Optimisation de la surveillance des sites industriels à risque par des robots mobiles

Marwa Gam

Normandie Université

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité AUTOMATIQUE, SIGNAL, PRODUCTIQUE, ROBOTIQUE

Préparée au sein de l'Université Le Havre Normandie

Optimisation de la surveillance des sites industriels à risque par des robots mobiles

Présentée et soutenue par
MARWA GAM

Thèse soutenue le 28/11/2022 devant le jury composé de		
M. HASSANE ABOUAISSA	MAÎTRE DE CONFERENCES (HDR), UNIVERSITE ARTOIS	Rapporteur du jury
M. SAID AMARI	MAÎTRE DE CONFERENCES (HDR), UNIVERSITE PARIS-SACLAY	Rapporteur du jury
MME ISABEL DEMONGODIN	PROFESSEUR DES UNIVERSITES, AIX-MARSEILLE UNIVERSITE	Membre du jury
M. ABDELKHALAK EL HAMI	PROFESSEUR DES UNIVERSITES, INSTITUT NATIONAL DES SCIENCES APPLIQUEES ROUEN NORMANDIE	Membre du jury
M. ACHRAF JABEUR TELMOUDI	MAÎTRE DE CONFERENCES (HDR), UNIVERSITE DE TUNIS (TUNISIE)	Membre du jury
M. DIMITRI LEFEBVRE	PROFESSEUR DES UNIVERSITES, Université Le Havre Normandie	Directeur de thèse

Thèse dirigée par DIMITRI LEFEBVRE (Groupe de Recherche en Electrotechnique et Automatique du Havre)

UNIVERSITÉ LE HAVRE NORMANDIE

591 ED PSIME

GREAH

Dédicaces

A celle qui a toujours cru en moi

A celle qui s'est sacrifié pour ma réussite

A celle qui m'a élevé avec amour et tendresse

A celle qui m'a soutenu et encouragé durant ces années d'études

A ma maman chérie.

A celle qui m'a bénie par ces prières

A celle qui m'a arrosé d'espoirs et de tendresse, à la source d'amour

A ma tante chérie Fattouma.

A celui qui m'a toujours supporté pour réussir

A celui qui m'a toujours guidé vers la bonne voie

A mon homme chéri Béchir.

A mon frère Anas et ma sœur Réfka qui m'ont toujours soutenu avec tant d'amour et de tendresse.

A ma copine Samar qui était toujours à mes côtés et qui m'a toujours soutenu avec amour dans les moments difficiles.

Aucune dédicace ne saurait exprimer mon amour éternel, mon respect et ma considération pour les sacrifices que vous avez consentis pour mon bien être et mon instruction.

A toute ma famille.

A tous mes ami(e)s.

A ceux que j'aime et qui m'aiment...

Remerciements

Mes trois années de thèse de doctorat étaient une aventure unique et passionnante. Une expérience enrichissante qui m'a beaucoup apporté sur le plan de la recherche intellectuelle. Cette thèse représente l'aboutissement d'un chapitre de ma vie. Je témoigne mon affection et ma profonde reconnaissance aux personnes qui ont contribué, de près ou de loin, à atteindre mes objectifs et à l'accomplissement de ma thèse.

Ainsi, je tiens à exprimer ma forte reconnaissance à mon directeur de thèse : M. Dimitri LEFEBVRE, d'abord pour la confiance qu'il m'a accordée en m'acceptant dans cette aventure, ensuite pour sa disponibilité de tous les instants, sa remarquable générosité et ses précieux conseils prodigués durant toutes ces années de thèse. Au-delà de mon encadrement qu'il a mené avec grande énergie, il a été un soutien permanent sur le plan personnel et professionnel. Je veux avec ces quelques mots lui exprimer toute ma gratitude. Ce travail est aussi le vôtre !

Merci à M. Édouard LECLERCQ et à M. Ahmed Ali Sofiane pour leur participation à mon comité de suivi de thèse et leurs suggestions toujours avisées.

Je tiens aussi à remercier Hassane Abouaissa et Said Amari pour avoir accepté d'être les rapporteurs de ma thèse et pour le temps qu'ils ont consacré, malgré leurs charges et leurs responsabilités, à l'évaluation de mes travaux avec soin. Je remercie également Mme Isabel Demongodin et M. Frederic Guinand pour avoir accepté d'examiner mon travail. Merci pour le temps et l'effort que vous m'avez consacrés et de m'avoir fait l'honneur de faire partie de mon Jury de thèse.

Cette thèse s'est déroulée au sein du Groupe de Recherche en Electrotechnique et Automatique du Havre (GREAH). Je remercie le directeur du GREAH : M. Georges BARAKAT de m'avoir accueilli au sein de son laboratoire. Je tiens à remercier tous mes ami(e)s, collègues, enseignants et personnels du laboratoire.

Je ne peux terminer sans adresser mes plus profonds et sincères remerciements aux êtres qui me sont le plus chers. Je pense bien évidement à ma très chère maman à qui je dois tout, à ma tante, mon frère et ma sœur qui m'ont toujours soutenu. Je n'aurais jamais pu aller aussi loin sans vous. Je vous remercie infiniment et j'espère vous rendre fiers. A mon homme qui m'a

toujours soutenu. Ton amour, ton regard et ton sourire donnent un sens à ma vie, cet aboutissement est aussi le tien. Il me tient aussi très à cœur de remercier mes amis : Samar DABGHI, Laila MESRAR, Oussama HAYANE, Sara HSAINI, Hamza CHAKRAA et Rania HAJ MANSOUR pour avoir été présents à des moments clés de cette thèse, et pour m'avoir encouragé par des gestes d'amitié et de bienveillance !

Table des matières

Liste des figures

Liste des Tableaux

Abréviations

ACO	Algorithme de colonies de fourmis
AGV	Véhicules à Guidage Automatique
UAV	Véhicules Aériens sans Pilote
WSN	Réseaux de Capteurs Sans Fil
BS	Recherche en faisceau
BAS	Recherche en faisceau A*
FBS	Recherche filtrée en faisceau
HFBS	Recherche en faisceau filtrée hybride
RBS	Recherche en faisceau avec récupération
GA	Algorithme génétique
GPS	Système de Positionnement Global
MRTA	Problème d'allocation de tâches multi-robots
MRR	Problème de routage multi-robot
MRS	Système multi-robots
MAS	Système multi-agents
MOLS	Recherche locale multi-objectif
MOMA	Algorithme mémétique multi-objectif
OAP	Problème d'affectation optimale
PN	Réseau de Petri
PL	Programmation linéaire
PPRT	Plan de Prévention des Risques Technologiques
SED	Système à événements discrets
SA	Recuit simulé
TSP	Problème du voyageur de commerce
mTSP	Problème du voyageur de commerce multiple
TSPTW	Problème du voyageur de commerce avec fenêtres de temps
MDHTSP	Problème de voyageur de commerce hétérogène à dépôts multiples
HMDMTSP	Problème de voyageur de commerce hétérogène multiple à dépôts multiples
TC	Conditions terminales
TS	Recherche Tabou
VRP	Problème de tournées de véhicules
CVRP	Problème de tournées de véhicules captifs
VRPTW	Problème de routage de véhicules avec fenêtres de temps
MDVRP	Problème de tournées de véhicules multi-dépôt
VRPSDPTW	Problème de tournées de véhicules avec livraison et ramassage simultanés et fenêtres temporelles

Chapitre 1 : Introduction générale

1.1.Contexte générale

Depuis la révolution industrielle, une grande attention est consacrée aux problèmes de sécurité des systèmes avec des impacts importants sur leur environnement. En effet, l'augmentation de la fréquence et de la gravité des événements accidentels imprévus et des actions malveillantes a des conséquences inacceptables sur l'image de l'industrie et l'environnement ainsi que de lourdes pertes matérielles et humaines [Angeli & Chatzinikolaou, 2004 ; Gockenbach & Borsi, 2008].

En plus, les industries manipulant des substances dangereuses sont souvent localisées dans des zones densément peuplées et des incidents sur ces sites peuvent entrainer des situations graves, comme le montre l'exemple des sites classés Seveso.

C'est pourquoi, les chercheurs et experts en sécurité ont déployé des efforts importants pour mieux comprendre et détecter précocement les événements inattendus qui peuvent survenir pendant le fonctionnement du système, identifier les vulnérabilités dans les systèmes de contrôle et développer de nouvelles solutions de sécurité [Kriaa et al., 2015]. [Stelzenmuller et al., 2013] ont mis en place un cadre de surveillance des zones soumises à des risques technologiques tels que les explosions, les fumées toxiques, les incendies, etc. Il existe également un autre type de risques, à savoir les risques naturels (tels que les inondations, les mouvements de terrain, les submersions, les tempêtes, les tremblements de terre, les éruptions volcaniques, les cyclones…) qui doivent être pris en compte. A ce titre, la sécurité et la prévention des risques ont fait récemment l'objet de nombreux travaux [Quarta et al., 2017 ; Johnsen, 2012].

Une approche de surveillance systématique des zones à haut niveau de risque est proposée dans notre travail de recherche pour assurer la sécurité en effectuant un ensemble de tâches de surveillance qui peut être évalué par des systèmes de capteurs intelligents portés par des agents mobiles. En effet, parmi les technologies de surveillance fondamentales, les agents mobiles peuvent être utilisés pour déplacer et déployer des capteurs dans les zones industrielles, afin de recueillir des informations et des mesures pendant le fonctionnement du système, de prévenir les risques et d'éviter l'apparition de pannes catastrophiques inattendues. Ils peuvent également être utilisés en situation de crise après qu'un incident se soit produit.
Dans ce contexte, ce travail de thèse fournit une réponse opérationnelle et méthodologique pour gérer les risques industriels en améliorant la surveillance des zones industrielles. L'objectif est

l'optimisation des patrouilles de surveillance avec des agents mobiles automatisés qui sont responsables de la surveillance. Ces agents sont constitués de véhicules à guidage automatique ou de véhicules aériens sans pilote qui portent divers capteurs pour collecter différentes mesures pendant les missions. Au-delà des spécificités de chaque classe d'agents, l'approche proposée est motivée par la nécessité d'inspecter des sites qui peuvent être dangereux ou difficiles d'accès. L'optimisation des missions est effectuée en respectant des contraintes fonctionnelles (par exemple, la priorité des opérations) et opérationnelles (par exemple, l'autonomie des agents) dans la double perspective de la configuration des patrouilles et de la planification des trajectoires, dans la mesure où ces aspects sont fortement corrélés. Dans ce travail de recherche, nous avons considéré en Normandie, France, deux sites classés Seveso comme la zone industrielle et portuaire de la ville du Havre et l'usine Lubrizol de la ville de Rouen.
Les questions auxquelles nous souhaitons répondre sont les suivantes.

- Combien d'agents mobiles sont nécessaires pour effectuer un ensemble donné de mesures ?
- Quels types de capteurs doivent équiper chacun de ces agents ?
- Comment définir la mission et la trajectoire de chaque agent ?

De telles questions sont étudiées comme un problème d'allocation de tâches multi-robots (MRTA), en particulier, un problème de routage multi-robot (MRR).

1.2.Contributions

Ce travail porte sur les méthodes d'optimisation afin de proposer une approche pour gérer et concevoir des patrouilles d'inspection composées d'agents mobiles associés à des ensembles de capteurs. Ces agents doivent accomplir les tâches de surveillance à moindre coût [Gam et al., 2020 ; Gam et al., 2021a ; Gam et al., 2022a].
Deux contributions principales sont présentées dans ce manuscrit :

La première contribution concerne l'optimisation par approche SED et l'algorithme de Dijkstra afin de réaliser les tâches de surveillance à un coût minimal et de trouver les meilleures trajectoires des agents mobiles. L'idée de base est de configurer le réseau en calculant le nombre d'agents et la distribution de capteurs sur les bases mobiles ; planifier pour chaque agent, la trajectoire optimale avec un coût minimal et coupler la configuration du réseau et la planification des trajectoires dans un problème d'optimisation globale. A cette fin, les réseaux

de Petri sont utilisés pour modéliser et formaliser le problème comme un problème d'estimation du marquage minimal [Gam et al., 2020 ; Gam et al., 2021a].

La deuxième contribution concerne les méthodes d'optimisation afin de proposer une approche basée sur la recherche en faisceau filtrée hybride et l'algorithme de Dijkstra dans le but d'éviter l'explosion combinatoire lors de l'exploration des solutions possibles. De plus, l'approche proposée prend en considération les contraintes possibles de précédence et d'énergie et peut traiter une grande variété de situations pratiques. Ainsi, le premier objectif de cette contribution est d'optimiser d'abord la configuration des patrouilles i.e., préciser la patrouille, notamment, le nombre de robots de chaque type. Ensuite, le deuxième objectif est d'optimiser les tâches de surveillance des robots mobiles et de planifier les trajectoires des robots afin de minimiser le coût global de la patrouille par rapport aux objectifs de surveillance. Les deux aspects sont combinés dans un seul problème d'optimisation et cette nouvelle perspective est certainement la contribution la plus importante de notre travail de recherche [Gam et al., 2022a].

1.3.Organisation du mémoire

Le reste du manuscrit est organisé en cinq chapitres.

Dans le deuxième chapitre, les missions de surveillance et les moyens d'inspection des sites à risque sont présentés. Ensuite, un état de l'art des contributions récentes du domaine est proposé. Puis, la différence entre les approches centralisées et distribuées ainsi qu'un aperçu sur les protocoles de communication sont présentés. Les hypothèses de travail, la formalisation et la modélisation des principaux éléments qui sont exploités dans notre travail sont aussi introduits et présentés.

Dans le troisième chapitre, les différents problèmes d'optimisation combinatoires en rapport avec notre travail ainsi que les différentes méthodes de résolution exactes et approchées sont détaillés. La différence entre les approches hors ligne, incrémentale et en ligne est présentée.

Dans le quatrième chapitre, l'utilisation des outils de modélisation des systèmes à évènements discrets (SED) comme les réseaux de Petri et les graphes d'états ainsi que les principales étapes de la résolution du problème sont détaillés. La méthode proposée utilise deux niveaux pour la modélisation de l'environnement à surveiller afin d'optimiser d'abord la configuration des patrouilles d'inspection en calculant le nombre de robots et la distribution de capteurs. Cette

méthode permet ensuite de résoudre le problème d'optimisation globale i.e., la détermination de la trajectoire optimale avec un coût minimal pour chaque robot et le couplage de la configuration du réseau et de la planification des trajectoires.

Dans le chapitre cinq, sur la base de la modélisation proposée dans le deuxième chapitre, une approche basée sur la recherche en faisceau filtrée hybride (HFBS) et l'algorithme de Dijkstra est proposée. Cette approche prend en compte les éventuelles contraintes de précédence et d'énergie et peut traiter une grande variété de situations pratiques. L'objectif de cette méthode est d'optimiser la configuration des patrouilles, l'allocation des tâches de surveillance des agents mobiles et leurs trajectoires afin de minimiser le coût global.

Pour terminer le mémoire, le chapitre six est consacré à la conclusion générale qui récapitule les résultats obtenus et propose des perspectives de poursuite de ce travail de recherche.

Chapitre 2 : Formulation et modélisation du problème

2.1. Introduction

La robotique mobile et, tout particulièrement, les systèmes multi-robots (ou Multi-robot system, MRS en Anglais) contribuent à améliorer la sécurité des systèmes avec un niveau de risque élevé. La robotique mobile est à la croisée de plusieurs disciplines techniques et scientifiques. Ses avantages attirent actuellement des chercheurs issus de nombreux domaines tels que la sécurité [Yi-Lin & Kuo-Lan ,2011], la recherche et le sauvetage [Nagatani et al., 2011], la surveillance [Marino et al., 2013], le déminage humanitaire [Khamis & ElGindy, 2012], la surveillance de l'environnement [Espina et al., 2011 ; Shkurti et al., 2012] et les soins de santé [Shiomi et al., 2013].

