agents avec prise en compte du partage de connaissances et des mécanismes de coopération. L'approche développée prend aussi en compte la délégation de tâche et la construction d'un état global du système. Les

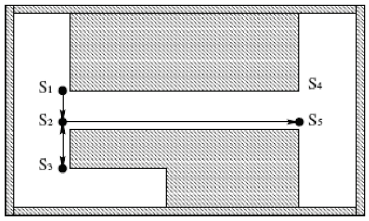


FIGURE 2.7 – Le point représente un agent poursuivant et sa trajectoire $(S_1, S_2, S_3, S_2, S_5)$ dans un espace euclidien 2D contenant des obstacles polygonaux, extraite de [PF02]

auteurs proposent un algorithme pour une patrouille de robots reposant sur une décomposition et une identification des *points critiques* de l'environnement. Si nous considérons un environnement muni d'obstacles, un point de cet environnement est considéré comme critique, si l'angle interne à l'obstacle formé des côtés reliés au point est inférieur à π.
L'algorithme proposé est structuré en quatre étapes et permet de construire une ronde optimale pour un poursuivant en termes de distance parcourue. Le point de départ du poursuivant est choisi arbitrairement. La première étape de l'algorithme permet l'identification des points critiques. A la deuxième étape, l'algorithme construit un graphe dont les noeuds sont les points critiques identifiés lors de la première étape. A partir d'un sommet donné, on crée une arête le reliant à un autre sommet chaque fois que ce dernier est visible depuis le point considéré. La graphe obtenu donne pour tout point critique de l'environnement, les déplacements possibles du poursuivant associé à la distance correspondante. A la troisième étape, l'algorithme permet la construction du graphe de surveillance à partir du graphe de visibilité. La dernière étape de l'algorithme détermine la ronde optimale permettant d'attrapper les évadés dans l'ensemble des points de l'espace qui n'appartiennent à aucun obstacle. Après avoir identifier tous les états finaux du graphe de surveillance, pour chaque état final, l'algorithme de Dijsktra est appliqué afin de déterminer la ronde optimale reliant les sommets finaux à l'état initial.
La mise en oeuvre de la coopération passe par l'identification des points de délégation. On appelle point de délégation un point critique qu'un poursuivant ne peut quitter sans risquer le passage d'un évadé. Il faut donc déléguer à cet endroit la tâche de surveillance à autre poursuivant. Au point de délégation, les robots doivent être capables de se concerter dans le but de décomposer la tâche collective de recherche d'évadé en un ensemble de sous-tâches. Un autre aspect à prendre en compte est la synchronisation entre robots. Un poursuivant qui détecte un point de délégation peut demander de l'aide aux autres poursuivants dont la tâche est terminée. Les poursuivants dont la tâche est

finie n'ont plus de contrainte et peuvent être réalloués à une autre tâche.
Le problème de la détermination du nombre minimum de poursuivants pour lequel il existe une trajectoire garantissant qu'un évadé s'il existe, sera trouvé en fonction d'un environnement donné est un problème NP-difficile [PF02]. Ainsi, l'optimisation de la ronde globale des poursuivants reste un problème ouvert. Pour les auteurs, il serait également intéressant d'étudier la mise en oeuvre de cette approche dans le cas de poursuivants possédant une vision restreinte de leur environnement.

[LGSC08] propose une approche qui permet à un opérateur de faire voler en simulation des drones et de réaliser des missions de surveillance et d'interception. La stratégie utilisée repose sur la tâche de patrouille par une auto-organisation des drones, qui a pour but de faire émerger des parcours de surveillance en fonction de leurs interactions avec l'environnement. Pour réaliser la tâche de patrouille, les auteurs adaptent le modèle EVAP ([CGS+07]) qui exploite uniquement le processus d'évaporation. Contrairement aux agents du modèle EVAP qui se déplacent dans un environnement modélisé par une grille, les drones ici se déplacent en coordonnées réelles, selon un cap et une vitesse donnée. Les auteurs ont conservé la matrice pour le dépot et la perception des phéromones. Les agents disposent d'un disque de perception et sont en mesure d'effectuer un choix parmi les cellules appartenant à ce disque. Ils couplent aux drones un réseau de capteurs, avec des capteurs dispersés sur l'environnement ou placés sur la frontière. Ces capteurs peuvent déclencher une alerte, en déposant une quantité de phéromone d'alarme. Lorsqu'une alarme est déclenchée, l'activité de l'opérateur change. Il passe d'une activité de surveillance à celle de poursuite des potentiels intrus qui auraient pénétré dans le périmètre. Sur une tâche ponctuelle, localisée et urgente comme l'interception d'intrus, l'intervention de l'opérateur apporte un important gain de performance. Les opérateurs ont été par exemple capables de positionner les drones sur les trajets présumés des intrus afin de les intercepter. Une fois positionnés, les drones sont bien guidés par la phéromone, mais s'ils la consomment sans détecter d'intrus, ils continuent leur tâche de surveillance. Suite aux difficultés exprimées par les opérateurs, une amélioration peut consister à l'introduction de nouveaux modes de contrôle dédiés à l'interception. Il serait par exemple intéressant d'introduire la possibilité de restreindre un sous-groupe d'agents à une certaine zone géographique tout en maintenant leur comportement autonome. Cela reviendrait à ajouter une composante de contrôle par politique au système. De telles zones pourraient même évoluer en fonction de la vitesse et de la direction probable d'un intrus.

2.7 Synthèse

Dans ce chapitre, nous avons présenté le problème de la patrouille ainsi que ses objectifs. Une formulation du problème de la patrouille a été présentée. Nous avons ensuite dressé un état de l'art sur le problème de la patrouille multi-agents, aussi bien en environnements connus qu'en environnemens inconnus. Quelque soit le type d'environnement, dans une approche de patrouille par un réseau de capteurs mobiles, les Sinks et les autres capteurs doivent se rencontrer le plus souvent possible pour que le centre de contrôle puisse récolter fréquemment les informations éparses issues des capteurs et réagir en conséquence à l'apparition éventuelle d'évènements. Tel que nous l'avons vu, l'utilisation des RC mobiles dans la tâche de patrouille impose le respect de contraintes inhérentes aux réseaux de capteurs.
Le chapitre qui suit présente une formulation du problème de détection d'évènements par un réseau de capteurs. Il sera suivi dans le chapitre 4 par la proposition d'une approche permettant d'obtenir des solutions approchées au problème ainsi formulé.

Chapitre 3

Formulation du problème de la patrouille multi-capteurs appliquée à la détection d'événements

3.1 Introduction

Dans le précédent chapitre, nous nous sommes intéressés au problème de la patrouille ainsi qu'à ses objectifs. Une formulation du problème de la patrouille multi-agents telle que rencontrée dans la littérature a été présentée. Nous allons dans le présent chapitre, et en particulier dans la prochaine section, traiter de la formulation du problème de patrouille lorsqu'un réseau de capteurs mobiles est utilisé. Puisque le problème met en exergue plusieurs objectifs, nous l'aborderons sous l'angle de l'optimisation multi-objectif. La section 3.3 sera dédiée à la présentation des principes sous-jacents à l'optimisation multi-objectif. Y seront présentés également la notion d'optimalité au sens de Pareto et le concept de dominance. Nous conclurons le chapitre par une synthèse.

3.2 Le problème de la patrouille multi-capteurs en tant que problème d'optimisation multi-objectif

L'objectif de cette section est de formaliser le problème de la patrouille multi-capteurs lorsque les capteurs sont utilisés pour une tâche de détection

d'évènements. Seront ainsi précisés les données du problème, les fonctions objectifs ainsi que les inconnues.

3.2.1 Présentation informelle du problème

La figure 3.1 présente un exemple d'environnement à surveiller où doit être déployé un groupe de capteurs composé d'un seul *Sink* pour la détection au plus tôt des évènements. Le scénario est le suivant :

- Un ensemble de capteurs est déposé dans une région de l'environnement.
- Cet ensemble de capteurs doit patrouiller dans l'environnement dans l'attente d'évènements (sources d'incendie, intrusions,...).
- Lorsqu'un évènement est détecté par des capteurs, ceux-ci acheminent l'alerte au *Sink* le plus proche, qui les transmet au centre de contrôle dans le but d'une intervention rapide.

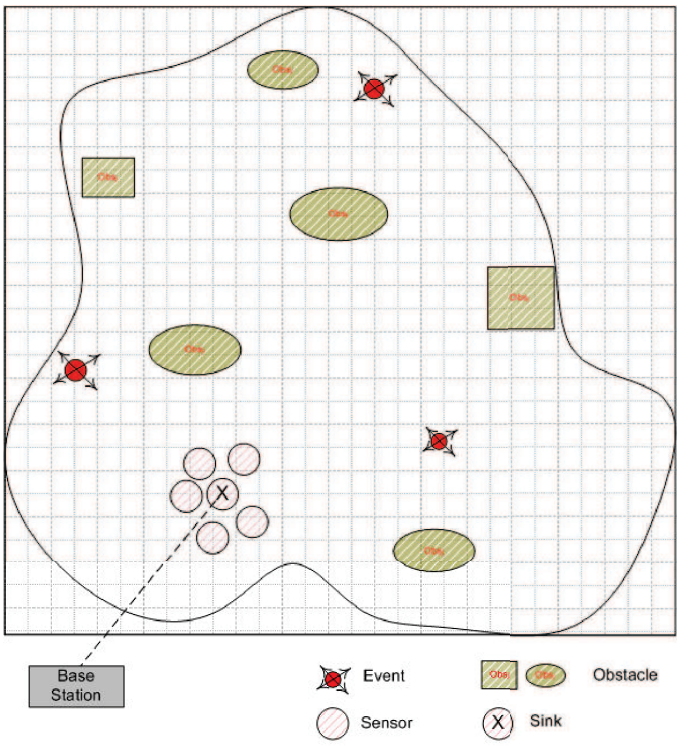


FIGURE 3.1 – Exemple d'environnement à surveiller

L'environnement dans lequel les agents doivent patrouiller peut-être décrit par un graphe $G = (V, E)$, où V représente l'ensemble des régions critiques de l'environnement et E l'ensemble des connexions entre ces régions.

Dans la figure ci-dessus, le graphe à patrouiller peut être obtenue en discrétisant l'environnement en cellules. Chaque cellule devient un noeud du graphe et des arcs sont crées pour permettre de passer d'un noeud vers un des 8 noeuds voisins.

3.2.2 Formalisation du problème

Le problème de la patrouille multi-capteurs en environnement connu peut être défini par le tuple $P = (G, N, S)$. $G = (V, E)$ est le graphe à patrouiller, N est la liste des noeuds de départ des $r = |N|$ capteurs, avec $r \leq |V|$. S est l'ensemble des capteurs Sinks qui peuvent transmettre directement les alertes d'événements au centre de contrôle.

Le problème à résoudre est celui de la détermination d'une stratégie de patrouille π permettant de minimiser la pire oisiveté du graphe, la pire oisiveté communicationnelle du réseau de capteurs, ainsi que l'énergie totale consommée par le réseau.

3.2.3 Critère de pire oisiveté du graphe

Rappelons que la pire oisiveté représente le plus grand délai de passage sur l'ensemble des noeuds du graphe. En d'autres termes, la valeur de la pire oisiveté du graphe est égale au plus grand intervalle de temps durant lequel un noeud ne reçoit la visite d'aucun agent. Intuitivement, minimiser la pire oisiveté revient à minimiser le temps de détection ou de perception d'un évènement sur l'ensemble du graphe, tandis que dans la minimisation de la moyenne oisiveté, la différence entre la plus petite et la plus grande oisiveté peut être significative, pouvant entraîner ainsi un délai de visite très long pour certains noeuds. Par conséquent, nous choisissons d'évaluer chaque stratégie de patrouille potentielle sur la base d'une minimisation de la pire oisiveté du graphe, car celle-ci semble plus adaptée que celle basée sur la minimisation de la moyenne oisiveté du graphe.

3.2.4 Critère de consommation d'énergie du réseau

Dans le problème de *patrouille multi-capteurs*, les agents sont des capteurs mobiles sans fil. Par conséquent nous devons prendre en compte les problématiques liées à leurs consommations en énergie. La quantité d'énergie consommée par un capteur est fonction de plusieurs paramètres : le temps de déplacement, le temps de communication, le temps de perception et le temps de calcul.

L'énergie consommée entre les instants $t-1$ et t par un capteur i, lorsque

celui-ci suit la stratégie de patrouille individuelle π_i de la stratégie de patrouille globale $\pi = (\pi_1, \cdots, \pi_i, \cdots, \pi_r)$, peut s'exprimer par l'équation :

$$E_i^{\pi}(t) = E_{com_i}^{\pi}(t) + E_{comp_i}^{\pi}(t) + E_{mov_i}^{\pi}(t) + E_{percept_i}^{\pi}(t) + E_{idle_i}^{\pi}(t) \qquad (3.1)$$

où $E_{com_i}^{\pi}(t)$, $E_{comp_i}^{\pi}(t)$, $E_{mov_i}^{\pi}(t)$, $E_{percept_i}^{\pi}(t)$ et $E_{idle_i}^{\pi}(t)$ représentent respectivement l'énergie de communication, de calcul, de déplacement, de perception et d'inactivité du capteur i pour la stratégie π entre les instants $t-1$ et t. Ces énergies sont exprimées en Watt-heure.

Dans [SdVS04], l'expression de l'énergie est proportionnelle à la quantité de trafic passant par chaque capteur. L'expression de l'énergie que nous considérons prend en compte les trafics locaux à chaque capteur, ce qui permet une évaluation de l'énergie consommée à chaque instant pour chaque capteur. Ainsi, l'expression 3.2 permet d'évaluer l'énergie consommée en communication du capteur i pour une stratégie de patrouille individuelle π_i entre les instants $t-1$ et t.

$$E_{com_i}^{\pi}(t) = e_i^{rec} \times \sum_{j \in v_i^t} p_{j,i}^t + e_i^{env} \times p_i^t \qquad (3.2)$$

où
e_i^{rec} représente l'énergie consommée par le capteur i lors de la réception d'un paquet.
e_i^{env} représente l'énergie consommée par le capteur i lors de l'émission d'un paquet.
$p_{j,i}^t$ représente le nombre de paquets relayés/retransmis par le capteur i en provenance des capteurs en communication directe avec i, entre les instants $t-1$ et t.
p_i^t représente le nombre de paquets partant du capteur i (le capteur i est considéré comme la source) vers les capteurs en communication directe avec i, entre les instants $t-1$ et t.
v_i^t représente l'ensemble des capteurs en communication directe avec le capteur i entre les instants $t-1$ et t.

L'expression 3.3 permet l'évaluation de l'énergie consommée en terme de calcul du capteur i pour une stratégie de patrouille π entre les instants $t-1$ et t.

$$E_{comp_i}^{\pi}(t) = n_{opa}^{i,t} \times e_{opa} + n_{opp}^{i,t} \times e_{opp} \qquad (3.3)$$

où
$n_{opa}^{i,t}$ et $n_{opp}^{i,t}$ représentent le nombre d'opérations respectivement d'addition/-soustraction et de multiplication/division effectuées par le capteur i entre les

instants $t-1$ et t.
e_{opa} et e_{opp} représentent l'énergie nécessaire pour réaliser une opération respectivement d'addition/soustraction et de multiplication/division.

[WIP+05] définit l'expression 3.4, qui permet l'évaluation de l'énergie consommée par un capteur mobile en mouvement sur une distance d. Dans cette expression, $P(t)$ représente la puissance consommée par un moteur. L'expression suppose aussi que le capteur détient deux moteurs.

$$E(d) = 2\int_0^T P(t)dt \tag{3.4}$$

où
T représente le temps de mouvement.

S'inspirant de l'expression 3.4, nous exprimons l'évaluation de l'énergie de déplacement par l'expression 3.5. Dans cette expression, $w_i^{j,t}$ indique pour une stratégie de patrouille individuelle π, le travail moyen effectué par le moteur j du capteur i entre les instants $t-1$ et t. M_i indique le nombre de moteurs du capteur i.

$$E^{\pi}_{mov_i}(t) = \sum_{j=1}^{M_i} w_i^{j,t} \tag{3.5}$$

L'expression 3.6 exprime pour une stratégie de patrouille individuelle π_i, l'évaluation de l'énergie de perception du capteur i. k_i^t représente le nombre de perceptions réalisées par le capteur i entre les instants $t-1$ et t. $e_i^{percept/m}$ représente l'énergie consommée par un capteur i lorsque le rayon de perception augmente d'une unité de longueur (d'un mètre). R_i représente le rayon de perception du capteur i.

$$E_{percept_i}(t) = k_i^t \times e_i^{percept/m} \times R_i \tag{3.6}$$

L'expression 3.7 exprime pour une stratégie de patrouille individuelle π_i, l'évaluation de l'énergie consommée par le capteur i pendant son inactivité entre les instants $t-1$ et t. Lorsque le capteur n'est ni en communication, n'effectue aucun calcul, aucun déplacement et ne fait aucune perception, il peut tout de même ajuster ou effectuer la rotation de ses bras et autres dispositifs, il peut aussi réorganiser ses données internes. e^t_{ajust} et e^t_{reorg} représentent les énergies consommées associées respectivement à l'exécution de l'une des tâches citées précédemment et α et β sont des coefficients multiplicateurs.

$$E^{\pi}_{idle_i}(t) = \alpha e^{t}_{ajust} + \beta e^{t}_{reorg} \tag{3.7}$$

Un des objectifs visés est de parvenir à minimiser la somme des énergies consommées par l'ensemble du réseau. Pour une stratégie de patrouille $\pi = (\pi_1, \cdots, \pi_r)$, nous exprimons l'énergie consommée par le réseau de capteurs par l'expression E^{π} ci-dessous.

$$E^{\pi} = \sum_{t=0}^{T} \left(\sum_{i=1}^{r} E^{\pi}_{i}(t) \right) \tag{3.8}$$

L'expression 3.8 permet d'évaluer l'énergie totale consommée par l'ensemble du réseau pendant toute la durée de simulation T lorsque chacun des r capteurs i suit sa stratégie individuelle π_i.

3.2.5 Critère de pire oisiveté communicationnelle

La détection des évènements par un groupe de capteurs mobiles demande, en plus de pouvoir réaliser la tâche de patrouille, de donner une possibilité de perception et de transmission au plus tôt des évènements perçus. Cette dernière considération nécessite, en plus des critères définis dans les sous-sections précédentes, de pouvoir minimiser un autre critère. Ce critère est le délai maximum de communication entre les Sinks et les autres capteurs de son groupe, et par conséquent entre le RC et la station de contrôle.

Définissons le délai maximum de communication entre un Sink et les autres capteurs du groupe par la notion d'*oisiveté communicationnelle*. L'oisiveté communicationnelle d'un capteur représente le nombre de pas de temps écoulé depuis la dernière communication entre un Sink et un capteur lorsqu'un événement a été détecté. Il est à noter que la communication entre un Sink et les capteurs de son groupe peut être directe ou indirecte. Lorsqu'elle est directe, un Sink se trouve dans le champs communicationnel du capteur. Une communication indirecte signifie que la communication entre un capteur et un Sink se fait par capteurs intermédiaires, formant ainsi une chaîne communicationnelle.
Posons $CI^{\pi}_{i}(t)$ l'oisiveté communicationnelle du capteur i au temps t lorsque celui-ci suit la stratégie de patrouille individuelle π_i faisant partie de la stratégie de patrouille globale π. Cette oisiveté reste à 0 pour chaque capteur $i \notin S$ qui n'est pas un Sink tant qu'il n'y a pas d'évenements. Elle s'incrémente pour chaque capteur $i \notin S$ dès l'apparition d'un événement. La pire oisiveté communicationnelle peut s'exprimer par l'équation suivante :

$$CI^{\pi} = \max_{k=1,\cdots,K} \max_{t=0,\cdots,T_k} \max_{i \notin S} CI_i^{\pi}(t) \tag{3.9}$$

où :

- K désigne le nombre de simulations effectuées pour avoir une évaluation fiable de l'oisiveté communicationnelle d'une stratégie. En effet, l'oisiveté communicationnelle dépend de l'instant où un évènement est apparu ainsi que de sa position. Une estimation de cette oisiveté sera d'autant plus précise qu'un nombre important de simulations sera réalisées, en faisant donc varier à chaque simulation la position (le noeud) et l'instant où l'évènement se déclare sur le graphe.
- T_k représente la durée totale de la k-ème simulation. T_k peut aussi représenter la durée au bout de laquelle tous les Sinks ont atteint leur seuil critique d'énergie pendant la k-ème simulation.