Dans ce travail de thèse, on s'intéresse à la surveillance des environnements à risques par l'utilisation des MRS. Il s'agit d'un type de système multi-agents (ou Multi-agent system, MAS en Anglais) dans lequel les robots évoluent dans un environnement physique ou simulé [Schneider, 2018]. De tels systèmes se caractérisent par des fonctions perceptuelles et décisionnelles qui permettent aux robots de se repérer et de se déplacer dans un environnement complexe [López et al., 2013]. Dans ce contexte, les Réseaux de Capteurs Sans Fil (ou Wireless Sensor Networks, WSN en Anglais) [Dou et al., 2017], les Véhicules à Guidage Automatique (ou Automates Guided Vehicle, AGV en Anglais) et les Véhicules Aériens sans Pilote (ou Unmanned Aerial Vehicle, UAV en Anglais), équipés de capteurs peuvent être considérés comme des solutions prometteuses pour recueillir des mesures utiles pendant le fonctionnement du système [Bahi et al., 2019 ; Kato et Wong, 2011 ; Ren et al., 2019 ; Guiochet et al., 2017 ; Lamine et al., 2020 ; Anjomshoaa et al., 2018].

Dans le reste de ce chapitre, nous allons présenter les missions de surveillance et les moyens d'inspection des sites à risque. Ensuite, nous allons présenter un état de l'art des contributions récentes du domaine. Nous montrerons par la suite la différence entre les approches centralisées et distribuées ainsi qu'un aperçu sur les protocoles de communication. La dernière partie du chapitre sera consacrée aux hypothèses de travail, à la formalisation de notre problème et à la modélisation des principaux éléments qui seront exploités dans la suite du mémoire.

2.2. Les missions de surveillance et moyens d'inspection des sites à risque

Un robot est défini comme « une machine physique équipée de capacités de perceptions, de décisions et d'actions qui lui permettent de se comporter de manière autonome dans son

environnement en fonction de ses fonctionnalités » [Filliat, 2011]. Cette définition est illustrée par le schéma de la Fig. 2. 1 pour montrer l'interaction d'un robot avec son environnement.

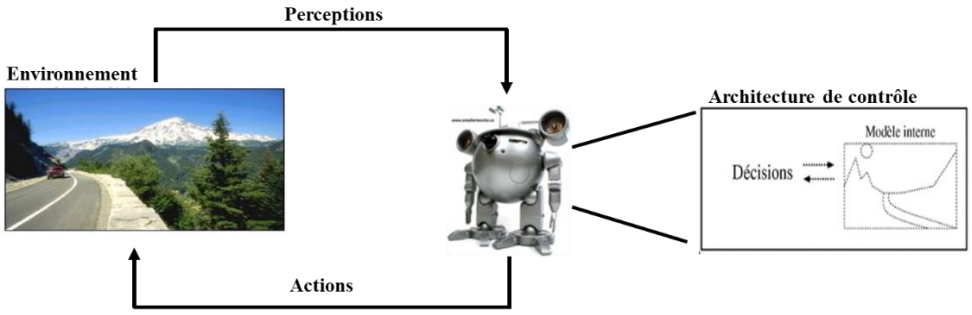


Figure 2. 1 : Interaction d'un robot avec son environnement

La robotique mobile est un domaine pluridisciplinaire à l'intersection de nombreuses disciplines telles que l'électronique, l'informatique, l'intelligence artificielle, l'automatique, la mécanique ou la mécatronique. De ce fait, le fonctionnement d'un système robotique dépend de la mission ou de la tâche à accomplir. Il est donc indispensable de définir un cadre formel pour décrire une mission robotique dans un environnement précis.

La complexité de la tâche et les difficultés environnementales influencent fortement le fonctionnement d'un robot. De plus, le flux d'informations à échanger lors d'une mission donnée doit être pris en considération. En effet, plus le robot possède d'informations précises sur l'environnement, plus il a de chance d'effectuer sa mission dans des bonnes conditions.

De plus, la mission peut être soumise à un ensemble d'exigences. Celles fixées d'une part en tant qu'objectifs pour la mission et celles fixées d'autre part par le cadre extérieur (interaction homme/robot). L'ensemble des exigences constituent le cahier des charges que l'utilisateur fixe pour spécifier la mission à réaliser. Parmi les critères fixés pour la mission, on trouve la robustesse, la proactivité, la réactivité, la durée de la mission, l'utilisation de ressources, les niveaux de performances et le taux de succès [Lampe, 2006].

La robotique mobile peut être exploitée ensuite dans des situations données et dans des domaines d'application précis, tels que :

- La robotique de loisir (compagnon, jouets) ;
- La robotique de service (bureaux, hôpital, maison) ;
- La robotique industrielle ou agricole (mines, récolte de productions agricoles, entrepôts logistiques) ;

- La robotique en environnement critique et dangereux (industriel, spatial, catastrophes naturelles, militaire).

Dans ce travail, nous nous intéressons à la robotique mobile utilisée en environnement industriel à risque pour éviter l'apparition d'incidents majeurs et améliorer la sécurité des systèmes [Biçen et al., 2014]. Au cours des deux dernières décennies, les systèmes de contrôle automatisés sont de plus en plus exploités comme outils fondamentaux de collecte de données par les scientifiques et développés pour la surveillance environnementale marine, terrestre et aérienne. Les AGV et les UAV font partie, au même titre que les WSN, des technologies de surveillance distribuées utilisées pour effectuer des tâches de surveillance réparties sur des sites industriels [Dunbabin et Marques, 2012].

2.2.1. Les réseaux de capteurs sans fil

Un réseau de capteurs sans fil (ou Wireless Sensor Networks, WSN en Anglais) permet de combiner la surveillance, les mesures environnementales, le suivi et le contrôle [Bahi et al., 2019]. Comme par exemple dans la Fig. 2. 2, le WSN possède de nombreuses applications [Li et Kara., 2017]. Les auteurs de [He et al., 2009] ont proposé un algorithme distribué pour maximiser la durée de vie du réseau. Dans [Bacco et al., 2017], les auteurs ont utilisé des nœuds de capteurs statiques et mobiles pour développer un réseau de surveillance innovant. Ils ont évalué cette technologie pour collecter, traiter et diffuser des informations pour étudier l'impact de la pollution et les conditions microclimatiques sur la qualité de vie dans les villes du futur. Les auteurs de [Jawhar et al., 2014] ont présenté différentes approches pour surveiller les infrastructures à l'aide d'un WSN afin de prolonger la durée de vie du réseau et de réduire la consommation d'énergie lors de la collecte et de la transmission des données.

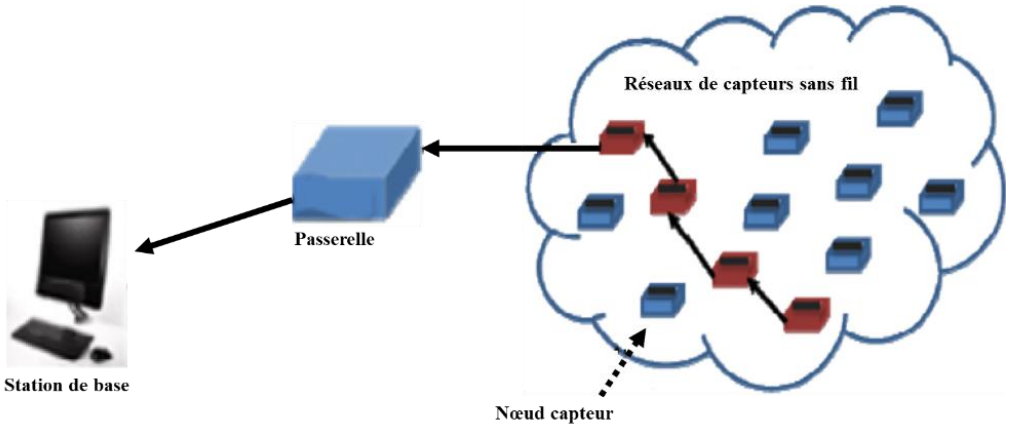


Figure 2. 2 : Exemple de WSN

2.2.2. Les véhicules aériens sans pilote

Les véhicules aériens sans pilote (ou Unmanned Aerial Vehicle, UAV en Anglais) présentent un outil non traditionnel de surveillance des populations et des lieux difficiles à atteindre (Fig. 2. 3). Les auteurs de [Ren et al., 2019] ont exploité la technologie UAV équipée de divers capteurs pour surveiller les zones minières. A travers cette application, ils ont montré l'efficacité des UAV en termes de coût, de précision, de flexibilité ainsi que de fiabilité d'acquisition des données. Dans [Liu et al., 2019], les auteurs ont développé un modèle de programmation en nombres entiers et deux heuristiques pour optimiser conjointement les itinéraires de vol et les emplacements des stations de base des UAV [Wu et al., 2020]. Dans [Ahmadzadeh et al., 2006], les auteurs ont présenté une stratégie basée sur la programmation en nombres entiers et un algorithme de planification de trajectoire pour une surveillance coopérative multi- UAV en tenant compte des contraintes spatio-temporelles. Dans le même contexte, [Khosiawan et al., 2018] ont proposé un système de planification pour UAV - AGV afin d'optimiser les trajets en tenant compte des contraintes temporelles.

Figure 2. 3 : Exemple de UAV

2.2.3. Les véhicules à guidage automatique

Le développement des véhicules à guidage automatique (ou Automated Guided Vehicles, AGV en Anglais) est également une technologie stimulante et émergente qui a attiré beaucoup d'attention dans la recherche et dans diverses applications au cours des dernières années (Fig. 2. 4). Pour les systèmes de transport intelligents, les AGV sont apparus comme une technologie rentable pour surmonter les difficultés de l'augmentation du trafic de véhicules [Liu et al., 2021]. Les auteurs de [Mohamad et al., 2018] ont mis en œuvre un algorithme génétique pour optimiser et améliorer les performances de l'environnement. De même, les AGV sont considérés comme une partie importante de la chaîne logistique dans l'industrie actuelle [Yao et al., 2018 ; De Ryck et al., 2020].

Contrairement aux travaux précédents, [Tanner, 2017] a développé une approche de surveillance coopérative pour coordonner des groupes d'AGV et d'UAV afin de localiser une cible mobile dans une zone donnée. Les AGV ont été le sujet de plusieurs autres travaux tels que [Nagatani et al., 2011 ; Liao et Su, 2011 ; Ruangpayoongsak et al., 2005] afin de surveiller et d'assurer la sécurité dans divers types d'environnements. Pour renforcer les patrouilles de surveillance dans les environnements dangereux, de nombreuses applications ont considéré l'optimisation des trajectoires comme un enjeu important pour les AGV. Par exemple, les auteurs de [Lai et al., 2020] ont formulé le problème de planification de trajectoires avec obstacles comme un problème de commande optimale en temps réel basée sur des réseaux de neurones. De plus, les auteurs dans [Yakovlev et al., 2020] ont étudié les algorithmes de planification de trajectoires et l'allocation de tâches afin d'être intégrés dans un système de contrôle pour le déploiement des AGV réels. En outre, l'algorithme de Dijkstra est utilisé dans [Zhong et al., 2020] pour réaliser une planification de trajectoires sans conflit, réduire le temps d'attente pour chaque tâche et améliorer l'efficacité opérationnelle du transport par AGV. Les auteurs de [Nishi et Tanaka, 2012] ont proposé un réseau de Petri décomposé en sous-réseaux de tâches et AGV. Ils ont considéré ainsi une méthode d'évitement des blocages pour assurer une répartition efficace et un routage sans conflit pour les systèmes AGV bidirectionnels.

Figure 2. 4 : Exemple d'AGV

Par rapport aux approches précédentes, on considère dans notre travail le problème d'optimisation des patrouilles de surveillance en faisant l'abstraction de la classe des agents (WSN, UAV ou AGV). De plus, nous étudions l'optimisation dans la double perspective de la configuration de patrouille et de la planification des trajectoires dans la mesure où ces aspects sont fortement corrélés.

2.3. Environnement

2.3.1. Environnement / site à risque

L'environnement est présenté comme le regroupement de plusieurs sites (Fig. 2. 5) susceptibles d'être affectées par les effets d'un évènement tel qu'un incendie, un accident ou une explosion. Les dommages se traduisent par des détériorations physiques (matérielles, corporelles, ou environnementales). L'environnement est soumis à deux catégories de risque

La première concerne les risques technologiques liés à la fabrication, le stockage et le transport par voie terrestre, fluviale ou maritime de produits dangereux. Parmi les produits considérés comme dangereux, on distingue principalement les substances énergétiques, radioactives ou chimiques. Ils sont susceptibles de provoquer des incendies, des dégagements de gaz et fumées toxiques ou radioactivité, des explosions, etc. Dans de telles situations, les dommages humains et matériels peuvent être considérables. Les sites SEVESO ont renforcé la prévention des accidents en imposant une gestion des risques par les industriels, sous l'autorité des États [Delvosalle et al., 1998].

La deuxième catégorie comprend les risques naturels comme les submersions, les inondations, les mouvements de terrain, les cyclones, les avalanches neigeuses, les tempêtes, les tremblements de terre, les séismes et éruptions volcaniques et les incendies de forêts.

Les efforts des chercheurs qui travaillent sur ces sujets visent à :

- Prévenir les actions malveillantes ou dangereuses,
- Développer des systèmes autonomes,
- Effectuer les tâches de surveillance de manière systématique et efficace.

Dans ce contexte, le Plan de Prévention des Risques Technologiques (PPRT) est élaboré. Ce PPRT est défini par la loi n° 2003-699 du 30 juillet 2003 relative à la prévention des risques technologiques et naturels et à la réparation des dommages. Le contenu et les dispositions de mise en œuvre du PPRT sont précisés par le décret n° 2005-1130 du 7 septembre 2005. L'objectif d'un PPRT est de fournir une réponse aux situations compliquées et difficiles en matière d'urbanisme et de mieux diriger et contrôler l'urbanisation future autour des établissements SEVESO. Il s'agit aussi de protéger les personnes. Compte tenu de l'état des connaissances et des pratiques d'une part, et de la vulnérabilité de l'environnement d'autre part, les établissements SEVESO doivent mettre en œuvre toutes les mesures de sécurité pour atteindre un niveau de risque aussi bas que possible [Tixier et al., 2002].

Dans ce contexte, les robots mobiles et les véhicules sans pilote équipés de capteurs apparaissent comme une solution prometteuse pour collecter des mesures utiles lors du

fonctionnement du système [López et al., 2013 ; Dou et al., 2017 ; Guiochet et al., 2017 ; Anjomshoaa et al., 2018 ; Lamine et al., 2020].

Figure 2. 5 : Un exemple d'environnement complexe à grande échelle

2.3.2. Cartographies

L'environnement peut être représenté en deux types de cartographies à savoir la carte métrique et la carte topologique.

La carte métrique représente l'environnement en deux dimensions à travers un ensemble d'objets associés à des positions dans un espace métrique. Cet espace est, le plus souvent, celui dans lequel s'exprime la position de l'agent mobile estimée par les données proprioceptives. Le principe est de détecter les objets et d'estimer leur position par rapport à l'agent mobile. Le calcul de la position de ces objets dans l'environnement se fait en prenant en considération la position estimée de l'agent.

La carte topologique permet de représenter l'environnement de l'agent mobile sous forme de graphe. Dans ce type de modélisation spatiale, la géométrie exacte de l'environnement n'est plus représentée directement mais l'espace est découpé en lieux ou sites qui correspondent aux nœuds d'un graphe. Une arête reliant deux nœuds peut être associée à une relation de connectivité entre deux sites. Elle indique la possibilité pour l'agent mobile de passer directement d'un site à un autre. La carte topologique mémorise en général les données [Kuipers et al., 1991 ; Thrun, 1999]. Dans une telle carte, un ensemble de sites et leurs liens de voisinage sont mémorisées.

Naturellement, la construction d'une carte topologique avec un modèle métrique est possible. Dans ce cas, les perceptions ne sont toutefois pas utilisées pour estimer la position relative des sites visités, mais seulement pour caractériser ces sites.

Les représentations hybrides sont également possibles pour avoir des caractéristiques à la fois topologiques et métriques [Filliat, 2011]. Dans la suite de notre travail, nous allons travailler sur les cartes topologiques.

2.4. Agents mobiles et flotte d'agents

Avec le développement des AGV, les robots mobiles ont parcouru un long chemin d'évolution dans la recherche et le développement. Afin de partager la charge de travail réalisée par des opérateurs humains, les robots mobiles peuvent être classés en trois types principaux en fonction de leur environnement d'exploitation : air, terre et eau (Fig. 2. 6). Dans des environnements peu ou pas structurés, la plupart de ces robots mobiles sont autonomes et effectuent diverses tâches étendant les capacités fonctionnelles humaines [Baylie, 2008].

Le choix d'un type de robot dépend de l'environnement d'exploitation et des exigences de la mission. Les robots aériens ou les drones effectuent, par exemple, des missions de livraison de produits en logistique et de photographie aérienne pour la surveillance dans les applications commerciales et de défense. Les robots terrestres transportent des matériaux, traversent et cartographient les mines, effectuent des interactions sociales dans des lieux publics, scannent et collectent des échantillons de roches dans l'espace. Les robots marins, opèrent sous l'eau, peuvent effectuer des missions de cartographie du fond marin, inspecter les épaves et les pipelines, et découvrent de niveaux gisements d'hydrocarbures ou minerais sous-marins [Khan, 2020].

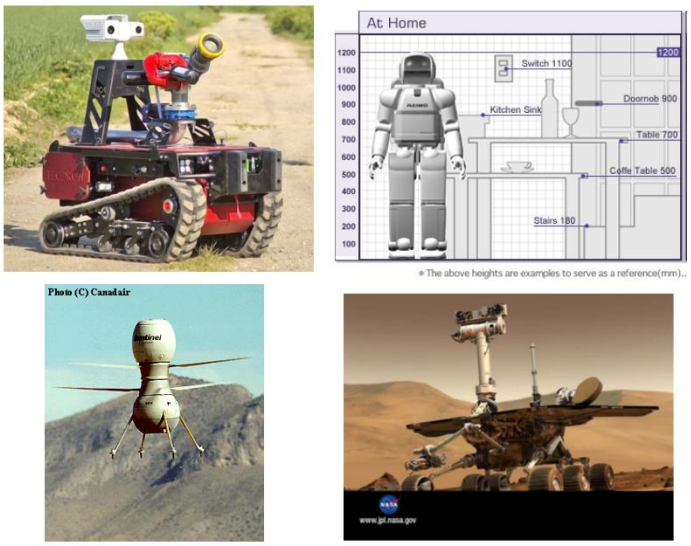


Figure 2. 6 : Les différents types de robots mobiles

2.4.1. Agents mobiles

Dans cette section, on se limite aux notions génériques d'agent mobile, notamment les plateformes mobiles et les capteurs utilisés par ces agents mobiles. De manière générale, on regroupe sous l'appellation agents mobiles, l'ensemble des robots à base mobile et les différents types de capteurs et de ressources.

Un AGV ou un UAV est considéré comme un agent artificiel et actif qui agit dans l'environnement. Un agent mobile est équipé de capacités de perceptions afin de prendre des décisions et d'agir dans l'environnement en utilisant ses effecteurs [Akli, 2007].

De plus, un agent mobile doit être capable de communiquer avec d'autres agents, de s'adapter à son environnement, de se protéger et de se déplacer d'un endroit à un autre. La coopération de plusieurs agents permet la construction d'un système de détection complet qui permet l'ajout et le retrait d'agents sans reconstruire l'ensemble du système [Labbassen, 2010].

Si les agents n'apportent pas fondamentalement de nouvelles capacités, ils ont néanmoins de nombreux avantages, notamment :

- Flexibilité : le nombre d'agents peut être adapté selon la taille du système d'information et du système surveillé.
- Fiabilité : en cas de mise hors service d'un agent, les autres agents peuvent remplacer l'agent défaillant. Le système de défense n'est pas compromis par la dégradation d'un seul agent. Un agent corrompu ne donne pas une image fausse de l'ensemble du système aux autres agents.
- Portabilité : les agents forment un système distribué qui permet de mieux détecter les attaques simultanées de plusieurs personnes réparties sur un réseau [Baylie, 2008].

2.4.1.1. Les plateformes mobiles

Par rapport à une plateforme fixe, une plateforme mobile est capable de se déplacer dans l'environnement, en fonction du type de locomotion dont elle dispose.

Une plateforme holonome a la capacité de se déplacer dans toutes les directions (en avant, sur les côtés et tourner sur elle-même) sur un plan à partir d'une position donnée. Elle offre ainsi plus de précision dans l'exécution des mouvements, ce qui simplifie le problème de planification de trajectoire [Khan, 2020].

De nombreuses plateformes simples ne sont pas holonomes comme le cas des voitures qui nécessitent davantage de manœuvres pour réaliser certaines trajectoires. Parfois, il est

nécessaire de faire un créneau pour réaliser un déplacement latéral. Ces contraintes doivent nécessairement être prises en compte lors de la planification de trajectoires. [Filliat, 2011].
Nous présentons rapidement dans cette section les différents types de plateformes mobiles utilisées en robotique, en focalisant sur les plateformes mobiles terrestres pour les environnements à risque.

- **Les plates-formes différentielles**

La configuration différentielle est l'une des configurations les plus utilisées pour les robots mobiles. Ce type de plateforme comporte deux roues commandées indépendamment. Une ou plusieurs roues sont ajoutées à l'avant ou à l'arrière du robot pour assurer sa stabilité (Fig. 2. 7). La spécification des vitesses des deux roues et la facilité de tourner sur place rendent la plate-forme plus facile à manipuler. Cette possibilité permet de traiter éventuellement le robot comme un robot holonome qui peut être utilisé dans des décombres et en milieu urbain. Dans ce cas, la commande du robot et la planification de déplacement peuvent être simplifiées [Filliat, 2011 ; Akli, 2007].

Figure 2. 7 : Exemple de plate-forme différentielle

- **Les plates-formes omnidirectionnelles**

Les plates-formes omnidirectionnelles sont quasiment holonomes puisqu'elles permettent de découpler le contrôle de la translation et de la rotation du robot.
Deux types de plateformes omnidirectionnelles existent :

- Les plateformes du premier type permettent une commande simple et relativement rapide puisque les changements de direction ne concernent que les roues et peuvent donc se faire très vite. Elles comportent trois ou quatre roues qui tournent à la même vitesse pour permettre la translation et un mécanisme qui permet d'orienter simultanément ces roues dans la direction souhaitée du déplacement (Fig. 2. 8). Le corps du robot lui-même n'effectue pas de rotation mais uniquement des translations. L'intérêt de cette plateforme est relativement limité en capacité de franchissement d'obstacles et son utilisation requièrent une surface plane.

Figure 2. 8 : Exemple de plate-forme omnidirectionnelle à roues orientables.

- Les plateformes du deuxième type utilisent des roues dites "suédoises", qui ne fournissent pas de résistance aux déplacements latéraux. La plateforme utilise trois roues dont les axes sont fixes (Fig. 2. 9). Les déplacements en rotation et dans toutes les directions sont obtenus en faisant varier individuellement les vitesses des roues. La plateforme tourne sur place lorsque les trois roues tournent dans le même sens, à la même vitesse. Lorsque deux roues tournent en sens opposés et une autre est fixe, la plateforme avance en direction de la roue fixe. Différentes combinaisons de vitesses permettent d'obtenir des déplacements quelconques [Filliat, 2011 ; Akli, 2007].

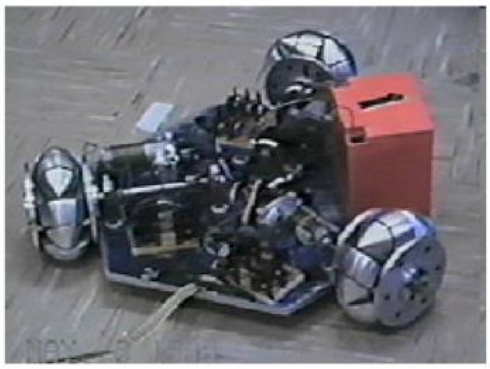


Figure 2. 9 : Exemple de plate-forme omnidirectionnelle à roues suédoises.

- **Les plateformes à pattes**

Des plateformes à deux, quatre ou six pattes peuvent également être utilisées pour se déplacer sur des terrains assez complexes. Les plateformes à six pattes sont relativement pratiques puisque le robot peut être en équilibre permanent sur au moins 3 pattes, ce qui facilite la commande. Les plateformes à deux ou quatre pattes sont plus complexes à commander, ce qui les rend relativement lentes. Ces plateformes sont rarement utilisées dans les applications qui ont besoin de précision en positionnement et en navigation. De telles plateformes commencent

à apparaître relativement à grande échelle comme le robot Nao de AldebarranRobotics (Fig. 2. 10). [Filliat, 2011].

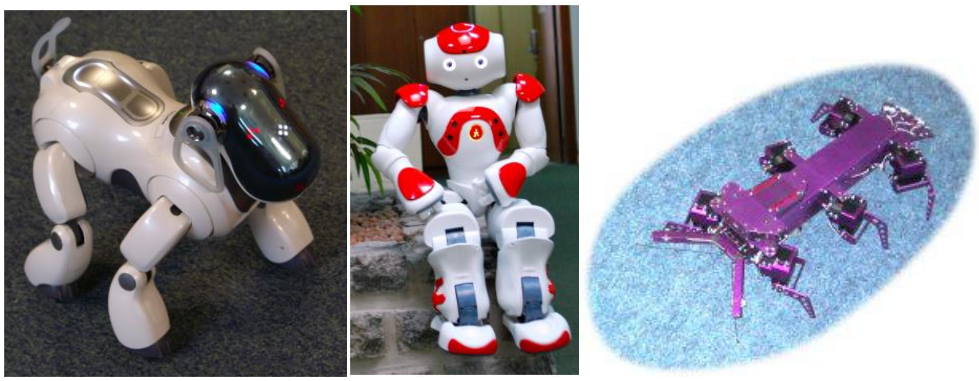


Figure 2. 10 : Exemples de robots à pattes (à gauche), Nao de AldebarranRobotics (au centre), Hexapode de AAI Canada (à droite).

- **Les plateformes de type véhicule**

Les plateformes de type véhicule sont non holonomes et néanmoins utilisées en robotique mobile. Ces véhicules sont difficiles à commander dans des environnements encombrés puisqu'ils doivent manœuvrer et ne peuvent pas tourner sur place (Fig. 2. 11). La réalisation des déplacements particuliers de ces plateformes est un problème de commande [Akli, 2007].

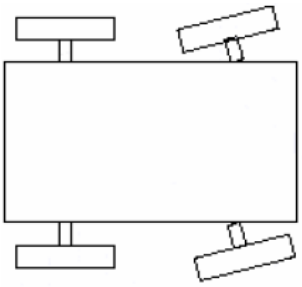


Figure 2. 11 : Plateforme de type véhicule

2.4.1.2. Les capteurs

Les capteurs sont définis comme des outils de perception qui permettent de gérer les relations entre le robot et son environnement. Ils permettent la mesure de variables qui servent à la commande du robot mais aussi à d'autres applications pour lesquelles le robot est considéré comme un capteur mobile. En particulier, on distingue deux types de capteurs [Akli, 2007 ; Filliat, 2011] :

- Les capteurs proprioceptifs qui mesurent l'état mécanique interne du robot comme les capteurs d'accélération, de vitesse et de position,

- Les capteurs extéroceptifs qui recueillent des informations sur l'environnement comme la mesure de distance, la détection de présence, etc.

➢ **Les capteurs proprioceptifs**

Les capteurs proprioceptifs sont utilisés par la commande du système. Ils interviennent notamment dans les boucles de régulation pour permettre à l'unité de commande de suivre correctement ou de mettre à jour la trajectoire en cours pour qu'elle soit conforme à celle exigée par la tâche. Parmi ces capteurs, on cite :

- L'odométrie qui permet d'estimer le déplacement à partir de la mesure de rotation des roues,
- Le radar qui sert à mesurer la vitesse du véhicule par effet doppler,
- Le gyromètre qui permet de mesurer l'orientation du corps sur lequel il est placé par rapport à un référentiel fixe et selon un ou deux axes,
- Le capteur de vitesse qui permet de mesurer la vitesse de rotation [Akli, 2007 ; Filliat, 2011].
- Les accéléromètres qui permettent de mesurer l'accélération [Akli, 2007 ; Filliat, 2011].

➢ **Les capteurs extéroceptifs**

Il existe deux types de capteurs à savoir les capteurs de contact et les capteurs sans contact. Les capteurs de contact exigent un contact avec l'objet sur lequel il va effectuer la mesure. Les capteurs sans contact prélèvent les mesures et les informations à distance à travers un rayonnement.

Parmi les types de capteurs extéroceptifs, nous citons :

- Les télémètres qui permettent de mesurer la distance par rapport à des cibles. Ils peuvent être : à infrarouge, à ultrasons ou laser.
- Le GPS (Système de Positionnement Global) qui fonctionne grâce à des balises placées sur des satellites en orbite terrestre.
- Les capteurs de proximité qui représentent une sous classe des capteurs sans contact. Ils sont caractérisés par l'absence de liaison mécanique entre le dispositif de mesure et l'objet cible. L'interaction entre le capteur et sa « cible » est réalisée par l'intermédiaire d'un capteur infrarouge ou plus souvent d'un champ (électrique, magnétique, électromagnétique) ou d'une caméra associée à un système d'analyse de l'image.

- Les capteurs de pression qui permettent de convertir les variations de pression en variations de tension électrique.

2.4.2. Flotte d'agents

La prise des mesures et l'affectation des tâches à un groupe d'agents mobiles est un problème complexe. De ce fait, les systèmes multi-robots sont composés d'agents qui collaborent pour effectuer ensemble les tâches. Certains objectifs qui sont impossibles à atteindre par un seul robot, deviennent réalisables grâce à cette collaboration. Les problèmes liés à l'organisation des agents mobiles au sein d'une flotte dépendent de la nature de la mission à réaliser.
L'utilisation des systèmes multi-robots est devenu plus populaire par rapport aux systèmes d'agent unique car elle permet :

- La résolution de tâches complexes : certaines tâches sont trop compliquées à réaliser par un seul robot, voire impossibles. Cette complexité peut également être due à la nature distribuée des tâches et/ou à la diversité des tâches en termes d'exigences différentes.
- L'augmentation de la performance : le temps d'exécution des tâches peut être considérablement réduit si de nombreux robots coopèrent pour effectuer les tâches en parallèle.
- L'accroissement de la fiabilité : la fiabilité du système augmente grâce à la redondance.
- La diminution du coût : avoir plusieurs robots simples revient souvent moins cher à mettre en œuvre que d'avoir un seul robot perfectionné.

Dans notre travail de recherche, nous nous intéresserons aux missions de surveillance effectuées par des agents mobiles dotés de capteurs permettant de réaliser les mesures souhaitées. Ces capteurs diffèrent d'un agent à un autre. Il existe alors deux types de flottes d'agents mobile :

- Flotte d'agents mobiles homogènes : les flottes d'agents homogènes, appelée aussi essaim, sont inspirées de la nature. Ceci a pour avantage de faciliter la programmation et de simplifier le coût de fabrication. Les auteurs de [Scheepers & Engelbrecht, 2016] ont conçu un algorithme qui utilise un nouvel optimiseur basé sur les techniques « essaim de particules » pour former des équipes multi-agents pouvant jouer au football de manière simple. Des algorithmes ont été présentés dans [Ahmadi & Stone, 2006] pour maintenir la connectivité de communication entre les robots et pour intégrer ou retirer des robots d'une équipe multi-robots tout en conservant la propriété biconnexe.