3.2.6 Synthèse des fonctions objectifs du problème

Le problème de la patrouille multi-capteurs consiste à déterminer une stratégie de patrouille π qui minimise les fonctions objectifs suivants :

$$\begin{cases} WI^{\pi} = \max_{t=0}^{T} \left(WI_t^{\pi}(G)\right)) = \max_{t=0}^{T} \left(\max_{i=1,\ldots,|V|} I_i^{\pi}(t)\right) \\ CI^{\pi} = \max_{k=1,\cdots,K} \max_{t=0,\cdots,T_k} \max_{i \notin S} CI_i^{\pi}(t) \\ E^{\pi} = \sum_{t=0}^{T} \left(\sum_{i=1}^{r} E_i^{\pi}(t)\right) \end{cases}$$

où :

- $I_i^{\pi}(t)$ est l'oisiveté du noeud i à l'instant t lorsque les capteurs suivent la stratégie de patrouille π.
- WI^{π} représente la pire oisiveté du graphe G à l'issue de la simulation.
- CI^{π} représente une estimation du délai maximum de communication entre les capteurs et les Sinks dès l'apparition d'un événement.
- E^{π} représente l'énergie totale consommée par l'ensemble du réseau.
- S représente l'ensemble des capteurs Sinks présents dans le réseau.
- T représente la durée totale de la simulation lorsque l'évaluation porte sur la pire oisiveté du graphe WI^{π} ou la consommation d'énergie E^{π}.
- $|V|$ représente le nombre de noeuds de l'environnement.
- r représente le nombre de capteurs du réseau.

3.3 Principes de l'optimisation multi-objectif

Nous résolvons au quotidien des problèmes d'optimisation plus ou moins complexes. La plupart des problèmes d'optimisation réels sont décrits à l'aide de plusieurs objectifs (ou critères) souvent contradictoires devant être optimisés simultanément [Bar03]. Nos déplacements, nos achats, notre organisation, sont la plupart du temps définis après avoir réfléchi au préalable aux multiples options dont nous disposons pour aboutir à la décision qui nous semble la plus appropriée.
Considérons l'exemple suivant inspiré d'une situation de tous les jours. Imaginons que l'on désire se déplacer d'une ville A vers une ville B.
On est généralement amené à vouloir minimiser le temps mis sur le parcours, tout en cherchant à emprunter le trajet le moins coûteux possible et le plus confortable possible. Une illustration est donnée à la figure 3.2, où nous supposons un déplacement en voiture.

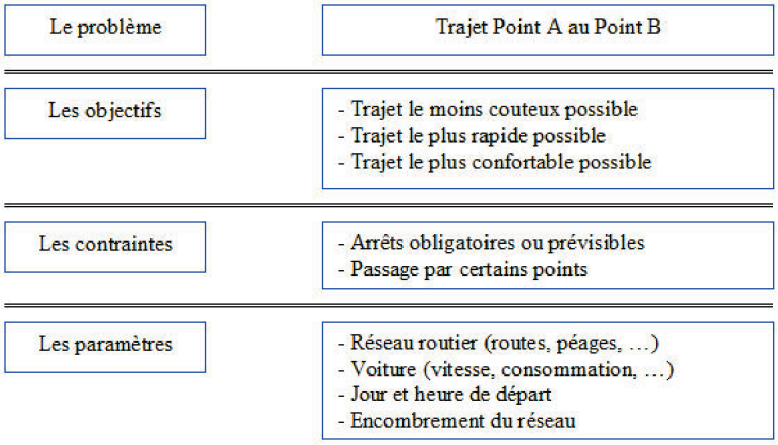


FIGURE 3.2 – Illustration d'un problème multi-objectif, inspirée de [Jé03] p.28

Comme le montre la figure 3.2, il y a différents objectifs à satisfaire. Il est à remarquer par exemple que le trajet le plus rapide n'est pas toujours le plus confortable. De même, le trajet le moins coûteux n'est pas toujours le plus confortable. Il faudra donc trouver un compromis entre ces différents objectifs. L'atteinte des objectifs doit prendre aussi en compte les contraintes et les paramètres liés au parcours.
Ces raisonnements, même s'ils paraissent anodins, font appel au concept de compromis, en ce sens que les décisions prises le sont rarement dans un contexte d'objectif unique. Généralement, plusieurs critères sont simultanément intégrés à la réflexion, afin de dégager un choix final présentant le

meilleur compromis entre tous les objectifs. Cela nous amène donc à examiner une catégorie de problèmes d'optimisation : les problèmes multi-critères ou multi-objectifs.

3.3.1 Définition et formulation

Un problème multi-objectif (PMO) ou multi-critère peut être défini comme un problème dont on recherche les paramètres qui satisfont un ensemble de contraintes et optimisent un vecteur de fonctions objectifs ([Ber01, Tei01, Bar03, Lie09]) ou plusieurs composantes d'un vecteur de coût ([BTNA06]). L'optimisation multi-objectif permet d'étudier les problèmes réels qui font intervenir de nombreux critères (souvent conflictuels) et contraintes. Dans ce contexte, la solution optimale recherchée n'est plus constituée d'un unique point, mais d'un ensemble de bons compromis satisfaisant toutes les contraintes. Ainsi, un problème d'optimisation multi-objectif sous contraintes peut être défini comme suit :

$$\begin{cases} \min\ f_i\,(X)\ \text{i=1,2,...,N} \\ \text{contraintes}\ g_j\,(X) \leq 0\ \text{j=1,2,...,m} \\ X \in \Re^n \end{cases}$$

N représente le nombre d'objectifs (ou critères), m le nombre de contraintes du problème, n représente le nombre de variables de décision et $X = (x_1, ..., x_n)$ représente le vecteur de décision.

Dans la formulation précédente, nous ne considérons que les contraintes d'inégalité. Sans perte de généralité, remarquons qu'une contrainte d'égalité de type $g(X) = 0$ est équivalente à deux contraintes d'inégalité $g(X) \leq 0$ et $-g(X) \leq 0$. Remarquons aussi que tout problème défini en terme de maximisation se ramène aisément à la formulation précédente moyennant quelques transformations mathématiques.

L'optimisation multi-objectif est le processus d'optimisation simultanément de deux ou plusieurs objectifs contradictoires sous réserve de certaines contraintes [Tei01, Bar03].

La qualité des solutions d'un problème multi-objectif peut être liée au concept de dominance, introduit ci-dessous.

3.3.2 Règles de dominance

Un problème de maximisation peut être aisément transformé en un problème de minimisation en considérant que maximiser une fonction f revient à

minimiser la fonction $-f$ dans l'espace de recherche. Ainsi, en optimisation multi-critère, on va chercher simultanément à minimiser plusieurs critères. Il convient donc de définir une relation d'ordre sur l'ensemble des solutions. Dans le cas des problèmes d'optimisation multi-objectifs, ces relations d'ordre sont appelées relations de dominance [Bar03]. On distingue plusieurs types de dominance, parmi lesquelles la dominance faible, la dominance stricte [Lie09], la $\epsilon-dominance$ [Pad09, WZ09], la $cône-dominance$ [CS02b], la dominance de Pareto [Lam00, Bar03, Lie09], etc. La dominance de Pareto a été introduite avec l'idée de distinguer les solutions potentiellement intéressantes des autres, avec le principe suivant : une solution potentiellement intéressante est une solution telle qu'on ne peut pas améliorer la performance sur un critère sans dégrader la performance sur au moins un autre critère [Gal08]. C'est cette relation de dominance que nous allons définir et utiliser dans cette thèse.

Définition 3.3.1 (Dominance)**.** *Soit* $X = (x_1, x_2, ..., x_n)$ *et* $Y = (y_1, y_2, ..., y_n)$ *deux vecteurs de décision. Y domine X (cette relation sera notée* $Y \succ X$*) dans un problème de minimisation si et seulement si :*

$$\begin{cases} \forall i \in 1, \ldots, N & f_i(Y) \leq f_i(X) \\ \exists j \in 1, \ldots, N & f_j(Y) < f_j(X) \end{cases}$$

La solution d'un problème multi-objectif est un ensemble des meilleurs compromis, appelé aussi surface de compromis ou front de Pareto. Il est composé des points qui ne sont dominés par aucun autre [Bar03, Gal08]. La détermination de cette surface de compromis utilise la relation de dominance entre les différents couples de solutions. Dans le but d'établir les relations de dominance entre les différents couples de solutions, une matrice de dominance D de taille $|V| \times |V|$ peut être construite, où V représente l'ensemble des solutions réalisables dans l'espace décisionnel. Cette matrice exprime pour un couple (X, Y) de solutions la dominance entre X et Y. Pour chaque couple de solutions (X, Y), une conclusion est émise quant à la nature de X et de Y, selon les règles suivantes [Jé03] :

$$\left| \begin{array}{l} D[X,Y] = 0 \\ D[Y,X] = 0 \end{array} \right\} \textit{implique} \text{ que X et Y sont équivalentes et on note } X \equiv Y.$$

$$\left| \begin{array}{l} D[X,Y] = 1 \\ D[Y,X] = 0 \end{array} \right\} \textit{implique} \text{ que X domine Y et on note } X \succ Y.$$

$$\left|\begin{array}{l} D\left[X,Y\right]=0 \\ D\left[Y,X\right]=1 \end{array}\right\} \textit{implique} \text{ que Y domine X et on note } Y \succ X.$$

Il est donc possible d'extraire de la matrice D le nombre $N_{X\succ}$ de solutions dominées par X. Ce nombre est défini par l'expression 3.10, où N_{sol} représente la cardinalité de l'espace ou de l'ensemble des solutions.

$$N_{X\succ} = \sum_{Y=1}^{N_{sol}} D\left[X,Y\right] \tag{3.10}$$

De façon similaire, on définit aussi le nombre de solutions $N_{\succ X}$ qui domine une solution X donnée. Ce nombre est défini par l'expression 3.11 :

$$N_{\succ X} = \sum_{Y=1}^{N_{sol}} D\left[Y,X\right] \tag{3.11}$$

Dans cette sous-section, nous avons montré comment construire la matrice de dominance D. Nous avons aussi montrer comment extraire de la matrice de dominance D le nombre $N_{X\succ}$ de solutions qu'une solution X domine. Les solutions $Pareto-optimales$ sont identifiables dans la matrice D par le fait que la colonne qui leur est associée ne possède que des termes nuls. Autrement dit, une solution X est Pareto-optimale si et seulement si $N_{\succ X} = 0$.

3.3.3 Le Front de Pareto

Le *front de Pareto* FP^* est l'ensemble des vecteurs de décision qui ne sont pas dominés [Bar03, GB08, Gal08]. FP^* est défini plus formellement par l'équation 3.12

$$FP^* = \{x \in V \mid \nexists y \in V,\ y \succ x\} \tag{3.12}$$

V représente l'ensemble des solutions réalisables dans l'espace décisionnel. Ainsi, le front de Pareto est l'ensemble des solutions de compromis.

Considérons un problème multi-objectifs formé de deux fonctions $g1$ et $g2$ à minimiser. Sans perte de généralité, supposons que l'ensemble des solutions réalisables dans l'espace décisionnel est donné par $\{A, B, C, D\}$, comme le montre la figure 3.3. Pour une solution donnée, l'axe des abscisses permet de projecter les valeurs associées à la composante $g1$, tandis que sur l'axe des ordonnées la composante $g2$ est projectée.

Nous présentons une illustration du front de Pareto à la figure 3.3. Aucun des points A, B et C ne domine l'autre et par conséquent l'ensemble formé des points A, B et C sont trois points du front de Pareto. Le point D n'appartient pas au front de Pareto car dans l'ensemble formé des points A, B et C, aucun point ne domine l'autre mais tous les trois dominent le point D.

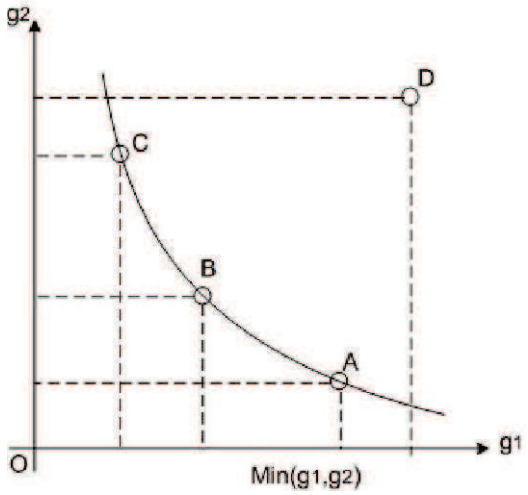


FIGURE 3.3 – Exemple de front de Pareto

Une solution X est considérée comme un optimum global au sens de Pareto s'il n'existe aucune solution Y qui domine X dans l'espace des objectifs. Une solution X est dite localement optimale au sens de Pareto s'il n'existe, dans le voisinage de X, aucune solution Y dominant X. Le voisinage d'une solution peut être défini comme une restriction de l'espace complet des objectifs.

3.4 Difficultés des problèmes d'optimisation multi-objectif

La difficulté principale de l'optimisation multi-objectif est liée à la présence de conflits entre les divers objectifs. En effet, les solutions optimales, pour un objectif donné, ne correspondent généralement pas à celles des autres objectifs pris indépendamment. De ce fait, il n'existe la plupart du temps aucun point de l'espace de recherche où toutes les fonctions objectifs sont optimales simultanément.
Une autre difficulté liée au problème multi-objectif est qu'il n'existe pas de définition de la solution optimale [Ber01]. De ce fait, le décideur peut sim-

plement exprimer le fait qu'une solution est préférable à une autre sans qu'il n'existe pour autant une solution meilleure que toutes les autres. La résolution d'un problème multi-objectif consiste alors à chercher un ensemble de solutions satisfaisantes et non la solution optimale, le choix final revenant au décideur.
D'après [Ber01], deux approches sont généralement utilisées pour solutionner les *PMO*. La première consiste à ramener un problème multi-objectif en tant que problème mono-objectif, en aggrégeant l'ensemble des objectifs en un seul objectif. Un premier risque lié à cette transformation est que le problème à résoudre peut être déplacé à celui du choix des paramètres d'aggrégation. Un autre risque est que le problème obtenu peut faire perdre tout son sens au problème initial. Puisque l'aggrégation intervient avant la résolution, un objectif peut primer sur les autres. La deuxième approche apporte des solutions au problème en considérant l'ensemble des objectifs sans appliquer d'aggrégation.
Dans la première approche, le décideur intervient au début de la définition du problème pour exprimer ses préférences afin d'effectuer une transformation du problème multi-objectif pour le mono-objectif. La transformation d'un problème multi-objectif en un problème mono-objectif nécessite des connaissances a priori du problème traité. L'optimisation d'un problème mono-objectif peut garantir l'optimalité de Pareto de la solution trouvée mais permet de trouver une seule solution. Les approches par aggrégation sont sensibles au paysage de la frontière de Pareto (convexité, discontinuité, etc.)[Meu02].
Dans la deuxième approche, les préférences sont appliquées dans l'ensemble des solutions proposées. Cette deuxième approche ne fixe pas les solutions comme la première, mais permet de fournir un ensemble de solutions offrant au décideur une plus vaste gamme d'actions à entreprendre ou à appliquer. Ainsi, l'une des principales difficultés liée à la prise en compte des préférences sur un problème d'optimisation multi-objectif est la subjectivité de ces mêmes préférences.

En plus des problèmes liées à la présence des contraintes, une autre difficulté rencontrée lors de la résolution d'un *PMO* est liée aux propriétés même des fonctions à optimiser. En fonction des problèmes à traiter, le front de Pareto peut avoir une configuration très complexe (nous pouvons citer : la convexité, la continuité, etc) [Dip10]. La convexité est le premier indicateur de la difficulté d'un problème multi-objectif [Bar03]. En effet, la plupart des méthodes ne permettent pas de résoudre des problèmes non convexes de manière optimale. Mais il existe d'autres indicateurs tout aussi importants, notamment la continuité, la multimodalité, la nature des variables de déci-

sion, etc [Bar03].
Ces indicateurs sur la compléxité d'un PMO sont décrits plus en détails ci-dessous :

Convexité : Un front de Pareto est dit convexe si, étant donné deux points distincts quelconques de ce front, le segment qui relie ces deux points est toujours contenu dans l'ensemble des solutions atteignables de l'espace des objectifs. La non convexité d'un espace entraîne l'existence de zones d'optimalité locale pouvant piéger les algorithmes de résolution. Les performances des métaheuristiques diffèrent suivant que la structure de la frontière Pareto est convexe ou concave. L'utilisation des méthodes d'agrégation est plus efficace lorsque la structure de la frontière est convexe que dans le cas où la frontière est concave [Meu02].

Continuité : La discontinuité d'un front peut également complexifier la résolution d'un problème multi-objectif. Cette discontinuité peut être liée à la présence des contraintes, ou à l'existence de zones optimales locales. Dans le premier cas, l'existence des contraintes peut créer le phénomène de discontinuité du front de Pareto, tandis que dans le second cas, la discontinuité du front de Pareto est principalement due à la forme même du front. La figure 3.4 présente dans un espace à deux objectifs (F_1, F_2), des exemples des fronts de Pareto présentant des discontinuités. On peut observer d'une part sur cette figure un front de Pareto avec une discontinuité due aux contraintes et d'autre part un autre front de Pareto avec une discontinuité due à la forme même du front.

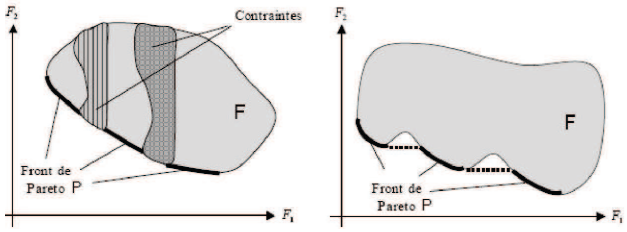


FIGURE 3.4 – Exemples des fronts de Pareto discontinus, extraite de [Jé03]

Multimodalité : La multimodalité du front de Pareto découle de l'existence de solutions localement Pareto-optimales [Meu02]. Les problèmes multi-modaux sont caractérisés par l'existence de plusieurs front locaux successifs, qui sont susceptibles d'empêcher la détermination du front

global. Ces fronts locaux agissent comme des attracteurs qui peuvent piéger les méthodes de résolution. La figure 3.5 présente dans un espace à deux objectifs (F_1, F_2), un exemple de front de Pareto avec un caractère multimodal. On peut observer sur cette figure des fronts de Pareto locaux et un front de Pareto global.