- Flotte d'agents mobiles hétérogènes : les flottes d'agents hétérogènes [Nishikawa et al., 2016 ; Simmons et al., 2000 ; Wurm et al., 2013] constituent la nouvelle tendance de la robotique mobile. Ce type de flottes comportent différents types de robots. Les auteurs de [Nishikawa et al., 2016] ont utilisé un algorithme génétique pour présenter une méthode de conception d'essaims robotiques hétérogènes. Ces robots autonomes ont été déployées pour effectuer des missions telles que la prévention des catastrophes, l'exploration sous-marine et le recueil de données géographiques. D'autres chercheurs dans [Simmons et al., 2000] ont décrit un système qui intègre la navigation autonome, la planification des tâches et une tâche de direction. Une interface graphique a aussi été conçue afin de contrôler plusieurs robots hétérogènes. Les tests ont montré la robustesse et l'efficacité du système, en particulier des stratégies de coordination. Une nouvelle approche a été présentée dans [Wurm et al., 2013] pour ajuster l'affectation des équipes hétérogènes de robots. Cette approche va au-delà des algorithmes d'affectation existants puisqu'elle prend explicitement en compte les actions symboliques. Cette méthode intègre aussi une approche de planification temporelle avec un planificateur traditionnel basé sur l'optimisation des coûts [Djellal, 2016].

En outre, dans nos travaux [Gam et al., 2020 ; Gam et al., 2021a], on a considéré la conception et la gestion de flottes d'agents mobiles composés de patrouilles de robots associés à des ensembles de capteurs qui dépendent des tâches à effectuer. En particulier, il s'agit de déterminer la configuration de la patrouille en calculant le nombre de robots et la distribution des capteurs pour que la patrouille puisse réaliser les tâches de surveillance à moindre coût.
Les problèmes d'allocation multi-tâches / multi-robots (ou Muti-Robots Task Allocation, MRTA en Anglais) est un problème difficile de la robotique mobile, en particulier lorsqu'il s'agit de robots hétérogènes peu fiables équipés de différents types de capteurs et ressources et qui doivent effectuer diverses tâches avec différentes exigences et contraintes d'une manière optimale. Ce problème peut être vu comme un problème de planification optimale où l'objectif est d'optimiser l'allocation d'un ensemble de robots à un ensemble de tâches sous des contraintes liées à la performance globale du système.

2.5. Réseaux et protocoles de communication

La communication entre les agents est indispensable. Chaque agent mobile doit connaitre en permanence l'état des autres agents et le degré d'accomplissement de la mission global. La

nature des communications dans une organisation centralisée ou décentralisée est un problème d'actualité.

Les approches d'allocation des tâches des systèmes multi-tâches / multi-robots peuvent être classées selon le paradigme organisationnel de la flotte. Ce paradigme montre comment les agents du système sont organisés en spécifiant les interactions entre les agents et les rôles de chaque agent au sein de la mission. Dans les sections suivantes, nous allons décrire les approches centralisées et décentralisées ainsi que leurs avantages et inconvénients.

2.5.1. Approche centralisée

Dans ce type de systèmes, chaque agent est connecté à un agent central (Fig. 2. 12-a). L'agent central distribue les tâches à effectuer et récolte les informations provenant de chaque agent [Horling & Lesser, 2004]. Cette approche évalue globalement l'état de la flotte et la remet à jour en traitant les informations provenant des capteurs. Elle assure aussi la sécurité des échanges d'information [Gerkey & Matarié, 2004].

Bien que les systèmes centralisés soient les plus utilisés [Brumitt & Stentz, 1998], ils présentent de nombreux inconvénients qui limitent leur utilisation. Le manque de robustesse est l'un des inconvénients les plus importants des systèmes centralisés ; si l'agent central échoue, tout le système échouera. De plus, l'évolutivité verticale du système est limitée puisque tous les agents sont connectés à l'agent central qui est considéré comme un goulot d'étranglement.

Dans la pratique, les approches entièrement centralisées peuvent être insolubles sur le plan numérique. De plus elles sont fragiles aux changements. En revanche, elles sont considérées comme la solution la mieux adaptée pour les problèmes d'allocation de tâches multi-robots où le nombre de robots et de tâches est petit, l'environnement est statique et l'information sur l'état global est disponible [Al-Yafi at al., 2009]. Les auteurs de [Coltin & Veloso, 2010] ont proposé un algorithme centralisé pour résoudre le problème d'allocation des tâches dans les systèmes multi-robots de façon à optimiser la durée de vie des capteurs du réseau. Dans [Liu & Kroll, 2012], les auteurs ont introduit l'approche centralisée pour résoudre un problème MRTA pour l'inspection dans une usine industrielle. Dans [Higuera & Dudek, 2013], l'approche MRTA basée sur la division équitable alloue une seule tâche globale entre un groupe de robots hétérogènes.

2.5.2. Approche décentralisée ou distribuée

La décentralisation répartit les tâches administratives et les pouvoirs entre les agents du système multi-agents [Horling & Lesser, 2004]. Dans un système distribué, il n'y a pas d'agent central

qui alloue les tâches aux autres agents. Chaque agent communique ses informations aux autres agents. Un agent du système décentralisé a besoin parfois d'échanger des informations avec d'autres agents afin de réaliser sa mission efficacement et en harmonie avec les autres agents. De nombreuses approches décentralisées sont proposées pour résoudre le problème MRTA. Les auteurs ont proposé dans [Giordani et al., 2010] une implémentation décentralisée de la méthode hongroise proposée pour résoudre le problème MRTA. Dans [Han-Lim et al., 2009], des approches décentralisées basées sur les enchères sont proposées pour résoudre le problème MRTA d'une flotte de robots mobiles autonomes. Une approche décentralisée de calcul évolutif est également proposée pour résoudre le problème MRTA en utilisant un algorithme génétique dans [Ping-an et al., 2009]. Une approche hiérarchique a été proposée dans [Khamis et al., 2011] comme approche décentralisée pour le problème MRTA.

Le principal avantage des systèmes décentralisés est leur robustesse : si l'un des agents tombe en panne, les autres agents continuent l'achèvement de leurs tâches [Elmogy, 2010]. Comme il n'y a pas d'agent central, de nouveaux agents peuvent être ajoutés en cas de panne par exemple. Cela signifie que l'évolutivité n'est plus un problème dans les systèmes décentralisés. Les approches décentralisées présentent aussi d'autres avantages par rapport aux approches centralisées, notamment la flexibilité et les faibles exigences de communication.

La Fig. 2. 12 montre la différence entre l'approche centralisée (Fig. 2. 12-a), décentralisée (Fig. 2. 12-b) et l'approche distribuée (Fig. 2. 12-c).

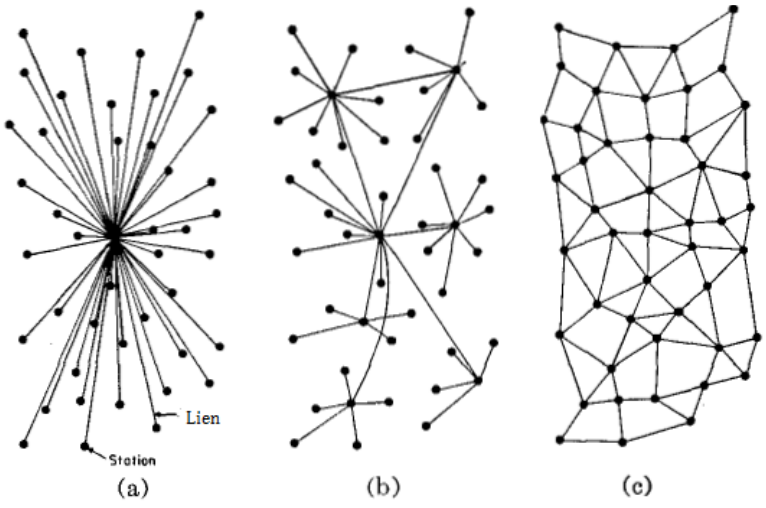


Figure 2. 12 : Types des approches : (a) centralisée (b) décentralisée (c) distribuée.

2.5.3. Protocoles de communication

La communication entre les robots mobiles peut se faire par plusieurs moyens. A l'origine, la communication se fait principalement par liaison série. Néanmoins, la communication peut se faire aussi via Wifi (802.11 b/g) ou Bluetooth (802.15.4).

La gestion des communications dans les MRS influence fortement leurs comportements : lorsque les communications entre les agents sont nombreuses, l'énergie nécessaire augmente et les problèmes de congestion des communications se manifestent. Le type de système multi-robots définit le modèle utilisé, à savoir système centralisé, graphe complet, …, et la fréquence de communication qui doit être implanté sur l'interface matérielle via un protocole de communication [Kancir, 2018].

Tableau 2. 1 : Comparaison des technologies de communication sans fils les plus courantes

Technologies	Poids	Coût	Consommation énergétique
WiFi	Moyen	Moyen	Moyenne
Zigbee	Léger	Faible	Faible
Bluetooth	Léger	Faible	Faible
Cellulaire	Léger	Haut	Moyenne
Acoustic	Lourd	Haut	Haute

2.6. Formalisation du problème étudié

Parmi les technologies de surveillance fondamentales, les véhicules à guidage automatique peuvent être utilisés pour déplacer et déployer des capteurs dans les zones industrielles. L'objectif est de recueillir des informations et des mesures pendant le fonctionnement du système, de prévenir les risques et d'éviter l'apparition de pannes catastrophiques inattendues. En d'autres termes, le problème d'allocation des tâches de surveillance via une flotte d'agents mobile étudie l'affectation tâche-agent afin d'atteindre les objectifs globaux du système [Korsah et al., 2013].

Les questions auxquelles nous cherchons à répondre sont les suivantes : Combien d'agents sont nécessaires pour effectuer les mesures ? Quels capteurs chaque agent doit-il porter ? Comment affecter chaque mesure de chaque site aux agents ? Quels sites doivent être visités par un agent donné ? Et dans quel ordre ?

Par conséquent, une approche de surveillance systématique hors ligne et centralisée est proposée dans la suite de notre travail. Notre objectif consiste à assurer la sécurité et la sûreté en effectuant un ensemble de tâches de surveillance avec des patrouilles composées d'agents mobiles équipés d'un ensemble de capteurs. Plus précisément, ce travail porte sur les méthodes d'optimisation afin de proposer une approche pour gérer et concevoir des patrouilles d'inspection composées d'agents mobiles associés à des ensembles de capteurs. Ces agents dépendent des tâches de surveillance pour les accomplir à moindre coût [Gam et al., 2020 ; Gam et al., 2021a ; Gam et al., 2022a]. Dans la suite, nous allons évoquer et développer les modèles utilisés.

2.6.1. Modélisation de l'environnement

Nous considérons un environnement constitué de N sites identifiés par leurs adresses a_i, $i = 1, \ldots, N$ (i.e., $A = \{a_1, a_2, \ldots, a_N\}$). Cet environnement est représenté par un maillage rectangulaire de taille $N = N_X \times N_Y$ cellules. Chaque cellule (x,y) définit un site et représente une zone spatiale qu'un agent peut visiter. Chaque cellule peut être visitée par plusieurs agents en même temps, c'est-à-dire que l'évitement des collisions est hors du sujet de notre travail. L'espace de travail peut comprendre des obstacles, des terrains plus ou moins accidentés ainsi que des chemins à sens unique.

A cet effet, un coût élémentaire c $(a_i, a_j) \geq 0$ est défini pour chaque déplacement entre deux cellules adjacentes $a_i = (x_i, y_i)$ et $a_j = (x_j, y_j)$. Les cellules $a_i = (x_i, y_i)$ et $a_j = (x_j, y_j)$ sont adjacentes si soit $x_i = x_j$ et $y_i = y_j + 1$ ou $y_i = y_j - 1$, soit $y_i = y_j$ et $x_i = x_j + 1$ ou $x_i = x_j - 1$, c'est-à-dire que les déplacements s'effectuent selon les axes x et y. Les spécifications des coûts élémentaires satisfont :

(i) le coût $c\ (a_i, a_j)$ est compris dans $]0, \infty[$;

(ii) $c\ (a_i, a_j) = \infty$ s'il n'y a pas de possibilité de passer de a_i à a_j ou si les cellules sont non adjacentes.

Il faut noter que les coûts ne sont pas nécessairement symétriques, c'est-à-dire que $c\ (a_i, a_j)$ peut être différent de $c\ (a_j, a_i)$. Pour les cellules non adjacentes a_i et a_j, on suppose que $c\ (a_i, a_j) = \infty$. La cellule d'adresse a_1 jouera le rôle particulier d'être le site de départ et aussi d'arrivée pour tous les agents de la patrouille.

Un tel environnement est formalisé avec un graphe pondéré et orienté, où chaque nœud représente une cellule, les déplacements possibles entre les cellules sont représentés par des transitions entre les nœuds du graphe et le poids $c\ (a_i, a_j)$ est associé à la transition permettant

de passer de a_i à a_j (les transitions de poids infini ne sont pas représentées) [Gam et al., 2020 ; Gam et al., 2021a ; Gam et al., 2022a].

2.6.2. Modélisation des agents mobiles

Les tâches de surveillance sont effectuées par un ensemble d'agents mobiles qui sont initialement positionnés dans le site a_1. Il existe N_r types d'agents mobiles en fonction des capteurs portés par les agents. En effet, chaque type *r* d'agent porte un sous-ensemble spécifique de capteurs *Sensor(r)* qui effectuent certaines mesures. On suppose que chaque capteur effectue une mesure donnée et on note par *Sensor* l'ensemble global des *q* capteurs (ou *q* mesures).
Nous considérons les hypothèses suivantes :

- Chaque site de la patrouille peut être visité par plusieurs agents en même temps ;
- Chaque site de l'environnement peut être visité plusieurs fois par le même agent ou par des agents différents ;
- Les agents peuvent faire plusieurs mesures sur un même site en même temps si nécessaire.

Chaque agent de type *r* possède un coût particulier qui est la somme du coût d'investissement et du coût d'exploitation. Le coût d'investissement dépend du robot mobile utilisé et des capteurs que porte l'agent et le coût d'exploitation dépend des trajectoires des robots mobiles. Dans ce travail, nous supposons que le coût d'exploitation pondère le coût de déplacement de site à site (le coût d'exploitation $c(r) = 1$ sera utilisé si aucun coût d'exploitation spécifique n'est considéré).
De plus, chaque agent de type *r* peut également satisfaire des contraintes concernant :

(i) Son autonomie par rapport à la consommation d'énergie ;
(ii) Le nombre maximal de capteurs qu'il peut transporter par rapport à son poids maximal ;
(iii) Un ordre partiel dans lequel les tâches (ie., les sites à visiter) doivent être effectuées.

D'une part, l'autonomie de chaque agent de type *r* est définie comme $T_{max}(r)$ et interprétée comme le temps de trajet maximal. Par conséquent, chaque agent de type *r* doit revenir sur le site a_1 avant d'atteindre son temps de parcours maximal $T_{max}(r)$. Lorsque les contraintes énergétiques ne sont pas prises en compte, alors $T_{max}(r) = \infty$. D'autre part, le nombre maximal de capteurs de chaque agent de type *r* est défini comme $S_{max}(r)$.

2.6.3. Modélisation des tâches de surveillance

Les tâches de surveillance doivent être effectuées dans certaines cellules spécifiques de l'environnement. L'ensemble des tâches de surveillance d'une mission *E* est défini par :

$E = \{(a_1, M(1)), (a_2, M(2)), \ldots, (a_{K-1}, M(K-1)), (a_K, M(K))\}$ (1)

avec :

- a_k, $k = 1,\ldots,K$: les adresses des cellules où une ou plusieurs mesures sont requises,
- $M(k) \subseteq Sensor$, $k = 1, \ldots, K$: l'ensemble des mesures à effectuer dans la cellule a_k (on a supposé que chaque mesure soit effectuée par un capteur spécifique). Dans la suite, nous désignons les cellules de *E* par le terme de sites. *E* comprend également le site a_1 où les agents commencent et terminent leur patrouille (il n'y a pas de mesure à prendre à cet endroit, donc $M(1) = \varnothing$).
- $\sigma(E)$: un ensemble de contraintes de précédence qui peut être défini sur *E*. Chaque contrainte est de la forme :

  $a_k < a_{k'}$, $k, k' \in \{1, \ldots, K\}$, signifiant que a_k doit être visité avant $a_{k'}$ et $\sigma(E) = \varnothing$ lorsqu'aucune contrainte de précédence n'existe.

2.7. Conclusion

Dans la première partie de ce chapitre, nous avons défini les missions de surveillance et moyens d'inspection des sites à risque. Par la suite, les différents éléments du problème ont été introduits, à savoir, l'environnement, les agents mobiles et les flottes d'agents. Nous avons aussi donné un aperçu sur les approches centralisées, décentralisées et les divers protocoles de communication qui peuvent être mis en œuvre. Enfin, des éléments de formalisation du problème ont été présentés dans la dernière partie du chapitre.

Dans le chapitre suivant, nous présentons quelques problèmes d'optimisation combinatoire en rapport avec notre travail et les différentes méthodes de résolution qui peuvent être utilisées.

Chapitre 3 : Méthodes d'optimisation combinatoires

3.1. Introduction

L'optimisation combinatoire est définie comme un cadre formel pour de nombreux problèmes et dans différents domaines [Belhoul, 2014]. C'est une branche de l'optimisation qui occupe une place très intéressante en informatique, en recherche opérationnelle et en mathématiques appliquées. Les problèmes d'optimisation combinatoire consistent à trouver la meilleure solution dans un nombre fini de solutions réalisables. Celles-ci se distinguent par un ensemble de conditions ou de propriétés, dites aussi contraintes, que doivent satisfaire toutes les solutions réalisables [Hao et al., 1999]. La résolution d'un tel problème permet de trouver une solution réalisable qui minimise (resp. maximise) une fonction objectif donnée [Solnon, 2008]. Les problèmes d'optimisation combinatoire sont souvent difficiles à résoudre [Belhoul, 2014]. En effet, la plupart de ces problèmes appartiennent à la classe des problèmes NP-difficiles et leur complexité est liée généralement à celle du problème de décision qui lui est associé. Afin de résoudre ce type de problème, de nombreuses méthodes de résolution ont été développées et classées en deux catégories principales : les méthodes exactes et les méthodes approchées [Hao et al., 1999].

Dans ce chapitre, nous présentons les différents problèmes d'optimisation combinatoires en rapport avec notre travail ainsi que les différentes méthodes de résolution exactes et approchées qui peuvent être utilisées. Nous montrerons par la suite la différence entre les approches hors ligne, incrémentale et en ligne.

3.2. Problèmes d'optimisation combinatoire

Le principe des problèmes d'optimisation combinatoire est de trouver la meilleure solution dans un ensemble discret nommé ensemble des solutions réalisables. Cet ensemble est généralement fini mais comporte un très grand nombre d'éléments. L'ensemble des solutions réalisables est décrit implicitement par des contraintes que les solutions doivent satisfaire. De plus, une fonction objective est introduite pour définir la notion de « meilleure » solution.
En outre, le nombre de solutions réalisables des problèmes d'optimisation combinatoires augmente de façon exponentielle en fonction de la taille du problème, ce qui exclut des méthodes de résolution basées sur l'examen de toutes les solutions réalisables [Belhoul, 2014].

3.2.1. Problème du voyageur de commerce (TSP)

Le problème du voyageur de commerce (ou Traveling Salesman Problem, TSP en Anglais) est un des problèmes NP-difficile les plus connus et étudiés en optimisation combinatoire. Le TSP a été formulé en 1954 [Dantzig et al., 1954]. Ce problème est celui d'un voyageur de commerce qui a *n* villes à visiter et qui souhaite mettre en place une tournée qui lui permette de passer une seule fois par chaque ville et de revenir à son point de départ (généralement nommé dépôt) pour un coût le plus bas possible. Le coût peut être la distance ou le temps [Mancel, 2004].

Il existe de nombreuses variantes du TSP comme le problème du voyageur de commerce multiple (ou Multiple Traveling Salesman Problem, mTSP en Anglais) qui consiste à résoudre l'allocation et l'optimisation d'un ensemble de routes pour *m* voyageurs de commerce. Ceux-ci doivent couvrir toutes les villes disponibles et retourner à leur ville de départ de telle sorte que chaque voyageur de commerce fasse un circuit en boucle [Khamis et al., 2015 ; Sarin et al., 2005 ; Bektas, 2006]. Depuis l'apparition du mTSP, de nombreuses variations sont apparues dans la littérature, telles que l'autorisation de multi-dépôts et l'ajout de contraintes spécifiques incluant l'ordonnancement et le nombre maximum de villes que chaque voyageur de commerce peut visiter [Khamis et al., 2015]. Une autre variante qui est obtenue par adjonctions de contraintes comme le TSP avec fenêtres de temps (ou Traveling Salesman Problem with Time Windows, TSPTW en Anglais). Cette variante consiste à effectuer la visite de chaque ville dans un intervalle de temps donné [Mancel, 2004 ; Gutin & Punnen, 2002 ; Laporte, 1992a]. Les auteurs de [Bae et al, 2019] ont présenté une nouvelle heuristique basée sur une technique primale-duale pour résoudre le problème de voyageur de commerce hétérogène à dépôts multiples (MDHTSP) tout en se concentrant sur l'allocation des tâches et les robots hétérogènes. D'autres auteurs ont présenté dans [Sundar & Rathinam, 2015] la programmation linéaire en nombres entiers mixtes et ont développé un algorithme exact basé sur une méthode branch and cut pour trouver une tournée optimale pour le problème de voyageur de commerce hétérogène à dépôts multiples (HMDMTSP). Dans [Öztürk & Kuzucuoğlu, 2015], une nouvelle approche basée sur la solution du problème du voyageur de commerce et la solution du plus court chemin de Dijkstra est proposée pour calculer la planification du chemin.
De plus, il existe de nombreuses applications de TSP qui sont des problèmes complexes comme les problèmes de séquencement de processus de fabrication ou d'optimisation de parcours en robotique et les problèmes de transport [Mancel, 2004].

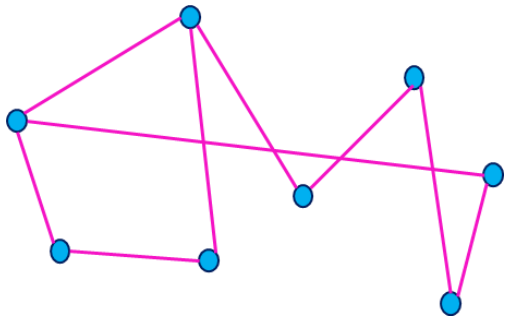


Figure 3. 1 : Exemple de solution du TSP

3.2.2. Problème de tournées de véhicules (VRP)

Le problème de tournées de véhicules (ou Vehicle Routing Problem, VRP en Anglais) fait partie des problèmes de la recherche opérationnelle et d'optimisation combinatoire qui existent depuis plusieurs décennies. Il a été défini pour la première fois par [Dantzig & Ramser, 1959]. Le VRP a été défini par extension du problème de base TSP en rajoutant différentes contraintes, hypothèses et objectifs. Il appartient à la classe des problèmes NP-complet [Chabot, 2015]. C'est un problème qui vise à résoudre l'allocation et à trouver un ensemble de trajectoires optimales pour une flotte de véhicules homogènes afin de livrer des articles aux clients [Mancel, 2004 ; Chao et al., 1996]. Pour traiter ce problème, une fonction de coût soumise à plusieurs contraintes doit être minimisée. Les auteurs dans [Cordeau et al., 2005] ont décrit le VRP comme la conception d'un ensemble d'itinéraires à moindre coût en vue de satisfaire la demande des clients tout en respectant la durée maximale pour chacune des routes et la capacité des véhicules.

De nombreuses variantes du VRP ont été introduites, citons le problème de tournées de véhicules captifs (CVRP) qui affecte chaque client à une route effectuée par un seul véhicule c'est-à-dire n'impose pas un nombre maximum de livraisons par véhicule [Ralphs, 2003], le problème de routage de véhicules avec fenêtres de temps (VRPTW) où les contraintes temporelles sont définies sous forme de fenêtres de temps et chacune de celles-ci est attribuée à chaque client, le problème de tournées de véhicules multi-dépôt (MDVRP) où les véhicules sont localisés dans plusieurs dépôts [Rachid, 2010 ; Chabot, 2015], etc.

Le VRP a été largement traité dans la littérature. Par exemple, [Wang et al., 2016] ont étudié dans l'industrie de la logistique, une variante pratique du problème de tournées de véhicules avec livraison et ramassage simultanés et fenêtres temporelles (VRPSDPTW). Ils ont mis en œuvre deux algorithmes, la recherche locale multi-objectif (MOLS) et l'algorithme

mémétique(1) multi-objectif (MOMA) pour résoudre un VRPSDPTW multi-objectif général (MO-VRPSDPTW) avec cinq objectifs. Les auteurs de [Usmani et al., 2010] ont présenté un modèle VRP intégré et ont développé une méthode de regroupement heuristique combinée avec une approche d'algorithme génétique GA pour résoudre le problème de routage multi-dépôts avec des véhicules hétérogènes. Selon le nombre de véhicules, les auteurs de [Nallusamy et al., 2009] ont utilisé une méthodologie de clustering des villes données et chaque cluster est attribué à un véhicule. Ils ont mis en œuvre le clustering K-Means et les algorithmes génétiques afin de générer un itinéraire optimisé pour chaque cluster/tour qui est considéré comme un VRP individuel. D'autres chercheurs ont présenté dans [Ghoseiri & Ghannadpour, 2010] une approche de programmation par objectifs et un algorithme génétique efficace adapté afin de modéliser et de résoudre le problème multi-objectif de VRPTW. De nombreux travaux ont mis en œuvre l'algorithme génétique afin de résoudre le VRPTW comme dans [Louis et al., 1999 ; Tan et al., 2001, 2006). Les auteurs de [Karakatič & Podgorelec, 2015] ont présenté également différentes approches, méthodes et opérateurs génétiques pour résoudre le MDVRP. Ils ont évalué l'efficacité de différentes variantes d'algorithmes génétiques sur des problèmes de référence standard.

Plusieurs applications de VRP ont été proposées dans l'industrie comme la livraison de journaux, de provisions, la collecte de déchets ou la tournée d'autobus scolaire, etc [Toth & Vigo, 2002 ; Cordeau et al., 2000 ; Braysy & Gendreau, 2005 ; Cordeau et al., 2001].

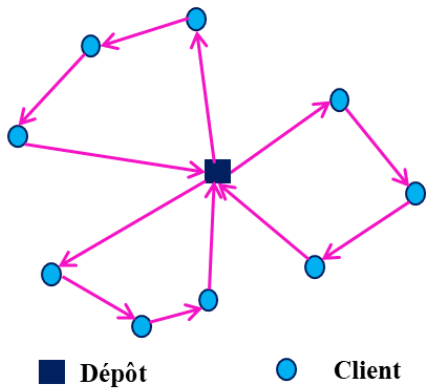


Figure 3. 2 : Exemple de solution du VRP

(1) Un algorithme mémétique est une hybridation entre les algorithmes génétiques et les algorithmes de recherche locale. L'idée principale de cet algorithme est d'utiliser le même processus de résolution que les algorithmes génétiques en ajoutant un opérateur de recherche locale en complément de celui de mutation.

3.2.3. Problème d'ordonnancement

Le problème d'ordonnancement consiste à ordonnancer, séquencer et répartir dans le temps un ensemble de tâches en prenant en compte les contraintes temporelles (contraintes d'enchaînement, délais, . . .) et les contraintes portant sur la disponibilité et l'utilisation des ressources requises par les tâches [Esquirol & Lopez, 1999]. Deux notions fondamentales interviennent dans un problème d'ordonnancement à savoir les tâches et les ressources. Indépendamment de leur disponibilité, les ressources sont connues a priori. Une tâche est un travail élémentaire dont la réalisation requière un certain nombre de ressources et d'unités de temps. Dans un contexte d'optimisation, on s'intéresse souvent à minimiser ou maximiser un critère, comme par exemple la durée totale de l'achèvement des tâches [Balas et al., 2008 ; Carlier & Chrétienne, 1982].

Les problèmes d'ordonnancement apparaissent dans divers domaines comme l'économie, l'informatique, l'industrie (problèmes d'ateliers, gestion de production), la construction (suivi de projet), l'administration (emplois du temps) [GOTHA, 1993].

3.2.4. Problème d'affectation linéaire

Les problèmes d'affectation traitent la question de savoir comment affecter n tâches à n machines de façon optimale. Ils se composent de deux composants : l'affectation en tant que structure combinatoire sous-jacente et une fonction objectif modélisant la « meilleure voie ». Les affectations (c'est-à-dire les couples tâche-machine) ont toutes un coût défini. L'objectif est de minimiser le coût total des affectations afin d'achever toutes les tâches. Le problème peut être décrit comme un graphe biparti $G = (V ; W ; E)$ où l'ensembles de sommets V correspond aux machines et l'ensemble de sommets W correspond aux tâches [Burkard & Eranda, 1999 ; Pentico, 2007]. [Kuhn, 1955] a proposé une méthode de calcul appelée algorithme hongrois pour résoudre ce problème avec une complexité $O(n^3)$. Par la suite, l'algorithme a été étendu par Munkres pour traiter le cas où le nombre de machines n'est pas le même que le nombre de tâches [Munkres, 2004 ; Bourgeois & Lassalle, 1971]. Bertsekas a proposé dans [Bertsekas, 1990, 1992] une "méthode intuitive qui fonctionne comme une véritable vente aux enchères où les agents se disputent les objets en augmentant leurs prix par appel d'offres".

3.2.5. Problème d'allocation de tâches multi-robots (MRTA)

Le problème d'allocation de tâches multi-robots (ou Multi-robot Task allocation, MRTA en Anglais) est l'un des problèmes les plus complexes des systèmes multi-robots, en particulier pour les robots mobiles hétérogènes équipés de différents types de capteurs. Le problème de l'allocation des tâches est un problème de décision dynamique [Korsah et al., 2013 ; Khamis et al., 2015]. Il doit être résolu de manière itérative dans le temps [Gerkey & Mataric, 2003].
Lorsque plusieurs robots doivent exécuter diverses tâches en tenant compte de contraintes et d'exigences différentes, la question à résoudre par le problème MRTA est la suivante : "Quel robot mobile doit exécuter quelles tâches ?". La réponse à la question clé de l'allocation des tâches consiste à trouver l'affectation d'un ensemble de tâches à une flotte de robots mobiles afin d'atteindre l'objectif global de manière efficace et fiable. Les approches d'allocation de tâches multi-robots peuvent être classées en fonction de la description du problème, du paradigme organisationnel utilisé, de la catégorie d'allocation de tâches, des techniques de résolution de problèmes et de la méthode de planification. [Gerkey & Mataric, 2004] ont introduit une analyse formelle et une taxonomie de la MRTA. Ils montrent comment la taxonomie qu'ils proposent peut être utilisée pour analyser et classifier les problèmes de MRTA, et aussi évaluer et comparer les solutions proposées.

L'étude de la taxonomie des problèmes MRTA est basée sur les caractéristiques principales des tâches, des robots et du temps. Trois axes servent à décrire les problèmes MRTA :

- **ST** vs **MT** : ST signifie que chaque robot est capable d'exécuter au maximum une tâche à la fois (ou Single-task robots, en Anglais), tandis que MT signifie que certains robots peuvent exécuter plusieurs tâches simultanément (ou multi-task robots, en Anglais).

- **SR** vs **MR** : SR signifie que chaque tâche nécessite exactement un robot pour la réaliser (ou Single-robot tasks, en Anglais), tandis que MR signifie que certaines tâches peuvent nécessiter plusieurs robots (ou multi-robot tasks, en Anglais).

- **IA** vs **TA** : L'assignement instantanée IA (ou Instantaneous assignment, en Anglais) signifie que les informations disponibles concernant les tâches, les robots et l'environnement ne permettent qu'une affectation instantanée des tâches aux robots, sans planification des affectations futures. À l'autre extrême, l'assignement à temps prolongé TA (ou time-

extended assignment, en Anglais) signifie que les tâches peuvent être planifiées sur un horizon de planification [Gerkey & Mataric, 2004 ; Nunes et al., 2017].