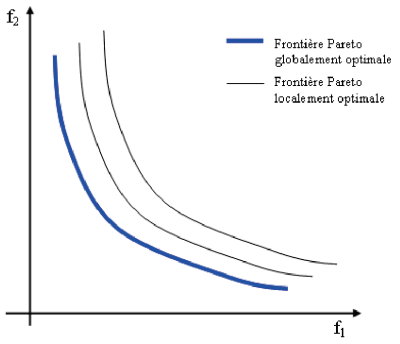


FIGURE 3.5 – Exemple de front de Pareto à caractère multimodal

Non uniformité des solutions : Un problème d'optimisation multi-objectif peut être caractérisé par une distribution non uniforme des solutions dans l'espace des objectifs. En effet, certaines zones de l'espace des objectifs peuvent présenter des densités de solutions plus importantes que d'autres. De ce fait, si les densités les plus fortes se situent loin du front optimal, les méthodes de résolution vont rencontrer plus de difficultés pour atteindre les solutions globales.

3.5 Etat de l'art des techniques d'optimisation multi-critères

Dans la littérature, on rencontre deux grandes classifications des méthodes de résolution des problèmes multi-objectifs. La première classification est orientée vers un point de vue utilisateur (le décideur). Dans la deuxième classification rencontrée dans la littérature, les approches sont structurées en fonction de leur façon de résoudre les fonctions objectifs.

En considérant la première classification, les principales méthodes existantes pour traiter les problèmes multi-objectifs peuvent être classées en trois grandes familles [Vel99, DF05, Lie09]. Ces trois familles de méthodes de ré-

solution de problèmes multi-objectifs sont fonction du moment où intervient le décideur pour le choix final de la solution à garder [DF05] :

- les méthodes dites a priori, pour lesquelles l'utilisateur définit le compromis qu'il désire réaliser avant de lancer la méthode de résolution. Elles transforment le problème multi-critère en un problème mono-critère, en pondérant l'ensemble des critères initiaux. Le compromis que l'on désire faire entre les objectifs est défini avant l'exécution de la méthode. La solution recherchée est obtenue en une seule exécution. Cette approche est rapide, mais il faut cependant prendre en compte le temps de modélisation du compromis ainsi que la possibilité pour le décideur de ne pas être satisfait de la solution trouvée.
- les méthodes progressives ou interactives, pour lesquelles l'utilisateur affine son choix des compromis au fur et à mesure du déroulement de l'optimisation. Le décideur intervient de façon à modifier certaines variables ou contraintes afin de diriger le processus d'optimisation. Comme indiqué dans [CS02a], ces méthodes ont l'inconvénient de monopoliser l'attention du décideur tout au long du processus d'optimisation. Cet aspect peut être pénalisant si l'évaluation des fonctions objectives est longue et que les sollicitations imposées au concepteur sont fréquentes.
- les méthodes dites a posteriori, pour lesquelles il n'est plus nécessaire, pour le concepteur, de modéliser ses préférences avant l'optimisation. Ces méthodes se contentent de produire un ensemble de solutions directement transmises au décideur. Il pourra alors a posteriori choisir une solution de compromis parmi celles obtenues lors de la résolution du problème.

La popularité actuelle des méthodes a posteriori résulte de l'utilisation des métaheuristiques capables d'explorer plusieurs solutions en parallèle et d'exploiter pleinement le concept de dominance au sens de Pareto, qui permet de s'affranchir des problèmes de mise à l'échelle des critères. Pour ces méthodes, deux phases importantes sont à considérer : la phase de recherche de l'ensemble des solutions Pareto optimales et la phase de choix de la (ou des) meilleure(s) solutions parmi ces solutions, ce qui relève de l'aide à la décision [DF05].

Pour ce qui est de la deuxième classification, les approches sont articulées en deux catégories : les approches non Pareto et les approches Pareto [Bar03]. Les approches non Pareto ne traitent pas le problème comme un véritable problème multi-objectif. Elles cherchent à ramener le problème initial à un ou plusieurs problèmes mono-objectifs. Par opposition aux méthodes non

Pareto, les approches Pareto ne transforment pas les objectifs du problème, ceux-ci sont traités sans aucune distinction pendant la résolution [Bar03]. Une articulation plus détaillée de cette classification est affinée avec [Ber01], qui propose une structure en trois principales méthodes à savoir les méthodes dites *agrégées*, puis celles fondées sur *Pareto*, et enfin les méthodes *non agrégées et non Pareto* .

3.5.1 Les méthodes agrégées

Parmi les techniques utilisées dans les méthodes agrégées, nous pouvons citer :
La moyenne pondérée qui consiste à additionner tous les objectifs en affectant à chacun d'eux un poids. Ce poids représente l'importance relative que le décideur accorde à l'objectif. Si on suppose un problème à n objectifs f1,f2,...,fn, le problème mono-objectif résultant peut être formulé par :
$min \sum_{i=1}^{n} w_i f_i(x)$ avec $w_i \geq 0$
w_i représente le poids affecté au critère i et $\sum_{i=1}^{n} w_i = 1$

Malgré la simplicité de mise en oeuvre, les difficultés rencontrées lors de la mise en oeuvre de cette approche sont les suivantes : d'une part, comment déterminer le poids de chaque objectif ? et d'autre part, comment définir l'interaction entre les différents objectif ?
Il faut tout de même mentionner que la somme pondérée se heurte à deux problèmes majeurs [Ber01] : la sensibilité à un changement d'échelle et à la compensation entre critères.

La sensibilité à un changement d'échelle : Le changement d'échelle est une transformation couramment utilisée en mathématique. S'il est utilisé dans une somme pondérée, il peut avoir un impact sur l'ordre des différentes décisions. L'exemple ci-dessous inspiré de [MPS94, Ber01], nous en donne une illustration.
Considérons trois actions (1,2 et 3) évaluées en fonction des critères coût et quantité de produit. On leur associe les poids respectifs de 3/5 et 2/5. Les colonnes *évaluation 1* et *évaluation 2* présentent les résultats des évaluations $coût \times 3/5 + produit \times coût0 \times 2/5$, où produit est exprimé respectivement en tonne et en kilogramme, coût0 représente le coût unitaire par produit que nous supposons ici égal respectivement à 100F et 10F. Comme illustré dans le tableau ci-dessous, une conversion des poids de tonne (t) à kilogramme (kg) dans la somme pondérée entraîne une inversion des rangs des actions.

TABLE 3.1 – Impacts du changement d'échelle sur la décision

	coût(F)	produit(t)	évaluation 1	évaluation 2	Rang1	Rang2
action 1	9000	60	7800	245400	3	1
action 2	6000	80	6800	323600	2	2
action 3	3000	90	5400	361800	1	3

La compensation entre critères : Une valeur basse d'un critère peut être compensée par les valeurs des autres critères, entraînant ainsi un mauvais choix par le décideur.

La méthode du *goal programming* est une méthode agrégée [CC61, Vel99], où le décideur fixe un but à atteindre pour chaque objectif. Ces valeurs sont ensuite ajoutées au problème comme des contraintes supplémentaires. La nouvelle fonction objectif est modifiée de façon à minimiser la somme des écarts entre les résultats et les buts à atteindre [Ber01]. Associons à chaque objectif f_i un but à atteindre T_i. Cette fonction objectif est alors définie par $min \sum_{i=1}^{k} |f_i(x) - T_i|$
Tout comme pour la somme pondérée, la méthode est facile à mettre en oeuvre. la définition des poids et des objectifs à atteindre est une question délicate qui détermine l'efficacité de la méthode.

3.5.2 Les méthodes Pareto

L'idée de base est que dans un problème multi-objectif, il existe un équilibre tel que l'on ne peut pas améliorer un critère sans détériorer au moins un des autres critères. Cet équilibre est encore appelé optimum de Pareto [Ber01]. Il s'agit donc d'utiliser le concept d'optimalité de Pareto afin de respecter l'intégralité de chaque critère [Gol89].
Dans la résolution des problèmes multi-objectif, la relation d'ordre entre les différents éléments de l'ensemble solution n'est pas totale, c'est-à-dire qu'une solution peut être meilleure qu'une autre sur certains objectifs et moins bonne sur les autres [Kam06]. C'est ainsi qu'on parle des solutions de compromis.

[FF93] propose la méthode MOGA (Multiple Objective Genetic Algorithm) qui utilise la dominance de Pareto pour rechercher les solutions d'un problème multi-objectif, où chaque individu de la population est rangé en

fonction du nombre d'individus qui le domine. Les auteurs utilisent une fonction de notation permettant de prendre en compte le rang de l'individu et le nombre d'individus ayant le même rang.

[GGP04a] propose de résoudre un *PMO* d'ordonnancement multi-objectif. Les auteurs décrivent les adaptations requises à l'optimisation à base des colonies de fourmis de sorte qu'elle puisse être utilisée pour obtenir des solutions de compromis. Les auteurs montrent comment solutionner efficacement un problème d'ordonnancement industriel multi-objectif en tenant compte des préférences énoncées par le décideur.
Dans la résolution du problème portant sur l'ordonnancement multi-objectif [GGP02, GPG02], trois objectifs ont été considérés selon un rangement *lexicographique* et l'*OCF* (Optimisation par Colonie de Fourmis) a démontré son efficacité pour modéliser et résoudre efficacement ce problème. Cette approche *lexicographique* d'aborder le problème a toutefois le désavantage de diriger la recherche dans une direction extrême et de considérer les autres objectifs seulement en cas d'égalité.
La recherche de solutions de compromis entre différents objectifs représente donc une direction intéressante à explorer. Toutefois, dans certaines situations pratiques, les temps de calculs peuvent s'avérer important si l'on cherche à générer entièrement l'ensemble Pareto-optimal [GGP04a]. II s'avère alors préférable de demander au décideur de préciser ses préférences pour les différents objectifs et de produire exclusivement les solutions Pareto-optimales correspondant à celles-ci. C'est ainsi que [GGP04b] propose une procédure générique pouvant être adaptée à différentes métaheuristiques pour la recherche de solutions de compromis, tout en faisant une application de la procédure à la recherche tabou (*Tabu Search*) hybridée avec la recherche par voisinage variable (*Variable neighborhood Search*) afin de traiter un problème d'ordonnancement bi-objectifs sur une machine unique avec réglages dépendant de la séquence.

3.5.3 Les méthodes non agrégées et non Pareto

Ces méthodes possèdent en général un processus de recherche qui traite séparément les différents objectifs. Parmi ces méthodes, nous pouvons citer :

Vector Evaluated Genetic Algorithm (VEGA)
VEGA est une extension d'un algorithme génétique simple pour la résolution d'un problème multi-objectif [Sch85, Cha05, Ber08]. Elle diffère d'un algorithme génétique simple par la façon dont s'effectue le sélection. Si nous considérons k objectifs et une population de n individus, on associe à chaque

objectif une sélection de n/k individus. On obtient ainsi k sous-populations contenant chacune n/k meilleurs individus pour chaque objectif. Ces k sous-populations sont melangées dans le but d'obtenir une nouvelle population de taille n. Les opérateurs génétiques de modification (croisement et mutation) sont appliqués enfin au processus.
La méthode VEGA a tendance à créer des sous-populations dont les meilleurs individus sont spécialisés pour un objectif particulier. L'évolution favorise ainsi de l'apparition des espèces. Cette méthode est très souvent utilisée car elle est facilement adaptable dans un algorithme génétique classique [Ber01].

Méthode lexicographique
Une telle méthode consiste à ranger au préalable tous les objectifs. Ces méthodes possèdent en général un processus de recherche qui traite séparément et séquentiellement les différents objectifs en fonction de leur ordre d'importance. Les objectifs précédemment traités deviennent des contraintes pour les prochains objectifs. [Fou85] propose une approche différente de cette méthode, avec une sélection aléatoire de la fonction objectif à traiter. Cela équivaut à une somme pondérée dans laquelle un poids correspond à la probabilité de sélection de la fonction objectif. Bien que les résultats qu'ils obtiennent soient satisfaisants, le risque essentiel ici est la grande importance attribuée aux objectifs classés en premier [Ber01]. La conséquence est que l'objectif le plus important produit une solution qui conditionne la convergence vers une zone restreinte de l'espace.

A Non Generational Genetic Algorithm
Les systèmes de classifieurs sont des mécanismes d'apprentissage basés sur un ensemble des règles condition/action [Ber01]. Chaque règle est notée en fonction du résultat de l'action produite et un algorithme génétique est utilisé pour générer de nouvelles règles [Gol89]. Un classifieur ([ZHLA02]) peut être utilisé comme un outil de reconnaissance qui reçoit une forme en entrée et donne des informations à propos de la classe de cette forme. C'est ainsi qu'une combinaison de classifieurs permet de fiabiliser la reconnaissance en utilisant la complémentarité qui peut exister entre les différents classifieurs [ZHLA02].

Un problème multi-objectif a des caractéristiques semblables à celles des systèmes classifieurs [Gol89]. Un classifieur possède une caractéristique principale appelé force, qui représente son adaptation à son environnement. Cette force évolue dans le temps de façon incrémentale en fonction des récompenses obtenues lors de réponses positives à des stimuli [Ber01]. Ce comportement

est transposé dans [VUC97], où des individus cherchent la frontière de Pareto d'un problème multi-objectif. L'algorithme développé est structuré en deux parties. La première partie s'occupe de la mise à jour de la fitness des individus. Dans un algorithme génétique, la fitness est la fonction d'évaluation d'un individu. La deuxième partie de l'algorithme permet de créer de nouveaux individus après sélection, croisement et mutation. L'évaluation de chaque individu est basé d'une part, sur l'évaluation de la dominance de l'individu au cours du temps, et d'autre part, sur l'évaluation du voisinage de l'individu par une fonction de calcul de la distance entre deux individus. La fonction de fitness de chaque individu est alors une combinaison linéaire de ces deux valeurs. Les auteurs réduisent ainsi un problème multitobjectif à un problème à deux objectifs à savoir minimiser la dominance, et puis minimiser l'importance de la niche (ensemble d'individus situés dans un espace restreint).
Cette méthode permet d'obtenir néanmoins de bons résultats. L'utilisation de nombreux paramètres en plus de ceux liés à l'algorithme génétique, rend difficile le réglage de l'algorithme.

Méthode élitiste
L'idée ici est d'utiliser les solutions non dominées trouvées à chaque génération et de la stocker dans une population externe. [IM96] couple à cela une méthode de recherche locale pour générer de meilleurs individus. L'utilisation d'une population externe pour stocker les individus non dominés et d'une recherche locale apporte à cette méthode une capacité élitiste très importante. Bien que la notion de dominance soit utilisée dans [IM96], la méthode développée par les auteurs ne saurait être classée dans les méthodes Pareto car la notion de dominance n'est pas utilisée dans le processus de sélection.

3.5.4 Discussion

Nous avons présenté dans cette section un état de l'art des techniques d'optimisation multi-critères. Nous distinguons globalement les approches Pareto et non Pareto. Les approches non Pareto transforment un problème d'optimisation multi-objectif en un ou plusieurs problèmes à un seul objectif. Cette transformation permet d'utiliser facilement les méthodes d'optimisation issues de l'optimisation à un objectif. Ainsi, les aproches non Pareto ont l'avantage de pouvoir réutiliser tous les algorithmes classiques dédiés aux problèmes d'optimisation à un seul objectif [Bar03]. L'utilisation de ces approches demande aussi que les objectifs soient commensurables, c'est-à-dire

qu'ils soient exprimés dans la même unité. Une telle approche ne peut être utilisée si dans l'ensemble des objectifs, on retrouve à la fois des objectifs qualitatifs et quantitatifs.

Contrairement aux approches non Pareto, les approches Pareto prennent relativement bien en compte les situations où il existe des objectifs qualitatifs et quantitatifs. Elles permettent également de prendre éventuellement en compte la modélisation des préférences du décideur.

3.6 Synthèse

Dans ce chapitre, nous avons proposé une formulation du problème de la patrouille multi-capteurs appliquée à la détection d'évènements en tant que problème d'optimisation multi-objectif. Les données du problème, les différentes fonctions objectifs ainsi que les inconnues ont été précisées. Nous avons aussi présenté le principe de l'optimisation multi-objectif, tout en faisant ressortir les difficultés qui y sont liées. Un état de l'art des techniques de résolution de problèmes multi-objectifs a également été présenté. Le chapitre qui suit propose une approche permettant d'obtenir des solutions approchées au problème de la patrouille multi-capteurs pour la détection d'événements.

Chapitre 4

Approche de résolution du problème de la patrouille multi-capteurs pour la détection d'évènements

4.1 Introduction

Nous proposons l'approche nommée *AMCPMC*, basée sur l'Optimisation par Colonies de Fourmis (OCF), pour résoudre de manière approchée le problème de la patrouille multi-capteurs formulé au chapitre précédent. Nous supposerons que l'environnement est connu et modélisé par un graphe. Le problème ici est celui de la détermination d'une stratégie de patrouille permettant de minimiser la pire oisiveté des noeuds du graphe, l'oisiveté communicationnelle des capteurs, ainsi que l'énergie totale consommée par le réseau de capteurs.
Nous supposerons l'existence de capteurs particuliers, appelés *Sinks*, au sein du réseau de capteurs. Ces capteurs sont capables de communiquer directement avec le centre de contrôle. Un Sink doit être en communication avec les autres capteurs du réseau le plus souvent possible pour garantir une réactivité suffisante du réseau face aux événements détectés. Les communications entre un Sink et les autres capteurs sont indispensables, parce qu'un Sink doit recenser les évènements perçus par l'ensemble du réseau afin de les transmettre au centre de contrôle. Les communications entre capteurs sont nécessaires afin de propager jusqu'à un Sink, des alertes associées aux évènements perçus localement.
Tel que mentionné dans l'un des chapitres précédents, les ressources des cap-

teurs sont limitées entre autres par leurs rayons communicationnels. Pour qu'une information arrive au centre de contrôle, il doit transiter par un Sink, d'où la nécessité de minimiser l'oisiveté communicationnelle afin que cette transmission soit aussi rapide que possible.

L'approche *AMCPMC* recherche, parmi l'ensemble des stratégies π possibles, la stratégie π^* qui minimise un certain ensemble de critères. En optimisation mono-critère, on écrit souvent : $\pi^* = \text{argmin}_{\pi \in \Pi} C^\pi$ où C est le critère à minimiser et Π l'ensemble des stratégies π possibles. Comparativement à l'expression utilisée en optimisation mono-critère, nous adopterons pour l'optimisation multi-critères l'écriture $\pi^* = \text{argmin}_{\pi \in \Pi} \{WI^\pi, CI^\pi, E^\pi\}$ où WI, CI et E représentent les critères à minimiser.

Dans ce chapitre, nous commencerons par présenter les caractéristiques et principes des approches *OCF*[1]. Ensuite, nous aborderons la description de notre approche de résolution. La section 4.4 clôturera ce chapitre par une synthèse.

4.2 Caractéristiques et principes des approches OCF

Les approches OCF sont inspirées du comportement collectif des fourmis. Ce comportement permet aux fourmis des colonies d'optimiser le chemin entre la nourriture et la fourmilière. Chaque fourmi d'une colonie, pendant le processus de construction de son chemin, dépose une trace de phéromone qui sera utilisée par les fourmis de leur colonie. Grâce aux dépôts successifs des phéromones et au phénomène d'évaporation, les fourmis empruntent progressivement un chemin plus court.
Dans l'optimisation par colonies de fourmis considérée, la recherche de nouvelles solutions est effectuée par les différentes colonies. La mémoire qui guide cette recherche correspond à une matrice de phéromone. Celle-ci est utilisée par les colonies pour construire de manière incrémentale les solutions. Une solution (stratégie) possible correspond à un ensemble de sous-solutions déterminées par chaque fourmi de la colonie.

Dans la communauté multi-agents, on distingue classiquement deux grandes catégories d'architecture : les architectures cognitives et les architectures réactives [Mic04, Dav08]. Dans une architecture cognitive, les décisions des

1. Optimisation par Colonies de Fourmis ou Ant Colony Optimization (*ACO*)

agents sont motivées par des buts explicites tout en faisant ressortir une certaine forme de rationalité basée sur une représentation symbolique ou logique du monde. Par contre, dans une architecture réactive, les agents sont purement réactifs car ils sont vus comme des entités qui ne font que réagir de manière mécanique aux stimuli qu'ils perçoivent. Si dans l'approche cognitive l'intelligence est attribuée aux agents individuels, l'approche réactive soutient que l'intelligence est une propriété globale du système multi-agent. L'approche par colonies de fourmis s'inscrit dans le contexte des approches réactives où l'intelligence est obtenue par les comportements d'agents simples en interaction. Ces interactions sont toujours effectuées à travers l'environnement, qu'elles soient directes [2] ou indirectes [3] [Gla11, Sim10]. Les agents réactifs n'ont pas de représentation de l'environnement ni des autres agents, et souvent peu de mémoire. Leurs actions et leurs perceptions sont purement locales [Dro05]. De ce fait, la multiplicité des interactions et leur caractère stochastique permettent d'obtenir, par auto-organisation et émergence de structures, des propriétés collectives robustes et adaptatives. Un exemple de cette approche est l'*optimisation par colonie de fourmis*. Dans ce système, chaque fourmi a un comportement simple, mais les interactions entre fourmis, réalisées par l'intermédiaire de phéromones, permettent de produire progressivement une solution proche, voire optimale.
Ces algorithmes s'inspirent des comportements collectifs de dépôt et de suivi de piste observés dans les colonies de fourmis. Une colonie d'agents simples (les fourmis) communiquent indirectement via des modifications dynamiques de leur environnement (les pistes de phéromone) et construisent ainsi une solution à un problème, en s'appuyant sur leur expérience collective.
Les techniques d'optimisation issues des colonies de fourmis (Ant Colony Optimization ou ACO en anglais) ont été appliquées à divers problèmes d'optimisation comme la coloration de graphes [CH97], le problème du voyageur de commerce [DG97], et aussi le problème de tournées de véhicules qui s'applique généralement sur les noeuds d'un réseau [BHS99b, BHS99a, RSD02].

[LPT03] présente la résolution du problème de tournée d'arcs à capacité par une approche OCF. Le CARP (Capacitated Arc Routing Problem) est un problème de tournées NP-difficile défini sur un graphe non orienté G = (V,E) où V est un ensemble de n noeuds et E un ensemble de m arêtes.

2. Les interactions sont directes lorsqu'un agent s'adresse directement à un autre. Il est à noter que ces interactions sont réalisées à travers l'environnement qui sert alors de canal de communication.

3. Les interactions sont indirectes lorsque les agents déposent l'information dans l'environnement. Cette information pourra être lue et exploitée par tout autre agent capable de la percevoir.

Une des contraintes est qu'on peut traverser une arête plusieurs fois mais son traitement doit être effectué par un seul véhicule et ceci lors d'une seule traversée. Le CARP consiste à déterminer un ensemble de tournées (cycles passant par le dépôt) tel que :

- chaque arête à traiter est effectivement traitée par une seule tournée ;
- la demande totale traitée par chaque tournée ne dépasse pas la capacité Q du véhicule ;
- le coût total des tournées (somme des coûts des arêtes traversées) est minimisé.

L'objectif des travaux qu'ils ont mené est de prouver que pour la résolution du dit problème, les algorithmes de fourmis sont capables de concurrencer d'autres méthodes à l'instar de la méthode tabou, le recuit simulé. Les temps de calcul pour une itération dans l'approche OCF a conduit les auteurs à réaliser un très faible nombre d'itérations par rapport au nombre d'itérations réalisées par l'algorithme génétique hybride de [LPRC01]. Cette grande différence dans le nombre d'itérations explique partiellement la dominance de l'algorithme génétique par rapport à l'algorithme de fourmis. Les résultats obtenus sur le CARP montrent que ces algorithmes fournissent des résultats comparables en qualités à ceux obtenus avec des métaheuristiques issues du recuit simulé [LPT03].

[ASG05] propose une résolution du problème du sac-à-dos multidimensionnel (*MKP* pour Multidimensional Knapsack Problem). Ce problème est NP-difficile et possède de nombreuses applications pratiques comme l'allocation des processeurs dans les systèmes distribués, le chargement des cargaisons, le découpage de stocks. L'objectif du MKP est de trouver un sous-ensemble d'objets qui maximise un profit total tout en satisfaisant certaines contraintes de capacité. [ASG05] propose un algorithme ACO générique pour résoudre le problème du sac-à-dos multidimensionnel. Pour les auteurs, cet algorithme est générique dans le sens où il est paramétré par les composants sur lesquels les traces de phéromone sont déposées. Ils ont proposé trois instanciations différentes de cet algorithme générique correspondant à trois façons de déposer et d'exploiter les traces de phéromone : *Vertex-AK* où la phéromone est déposée sur les objets, *Path-AK* où la phéromone est déposée sur les couples d'objets sélectionnés consécutivement, et *Edge-AK* où la phéromone est déposée sur les paires d'objets sélectionnés dans une même solution. L'idée est de construire des solutions de façon incrémentale, par ajouts successifs d'objets à une solution partielle. A chaque itération, l'objet à ajouter est choisi selon une probabilité dépendant de traces de phéromone et d'une information heuristique locale. Les auteurs étudient l'influence de la phéromone sur le processus de résolution. Le dépôt de phéromone sur tous les

couples de sommets sélectionnés (*Edge-AK*) permet d'obtient de meilleurs résultats comparativement aux autres.

Contrairement aux approches OCF qui utilisent une seule colonie pendant le processus d'optimisation, d'autres par contre utilisent plusieurs colonies [GTA99, PFM07, LC06, LK08]. Nous nous sommes intéressés à l'approche multi-colonies pour plusieurs raisons : premièrement, elle offre une plus grande flexibilité en terme de possibilités de déploiement des fourmis. Deuxièmement, elle permet d'obtenir une multiplicité des solutions potentielles au problème au cours du processus d'optimisation. Troisièmement, elle permet d'obtenir plus rapidement de bonnes solutions, en raison de la compétition instaurée au niveau des colonies.
Une limite liée à l'approche multi-colonies est l'augmentation du temps d'exécution nécessaire pour trouver une solution comparativement à une approche mono-colonie. On note aussi l'utilisation de beaucoup plus de ressources (mémoire et processeur) due d'une part au nombre et à la taille des listes tabou, ainsi qu'à la taille du graphe de patrouille, et d'autre part, le nombre des fourmis et colonies influence considérablement sur le temps d'exécution.
Dans un contexte multi-colonies, un certain nombre de difficultés vont être rencontrées. Pour un problème donné, comment définir le nombre de colonies approprié et le nombre de fourmis par colonie. Combien de graphes doivent être utilisés (un par colonie, un pour toutes les colonies), faut-il utiliser plusieurs types de phéromones (un par colonie) et si oui, comment ces phéromones peuvent-elles être transmises d'une colonie à l'autre pour améliorer la résolution, combien faut-il de listes tabou, etc.

Une approche multi-colonies est développée dans [LC06, LK08] afin de résoudre le problème multi-agents de la patrouille. Les auteurs considèrent que l'environnement de patrouille est modélisé par un graphe $G = (V,E)$, où V est l'ensemble des sommets du graphe et E l'ensemble des arêtes. Dans [LC06, LK08], l'approche OCF est utilisée afin de déterminer par simulation les stratégies à appliquer aux différents agents patrouilleurs. Les auteurs procèdent à un déploiement de différentes colonies. La meilleure colonie est sélectionnée et correspond à celle qui minimise la pire oisiveté du graphe. Les fourmis de la meilleure colonie déposent les phéromones sur les noeuds du graphe. Ce dépôt de phéromone amènera les autres fourmis des autres colonies à marquer et à suivre probablement la solution de la meilleure colonie. Les auteurs ont étudiés deux approches de départ des agents. Dans [LC06], les agents partent d'un même noeud initial, tandis que dans [LK08], une étude comparative est faite entre le départ d'un même noeud initial et celui à partir des différents noeuds du graphe (dispersion des agents à travers

le graphe). La stratégie par dispersion initiale des agents permet d'obtenir globalement de meilleurs résultats expérimentaux de patrouille.

Nous présentons à la figure 4.1, une synthèse de l'approche multi-colonie développée dans [LC06, LK08]. L'initialisation correspond à une mise à zéro de la matrice des phéromones.

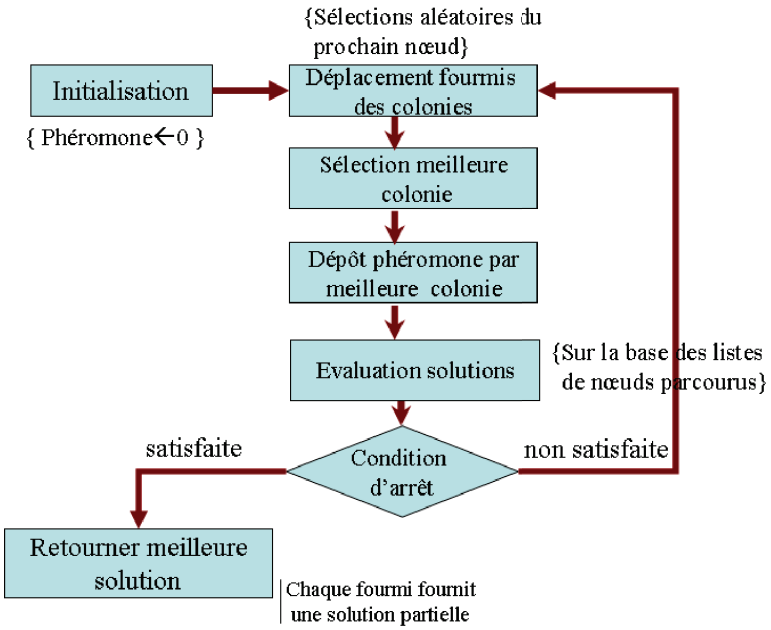


FIGURE 4.1 – Schéma général OCF

Après la phase d'initialisation, on procède par la suite à un déploiement des différentes colonies. Le déplacement des fourmis des différentes colonies est basé sur une sélection aléatoire du prochain noeud à visiter. La probabilité de sélection d'un noeud par la fourmi k de la colonie l est définie par l'expression de $p_{ij}^{k,l}$ ci-dessous :

$$p_{ij}^{k,l}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha}\ [\eta_{ij}]^{\beta}}{\sum_{u \in autorisé_l} [\tau_{iu}(t)]^{\alpha}\ [\eta_{iu}]^{\beta}} & \text{if } j \in autorisé_l \\ 0 & \text{sinon} \end{cases} \tag{4.1}$$

où
$autorisé_l = \{V - \sum_{i=1}^{r} tabou_{i,l}\}$ représente l'ensemble des noeuds non visités par les fourmis de la colonie l, et $tabou_{i,l}$ représente l'ensemble des noeuds déjà visités par la fourmi i de la colonie l.

V représente l'ensemble des noeuds du graphe.
$\tau_{ij}(t)$ représente l'intensité des phéromones sur l'arête (i, j) au cycle t.
$\eta_{ij} = 1/c_{ij}$ indique la visibilité du noeud j par rapport au noeud i, c'est-à-dire l'inverse de la distance c_{ij} entre les noeuds i et j.
α et β sont considérés comme des paramètres pour le contrôle respectivement de l'intensité des phéromones et de la visibilité.

La mise à jour de l'intensité des phéromones est défini par l'équation 4.2.

$$\tau_{ij}(t+1) = \rho\, \tau_{ij}(t) + \Delta_{\tau_{ij}} \tag{4.2}$$

où
$1 - \rho$ est le coefficient d'évaporation.
$\tau_{ij}(t+1)$ et $\tau_{ij}(t)$ représentent les intensités des phéromones sur l'arête (i,j) au temps de cycle t+1 et t, respectivement.
$\Delta_{\tau_{ij}}$ est la quantité de phéromone déposée sur l'arête (i,j) par toutes les fourmis de la colonie à ce cycle.

Après un déplacement sans dépôt des phéromones, on procède par la suite à un dépôt de phéromone par la meilleure colonie. Lors de leur tournée, l'ensemble des fourmis de la meilleure colonie dépose une quantité de phéromone $\Delta_{\tau_{ij}}$ sur les arêtes (i,j) du graphe visité par les fourmis de la meilleure colonie. L'équation 4.2 traduit cette mise à jour des phéromones. L'objectif de ce dépôt est d'amener les autres fourmis des autres colonies à suivre éventuellement l'itinéraire de la meilleure colonie.
L'objectif principal visé ici est de déterminer une stratégie de patrouille qui minimise la pire oisiveté du graphe. Pour cela, à chaque colonie correspond une solution potentielle au problème, c'est-à-dire une stratégie de patrouille multi-agent. Lors de la récupération d'une solution, chaque fourmi fournit une partie de la solution au problème. L'assemblage des différentes sous-solutions permet d'avoir la solution au problème original.
Contrairement aux approches de résolution du problème classique de la patrouille multi-agents qui consiste à déterminer une stratégie de patrouille qui minimise la pire oisiveté environnementale, nous nous sommes intéressés à la détermination des stratégies de patrouille applicables dans le contexte des réseaux de capteurs. Les stratégies déterminées doivent permettre de minimiser la pire oisiveté environnementale, l'oisiveté communicationnelle et l'énergie consommée par l'ensemble du réseau. C'est ainsi que nous proposons dans la suite de ce chapitre notre procédure de résolution du problème de la patrouille multi-capteurs.

4.3 Approche de résolution du problème de la patrouille multi-capteurs

Parmi les approches de résolution de problèmes multi-objectifs présentées dans le chapitre précédent, nous avons opté pour une méthode *a posteriori*. L'une des principales raisons est que ces méthodes exploitent pleinement le concept de dominance au sens de Pareto, permettant ainsi de s'affranchir des problèmes de mise à l'échelle des critères. De plus, contrairement aux méthodes progressives, elles ne monopolisent pas l'attention du décideur tout au long du processus d'optimisation. Nous présentons ici notre approche : *AMCPMC*(Approche Multi-Colonies pour les problèmes de Patrouille Multi-Capteurs). Cette approche consiste à utiliser une approche par Optimisation à base des Colonies de Fourmis (OCF) pour la résolution du problème de la patrouille multi-capteurs.
Pour concevoir l'approche de résolution *AMCPMC*, nous avons émis l'hypothèse que la consommation d'énergie du réseau de capteurs est principalement liée aux déplacements, qui peuvent être minimisés lorsque la pire oisiveté du graphe l'est. En d'autres termes, minimiser la pire oisiveté du graphe revient à minimiser indirectement la consommation globale en énergie du réseau lorsque la consommation en énergie du réseau causée par les communications, les calculs, les perceptions et l'inactivité est négligeable par rapport à celle causée par les déplacements. Par conséquent, seulement deux critères seront minimisés : la pire oisiveté du graphe et l'oisiveté communicationnelle des capteurs.

4.3.1 Structures des stratégies de patrouille considérées

Avant de détailler l'algorithme *AMCPMC*, nous voudrions apporter une précision quant à la structure même des stratégies de patrouille déterminées par notre algorithme. Nous supposons qu'une stratégie de patrouille multi-capteurs est définie par $\pi = (\pi_1, \cdots, \pi_r)$. Chaque stratégie individuelle π_i est composée d'un cycle de patrouille et éventuellement d'un pré-cycle. Le cycle de patrouille de π_i représente la liste des noeuds qui seront visités indéfiniment par le capteur i. Le premier noeud et le dernier noeud de cette liste sont les mêmes, d'où le nom de cycle. Le pré-cycle de π_i représente la liste des noeuds visités par le capteur i pour rejoindre son cycle de patrouille. Par exemple, lorsque les agents patrouilleurs sont localisés au même endroit dans leur environnement, autrement dit lorsqu'ils partent tous du même noeud initial, les stratégies de patrouille multi-agent optimales au sens de WI sont

rarement celles où chaque stratégie individuelle est telle que le cycle de patrouille débute et se termine sur ce noeud initial.
Souvent, une stratégie de dispersion des agents sur les noeuds du graphe est nécessaire dans une phase préalable à la patrouille en elle-même. Cette stratégie de dispersion, employée dans [LK08] pour déterminer les pré-cycles des stratégies de patrouille, est décrite ci-dessous.

Mise en oeuvre de la dispersion des agents à travers le graphe

Supposons que n est le nombre de noeuds du graphes à patrouiller et r le nombre d'agents patrouilleurs. Le problème de la dispersion des agents est de trouver une liste l^* (cf. équation 4.3) composée de r noeuds qui maximisent la somme des plus petites distances entre les r noeuds. Ce problème de dispersion a été décrit par *F. Lauri* dans [LK08].

$$l^* \in \underset{l \in L}{\operatorname{argmax}}\, d(l) \tag{4.3}$$

où $d(l)$ est la somme des plus petites distances entre les noeuds présents dans l (voir l'équation 4.4) et L représente l'ensemble des solutions possibles.

$$d(l) = \sum_{i \in l} min_{j \in l, i \neq j} c(i,j) \tag{4.4}$$

Dans l'équation 4.4, $c(i,j)$ est le coût (distance ou temps de parcours) du noeud i au noeud j.
Le problème de la dispersion des agents revient alors à déterminer une liste de r noeuds les plus éloignés possibles. La solution recherchée doit maximiser la somme des distances minimales entre les r noeuds. L'objectif principal visé est d'avoir les noeuds initiaux de départ de patrouille (des agents), les plus éloignés possibles.
Une approche de résolution du problème de dispersion par le biais d'un algorithme évolutionnaire (AE) est présentée dans [LK08]. Le but de l'approche par AE est de déterminer le meilleur ensemble de noeuds assez distancés qui permette une dispersion efficiente des différents agents à travers le graphe. L'approche par AE est couplée à l'OCF dans [LK08] afin de résoudre de façon efficiente le problème de la patrouille multi-agents.
Pour le problème classique de la patrouille multi-agents, la prise en compte de l'aspect dispersion dans le départ des agents permet d'obtenir de meilleurs résultats que l'OCF classique avec le départ des agents d'un même noeud initial [LK08]. Ainsi, cette phase est utilisée pour permettre de déterminer des stratégies avec des précycles (séquences de noeuds visités une seule fois

avant le cycle). Ces stratégies peuvent être plus efficaces sur certaines topologies de graphes.
Par une méthode de résolution exacte, l'équation 4.3 nécessiterait l'exploration d'un nombre de solutions éventuelles égal au coefficient binomial $C(n, r)$ où n représente le nombre de noeuds du graphe et r le nombre d'agents patrouilleurs, correspondant encore à la taille de la liste l^*. L'application d'une telle approche impliquerait par conséquent un temps exponentiel. L'utilisation d'un AE permet de trouver rapidement de bonnes solutions, en de temps d'exécution raisonnable.
Dans la résolution par un AE, chaque individu (solution potentielle) est représenté par une liste l formée de r noeuds définie par $l = (n_1, n_2, \ldots, n_r)$ avec $n_{i+1} > n_i$ pour tout $i = 1, 2, \ldots, r-1$. Aufur et à mesure des itérations, les opérateurs évolutionnaires à savoir *croisement*, *mutation* et *sélection* sont appliqués aux différentes populations. Pour plus d'informations, ces différents opérateurs sont explicités par *F. Lauri* dans [LK08].