Dans cette thèse, nous examinons une classe de problèmes MRTA connue sous le nom du problème de routage multi-robot (ou Multi-robot Routing, MRR en Anglais). Dans une mission de MRR, une flotte de robots mobiles doit se déplacer et visiter un certain nombre de points d'intervention répartis sur une zone géographique afin d'effectuer un ensemble de tâches [Lagoudakis et al., 2005]. Lorsque les robots se rendent à l'endroit qui leur a été attribué, ils doivent éviter les obstacles tels que les bâtiments, les murs, etc. Les routes des robots vers les cibles doivent optimiser des critères tels que la distance ou le temps de déplacement. La mission est considérée comme terminée lorsque tous les cibles ont été visités.
Pour des missions de routage simples dans lesquelles les emplacements des tâches ne sont visités que par un seul robot et qu'il n'y a pas de contraintes comme l'ordre de précédence entre les tâches, le problème de routage multi-robots devient similaire au problème du voyageur de commerce multiple (mTSP) [Bektas, 2006] ou problème de routage de véhicules (VRP) [Laporte, 1992b ; Schneider, 2018].

En prenant en compte des contraintes de précédence, temporelles et de synchronisation, le seul problème MRTA qui peut être modélisé comme VRP ou TSP est [ST-SR-TA]. Dans MRTA, les robots peuvent être hétérogènes avec des différentes capacités et aptitudes [Khamis et al., 2011 ; Gam et al., 2021a], ils peuvent démarrer à partir de différents dépôts, et ils peuvent avoir besoin de communiquer entre eux. De plus, les VRP considèrent qu'un nombre infini de véhicules homogènes est disponible [Nallusamy et al., 2009 ; Nunes et al., 2017]. Il est à noter que les problèmes MRTA sont de complexité croissante en fonction de leur catégorie, [MT-MR] se réfère aux problèmes les plus difficiles pour lesquels seules quelques contributions existent [Zhang & Smith, 2020].

Les problèmes de MRTA existent dans divers domaines d'application. Les contributions récentes sont reportées dans le tableau 3. 1. Les auteurs de [Saeedvand et al., 2019] ont remplacé le travail humain par des robots humanoïdes pour le sauvetage, l'exploration et la défense dans des environnements dangereux. Les auteurs de [Alitappeh & Jeddisaravi, 2022] ont abordé l'application de l'exploration de l'environnement afin de distribuer les tâches sur l'ensemble des points dans les sous-régions et de minimiser le coût global du système. En outre, l'étude de [Jose & Pratihar, 2016] a présenté une application où des robots parcourent le chemin minimum

afin d'inspecter des sites dans une usine. Les auteurs de [Turner et al., 2018] ont étendu un algorithme existant d'allocation de tâches distribuées avec une nouvelle méthode pour maximiser le nombre d'allocations de tâches dans un système multi-robots distribué en tenant compte des contraintes de temps. Les auteurs de [Atay & Bayazit, 2006] ont considéré une application pour détecter et contrôler plusieurs régions dans un environnement inconnu en utilisant plusieurs robots hétérogènes. Afin de déterminer l'algorithme le plus approprié pour résoudre le problème MRTA, les auteurs ont introduit dans [Shelkamy et al., 2020] deux approches stochastiques différentes pour montrer leurs performances en termes de temps de convergence et de distance minimale. Dans les systèmes de surveillance mobiles, les auteurs de [Khamis et al., 2011] ont utilisé l'allocation de tâches complexes pour attribuer un ensemble de tâches de surveillance à un ensemble d'agents de détection mobiles. Les auteurs de [Gam et. al., 2020] ont résolu la spécification des patrouilles de surveillance avec des modèles de réseaux de Petri en utilisant des techniques d'estimation du marquage initial.

Tableau 3. 1 : Classification des références MRTA en fonction des types de problèmes et des méthodes utilisées

Taxonomie MRTA	Références	Méthode
ST - SR - TA	[Alitappeh & Jeddisaravi, 2022]	Algorithme génétique
	[Jose & Pratihar, 2016]	
ST- SR - TA	[Saeedvand et al., 2019]	
ST - SR - IA	[Shelkamy et al., 2020]	
ST- SR - TA	[Atay & Bayazit, 2006]	Programmation linéaire en nombres entiers mixtes
MT - MR - IA	[Zhang & Smith, 2020]	Théorie de l'information invariante
ST - SR - TA	[Khamis et al., 2011]	Approche basée sur le marché
ST - SR - TA	[Turner et al., 2018]	Algorithme d'impact sur les performances (PI-MaxAss)

La plupart des contributions sur MRTA ont un grand potentiel pour les applications pratiques et une bonne performance à faible coût [Alitappeh & Jeddisaravi, 2022 ; Jose & Pratihar, 2016; Saeedvand et al., 2019 ; Shelkamy et al., 2020]. Les auteurs de [Atay & Bayazit, 2006 ; Zhang & Smith, 2020] ont fourni un outil de modélisation puissant pour calculer les trajectoires

optimales des robots et maximiser la fonction objectif tout en respectant les contraintes. [Khamis et al., 2011] ont considéré une approche hybride qui combine les éléments centralisés et distribués. [Turner et al., 2018] ont également optimisé le temps d'attente moyen afin d'attribuer le nombre maximum de tâches à chaque véhicule avec une méthode de complexité polynomiale. Cependant, la plupart des résultats cités ci-dessus ont aussi certaines limites comme l'augmentation exponentielle de leur complexité, des algorithmes lents pour atteindre la solution optimale et coûteux en temps de calcul ce qui signifie que ces approches ne sont pas applicables pour notre problème [ST- MR – TA]. Par rapport aux approches précédentes, la méthode de recherche en faisceau utilisée dans notre travail a une complexité numérique raisonnable. Ainsi, elle reste réalisable pour les systèmes à grande échelle.

3.2.6. Complexité des méthodes d'optimisation

La complexité des méthodes d'optimisation peut être évaluée selon les critères suivants :

- Durée d'exécution ; une approche de résolution est plus efficace qu'une autre si pour les mêmes données, le temps de calcul est plus faible.
- Espace mémoire ; une approche est plus efficace qu'une autre si elle consomme moins d'espace mémoire pour résoudre le même problème.
- Robustesse ; une approche est plus robuste qu'une autre si elle maintient ses performances en dépit de l'occurrences de perturbations.

Le but de la théorie de la complexité [Cook, 1971] est d'estimer le nombre d'instructions à effectuer pour résoudre les instances des problèmes étudiés. Cette étude permet d'établir une classification des problèmes selon les niveaux de difficulté de leur résolution [Papadimitriou, 1994]. Les classes de complexité ont été présentées pour les problèmes de décision, c'est-à-dire les problèmes avec une question dont la réponse « oui » ou « non ». En fonction du degré de complexité en termes de temps d'exécution, un problème d'optimisation peut être placé dans l'une des quatre classes suivantes [Sakarovitch, 1984] :

- Les problèmes indécidables : pour lesquels il n'existe pas de méthode de résolution.
- Les problèmes de la classe P (appelés problèmes polynomiaux) : pour lesquels il existe un algorithme de complexité polynomiale en temps d'exécution pour leur résolution.
- Les problèmes de la classe NP (appelés problèmes NP-difficiles) : ne peuvent pas être résolu en temps polynomial par des algorithmes déterministes [Lopez & Esquirol,

1999 ; Carlier & Chrétienne, 1988]. La résolution des problèmes NP peut nécessiter un nombre important (voire exponentiel) de solutions mais l'examen de chaque solution se fait en temps polynomial. Ces problèmes sont résolus par des méthodes approchées dont l'optimalité n'est pas prouvable.

- Les problèmes NP-Complets : sont les problèmes les plus difficiles qui appartiennent à la classe NP et sont résolus généralement en un temps exponentiel. Ces problèmes sont également compliqués à résoudre par rapport à tout autre problème de type NP.

Pour résoudre un problème d'optimisation de manière efficace, il faut prendre en considération la spécificité de chaque problème. A cet égard, il existe tant de méthodes que de problèmes d'optimisation. La difficulté principale se résume dans le choix d'une méthode efficace qui fournit, pour un problème d'optimisation donné, une solution de bonne qualité suffisamment proche de la solution optimale avec un temps de calcul raisonnable.

Dans la suite, nous passons en revue les différentes méthodes de résolution des problèmes d'optimisation qui sont classés principalement en deux catégories : les méthodes exactes où la complétude de la résolution (l'algorithme explore toutes les solutions de l'arbre de recherche) et la solution optimale sont garanties pour les problèmes de petite taille et les méthodes approchées où la complétude est perdue (l'algorithme explore partiellement l'arbre de recherche) pour gagner en efficacité et l'optimalité est obtenue en un temps raisonnable pour les problèmes de grande taille [Hao et al., 1999].

3.3. Les méthodes de résolution exactes

Une méthode exacte est définie comme une méthode qui fournit une solution optimale pour un problème d'optimisation. Le principe de base de cette catégorie de méthodes consiste généralement à recenser, toutes les solutions dans l'espace de recherche. L'optimalité est évaluée par rapport aux critères de performance fixés par le contrôleur [Gargouri, 2003]. Les méthodes exactes garantissent de trouver la solution optimale du problème et en prouvent l'optimalité.

Généralement, il est impossible de résoudre un problème d'optimisation NP-difficile avec un algorithme en temps polynomial. Cependant, des méthodes efficaces peuvent être développées pour certains énoncés. Pour quelques sous-problèmes, la programmation linéaire peut être appliquée, pour d'autres, la programmation dynamique ou encore la Procédure par Séparation et Evaluation (ou Branch and Bound, en Anglais) [Baptiste & Le Pape, 1996 ; Le Pape, 1995].

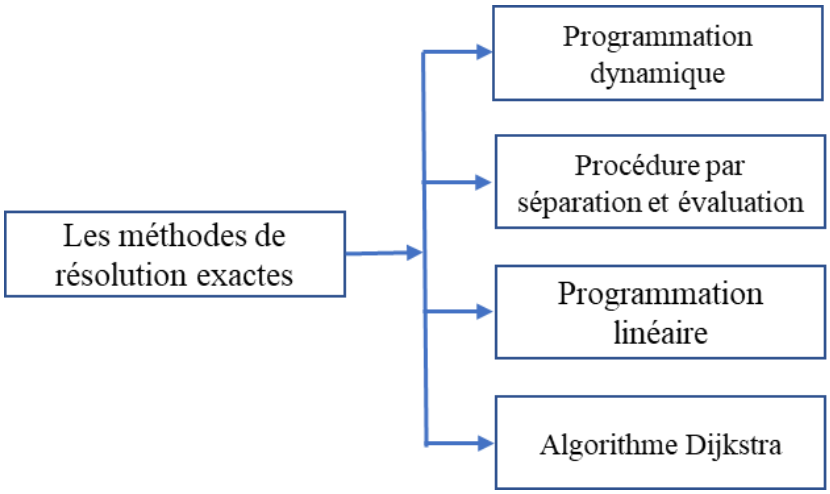


Figure 3. 3 : Les méthodes de résolution exactes

3.3.1. Programmation dynamique

La programmation dynamique est définie comme étant une méthode d'optimisation qui permet de résoudre un problème en le décomposant en sous-problèmes. Cette méthode est inspirée du principe de Bellman : « Si un point M appartient à un trajet optimal entre deux points N et P, alors la portion de ce trajet allant de N à M est le trajet optimal entre N et M » [Bellman, 1986]. Le concept est de construire d'abord les sous-trajets optimaux ce qui permet par la suite de construire, par récurrence, le trajet optimal pour l'ensemble du problème.

La programmation dynamique peut être coûteuse en termes de temps d'exécution et de mémoire. Toutefois, elle peut générer une solution optimale en temps polynomial pour les problèmes de petite taille [Aggoune, 2002].

3.3.2. Procédure par séparation et évaluation (Branch & Bound)

La méthode *Branch and Bound* est destinée très souvent aux problèmes de recherche opérationnelle. Elle a été explorée pour la première fois par [Dantzig et al., 1954] pour résoudre le TSP. Elle est basée sur le principe de l'énumération implicite des solutions réalisables. Plutôt que d'énumérer toutes les solutions réalisables du problème, l'idée est de calculer le coût pour chacune et donner le minimum afin de restreindre progressivement l'ensemble des solutions réalisables. L'approche consiste à trouver la meilleure solution réalisable en alternant les étapes de séparation et d'évaluation de manière à éliminer les mauvaises solutions et ce, jusqu'à obtenir l'optimalité de la solution [Collette & Siarry, 2002 ; Belhoul, 2014]. Cette méthode est bien connue pour résoudre les problèmes d'optimisation combinatoire [Lawler, 1966 ; Pruul et al., 1988 ; Brusco & Stahl, 2005].

3.3.3. Programmation linéaire

La programmation linéaire (PL) est un outil puissant en recherche opérationnelle qui s'appuie sur des techniques de modélisation et de résolution pour résoudre des problèmes d'optimisation dans divers domaines d'application [Sakarovitch, 1984]. La méthode est basée sur une modélisation mathématique du problème afin d'optimiser une fonction objectif linéaire qui dépend des variables de décisions [Mellouli et al., 2004]. Ces variables sont soumises à un ensemble de contraintes formulées sous forme d'inéquations et/ou d'équations linéaires. Le but de la programmation linéaire consiste à trouver une solution maximisant (resp. minimisant) la fonction objectif.

3.3.4. Algorithme de Dijkstra

L'algorithme de Dijkstra est l'un des algorithmes les plus courants pour trouver les plus courts chemins entre deux nœuds donnés d'un graphe orienté ou non orienté [Arjun et al., 2015]. Cependant, il n'est pas adapté à un environnement dynamique.
En outre, l'algorithme de Dijkstra effectue une recherche aveugle qui peut faire perdre beaucoup de temps [Sanan and al., 2013]. Il a été appliqué dans différents domaines et en particulier pour les problèmes de transport [Soltani et al., 2002 ; Chen et al., 2014 ; Nagar & Tawfik, 2007]. Le principe de l'algorithme est de considérer un nœud source (nœud initial). L'algorithme va attribuer certaines valeurs de distance par rapport au nœud source et va tenter de les améliorer étape par étape.

Les méthodes précédentes examinent implicitement l'ensemble de l'espace de recherche pour aboutir à la solution optimale. L'utilisation des méthodes exactes est donc particulièrement intéressante dans le cas des problèmes d'optimisation de petite taille. Mais, lorsque le temps de calcul nécessaire pour atteindre une solution optimale devient important, les méthodes approchées fournissent une solution plus efficace en un temps de calcul acceptable.

3.4. Les méthodes de résolution approchées

Les méthodes approchées représentent une alternative intéressante pour traiter les problèmes d'optimisation de grande taille dans lesquels une solution optimale ne peut pas être obtenue en un temps raisonnable [Liouane, 1998]. Ces méthodes sont connues depuis longtemps et utilisées par de nombreux praticiens. Leur principe est de fournir une solution de bonne qualité en un temps polynomial. Les méthodes approchées peuvent être divisées en deux catégories : les

méthodes basées sur des heuristiques et les méthodes basées sur des métaheuristiques. Dans cette section, les principales heuristiques et métaheuristiques pour les problèmes d'optimisation combinatoire sont présentées.

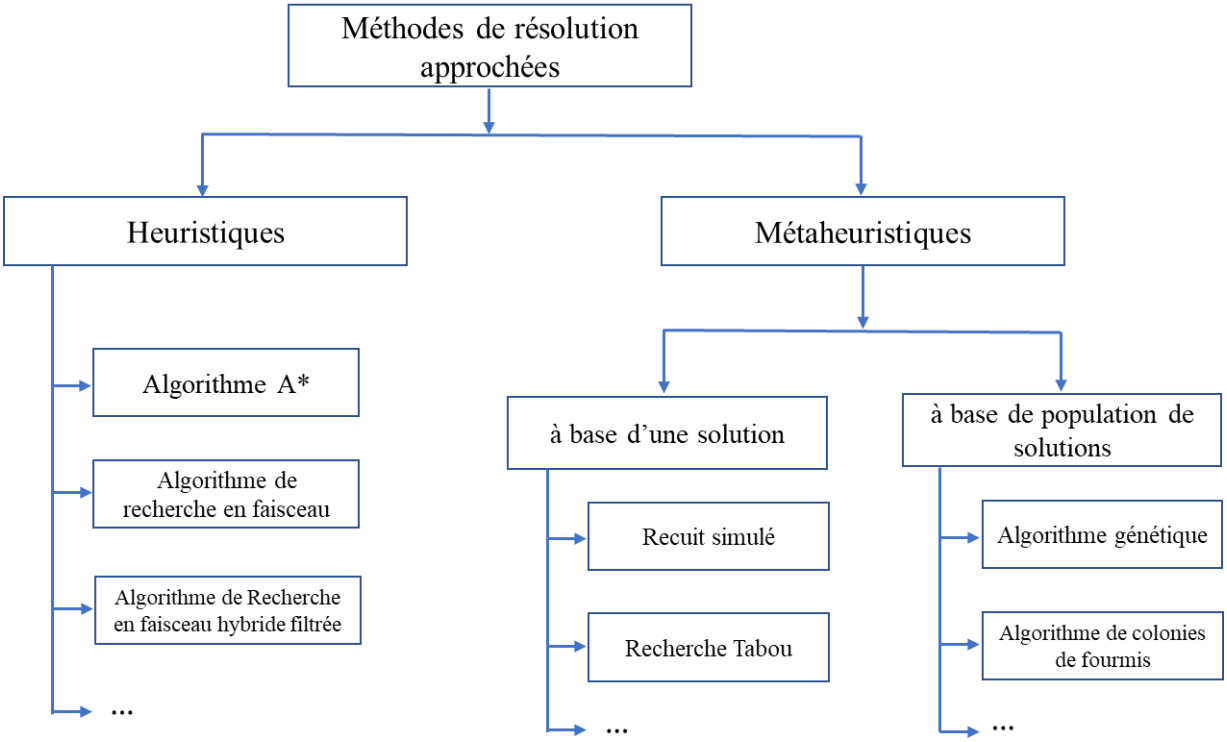


Figure 3. 4 : Classification des méthodes de résolution approchées

3.4.1. Les heuristiques

Les méthodes heuristiques sont empiriques et basées sur l'expérience et sur l'analogie. En tenant en compte d'une fonction objectif, elles permettent de fournir généralement des résultats réalisables de bonne qualité en un temps raisonnable (complexité polynomiale). Généralement, une heuristique est spécifique à un problème précis et ne peut pas être répétée. Ce type de méthodes se situe à mi-chemin entre les méthodes exactes de recherche opérationnelle et les métaheuristiques [Kefi, 2008]. Les méthodes heuristiques permettent d'intégrer des stratégies de décision (FIFO, SPT, LPT, EDF, LRF, HPF) afin de construire une solution souvent très proche de la solution optimale. En outre, il existe d'autres heuristiques utilisées essentiellement pour la recherche de trajets dans un graphe [Guan & Cheung, 2004]. Parmi les méthodes de recherche de graphe les plus célèbres, on peut mentionner l'algorithme A* (A star) et l'algorithme de recherche en faisceau et ses diverses variantes.

- ***La méthode A****

L'algorithme de recherche A étoile A* (ou A star en Anglais) est un algorithme itératif permettant de rechercher un chemin dans un graphe entre un nœud initial et un nœud final donné [Hart et al., 1968]. Cette méthode ne garantit pas la solution optimale mais renvoie très rapidement une bonne solution (qui peut être optimale dans certains cas).

La méthode utilise une évaluation heuristique pour chaque nœud afin d'estimer le meilleur chemin qui le traverse, classe les nœuds dans l'ordre croissant, puis visite les nœuds dans cet ordre. Il s'agit d'une méthode simple qui ne nécessite pas de prétraitement [Cherif, 2021].

De par sa simplicité, A* est une méthode célèbre dans des applications comme les jeux vidéo [Rabin, 2000] qui privilégient le temps de calcul à la précision des résultats.

- ***Algorithme de recherche en faisceau***

L'algorithme de recherche en faisceau BS (ou Beam Search, en Anglais) est une méthode heuristique pour résoudre les problèmes d'optimisation. C'est une heuristique de recherche qui permet d'explorer un arbre et de traiter un ensemble limité de fils pour chaque nœud [Wu et al., 2011].

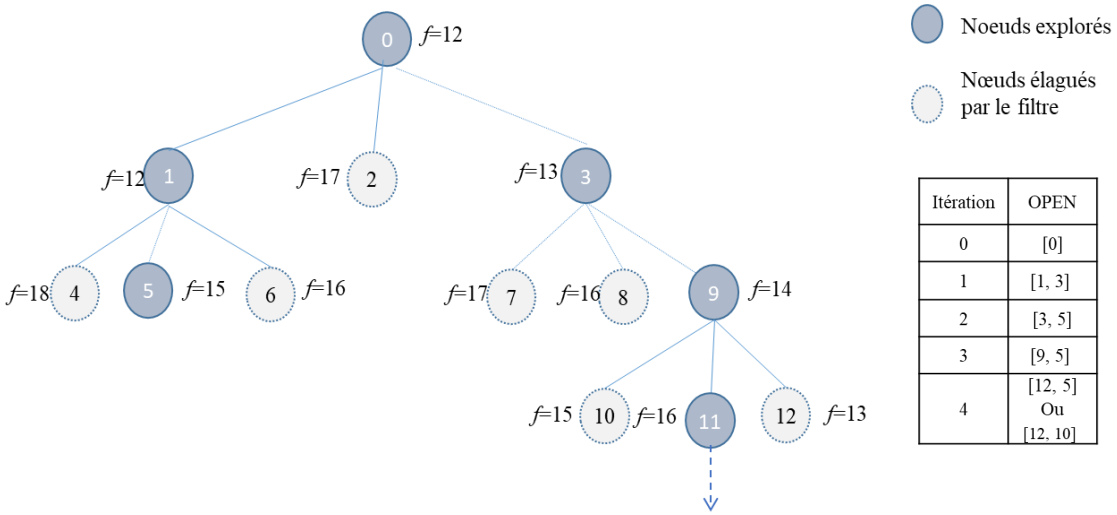
La méthode utilise d'abord l'algorithme de parcours en largeur [Cherif et al., 2019] afin d'explorer l'arbre de recherche. Elle limite l'espace de recherche en explorant uniquement les meilleurs successeurs à chaque niveau de l'arbre. En effet, à chaque itération (niveau de l'arbre), elle génère tous les successeurs du nœud actuel en les classant par ordre croissant selon leur coût. La fonction heuristique évalue le coût pour atteindre un nœud final donné (ou un ensemble de nœuds finaux) qui satisfait certaines conditions terminales (TC) spécifiques à partir d'un nœud initial donné S_0. En conséquence, la méthode ne mémorise qu'un nombre limité β (dit largeur du faisceau) de successeurs (les plus prometteurs avec le meilleur coût) et le reste des successeurs est élagué de manière permanente. Seuls les nœuds retenus sont développés à l'étape suivante [Gam et al., 2022a].

L'algorithme de recherche en faisceau utilise une fonction d'évaluation f qui est calculée à chaque nœud S du graphe par :

$$f(S_0, \sigma, S, TC) = g(S_0, \sigma, S) + h(S, TC) \quad (1)$$

où $g\ (S_0, \sigma, S)$ est le coût réel de S_0 à S avec la trajectoire définie par le chemin σ, et $h\ (S, TC)$ est une estimation du coût de S au nœud le plus proche qui satisfait TC. La fonction heuristique $h\ (S, TC)$ doit sous-estimer le coût réel [Mejía & Lefebvre, 2019 ; Lefebvre & Basile, 2021 ; Gam et al., 2022a].

Avec une largeur du faisceau infinie, tous les nœuds sont pris en compte et la méthode devient identique à l'algorithme de parcours en largeur. La réduction de la largeur du faisceau limite la mémoire nécessaire pour effectuer la recherche. Par conséquent, la recherche en faisceau présente une optimisation de l'algorithme de recherche de parcours en largeur.
Si aucune solution n'est obtenue pour une valeur de β donnée, dans ce cas-là, la procédure est répétée avec une plus grande taille de faisceau.
L'exploration partielle de l'arbre de recherche n'assure pas l'obtention d'une solution (la complétude n'est pas garantie). De plus, l'algorithme ne garantit pas non plus de trouver la meilleure solution (l'optimalité n'est pas garantie).
La recherche en faisceau est généralement utilisée pour les systèmes de grande taille lorsque l'espace mémoire est insuffisant pour stocker l'ensemble de l'arbre de recherche [Tillmann & Ney, 2018].



Itération	OPEN
0	[0]
1	[1, 3]
2	[3, 5]
3	[9, 5]
4	[12, 5] Ou [12, 10]

Figure 3. 5 : Exploration de l'arbre de recherche avec l'algorithme BS pour $\beta = 2$

La Fig. 3. 5 montre l'exploration de l'arbre de recherche en utilisant l'algorithme BS avec un largeur du faisceau $\beta = 2$. L'algorithme sélectionne, à chaque niveau, les β meilleurs candidats afin d'être explorés dans le niveau suivant. Le terme f à côté de chaque nœud i est le coût estimé du chemin du nœud S_0 au nœud terminal à travers ce nœud. Les chiffres à l'intérieur des nœuds représentent l'ordre de génération.

Le nœud S_0 génère trois nœuds fils (S_1, S_2 et S_3) et les nœuds S_1 et S_3 générés avec les plus petites valeurs de f sont conservés grâce au filtre β =2. La liste des candidats à explorer est [S_1, S_3]. Ensuite, S_1 est exploré en générant trois nœuds fils (S_4, S_5 et S_6). Les nœuds S_5 et S_3 générés avec les plus petites valeurs de f sont conservés grâce au filtre β et triés dans la liste des candidats : [S_3, S_5] et S_4, S_6 sont éliminés en raison de la capacité de la liste induite par le filtre. Pareil pour le nœud S_3 qui est exploré en générant trois nœuds fils (S_7, S_8 et S_9). Seuls les nœuds S_5 et S_9 sont conservés par le filtre et la liste des candidats devient [S_9, S_5]. À ce niveau de la recherche, le nœud S_9 qui a la valeur f la plus basse est exploré en générant d'autres nœuds. Grâce à cette stratégie, la recherche évolue de manière contrôlée vers le nœud cible.
Dans ce contexte, plusieurs variantes de recherche en faisceau ont été étudiées et proposées pour la résolution de problèmes d'optimisation [Gam et al., 2022a ; Luo et al., 2015 ; Mejía & Niño, 2017].

- ***Recherche en faisceau filtrée hybride (HFBS)***

La méthode de recherche A* nécessite un temps et une mémoire exponentiels (Russell et Norvig, 1995). Pour cette raison, L'algorithme de recherche en faisceau s'est avéré efficace pour un certain nombre de problèmes combinatoires (avec une complexité polynomiale) [Ow & Morton, 1988]. L'élagage de l'arbre de recherche est une pratique courante, mais au détriment de l'optimalité, et peut aboutir à l'élimination de bons candidats dans l'arbre de recherche. Avoir un coût raisonnable est un avantage très important pour résoudre les problèmes d'optimisation de grande taille, contrairement aux méthodes exactes qui n'arrivent pas à les résoudre.

De nombreuses améliorations ont été développées dans ce but, et en particulier les extensions de l'algorithme de recherche en faisceau (BS) ont été présentées comme la recherche en faisceau filtrée qui utilise une fonction simple pour pré-évaluer et sélectionner les nœuds avant d'appliquer une fonction plus complexe (dite une fonction d'évaluation) aux nœuds restants [Ow & Morton, 1988] et la recherche en faisceau avec récupération (RBS) qui applique à chaque nœud un autre test qui vérifie si le nœud actuel est dominé par un autre nœud [Croce et al., 2004]. Plusieurs variantes de BS ont été présentées pour améliorer notamment la sélection des nœuds à explorer et la vitesse de la recherche. Ces variantes combinent la stratégie d'exploration de la méthode A* avec la méthode d'élagage de recherche en faisceau.

La méthode de recherche en faisceau BAS (ou Beam A* search, en Anglais) est la première variante qui a été introduite par [Mejía & Odrey, 2005]. Elle consiste à explorer un nombre

maximum de nœuds à chaque niveau du graphe. Ainsi, à une itération donnée, seuls les nœuds dont le niveau n'a pas été complètement développé restent dans une liste notée OPEN : tous les autres nœuds sont supprimés. Un inconvénient de la méthode BAS est que dans les premières itérations de la recherche, la plupart des nœuds explorés proviennent d'un nœud parent commun. A ce stade, la recherche n'explore que les nœuds d'une zone limitée de l'arbre de recherche [Mejía & Odrey, 2005].

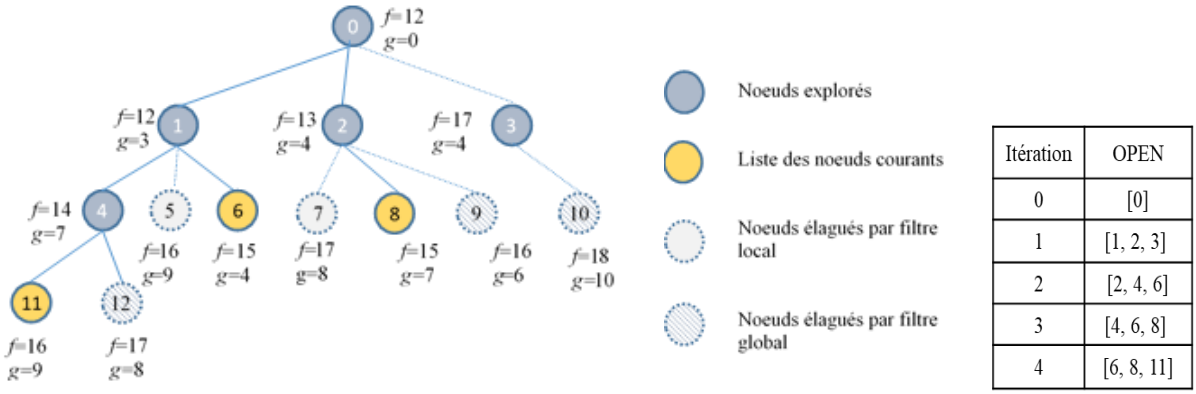
Une autre variante appelée la recherche en faisceau filtré FBS (ou Filtered Beam Search, en Anglais) a été introduite afin de surmonter les limites de l'algorithme BAS [Ow & Morton, 1988 ; Mejía & Niño, 2017 ; Lefebvre & Mejía, 2018]. L'idée est d'explorer des nœuds dans différentes branches pour résoudre le problème d'exploration d'une zone limitée de l'arbre de recherche avec l'algorithme BAS. Un mécanisme de filtrage double a été mis en œuvre pour filtrer les nœuds non prometteurs : un filtre global β_g définit un nombre maximum de nœuds pouvant être développés à chaque niveau du graphe et un filtre local β_l définit le nombre maximum de nœuds successeurs à partir de n'importe quel nœud parent donné. Le principal avantage de l'algorithme FBS est sa complexité polynomiale en temps et en espace par rapport à l'algorithme A*. Par contre, l'inconvénient de cette méthode est le choix des candidats à explorer. En effet, la sélection du candidat à explorer se fait entre des candidats de niveaux différents ce qui signifie qu'ils n'ont pas les mêmes chances d'être sélectionnés [Mejía & Niño, 2017 ; Lefebvre & Mejía, 2018]. Cela peut conduire à une recherche en profondeur avec la possibilité d'avoir un risque d'un mauvais choix de nœud, ce qui peut entrainer un retard dans la recherche.

Afin de surmonter les limites de BAS et FBS, une autre variante appelée la recherche en faisceau filtrée hybride (ou Hybrid Filtered Beam Search, HFBS en Anglais) a été introduite [Mejía & Niño, 2017]. Cet algorithme rassemble les principes des algorithmes BS [Sabuncuoglu & Bayiz, 1999], BAS [Mejía & Montoya, 2009] et FBS [Ow & Morton, 1988]. L'idée est d'explorer des nœuds de diverses branches à des niveaux peu profonds et des nœuds issus uniquement des meilleures branches retenues à des niveaux plus profonds. De même que la méthode FBS mentionnée précédemment, l'algorithme HFBS utilise à la fois un filtre global avec le paramètre βg, qui limite le nombre de nœuds à chaque niveau de l'arbre et un filtre local qui ne conserve que les β_l meilleurs successeurs pour chaque nœud développé [Mejía & Niño, 2017 ; Gam et al., 2022a]. Cette variante assure la diversification de la population de candidats

en explorant, au premier niveau de l'arbre de recherche, les différents candidats de même profondeur selon β_g et en explorant les meilleurs candidats des niveaux plus profonds. Nous utiliserons cette variante dans la suite de notre travail.