4.3.2 Algorithme *AMCPMC*

L'algorithme *AMCPMC* est un algorithme utilisant un front de Pareto pour mémoriser les meilleures stratégies de patrouille. Il appartient donc à la classe des algorithmes a posteriori, qui permet ensuite au décideur de choisir parmi un ensemble de solutions copmpromis, celle qui satisfait le mieux à ses exigences. Cet algorithme prend en entrée le graphe G, la liste N des noeuds de départ des agents, l'ensemble S des capteurs qui sont des Sinks, le nombre n de colonies, le nombre T_{MAX} d'itérations à utiliser pour déterminer les meilleures stratégies de patrouilles ainsi qu'un booléen $DisperseAgents$. Ce booléen indique si les stratégies de patrouille sont constituées ou non de pré-cycles. Si ce booléen est armé, les pré-cycles sont calculés à l'issue d'une dispersion des agents sur les noeuds les plus éloignés du graphe G. Nous avons fait le choix de permettre au décideur d'introduire cette connaissance pour restreindre l'ensemble de recherche des stratégies de patrouille. Soient $n_i \in N$ le noeud de départ de l'agent i et d_i le noeud que l'agent i doit rejoindre lors de la phase de dispersion. Le pré-cycle associé à l'agent i est la liste des noeuds à visiter pour parcourir le chemin le plus court entre n_i et d_i. Nous supposons également dans cet algorithme qu'à chaque agent i est associée une stratégie individuelle de patrouille de la forme $\pi_i = (p_i, c_i)$, où p_i est le pré-cycle et c_i est le cycle, tels que le dernier noeud spécifié dans p_i et le premier noeud spécifié dans c_i sont les mêmes. Les étapes principales de cet algorithme sont présentées à la page 69.

Algorithme 2 Algorithme *AMCPMC*

Entrées: G : graphe, N : liste des noeuds de départ des $r = [N|$ agents, S : ensemble des capteurs Sinks, n : nombre de colonies, T_{MAX} : nombre d'itérations maximum de l'algorithme, $DisperseAgents$: Booléen indiquant si les agents doivent subir ou non une dispersion sur les noeuds les plus éloignés du graphe.

Sorties: F : Front de Pareto des meilleures stratégies de patrouille.

1: **Si** $DisperseAgents$ **Alors**
2: Déterminer les pré-cycles des agents en considérant une dispersion des agents sur les noeuds les plus éloignés du graphe G (voir section 4.3.1). Le dernier noeud de chaque pré-cycle est le noeud de début du cycle.
3: **Sinon**
4: Les pré-cycles des agents sont initialisés en tant que listes composées uniquement du noeud de départ indiqué dans la liste N.
5: **FinSi**
6: Associer à chaque colonie d'identifiant pair le critère WI et à chaque colonie d'identifiant impair le critère CI.
7: Initialiser les 2 graphes de phéromones G_{WI} et G_{CI} et le front de Pareto F.
8: $T \leftarrow 0$
9: **TantQue** $T < T_{MAX}$ **Faire**
10: Positionner les r fourmis de chaque colonie sur le dernier noeud de leurs pré-cycles.
11: Initialiser les n listes tabu (une par colonie).
12: Déterminer les n stratégies de patrouille en déplaçant les r fourmis dans chacune des n colonies. Le graphe de phéromones utilisé par une colonie dépend du critère privilégié et déterminé à la ligne 6.
13: Mettre à jour chaque graphe de phéromones. Le graphe de phéromones G_{WI} (respectivement G_{CI}) est mis à jour avec la meilleure colonie choisie parmi les identifiants pairs (respectivement impairs).
14: Calculer pour chacune des n stratégies de patrouille les 2 critères WI et CI, en utilisant les $|S|$ Sinks pour l'évaluation de CI.
15: Mettre à jour le front de Pareto F avec les meilleures stratégies de patrouille.
16: $T \leftarrow T + 1$
17: **FinTantQue**
18: **Retourner** Front de Pareto F.

La recherche de solutions de compromis est rendu possible grâce à l'utilisation de deux graphes de phéromones, que nous appelerons G_{WI} et G_{CI}. Le graphe G_{WI} permet de rechercher les meilleures stratégies de patrouille

minimisant le critère WI. Le graphe G_{CI} permet de rechercher les meilleures stratégies de patrouille minimisant CI. Ces deux graphes sont initialisés avec les mêmes noeuds et les mêmes arcs que le graphe G. Ils sont ensuite rendus complets, pour faciliter la recherche de solutions, en ajoutant des arcs entre chacune des paires de noeuds n'existant pas dans G. Le poids associé à chaque arc (s, e) ajouté est le coût du chemin le plus court entre les noeuds s et e. Nous considérons que les colonies de fourmis d'identifiant pair minimisent le critère WI alors que celles d'identifiant impair minimisent le critère CI (ligne 6). Après que les deux graphes de phéromones soit initialisés avec une quantité faible de phéromones sur leurs arcs et que le front de Pareto soit initialisé à l'ensemble vide (ligne 7), T_{MAX} itérations se succéderont pour déterminer un ensemble de solutions de compromis (lignes 9 à 17), qui seront finalement retournées dans le front de Pareto F (ligne 18). A chaque itération, n stratégies de patrouille sont déterminées à l'issue des déplacements de r fourmis dans chacune des n colonies (ligne 12). Les fourmis des colonies d'identifiant pair utilisent le graphe G_{WI} pour leurs déplacements, celles des colonies d'identifiant impair utilisent le graphe G_{CI}. A noter qu'une colonie a élaboré une stratégie de patrouille une fois que l'ensemble des noeuds visités par la colonie est l'ensemble V des noeuds du graphe G, c'est-à-dire une fois que la liste tabu associée à la colonie contient tous les noeuds V. A chaque itération, les n listes tabu sont initialisées à vide (ligne 11). Lorsque les n stratégies de patrouille sont disponibles, celles-ci sont utilisées pour mettre à jour les phéromones sur les graphes G_{WI} et G_{CI} (ligne 13) et elles sont évaluées sur les deux critères (ligne 14). La meilleure colonie choisie parmi les identifiants pairs met à jour les phéromones du graphe G_{WI} et la meilleure colonie choisie parmi les identifiants impairs met à jour les phéromones du graphe G_{CI} (ligne 13). Le front de Pareto est mis à jour avec les meilleures stratégies de patrouille (ligne 15) avant de commencer une nouvelle itération.

4.4 Synthèse

Dans ce chapitre, nous avons proposé un algorithme basé sur des colonies de fourmis, pour résoudre de manière approximative le problème de la patrouille multi-capteurs appliquée à la détection des évènements. Après avoir exposé les caractéristiques et les principes des approches basées sur les colonies de fourmis, nous avons présenté notre technique **AMCPMC**, qui permet de calculer des solutions de compromis en maintenant un front de Pareto des meilleures stratégies multi-capteurs trouvées au cours du processus de recherche.

Dans le but d'évaluer la pertinence et l'efficacité de notre algorithme,

nous procédons dans le chapitre suivant à des expérimentations en utilisant plusieurs scenarii de patrouille.

Chapitre 5

Expérimentations et résultats des simulations

5.1 Introduction

Dans le chapitre précédent, nous avons proposé une approche pour résoudre de manière approximative le problème de la patrouille multi-capteurs. Dans le présent chapitre, nous nous intéressons essentiellement à la validation de notre approche sur la base de résultats expérimentaux.
Dans un premier temps, nous commencerons par présenter les résultats expérimentaux, puis nous procéderons à une analyse de ces résultats. Enfin, nous terminons par la section 5.6 qui présente une synthèse du chapitre.

5.2 Protocole d'expérimentations

Avant de présenter les résultats expérimentaux et de fournir une analyse de ces résultats, nous allons préciser tout d'abord dans quelles conditions nous avons réalisées nos expériences de simulation.

Pour commencer, nos simulations fûrent réalisées sur un ordinateur équipé du système d'exploitation Microsoft Windows 7 Édition Familiale Premium, d'un processeur Intel(R) (Atom(TM) CPU N570 @ 1.66GHz, 1.66GHz, 2 coeurs) et de 2 Go.

Les stratégies de patrouille multi-capteurs ont été évaluées sur six topologies de graphes de complexités variables (cf. tableau 5.1). Une représentation graphiques de chaque topologie de graphe est donnée dans l'Annexe A.

TABLE 5.1 – Topologies de graphes

Name	# nodes	# edges	Degree
Map A	50	106	7
Map B	50	69	5
Island	50	84	6
Cercle	56	56	2
Corridor	70	69	2
Grille	80	142	4

Des populations de 2, 5, 10 et 15 capteurs fûrent successivement utilisées sur chaque topologie de graphe. Les colonies de fourmis sont donc composées d'autant de fourmis qu'il y a de capteurs considérés. Pour chaque population de capteurs, un seul est considéré comme étant le Sink. Les paramètres présentés au tableau 5.2 ont été employés dans toutes nos simulations. Ces paramètres sont ceux qui nous ont permis d'obtenir les meilleurs résultats, parmi les différents autres paramètres testés.

TABLE 5.2 – Paramètres de l'algorithme *AMCPMC*

Paramètres	Valeurs
Nombre d'itérations (T)	50000
Nombre de colonies	50
Quantité initiale de phéromone	1.0
Coefficient d'évaporation $(1 - \rho)$	0.5
Contrôle intensité phéromone (α)	5.0
Contrôle visibilité (β)	1.0

Les résultats qui suivent montrent une évaluation des stratégies de patrouille déterminées par l'algorithme *AMCPMC* en terme d'oisiveté environnementale (critère WI) et d'oisiveté communicationnelle (critère CI). Nous nous sommes aussi intéressés à l'évaluation a posteriori des délais entre l'apparition et la détection d'un évènement.

5.3 Etude sur la base des critères WI et CI

5.3.1 Environnement MapA

La figure 5.1 présente les solutions de compromis déterminées par *AMCPMC* pour l'environnement MapA. Nous pouvons remarquer sur cette figure que le fait d'utiliser 10 ou 15 capteurs permet d'effectuer des patrouilles plus efficaces en terme des critères WI et CI que lorsque 2 ou 5 capteurs sont utilisés. Par contre, lorsque 15 capteurs sont utilisés plutôt que 10 capteurs, une stratégie meilleure pour 10 capteurs a été déterminée par rapport aux stratégies obtenues avec 15 capteurs. Nous pensons que ce résultat est dû à une convergence trop rapide de l'algorithme, qui n'a pas réussi à trouver des stratégies plus efficaces.

La taille des fronts de Pareto des différentes populations de capteurs nous montre également qu'il est beaucoup plus difficile de trouver de bonnes stratégies lorsque peu de capteurs sont impliqués (en particulier 2 ou 5 capteurs) que lorsqu'un plus grand nombre est utilisé.

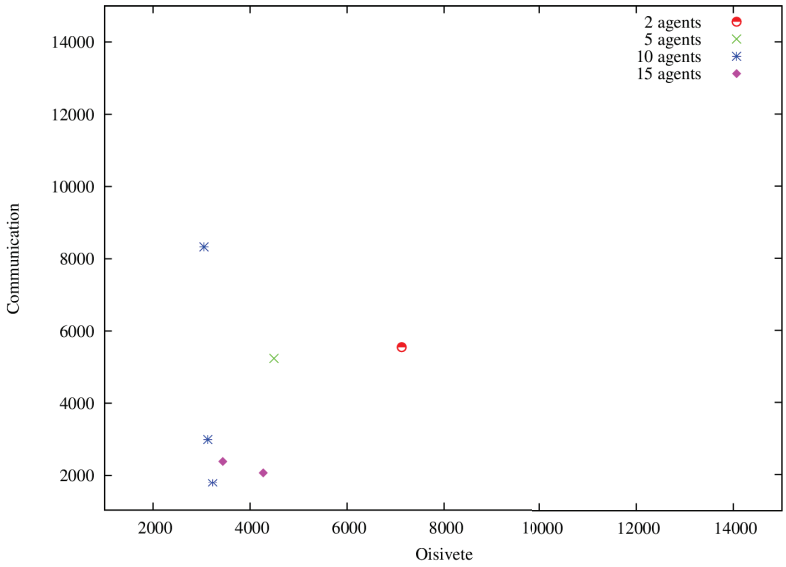


FIGURE 5.1 – Solutions de compromis pour l'environnement MapA

Si l'on analyse les meilleures stratégies déterminées par l'algorithme, on constate que, pour cet environnement, les noeuds à visiter régulièrement ont été réparties plus ou moins équitablement entre les capteurs. Certains noeuds peuvent être visités plusieurs fois par des capteurs différents tandis que d'autres noeuds ne seront visités que par un capteur en particulier. Il est

donc normal d'obtenir des solutions dont l'oisiveté communicationnelle est plus grande lorsque moins de capteurs sont impliqués, puisque moins il y a de capteurs, plus le temps requis pour que les capteurs et le Sink synchronisent leur passage et puissent transmettre l'alerte au Sink est grand.

5.3.2 Environnement MapB

Nous présentons dans la figure 5.2 les solutions de compromis déterminées par *AMCPMC* pour l'environnement MapB. Dans cet environnement aussi, il est préférable d'utiliser plutôt 10 ou 15 capteurs que 5 et plutôt 5 capteurs que 2 capteurs pour patrouiller plus efficacement. Par ailleurs, les patrouilles impliquant 10 capteurs sont aussi meilleures que celles impliquant 15 capteurs. Nous pensons que ce résultat est aussi dû à une convergence trop rapide de l'algorithme, qui n'a pas réussi à trouver des stratégies plus efficaces.

La taille des fronts de Pareto des différentes populations de capteurs nous indique ici qu'il est plus difficile de déterminer des stratégies de patrouille pour un nombre important de capteurs que pour un nombre plus réduit.

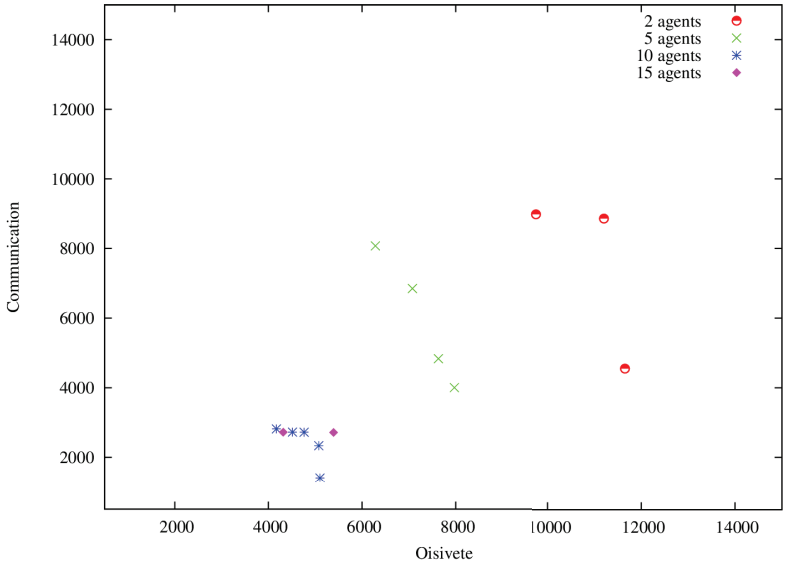


FIGURE 5.2 – Solutions de compromis pour l'environnement MapB

Soulignons tout d'abord le fait que ce graphe est constitué des mêmes noeuds que le graphe MapA et qu'il comprend quatre sous-graphes connectés par un noeud commun (le noeud 29, voir figure A.2, page 101). Si l'on analyse les meilleures stratégies déterminées par l'algorithme, on se rend compte que, pour cet environnement, les noeuds à visiter régulièrement semble avoir subi

un partitionnement, les noeuds d'une partition étant alors associés à un capteur particulier ; le noeud 29 est quant à lui visité régulièrement par plusieurs agents. Par exemple, avec 5 capteurs, 2 capteurs s'occuperont de visiter les noeuds d'une partition, les 3 autres capteurs étant attachés à visiter régulièrement les noeuds de 3 partitions différentes et le noeud 29 apparaîtra dans le cycle de patrouille de plusieurs capteurs, voire des 5 capteurs. Ici encore, il est donc normal d'obtenir des solutions dont l'oisiveté communicationnelle est plus grande lorsque moins de capteurs sont impliqués, puisque moins il y a de capteurs, plus le temps requis pour que les capteurs et le Sink synchronisent leur passage au noeud 29 et puissent transmettre l'alerte au Sink est grand.

5.3.3 Environnement Island

La figure 5.3 présente les solutions de compromis déterminées par *AMCPMC* pour l'environnement Island. Sur cette figure, nous pouvons remarquer que notre algorithme détermine des stratégies de patrouille qui sont plus performantes lorsque 10 ou 15 capteurs sont utilisés plutôt que 5, ou lorsque 5 capteurs sont impliqués plutôt que 2 capteurs.

La taille des fronts de Pareto des différentes populations de capteurs nous indique ici qu'a priori, il est plus difficile de déterminer des stratégies de patrouille pour un nombre faible de capteurs que pour un nombre plus conséquent.

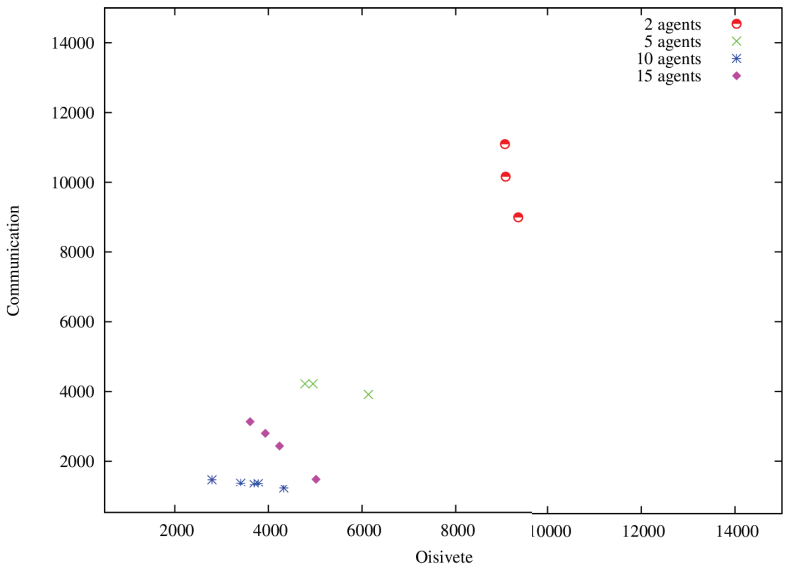


FIGURE 5.3 – Solutions de compromis pour l'environnement Island

Comme illustré par la figure A.6, page 103, l'environnement Island est constitué de 9 îlots connectés entre eux. Chaque îlot est composé de 4 à 7 noeuds. Dans cet environnement, les stratégies les plus satisfaisantes qui ont été déterminées par l'algorithme sont celles qui répartissent les îlots sur les capteurs en s'assurant que chaque capteur visite un noeud éloigné à un îlot pour propager éventuellement une alerte à un autre capteur au besoin. Comme pour les autres environnements, moins il y a de capteurs impliqués dans la patrouille, plus le temps requis pour que les capteurs et le Sink synchronisent leur passage à des noeuds proches et puissent transmettre l'alerte au Sink est grand.

5.3.4 Environnement Cercle

Nous présentons à la figure 5.4 les solutions de compromis déterminées par *AMCPMC* pour l'environnement Cercle. Dans cet environnement encore, plus on dispose de capteurs, plus performantes sont a priori les stratégies de patrouille déterminées par l'algorithme.

La taille des fronts de Pareto des différentes populations de capteurs nous indique que pour cet environnement, il est aussi difficile a priori de déterminer des stratégies de patrouille pour un nombre important de capteurs que pour un nombre plus faible. Nous pensons que le faible degré de ce graphe (de degré 2) restreint considérablement le nombre de stratégies de patrouille satisfaisantes. Ces stratégies restent cependant difficiles à trouver car l'ensemble des stratégies de patrouille reste lui très grand.

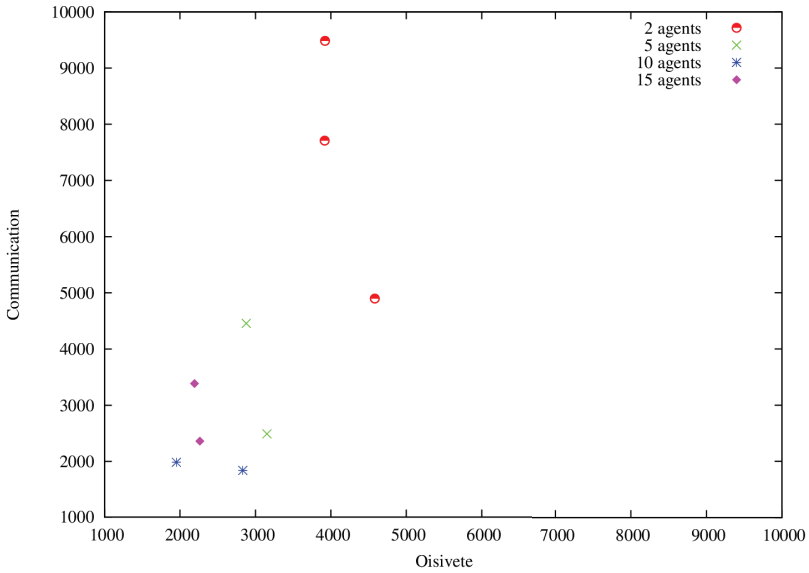


FIGURE 5.4 – Solutions de compromis pour l'environnement Cercle

Les stratégies déterminées dans cet environnement par notre algorithme sont composées de cycles de patrouille où les capteurs opèrent des va-et-vient sur des noeuds géographiquement proches. La même conclusion que précédemment peut être émise quant à l'évolution de l'oisiveté communicationnelle en fonction du nombre de capteurs.

5.3.5 Environnement Corridor

La figure 5.5 présente les solutions de compromis déterminées par *AMCPMC* pour l'environnement Corridor. Il est plus difficile de tirer une conclusion définitive concernant l'évolution de la performance des stratégies multi-capteurs en fonction du nombre de capteurs impliqués pour cet environnement. En analysant les résultats obtenus, il semblerait en effet que la performance des stratégies dépend assez peu du nombre de capteurs impliqués. Cet environnement Corridor, comme l'environnement Cercle, est de degré 2. Les stratégies satisfaisantes pour cet environnement sont donc très difficiles à trouver du fait de leur faible nombre. Les solutions les plus satisfaisantes n'auraient donc pas été trouvées en raison d'une convergence trop rapide de notre algorithme.

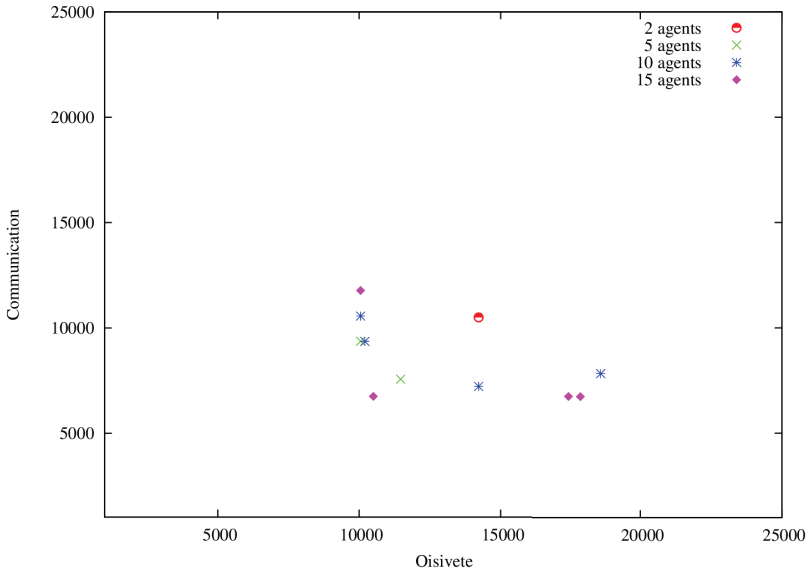


FIGURE 5.5 – Solutions de compromis pour l'environnement Corridor

En analysant les stratégies trouvées par notre algorithme, nous nous sommes aperçus que celles-ci, comme dans le cas de l'environnement précédent, sont composées de cycles de patrouille où les capteurs opèrent des va-et-vient sur des noeuds géographiquement proches. Ces stratégies ne sont pas les plus satisfaisantes pour cet environnement, ce qui peut expliquer les résultats plutôt moyens.

5.3.6 Environnement Grille

Les solutions de compromis déterminées par *AMCPMC* pour l'environnement Grille sont présentées à la figure 5.6. Dans cet environnement, les patrouilles impliquant 10 capteurs sont aussi meilleures que celles impliquant 15 capteurs selon le critère. On note tout de même d'assez bonnes stratégies avec le cas de 15 capteurs sur le plan communicationnel. Les stratégies les plus satisfaisantes qui ont été déterminées par l'algorithme sont celles qui répartissent assez équitablement les noeuds à visiter aux différents capteurs. La répartition des noeuds observée dans les stratégies satisfaisantes est favorisée par la structure uniforme de cet environnement.

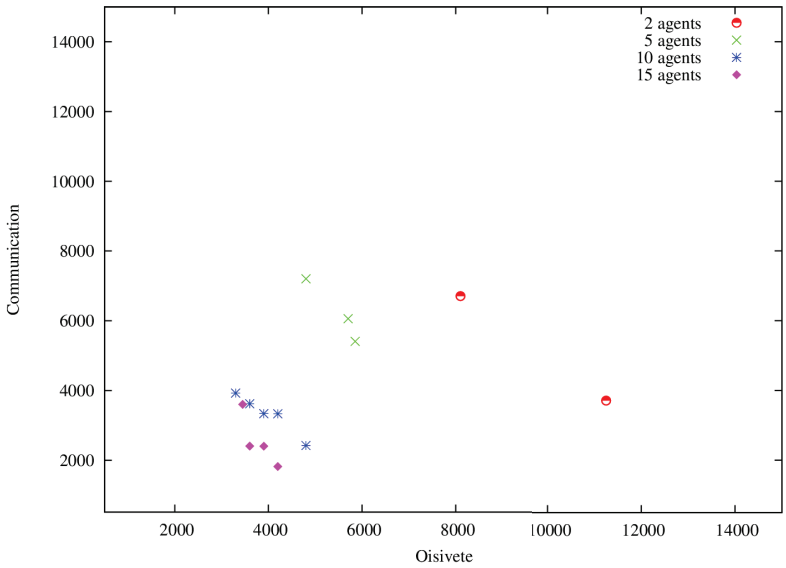


FIGURE 5.6 – Solutions de compromis pour l'environnement Grille

5.4 Evaluation des délais de détection

Nous nous intéressons dans ce qui suit à l'évaluation des délais entre l'apparition d'un évènement et sa détection. Nous avons utilisé les mêmes instants et positions des évènements pour évaluer le critère CI pour toutes les stratégies. 5 simulations furent utilisées pour chaque population de capteurs et sur chaque graphe pour évaluer ces délais. Dans les tableaux de résultats qui suivent, $D_1(t_i)$ indique le délai moyen de détection associé à un évènement apparu à l'instant t_i et $D_2(t_i)$ indique le délai moyen nécessaire à la réception et à la transmission par le Sink des alertes associées aux évènements perçus. Les valeurs respectives de t_1 et t_2 sont de 200 et 3000.

5.4.1 Environnement MapA

Dans un premier temps, nous nous sommes intéressés aux délais entre l'apparition et la détection d'un évènement dans l'environnement MapA. Pour un nombre variable de capteurs, les meilleures pires oisivetés communicationnelles, c'est-à-dire les plus grand délais entre l'apparition d'un évènement, sa détection et sa transmission au plus tôt au Sink, ont été évaluées et nous obtenons les résultats suivants (cf. tableau 5.3).

TABLE 5.3 – Délais entre l'apparition et la détection d'un évènement en environnement MapA

Nombre Capteurs	$D_1(t_1)$	$D_2(t_1)$	$D_1(t_2)$	$D_2(t_2)$
2	262	282	278	319
5	185	187	147	184
10	53	60	145	307
15	30	32	84	162

D'après le tableau 5.3, les délais moyens de détection des évènements sont obtenus avec 15 agents-capteurs. Ces résultats s'expliquent par la meilleure couverture de l'environnement lorsque le nombre d'agents-capteurs est grand. Pour l'environnement MapA, plus le groupe est formé de capteurs, meilleure est la couverture. Ainsi, les délais diminuent de façon générale avec le nombre croissant d'agents.

5.4.2 Environnement MapB

Nous présentons dans le tableau 5.4, les délais moyens de détection des évènements pour l'environnement MapB. Les délais avec un réseau formé de deux agents-capteurs sont plus élevés qu'avec les autres groupes. Pour le cas de dix agents-capteurs, un délai très faible est suffisant pour transmettre un évènement apparu à t_1. Cela voudrait dire que l'évènement a été probablement détecté par le Sink. Lorsque les délais sont grands, l'évènement détecté arrive au Sink par le biais de communications indirectes.

TABLE 5.4 – Délais entre l'apparition et la détection d'un évènement en environnement MapB

Nombre Capteurs	$D_1(t_1)$	$D_2(t_1)$	$D_1(t_2)$	$D_2(t_2)$
2	179	185	198	217
5	31	57	55	207
10	1	13	256	364
15	26	33	168	190

5.4.3 Environnement Island

Le tableau 5.5 présente les délais moyens entre l'apparition d'un évènement, sa détection et sa transmission au plus tôt au Sink dans l'environnement Island. Lorsque le réseau est formé de quinze agents-capteurs, un délai faible est suffisant pour transmettre un évènement apparu à t_1. Dans ce cas, l'évènement a été détecté soit directement par le Sink, soit par un capteur situé dans le rayon communicationnel du Sink. Nous observons également que les délais sont naturellement assez élevés avec des groupes formés seulement de deux agents-capteurs.

TABLE 5.5 – Délais entre l'apparition et la détection d'un évènement en environnement Island

Nombre Capteurs	$D_1(t_1)$	$D_2(t_1)$	$D_1(t_2)$	$D_2(t_2)$
2	549	551	11	230
5	30	38	224	314
10	8	15	10	20
15	3	18	201	277

5.4.4 Environnement Cercle

Nous présentons (cf. tableau 5.6), les délais moyens entre l'apparition d'un évènement, sa détection et sa transmission au plus tôt au Sink dans l'environnement Cercle. Lorsque le réseau est formé de dix agents-capteurs, un délai égal également encore faible est suffisant pour transmettre un évènement apparu à t_1. Les groupes formés de cinq, dix et quinze capteurs offrent de bons délais pour des évènements apparus à t_1. Si l'évènement apparaît plutôt à t_2, les groupes formés de cinq et dix agents-capteurs réagissent également plutôt rapidement.

TABLE 5.6 – Délais entre l'apparition et la détection d'un évènement en environnement Cercle

Nombre Capteurs	$D_1(t_1)$	$D_2(t_1)$	$D_1(t_2)$	$D_2(t_2)$
2	208	265	160	368
5	7	11	168	348
10	2	8	32	118
15	5	6	94	98

5.4.5 Environnement Corridor

Le tableau 5.7 présente les délais moyens entre l'apparition d'un évènement, sa détection et sa transmission au plus tôt au Sink pour le cas de l'environnement Corridor. Les meilleurs délais sont obtenus lorsque le réseau est formé de cinq ou dix agents-capteurs, où un délai relativement faible est suffisant pour transmettre un évènement apparu à t_1.

TABLE 5.7 – Délais entre l'apparition et la détection d'un évènement en environnement Corridor

Nombre Capteurs	$D_1(t_1)$	$D_2(t_1)$	$D_1(t_2)$	$D_2(t_2)$
2	25	62	100	300
5	3	44	360	435
10	1	21	120	332
15	46	55	366	369

5.4.6 Environnement Grille

Nous présentons dans le tableau 5.8, les délais moyens entre l'apparition d'un évènement, sa détection et sa transmission au plus tôt au Sink dans l'environnement Grille. Pour le cas de quinze agents-capteurs, un évènement apparu à t_1 est transmis avec un délai faible. Cela voudrait dire que l'évènement a sûrement été détecté par le Sink. De façon générale, plus le délai est petit, plus le capteur ayant détecté l'évènement est soit sur la trajectoire du Sink, soit proche du Sink. Lorsque les délais sont grands, cela voudrait dire que l'évènement détecté arrive au Sink par le biais de communications indirectes.

TABLE 5.8 – Délais entre l'apparition et la détection d'un évènement en environnement Grille

Nombre Capteurs	$D_1(t_1)$	$D_2(t_1)$	$D_1(t_2)$	$D_2(t_2)$
2	109	234	102	188
5	105	128	30	315
10	1	16	10	105
15	1	7	200	250

5.5 Temps de calcul

La figure 5.7 et le tableau 5.9 présentent les temps d'exécution nécessaires pour la détermination des solutions issues des simulations sur les différents environnements, en fonction du nombre d'agents capteurs.

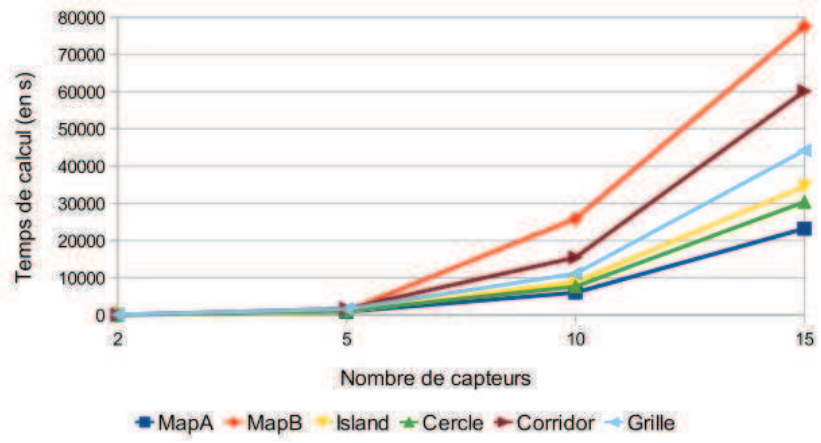


FIGURE 5.7 – Temps de calcul des simulations

TABLE 5.9 – Temps d'exécution des simulations

Graphe	Temps moyen d'exécution (en s) en fonction du nombre de capteurs			
	2	5	10	15
MapA	38	822	6024	23196
MapB	86	1033	25856	77568
Island	50	906	8885	34540
Cercle	24	1081	7621	30384
Corridor	46	1719	15465	60141
Grille	73	1610	11100	44207

La complexité en temps de calcul de l'algorithme *AMCPMC* pour chaque environnement semble être en $O(x^3)$, où x est le nombre de capteurs. En effet, pour chaque environnement, le temps de calcul s'accroît selon un polynôme du troisième degré avec le nombre de capteurs.

5.6 Synthèse

Nous avons implémenté l'approche décrite dans le chapitre précédent, puis nous l'avons évaluée sur la base d'expériences de simulations informatiques, en utilisant différents scenarii. Ce chapitre expose les résultats obtenus par notre approche. Nous pouvons constater que ces résultats sont plutôt satisfaisants et encourageants, et qu'ils permettent de valider à la fois la pertinence et l'efficacité de notre approche.

Chapitre 6

Conclusion générale

6.1 Bilan

Dans cette thèse, nous nous sommes intéressés à l'utilisation de la patrouille à l'aide d'un réseau de capteurs pour la surveillance d'un environnement en vue de détecter rapidement des évènements critiques pouvant y survenir.

Une formulation du problème de la patrouille multi-agents en environnements connus et une formulation du problème de la patrouille multi-agents en environnements inconnus ont été présentées telle que rencontrées dans la littérature. Les approches proposées dans l'état de l'art pour les résoudre ont été décrites. Nous nous sommes par la suite intéressés à formaliser le problème de la patrouille multi-capteurs lorsque ceux-ci sont utilisés en vue de détecter rapidement des évènements. Ce problème diffère de celui de la patrouille multi-agents sur plusieurs aspects. Premièrement, les capteurs sont des agents disposant de ressources limitées. Deuxièmement, les capteurs ayant détecté un évènement doivent propager leur alerte à un capteur particulier, appelé *Sink*, qui peut à son tour alerter le centre de contrôle. Ce dernier pourra alors éventuellement décider d'intervenir dans la région où l'alerte a été déclarée. Ces différences impliquent de les prendre en compte dans la formulation du problème de patrouille lorsqu'un réseau de capteurs est impliqué. Cette prise en compte nécessite de définir plusieurs fonctions objectifs. Etant donné que le problème de la patrouille multi-capteurs est multi-objectif, nous l'avons abordé en utilisant une technique qui peut s'adapter facilement dans le domaine de l'optimisation multiobjectif. Notre technique, appelée *AMCPMC*, est basée sur des colonies de fourmis en compétition. Les résultats expérimentaux menés dans le but d'évaluer la pertinence de notre approche montrent

qu'elle est capable de déterminer des solutions acceptables.

6.2 Perspectives et travaux futurs

Les résultats de notre travail sont encourageants, mais il pourrait donner lieu à plusieurs améliorations à différents niveaux. Nous définissons ci-dessous plusieurs axes d'approfondissement pour la suite de ce travail.

Une première perspective à court terme serait de compléter les expériences menées dans le chapitre précédent, en intégrant les résultats obtenus par notre approche en faisant varier un plus grand nombre de paramètres, et en particulier le nombre de colonies, le nombre d'itérations et le nombre de *Sinks* impliqués dans la patrouille. Dans le chapitre précédent, nous avons présenté les résultats obtenus à partir d'expériences menées avec un seul Sink, en utilisant 50000 itérations et 50 fourmis par colonie. Il serait intéressant d'étudier l'influence de ces paramètres sur les performances des stratégies de patrouille déterminées.

Une des perspectives envisageables à court et moyen terme concerne la validation de notre approche en utilisant un réseau physique réel de capteurs mobiles sans fil. Ces expérimentations permettraient en effet de récolter de précieuses informations relatives à l'efficacité de notre approche sur des capteurs réels, et d'entrevoir les forces et les faiblesses de notre approche. Préalablement à ces expérimentations, nous pourrions envisager d'intégrer le critère relatif à la consommation d'énergie dans notre approche, en ajoutant un troisième graphe de phéromones utilisés pour privilégier les stratégies optimisant ce critère. Ce critère pourrait alors être plus précisément mesuré sur la base des expériences de patrouille menées avec les capteurs réels.

Une autre perspective, plus théorique cette fois-ci, serait d'étudier et de proposer des métaheuristiques alternatives, autres que celles basées sur des colonies de fourmis, afin de déterminer de meilleures stratégies de patrouille multi-capteurs. Les métaheuristiques utilisées pourraient être des algorithmes évolutionnaires, des algorithmes à base d'essaims particulaires, des techniques basées sur l'entropie croisée, ou une combinaison de ceux-ci, par exemple. La nouvelle approche ne devrait pas introduire de biais sur les stratégies de patrouille. Une des limitations actuelles de notre approche, qui devrait être levée, réside en effet dans la détermination de stratégies de patrouille ayant une structure particulière. Ces stratégies comportent en effet toutes des cycles de patrouille où les noeuds de début et de fin de cycle sont soit les noeuds de départ des agents, soit les noeuds obtenus après une phase de dispersion des agents sur les noeuds du graphe. Ce travail serait alors essentiel pour confronter l'approche actuelle à une autre, et évaluer plus précisément son

efficacité.

Bibliographie

[Aa02] I. F. Akyildiz and al. Wireless sensor networks : a survey, Computer Networks. In *Elvesier*, pages 393–422, 2002.

[Aba96] F. R. Abate. The Oxford Dictionary and Thesaurus : The Ultimate Language Reference for american Readers. In *Oxford Univ. Press*, 1996.

[AKK04] J. N. Al-Karaki and A. E. Kamal. Routing Techniques in Wireless Sensor Networks : A Survey. In *IEEE Wireless Communications*, 2004.

[ARS+04] A. Almeida, G. Ramalho, H. Santana, P. Tedesco, T. Menezes, V. Corruble, and Y. Chevaleyre. Recent Advances on Multi-Agent Patrolling. In *Springer-Verlag, Berlin Heidelberg*, pages 474–483, 2004.

[ASG05] I. Alaya, C. Solnon, and K. Ghédira. Optimisation par colonies de fourmis pour le problème du sac à dos multidimensionnel. In *Technique et science informatiques*, 2005.

[ATC07] I. F. Akyildiz, M. Tommaso, and R. Kaushik Chowdhury. A survey on wireless multimedia sensor networks, Computer Networks 51. In *Elvesier*, pages 921–960, 2007.

[BA98] T. Balch and R. Arkin. Behavior-based formation control for multirobot systems. In *In IEEE Transactions on Robotics and Automation, 14(12)*, 1998.

[Bar03] Vincent Barichard. *Approches hybrides pour les problèmes multiobjectifs*. PhD thesis, Université d'Angers, 2003.

[BD01] J.P. Briot and Y. Demazeau. Principes et architecture des systèmes multi-agents. In *collection IC2, Hermès*, 2001.

[BE02] D. Braginsky and D. Estrin. Rumor Routing Algorithm For Sensor Networks. In *WSNA'02, ACM*, 2002.

[BEGH01] N. Bulusu, D. Estrin, L. Girod, and J. Heidemann. Scalable Coordination forWireless Sensor Networks : Self-Configuring Loca-

lization Systems. In *6th International Symposium on Communication Theory and Applications, ISCTA'01*, 2001.

[Ber01] Alain Berro. *Optimisation multiobjectif et stratégie d'évolution en environnement dynamique*. PhD thesis, Université des Sciences Sociales Toulouse I, 2001.

[Ber08] Alain Berro. Algorithme évolutionnaire pour l'optimisation multiobjectif. In *LAAS, UT1 Sciences Sociales Toulouse I*, 2008.

[BFM+00] W. Burgard, D. Fox, M. Moors, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. In *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), San Francisco, CA*, 2000.

[BHD94] R. Beckers, O. E. Holland, and J. L. Denebourg. From local actions to global tasks : stigmergy and collective robotics. In *In Artificial Life, Brooks R., Maes P. (eds), Cambridge, Mass., MIT Press*, pages 181–189, 1994.

[BHS99a] B. Bullnheimer, R.F. Hartl, and C. Strauss. An improved ant system algorithm for the Vehicle Routing Problem. In *Annals of Operations Research*, pages 319–328, 1999.

[BHS99b] B. Bullnheimer, R.F. Hartl, and C. Strauss. Applying the ant systems to the vehicle routing problem. In *Meta-Heuristics : Advances and Trends in Local Search Paradigms for Optimization, Kluwer*, 1999.

[BsHJG06] S. Byungrak, Yong sork Her, and K. Jung-Gyu. A Design and Implementation of Forest-Fires Surveillance System based on Wireless Sensor Networks for South Korea Mountains. In *IJCSNS International Journal of Computer Science and Network Security, Vol.6 No.9B*, 2006.

[BTNA06] M. Basseur, E. Talbi, A. Nebro, and E. Alba. Metaheuristics for Multiobjective Combinatorial Optimization Problems : Review and recent issues. In *INRIA Report-ISSN 0249-6399*, 2006.

[CC61] A. Charnes and W. Cooper. Management models and industrial applications of linear programming. In *vol1, John Wiley, New-York*, 1961.

[CDM91] A. Colorni, M. Dorigo, and V. Maniezzo. Distributed Optimization by ant colonies. In *First ECAL*, pages 134–142, 1991.

[CDM92] A. Colorni, M. Dorigo, and V. Maniezzo. An Investigation of some properties of an ant algorithm. In *PPSN*, pages 509–520, 1992.

[CEBP99] D. A. Ciccimaro, H. R. Everett, M. H. Bruch, and C. B. Phillips. A Supervised Autonomous Security Response Robot. In *American Nuclear Society 8th International Topical Meeting on Robotics and Remote Systems (ANS'99), Pittsburgh*, 1999.

[CGS+07] H. N. Chu, A. Glad, O. Simonin, F. Sempé, A. Drogoul, and F.Charpillet. Swarm Approaches for the Patrolling Problem, Information Propagation vs. Pheromone Evaporation. In *19th IEEE International Conference on Tools with Artificial Intelligence, ICTAI, vol. 1*, pages 442–449, 2007.

[CH97] D. Costa and A. Hertz. Ants can colour graphs. In *JORS,48(3)*, pages 295–305, 1997.

[Cha05] A. B. A. Chamseddine. Optimisation multi-objectif évolutionnaire. In *Ecole Polytechnique de Tunisie*, 2005.

[Che02] Y. Chevaleyre. Le problème de la patrouille. In *Rapport de stage postdoctoral, Université Fédérale de Pernambuco, Brésil*, 2002.

[Che05] Y. Chevaleyre. The Patrolling Problem. In *Annales du LAMSADE, numéro 4, Paris-Dauphine University, France*, 2005. available in french at http ://www.lamsade.dauphine.fr/~chevaley/papers/anna_patro.pdf.

[Che06] Y. Chevaleyre. Le problème multi-agents de la patrouille. In *Annales du LAMSADE, Université Paris-Dauphine, France*, pages 121–144, 2006.

[Chu07] H. N. Chu. Approches collectives pour le problème de la patrouille multi-agents. In *Laboratoire Lorraine de Recherche en Informatique et ses Applications*, 2007.

[CS02a] Y. Colette and P. Siarry. Optimisation multiobjectif. Technical report, Edition Eyrolles, 2002.

[CS02b] Y. Collette and P. Siarry. Optimisation multiobjectif. In *Eyrolles*, 2002.

[Dav08] Meignan David. *Une approche organisationnelle et multi-agent pour la modélisation et l'implantation de métaheuristiques : Application aux problèmes d'optimisation de transports*. PhD thesis, Université de Technologie de Belfort-Montbéliard, 2008.

[DB05] M. Dorigo and C. Blum. Ant colony optimization theory : A survey. In *Theoretical Computer Science 344, Elsevier*, pages 243–278, 2005.

[DF05] Clarisse Dhaenens-Flipo. Optimisation combinatoire multi-objectif : apport des méthodes coopératives et contribution à l'extraction de connaissances. In *HDR, Université des Sciences et Technologies de Lille*, 2005.

[DG97] M. Dorigo and L. M. Gambardella. Ant colony system : a cooperative learning approach to the travelling salesman problem. In *TEC*, volume 1, pages 53–66, 1997.

[Dip10] Jean Dipama. *Optimisation multi-objectif des systèmes énergétiques*. PhD thesis, Ecole Polytechnique de Montréal, 2010.

[DMC96] M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System : Optimization by a colony of cooperating agents. In *IEEE Transactions on Systems, Man, and Cybernetics (Vol.26, no.1)*, pages 1–13, 1996.

[DPG01] L. Doherty, K. S. Pister, and L. El Ghaoui. Convex position estimation in wireless sensor networks. In *IEEE INFOCOM, vol 3*, pages 1655–1663, 2001.

[Dro93] Alexis Drogoul. *De la Simulation Multi-Agent à la Résolution Collective de Problèmes*. PhD thesis, Université Paris VI, 1993.

[Dro05] A. Drogoul. L'intelligence (Traité des Sciences cognitives), chapter Les systèmes multi-agents. In *Hermes Science*, 2005.

[EAK07] Y. Elmaliach, N. Agmon, , and G. A. Kaminka. Multi-robot area patrol under frequency constraints. In *IEEE International Conference on Robotics and Automation (ICRA-07)*, 2007.

[EG99] H. Everett and D. Gage. From laboratory to warehouse : security robots meet the real world. In *International Journal of Robotics Research*, pages 760–768, 1999.

[Fer95] J. Ferber. Les Systèmes Multi-Agents : vers une intelligence collective. In *iia, InterEdition*, 1995.

[FF93] C. M. Fonseca and P. J. Fleming. Genetic algorithms for multiobjective optimization : Formulation, discussion and generalization. In *Fifth International Conference on Genetic Algorithm, San Mateo, California*, pages 416–423, 1993.

[FLKT09] E. T. Fute, F. Lauri, A. Koukam, and E. Tonye. The Coverage Problem in Wireless Sensor Networks by Holonic Multi-agent Approach. In *International Journal of Computing and ICT Research, ACM, Vol.3, No.1*, pages 32–41, 2009.

[FM98] Schneider M. Fontan and J. Maja Mataric. Territorial multi-robot task division. In *IEEE Transactions of Robotics and Automation, 14(5)*, 1998.

[Fok04] Chien-Liang Fok. Efficient Fire Detection and Tracking Using Mobile Agents in a Wireless Sensor Network. In *MURI ONR 3-Year Review, Baltimore*, 2004.

[Fou85] Fourman. Compaction of symbolic layout using genetic algorithm. In *First International Conference on Genetic Algorithm*, pages 141–153, 1985.

[FTKL09a] E. T. Fute, E. Tonye, A. Koukam, and F. Lauri. Multi-Agent Patrolling Strategy : Application to the Exploration Environment Problem. In *International Journal of Automation, Robotics and Autonomous Systems, ICGST-ARAS, Vol.9, Issue 2*, 2009.

[FTKL09b] E. T. Fute, E. Tonye, A. Koukam, and F. Lauri. Stratégie de Patrouille Multi-Agents : Application au Problème d'Exploration d'un Environnement. In *5th IEEE International Conference : Sciences of Electronic, Technologies of Information and Telecommunications*, 2009.

[Gal08] Lucie Galand. *Méthodes exactes pour l'optimisation multicritère dans les graphes : recherche de solutions de compromis*. PhD thesis, Université de Paris VI, 2008.

[GB08] Tristram Gräbener and A. Berro. Optimisation multiobjectif discrète par propagation de contraintes. In *Actes JFPC*, 2008.

[GBSC09] A. Glad, O. Buffet, O. Simonin, and F. Charpillet. Auto-organisation dans les algorithmes fourmis pour la patrouille multi-agent. In *JFPDA'09*, 2009.

[GGP02] C. Gagné, M. Gravel, and W. L. Price. Algorithme d'optimisation par colonie de fourmis avec matrices de visibilité multiples pour la resolution d'un problème d'ordonnancement industriel. In *Information Systems and Operational Research (INFOR)*, volume 40(2), pages 259–276, 2002.

[GGP04a] C. Gagné, M. Gravel, and W. L. Price. Optimisation Multi-Objectifs à l'aide d'un Algorithme de Colonie de Fourmis. In *INFOR*, volume 42(1), pages 23–42, 2004.

[GGP04b] C. Gagné, M. Gravel, and W. L. Price. Using metaheuristic compromise programming for solution of multiple objective scheduling problems. In *Journal of the Operational Research Society*, 2004.

[Gla11] Arnaud Glad. *Etude de l'auto-organisation dans les algorithmes de patrouille multi-agent fondées sur les phéromones digitales*. PhD thesis, Université Nancy 2, 2011.

[Glo07] Glossaire. http ://www.maison-domotique.com/glossaire.php. In *Glossaire Domotique et NTIC*, 2007. visité le 19 juin 2007.

[GM02] B. P. Gerkey and M. J. Mataric. Pusher-watcher : An approach to fault-tolerant tightly-coupled robot coordination. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2002), Washington, DC*, pages 464–469, 2002.

[Gol89] D. E. Goldberg. Genetic algorithms in search, optimization, and machine learning. In *Addison-Wesley, Reading, Massachusetts*, 1989.

[GPG02] M. Gravel, W. L. Price, and C. Gagné. Scheduling continuous casting of aluminum using a multiple-objective ant colony optimization metaheuristic. In *European Journal of Operational Research*, volume 143(1), pages 218–229, 2002.

[GR01] Yoav Gabriely and Elon Rimon. Spanning-tree based coverage of continuous areas by a mobile robot. In *Ann. Math. Artif. Intell.*, pages 77–98, 2001.

[GSBC08] A. Glad, O. Simonin, O. Buffet, and F. Charpillet. Theoretical study of ant-based algorithms for multi-agent patrolling. In *Proceedings of the Eighteenth European Conference on Artificial Intelligence,(ECAI'08)*, 2008.

[GTA99] L. M. Gambardella, E. Taillard, and G. Agazzi. MACS-VRPTW : A multiple ant colony system for vehicle routing problems with time windows. In *Technical Report, Lugano, Switzerland, 1999*, pages 1–17, 1999.

[Gui06] Noël Guillaume. *Indexation dans les bases de données capteurs temps réel*. PhD thesis, Institut National des Sciences Appliquées de Lyon, 2006.

[HT03] Chi-Fu Huang and Yu-Chee Tseng. The Coverage Problem in a Wireless Sensor Network. In *WSNA'03, San Diego*, 2003.

[IM96] H. Ishibuchi and T. Murata. Multi-objective genetic local search algorithm. In *International Conference on Evolutionary Computation, Nagoya, Japan*, pages 119–124, 1996.

[Jé03] Regnier Jérémi. *Conception de systèmes hétérogènes en Génie Électrique par optimisation évolutionnaire multicritère*. PhD thesis, Institut Nationale Polytechnique de Toulouse, 2003.

[Kam06] Zidi Kamel. *Système interactif d'aide au déplacement multimodal*. PhD thesis, Ecole Centrale de Lille, Université des Sciences et Technologies de Lille, 2006.

[Lam00] Jouni Lampinen. Multiobjective nonlinear pareto-optimization. Technical report, University of Technology, Laboratory of Information Processing, 2000.

[Lau05] Fabrice Lauri. Etat de l'art des stratégies multi-agents de patrouille. Technical report, LORIA-INRIA, 2005.

[LC06] F. Lauri and F. Charpillet. Ant Colony Optimization applied to the Multi-Agent Patrolling Problem. In *SIS, Indianapolis, Indiana, USA*, 2006.

[LCS05] F. Lauri, F. Charpillet, and D. Szer. Analyse théorique du problème de la patrouille multi-agent en utilisant le cadre des processus décisionnels de Markov. In *LORIA-INRIA, Campus Scientifique*, 2005.

[LGSC08] F. Legras, A. Glad, O. Simonin, and F. Charpillet. Partage d'autorité dans un essaim de drones auto-organisé. In *Systèmes Multi-agents, JFSMA 08*, 2008.

[Lie09] Arnaud Liefooghe. *Métaheuristiques pour l'optimisation multiobjectif*. PhD thesis, Université Lille 1 - Sciences et Technologies, 2009.

[LK08] F. Lauri and A. Koukam. A Two-Step Evolutionary and ACO Approach for Solving the Multi-Agent Patrolling Problem. In *WCCI, Hong-Kong, China*, 2008.

[LPJF03] Xiang-Yang Li, Peng-JunWan, and Ophir Frieder. Coverage in Wireless Ad-hoc Sensor Networks. In *Computers, IEEE Transactions, vol 52, Issue 6*, pages 753–763, 2003.

[LPRC01] P. Lacomme, C. Prins, and W. Ramdane-Chérif. Competitive genetic algorithms for the Capacitated Arc Routing Problem and its extensions. In *Applications of evolutionary computing, Lecture Notes in Computer Science 2037, Springer*, pages 473–483, 2001.

[LPT03] P. Lacomme, C. Prins, and A. Tanguy. Optimisation par colonies de fourmis pour les tournées sur arcs. In *Conférence Francophone de MOdélisation et SIMulation, MOSIM'03, Toulouse (France)*, 2003.

[Meu02] Hervé Meunier. *Algorithmes évolutionnaires parallèles pour l'optimisation multi-objectif de réseaux de télécommunications mobiles*. PhD thesis, Université des Sciences et Technologies de Lille, 2002.

[Mic04] Fabien Michel. *Formalisme, outils et éléments méthodologiques pour la modélisation et la simulation multi-agents*. PhD thesis, Université Montpellier II, 2004.

[MKQP01] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak. Exposure in Wireless Ad Hoc Sensor Networks. In *7th Annual International Conference on Mobile Computing and Networking*, pages 139–150, 2001.

[MM03] A. Muñoz-Melendez. *Coopération située : Une approche constructiviste de la conception de colonies de robots*. PhD thesis, Université Pierre et Marie Curie, 2003.

[MPS94] L. Y. Maystre, J. Pictet, and J. Simos. Méthode multicritères electre : Description, conseils pratiques et cas d'application à la gestion d'environnement. In *Presses polytechniques et universitaires romandes, Lausanne*, 1994.

[MRZD02] A. Machado, G. Ramalho, J. D. Zucker, and A. Drogoul. Multi-Agent Patrolling : an Empirical Analysis of Alternatives Architectures. In *in Proc. of 3rd MABS*, pages 155–170, 2002.

[MRZD03] A. Machado, G. Ramalho, J. D. Zucker, and A. Drogoul. Multi-Agent Patrolling : an Empirical Analysis of Alternatives Architectures. In *Springer-Verlag, Berlin*, 2003.

[Pad09] Nikhil Padhye. Comparison of archiving methods in multi-objective particle swarm optimization (mopso) : Empirical study. In *GECCO'09, ACM, New York, USA*, 2009.

[PF02] D. Pellier and H. Fiorino. Approche multi-agent pour l'exploration coordonnée d'un environnement labyrinthique 2D. In *Journée Française d'Intelligence Artificielle Distribuée et des Systèmes Multi-agents (JFIADSMA'02), Lille*, pages 180–192, 2002.

[PFM07] P. Pellegrini, D. Favaretto, and E. Moretti. Multiple Ant Colony Optimization for a Rich Vehicle Routing Problem : a Case Study. In *Department of Applied Mathematics, University Ca' Foscari of Venice, Italy*, pages 1–31, 2007.

[Pou11] Cyril Poulet. Étude du Problème de la Patrouille Multi-Agents en Système Ouvert. In *RJCIA'11*, 2011.

[RSD02] M. Reimann, M. Stummer, and K. Doerner. A savings based Ant System for the Vehicle Routing Problem. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO, Morgan Kaufmann, New York)*, pages 1317–1325, 2002.

[Sam08] Ludovic Samper. *Modélisations et Analyses de Réseaux de Capteurs*. PhD thesis, INPG, 2008.

[Sch85] David Schaffer. Multiple objective optimisation with vector evaluated genetic algorithm. In *First International Conference on Genetic Algorithm*, pages 93–100, 1985.

[SD02] T. Stützle and M. Dorigo. A Short Convergence Proof for a Class of Ant Colony Optimization Algorithms. In *IEEE Transactions on Evolutionary Computation, Vol.6, no.4)*, pages 358–365, 2002.

[SdVS04] J. B. Seung, Gustavo de Veciana, and Xun Su. Minimizing Energy Consumption In Large-scale Sensor Networks Through Distributed Data Compression And Hierarchical Aggregation. In *IEEE Journal on Selected Areas in Communications*, pages 1–10, 2004.

[Sem04] François Sempé. *Auto-organisation d'une collectivité de robots : Application à l'activité de patrouille en présence de perturbations*. PhD thesis, Université Pierre et Marie Curie, 2004.

[Sim01] Olivier Simonin. *Le modèle satisfaction-altruisme : coopération et résolution de conflits entre agents situés réactifs, application à la robotique*. PhD thesis, Université Montpellier II, 2001.

[Sim10] Olivier Simonin. Contribution à la résolution collective de problème - Modèles d'auto-organisation par interactions directes et indirectes dans les SMA réactifs et robotiques. *HDR, Université Henri Poincaré (Nancy I)*, 2010.

[SRa04] H. Santana, G. Ramalho, and al. Multi-Agent Patrolling with Reinforcement Learning. In *3rd AAMAS*, pages 1122–1129, 2004.

[SRT10] G. Sampaio, G. Ramalho, and P. Tedesco. A technique inspired by the law of gravitation for the timed multi-agent patrolling. In *22nd IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 113–120, 2010.

[Tei01] Jurgen Teich. Pareto-Front Exploration with Uncertain Objectives. In *Springer-Verlag, Berlin Heidelberg*, pages 314–328, 2001.

[Vel99] D. A. Van Veldhuizen. *Multiobjective Evolutionnay Computation : Classifications, Analyses, and New Innovation*. PhD thesis, Air Force Institute of Technology, AFTI/DS/ENG/99-01, 1999.

[VUC97] M. Valenzuela and E. Uresti-Charre. A non-generational genetic algorithms for multiobjective optimization. In *Seventh International Conference on Genetic Algorithm*, pages 658–665, 1997.

[WCB00] R. H. Wendi, A. Chandrakasan, and H. Balakrishnan. Energy-Efficient Communication Protocol forWireless Microsensor Networks. In *33rd Hawaii International Conference on System Sciences, IEEE*, 2000.

[WIP+05] G. Wang, M. J. Irwin, B. Piotr, Haoying Fu, and Tom La Porta. Optimizing Sensor Movement Planning for Energy Efficiency. In *ISLPED'05, San Diego, California, USA*, 2005.

[WLB96] I. A. Wagner, M. Lidenbaum, and A. Bruckstein. Cooperative Covering by Ant-Robots Using Evaporating Traces. In *Technical Report CIS9610, CS Dept., Technion, Haifa, Israël*, 1996.

[WLB99] I. A. Wagner, M. Lindenbaum, and A. M. Bruckstein. Distributed Covering by Ant-Robots Using Evaporating Traces. In *IEEE Transactions on Robotics and Automation, (vol 15,num 5)*, pages 918–933, 1999.

[WZ09] G. Wenyin and C. Zhihua. An improved multiobjective differential evolution based on pareto-adaptive epsilon-dominance and orthogonal design. In *European Journal of Operational Research 198(2)*, pages 576–601, 2009.

[YB96] H. Yamaguchi and G. Beni. Distributed autonomous formation control of mobile robot groups by swarm-based pattern generation. In *in Proc. of the Second Int. Symp. on Distributed Autonomous Robotic Systems DARS-96, Springer Verlag*, pages 141–155, 1996.

[YCX06] Liang Yuan, Weidong Chen, and Yugeng Xi. Energy-Efficient Aggregation Control for Mobile Sensor Networks. In *Springer-Verlag Berlin Heidelberg*, pages 188–193, 2006.

[YDCW05] Shuhui Yangy, Fei Daiz, Mihaela Cardeiy, and Jie Wuy. On Multiple Point Coverage in Wireless Sensor Networks. In *Mobile Adhoc and Sensor Systems Conference, IEEE*, 2005.

[YF04] O. Younis and S. Fahmy. HEED : A Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad Hoc Sensor Networks. In *IEEE Transactions on Mobile Computing, vol 3, no 4*, pages 366–379, 2004.

[YWM05] Liyang Yu, Neng Wang, and Xiaoqiao Meng. Real-time Forest Fire Detection with Wireless Sensor Networks, IEEE. In *proc IEEE MABS*, 2005.

[ZHLA02] H. Zouari, L. Heutte, Y. Lecourtier, and A. Alimi. An overview of classifier combinaison methods in pattern recognition. In *RFIA, vol.2, Angers, France*, pages 499–508, 2002.

[ZSDT02] R. Zlot, A. Stentz, M. B. Dias, and S. Thayer. Multi-Robot Exploration Controlled by a Market Economy. In *In Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA), Washington, DC*, pages 3016–3023, 2002.

Annexe A

Topologies de graphes

Les différentes topologies de graphes utilisées pendant la validation expérimentale de notre approche sont représentées sur les figures A.1 à A.6.

FIGURE A.1 – MapA : Graphe de 50 noeuds et 104 arêtes.

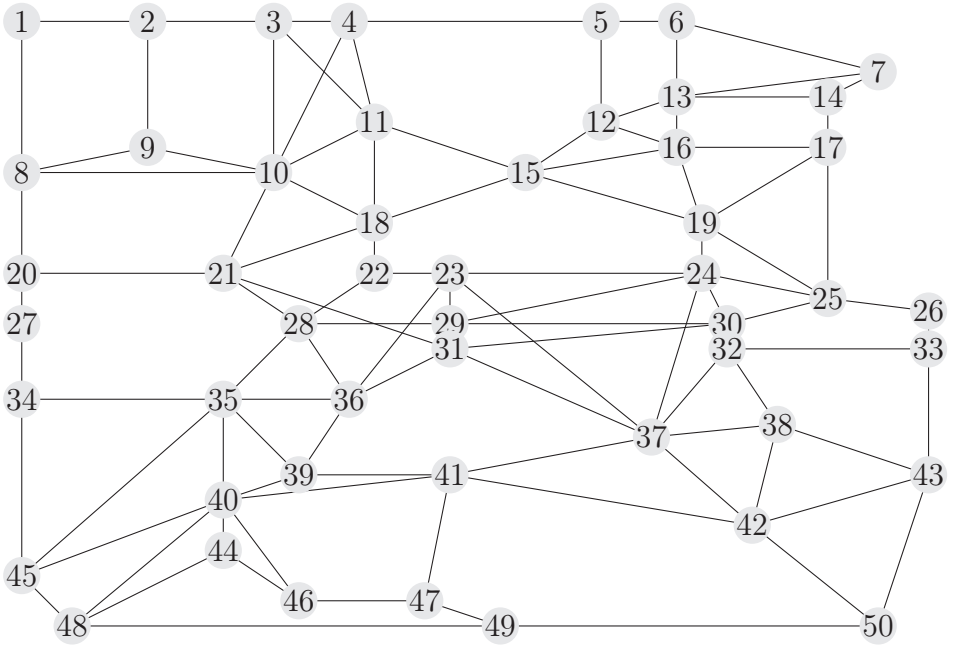


FIGURE A.2 – MapB : Graphe de 50 noeuds et 69 arêtes

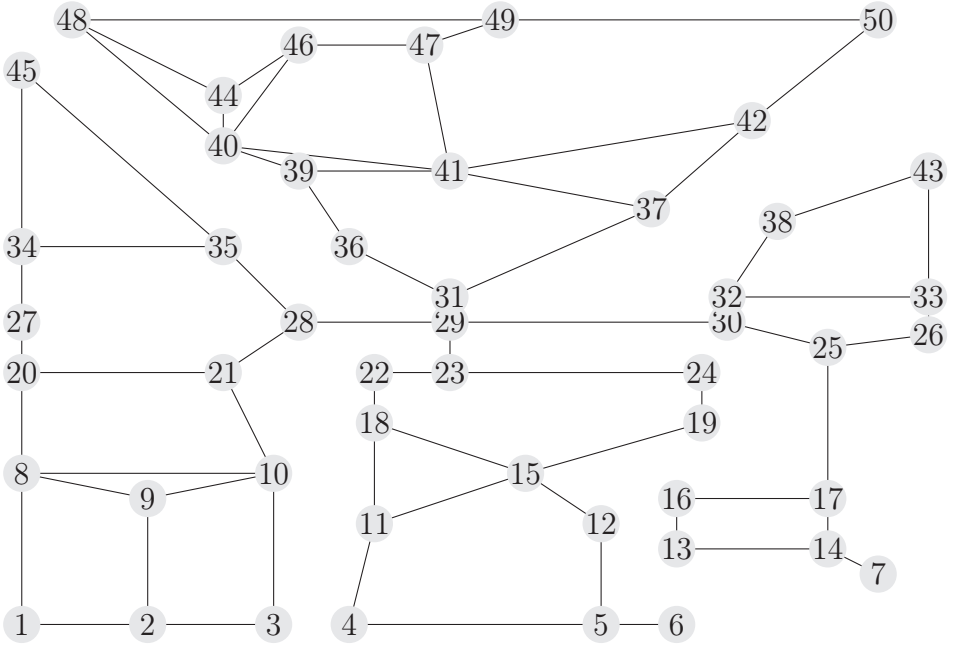


FIGURE A.3 – Cercle : Graphe de 56 noeuds et 56 arêtes

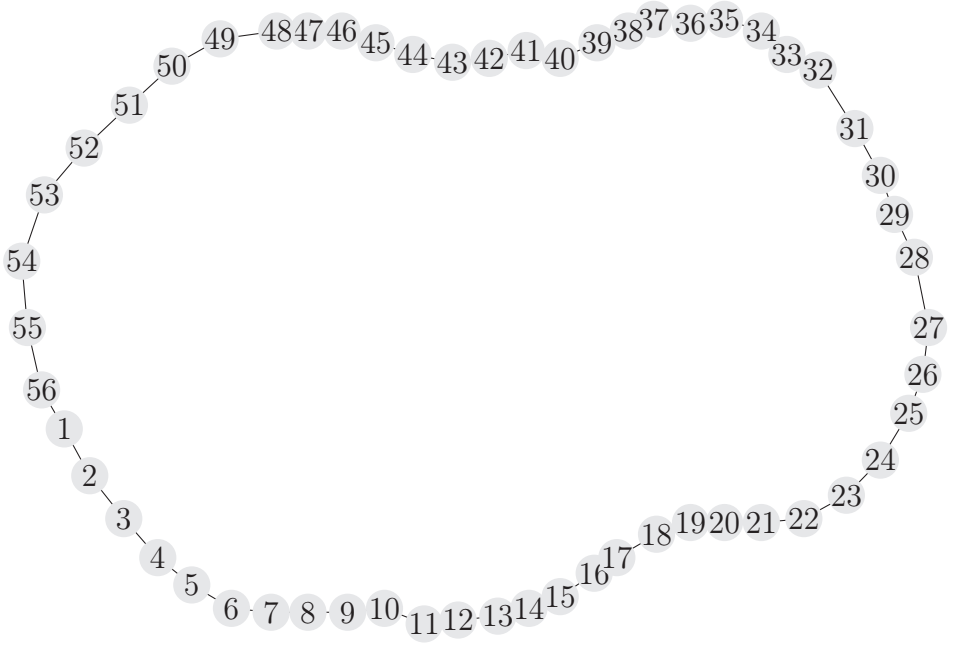


FIGURE A.4 – Corridor : Graphe de 70 noeuds et 69 arêtes

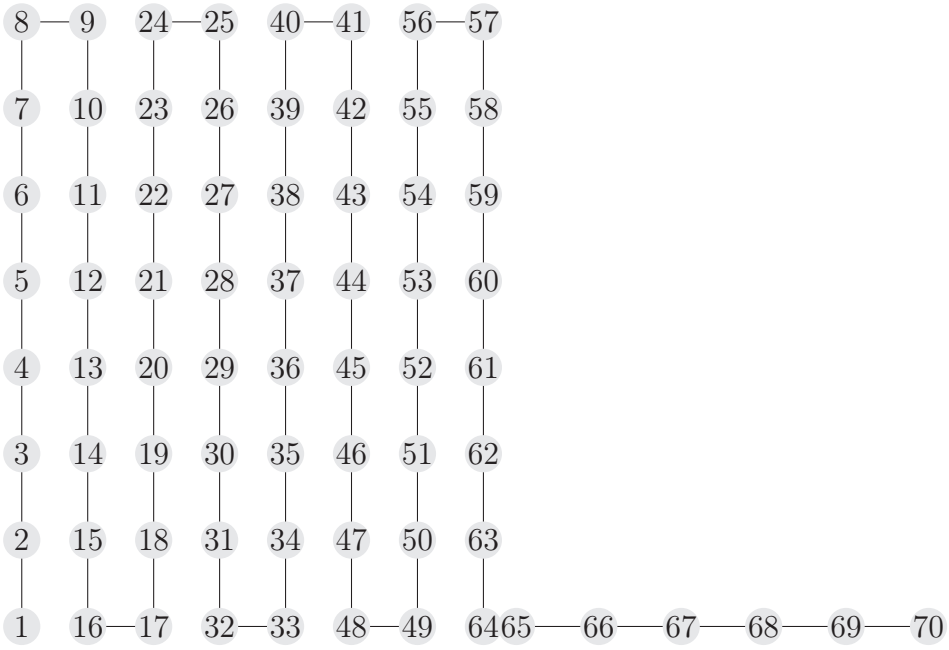


FIGURE A.5 – Grille : Graphe de 80 noeuds et 142 arêtes

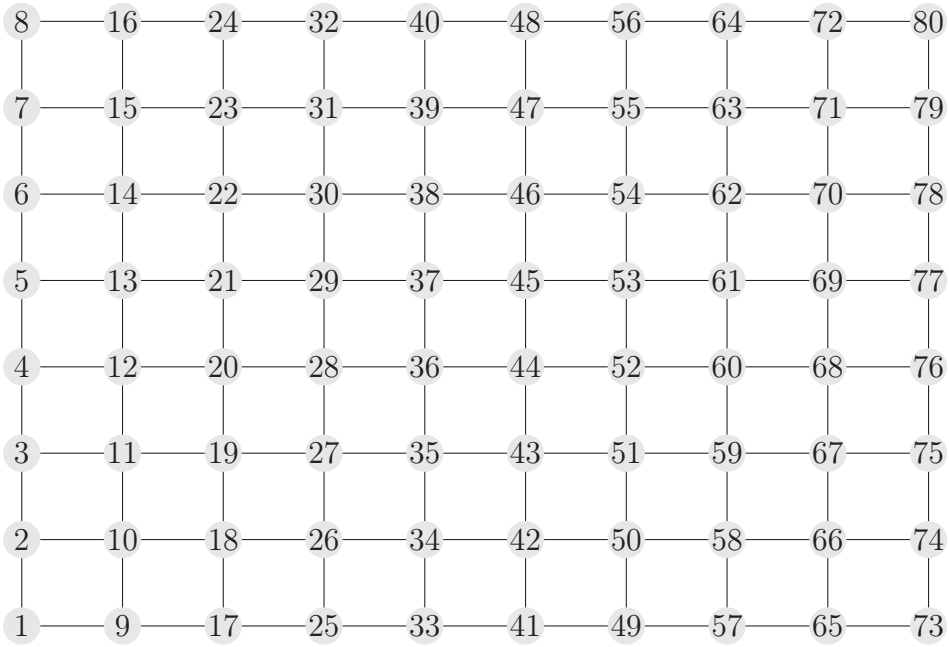
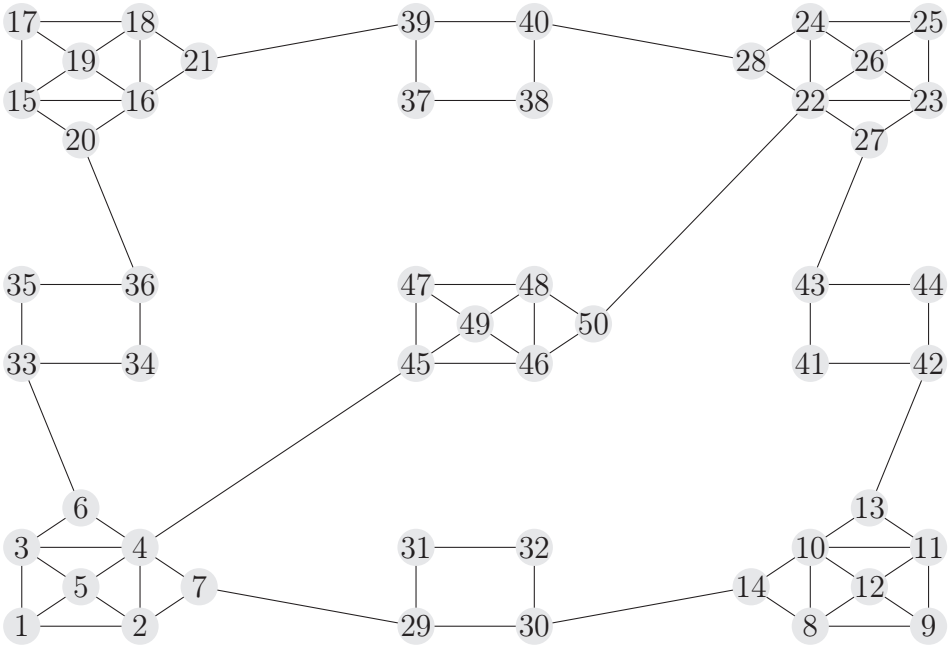


FIGURE A.6 – Island : Graphe de 50 noeuds et 84 arêtes



Résumé :

Pouvoir lutter efficacement contre certains fléaux comme les incendies de forêt, les feux de brousse ou les catastrophes naturelles constitue un enjeu majeur dans plusieurs villes du monde. Avec l'avènement de la technologie de pointe représentée par les réseaux de capteurs, la détection de ces phénomènes devient plus aisée. En effet, des capteurs peuvent être déployés dans des zones difficiles d'accès et s'ils sont suffisamment nombreux pour couvrir la totalité de l'environnement à surveiller, une alerte peut être directement donnée par le capteur ayant détecté un certain type d'évènement (feu, secousse sismique...). Le centre de contrôle ayant reçu l'alerte peut ensuite décider d'intervenir sur la zone en cause. Nos travaux se situent dans ce cadre de la détection de phénomènes par un réseau de capteurs, en supposant que l'environnement est connu et que les capteurs sont mobiles, sans fil et en nombre insuffisant pour couvrir la totalité de l'environnement à surveiller. Parler de surveillance par un nombre faible d'entités mobiles nécessite de parcourir régulièrement certaines zones critiques de l'environnement, ce qui peut s'apparenter à une tâche de patrouille.
Dans le cadre de cette thèse, nous nous sommes focalisés sur la détermination de stratégies de patrouille multi-capteurs appliquée à la détection d'évènements. Un problème similaire au nôtre est celui de la patrouille multi-agents dans un environnement connu. Ce problème consiste à faire visiter régulièrement les noeuds d'un graphe (représentant l'environnement) par des agents. Les capteurs peuvent être considérés comme des agents ayant des ressources limitées, en terme d'énergie en particulier. Le cadre de la patrouille multi-agents et les techniques proposées pour le résoudre ne peuvent pas être utilisés ici. Après avoir formulé mathématiquement le problème de la patrouille multi-capteurs appliquée à la détection d'évènements, nous proposons une technique de résolution approchée basée sur des colonies de fourmis. Des simulations ont été réalisées en considérant différents scenarii (topologies d'environnement, populations de capteurs, apparitions des événements) afin d'évaluer la pertinence de notre approche. Les résultats expérimentaux montrent que notre approche permet de déterminer des stratégies de patrouille satisfaisantes dans la majorité des scenarii.

Mots-clés : Patrouille multi-capteurs, colonie de fourmis, détection d'évènements, réseaux de capteurs.

SPIM

■ École doctorale SPIM - Université de Technologie Belfort-Montbéliard
F - 90010 Belfort Cedex ■ tél. +33 (0)3 84 58 31 39
■ ed-spim@univ-fcomte.fr ■ www.ed-spim.univ-fcomte.fr

utbm
université de technologie
Belfort-Montbéliard