Les limites de l'algorithme HFBS sont (1) l'obtention de la solution, même si elle existe, n'est pas garantie et (2) la détermination de la meilleure combinaison (β_l, β_g) n'est pas facile à obtenir. Plus particulièrement, il n'existe pas de propriété de monotonie de la performance en fonction de β_l et β_g [Gam et al., 2022a].

Bien que la recherche explore initialement les candidats de différentes branches, elle ne maintient pas cet équilibre pour les étapes ultérieures de la recherche : la recherche devient plus intense sur les meilleures branches, ce qui peut conduire à une recherche approfondie avant de se rendre compte qu'un mauvais choix a été fait, comme le montre dans l'exemple suivant.



Itération	OPEN
0	[0]
1	[1, 2, 3]
2	[2, 4, 6]
3	[4, 6, 8]
4	[6, 8, 11]

Figure 3. 6 : Exploration de l'arbre de recherche avec la méthode HFBS

La Fig. 3.6 illustre les stratégies d'expansion de l'algorithme HFBS avec β_g =3 et β_l =2. Les termes *g* et *f* à côté de chaque nœud *i* sont respectivement le coût du chemin du nœud S_0 à *i* et le coût estimé du chemin du nœud S_0 au nœud terminal à travers ce nœud. Les chiffres à l'intérieur des nœuds représentent l'ordre de génération.

Le nœud S_0 génère trois nœuds fils (S_1, S_2 et S_3) et tous ces nœuds générés avec les plus petites valeurs de *f* sont conservés grâce au filtre global βg =3 (à cette étape initiale, le filtre local n'est pas utilisé). La liste des candidats à explorer est [S_1, S_2, S_3]. Ensuite, S_1 est étendu en générant trois nœuds fils (S_4, S_5 et S_6) et seuls les deux meilleurs nœuds fils sont conservés (S_4 et S_6) en raison du filtre local β_l =2. La liste des candidats triés par rapport à *f* est mise à jour : [S_2, S_4, S_6, S_3] et S_3 est éliminé en raison de la capacité de la liste induite par le filtre global. S_2 est étendu en générant trois nœuds fils (S_7, S_8 et S_9). Seuls les nœuds S_8 et S_9 sont conservés par le filtre local et la liste des candidats devient [S_4, S_6, S_8, S_9] et à nouveau filtrée et S_9 est éliminé. À ce

niveau de la recherche, le nœud S_4 qui a la valeur *f* la plus basse est étendu en générant les nœuds S_{11} et S_{12} et de nouveau le filtre global garde les 3 meilleurs candidats [S_6, S_8, S_{11}]. La recherche continue jusqu'à ce que le nœud cible soit trouvé [Gam et al., 2022a].

3.4.2. Les métaheuristiques

Les métaheuristiques sont des méthodes d'optimisation visant à résoudre plusieurs problèmes d'optimisation de nature différentes issus particulièrement des domaines de l'ingénierie, la recherche opérationnelle ou de l'intelligence artificielle. Ces méthodes complexes permettent de combiner plusieurs approches heuristiques et d'avoir généralement une solution réalisable de très bonne qualité. Il s'agit principalement d'algorithmes itératifs stochastiques, qui progressent vers un optimum global.

La plupart des métaheuristiques sont représentées principalement par les méthodes de voisinage [Taillard et al., 2001] c'est-à-dire à partir d'une solution initiale générée aléatoirement, un processus itératif est mis en œuvre pour remplacer la solution initiale par l'un de ses voisins afin d'améliorer le résultat. Parmi les méthodes de voisinage, on peut trouver la recherche tabou, le recuit simulé, les algorithmes génétiques, etc. Le critère d'arrêt pour ces algorithmes peut être une valeur seuil de la fonction objectif, un temps de calcul maximal ou un nombre d'itérations. Dans ce cas-là, l'algorithme renvoie la meilleure solution trouvée.

Les métaheuristiques proposent des solutions approchées pour résoudre des problèmes d'optimisation de grande taille dans de nombreuses [Osman & James, 1996 ; Hao et al., 1999].

Les métaheuristiques peuvent être divisées principalement en deux classes (Fig. 3.4) :

- Les métaheuristiques à solution unique
- Les métaheuristiques à population de solutions ou solutions multiples

➢ *Les métaheuristiques à solution unique*

Les métaheuristiques à solution unique sont également appelées méthodes de recherche locale ou méthodes de trajectoire. Elles permettent de construire une trajectoire dans l'espace des solutions en commençant par une solution initiale, générée par une heuristique ou aléatoirement, en essayant de se rapprocher des solutions optimales.

Le recuit simulé [Kirkpatrick et al., 1983] et la recherche Tabou [Glover, 1989 ; Glover, 1990] sont parmi les méthodes les plus connues.

❖ **Le recuit simulé**

Le recuit simulé SA (ou Simulated Annealing, en Anglais) a été introduite par [Cerný, 1985 ; Kirkpatrick et al., 1983] pour résoudre les problèmes d'optimisation combinatoire. Cette métaheuristique est basée sur une technique appliquée depuis longtemps par les métallurgistes afin d'assurer la solidification du métal dans une structure d'énergie minimale [Metropolis et al., 1953].

Le recuit simulé est une méthode stochastique qui s'appuie sur des travaux faites par [Metropolis et al., 1953] afin de simuler l'évolution d'un système de recuit physique. Le principe du recuit simulé est de chercher itérativement des solutions avec un faible coût. L'objectif est d'accepter des solutions intermédiaires en dégradant la fonction objectif. L'acceptation des solutions moins efficaces que la solution actuelle permet à l'algorithme de trouver des optima locaux [Hao et al., 1999]. Le recuit simulé a été appliqué pour résoudre les problèmes d'allocation de tâches multi-robots (MRTA) [Hao et al., 1999 ; Khamis et al., 2015 ; Shi et al., 2022].

❖ **La recherche Tabou**

La recherche Tabou TS (ou Tabu Search, en Anglais) est une méthode de recherche locale itérative combinée avec un ensemble de techniques pour éviter d'être piégé dans la répétition d'un cycle ou un minimum local. Elle a été développée principalement par [Glover, 1986] et indépendamment par [Hansen, 1986 ; Glover & Laguna, 1997]. Elle est une métaheuristique basée sur la création et l'évaluation d'un voisinage [Kammarti et al., 2005 ; Karray et al., 2008]. Elle fait appel à un ensemble de mécanismes et de règles afin de guider la recherche locale de manière intelligente et de surmonter le problème d'optima locaux.
La recherche Tabou a été utilisée pour résoudre les problèmes d'optimisation combinatoire [Glover, 1989, 1990, 1997 ; Battiti & Tecchiolli, 1994].

➢ ***Les métaheuristiques à population de solutions***

Les méthodes à population de solutions traitent un ensemble de solutions. Ce sont des algorithmes évolutionnaires qui permettent d'améliorer, au fil des itérations, une population de solutions afin d'obtenir une solution globale. L'avantage des méthodes à solutions multiples est de considérer la population comme facteur de diversité.
Les algorithmes génétiques [Holland, 1975] et l'algorithme de colonies de fourmis [Colorni et al., 1991] sont parmi les métaheuristiques les plus connues.

❖ **Les algorithmes génétiques**

Les algorithmes génétiques GA (ou Genetic Algorithm, en Anglais) sont des algorithmes d'optimisation stochastique développés par [Holland, 1975]. [Goldberg, 1989] a aussi largement contribué à les enrichir. Ces méthodes ont été fondées sur les mécanismes de la génétique et de la sélection naturelle pour obtenir un fonctionnement simple. Elles sont basées sur un codage de l'information sous forme de chaînes binaires de taille déterminée et sur un ensemble d'opérateurs génétiques : la sélection, le croisement et la mutation. En effet, elles partent d'une population de solutions initiales et potentielles (dites chromosomes). Cet ensemble de solutions peut être généré avec des méthodes heuristiques ou aléatoirement. Les solutions potentielles appelées également individus sont évaluées par une fonction fitness (fonction d'évaluation) [Barichard, 2003].
L'idée de l'algorithme est de sélectionner les meilleurs individus (parents) pour la reproduction. Ensuite, un opérateur de croisement est utilisé sur les parents pour produire des nouveaux individus (enfants). A cette étape, le croisement ne recombine que les valeurs (gènes) existantes entre les parents et n'introduit pas de nouvelles valeurs chez les individus enfants. Pour cela, l'opérateur de mutation est appliqué afin de subir une transformation individuelle et de changer de façon aléatoire le code génétique d'un chromosome [Mesghouni, 1999]. Il convient de noter que la mutation est considérée dans les GA comme un opérateur secondaire par rapport au croisement. Certaines solutions se reproduisent, d'autres disparaissent et seules les meilleures solutions sont conservées pour constituer la population de la génération suivante. Ainsi, le cycle est répété jusqu'à qu'à l'obtention de la solution satisfaisante [Hao et al., 1999].

Les algorithmes génétiques ont été utilisés pour résoudre les problèmes de MRTA [Alitappeh & Jeddisaravi, 2022 ; Jose & Pratihar, 2016 ; Saeedvand et al., 2019 ; Shelkamy et al., 2020] et ils ont été appliqués aussi dans divers domaines tels que la distribution (le problème du voyageur de commerce) [Maha & Ahmed, 2020] et les tournées (le problème de tournées de véhicules) [Nallusamy et al., 2009 ; Ghoseiri & Ghannadpour, 2010 ; Karakatič & Podgorelec, 2015].

❖ **Algorithme de colonies de fourmis**

Les algorithmes de colonies de fourmis ACO (ou Ant Colony Optimization, en Anglais) sont des algorithmes d'optimisation inspirés du comportement des fourmis. Ils ont été introduits par [Colorni et al., 1991] et appliquées pour la première fois au TSP. Ce sont des algorithmes

itératifs qui permettent à tous les individus de la population de partager un savoir commun afin d'orienter leurs choix futurs. Ce savoir commun permet également de montrer aux autres individus les choix à suivre ou à ignorer.
Le principe de cette métaheuristique est d'élaborer des chemins qui se révèlent souvent être les plus courts pour aller du nid vers une source de nourriture. En effet, chaque fourmi dépose sur son parcours une quantité de phéromone pour communiquer avec les autres fourmis. Les fourmis choisissent les chemins avec les plus fortes concentrations de phéromones.

L'objectif de ces méthodes est de reproduire le comportement des fourmis dans le but d'obtenir la meilleure solution possible pour différents types de problèmes [Colorni et al., 1991 ; Colorni et al., 1992].

3.5. Les affectations hors ligne / incrémentale / en ligne

MRTA est défini comme une instance du problème d'affectation optimale (ou Optimal Assignment Problem, OAP en Anglais) [Gerkey & Mataric, 2004]. Il peut être formulé comme un problème d'ordonnancement. La taxonomie de [Gerkey & Mataric, 2004] décrit trois variantes du problème MRTA, à savoir l'affectation hors ligne, l'affectation incrémentale et l'affectation en ligne.

- Affectation hors ligne : l'ensemble de tâches est supposé connu a priori. Dans ce cas-là, la solution du problème est l'assignement des tâches en une seule fois.

- Affectation incrémentale : est une instance itérative de l'affectation hors ligne. Lorsqu'une nouvelle tâche entre dans le système, l'attribution de la nouvelle tâche est effectuée, et les robots auxquels une tâche a été affectée lors de l'itération précédente deviennent disponibles pour la nouvelle tâche entrante. En d'autres termes, la réaffectation des tâches aux robots est autorisée. À chaque itération, tous les robots libèrent leurs tâches précédentes et seront affectés à de nouvelles tâches.

- Affectation en ligne : La réaffectation des robots n'est pas autorisée. Un robot ne peut recevoir une nouvelle tâche que s'il termine l'exécution de sa tâche précédente ou si aucune tâche ne lui a été affectée.

3.6. Conclusion

Dans ce chapitre, les problèmes d'optimisation combinatoires ont été présentés, en particulier, le problème d'allocation de tâches multi-robots (MRTA). Par la suite, les méthodes de résolution exactes ont été introduites, ainsi que les méthodes de résolution approchées à savoir les heuristiques et les métaheuristiques. Celles-ci ont été classées en deux catégories : les métaheuristiques à solution unique et à population de solutions. Enfin, Nous avons donné un aperçu sur les différentes affectations du problème MRTA.

Dans le chapitre suivant, nous allons procéder tout d'abord à la modélisation du problème en utilisant l'approche des systèmes à évènements discrets (SED). Ensuite, la configuration et la planification des trajectoires seront proposées pour optimiser les missions de surveillance par approche SED.

Chapitre 4 : Optimisation par approche SED

4.1. Introduction

Le réseau de Petri est utilisé comme outil de modélisation pour surveiller l'environnement et pour décrire et calculer les trajectoires des robots. Plus précisément, nous utilisons les réseaux de Petri pour modéliser et formaliser le problème comme un problème d'estimation de marquage minimal [Giua, 1997 ; Giua & Seatzu, 2002 ; Giua et al., 2003 ; Giua et al., 2007 ; Li et al., 2013 ; Cassandras & Lafortune, 2008 ; Murata, 1989]. Les objectifs sont reformulés comme un problème d'optimisation du marquage initial et un calcul de séquence de tir.

Les outils de modélisation des systèmes à évènements discrets (SED) utilisés tout au long de ce chapitre sont présentés dans la première section. La deuxième section s'intéresse aux hypothèses et aux objectifs du problème. Dans la troisième section, les principales étapes de la résolution du problème sont présentées. Cette section détaille les méthodes utilisées d'abord pour optimiser la configuration et ensuite pour résoudre le problème d'optimisation globale. Une modélisation et une résolution simplifiées du problème sont proposées, dans la quatrième section ainsi qu'une application pour une étude de cas de la zone industrielle et portuaire de la ville du Havre.

4.2. Modélisation par SED

Cette première partie du chapitre présente les concepts de base de modélisation des SED à savoir le diagramme d'état et les réseaux de Petri.

4.2.1. Diagramme d'état

Les diagrammes d'état sont utilisés en informatique et dans plusieurs autres domaines pour décrire et modéliser les systèmes ayant des comportements interactifs complexes. Ils sont utilisés pour effectuer une analyse formelle du système. Les diagrammes d'état ont une bonne sémantique structurelle mais il leur manque une sémantique comportementale formelle [Claude Elwood, 1948 ; Taylor, 1967].

Le diagramme d'état est constitué d'états, de transitions, d'événements et de conditions (fig. 4. 1). Il décrit le comportement interne d'un objet en utilisant une machine à états finis. Ce comportement est décrit et représenté par une série de transitions entre états et par les actions que le système ou ses composants résident en réponse à un événement.

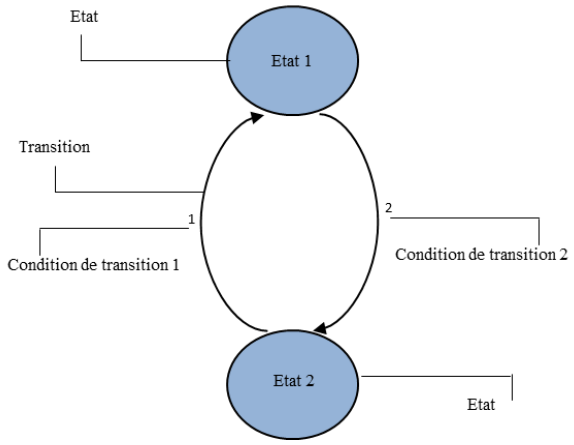


Figure 4. 1 : Un exemple de machine à états finis

4.2.2. Réseau de Petri

Un réseau de Petri PN (ou Petri Net, en Anglais) autonome et discret est représenté par un graphe biparti et orienté composé de deux types de nœuds (ou sommets) : un ensemble de places et un ensemble de transitions dont aucun arc ne relie deux sommets de même type. Les places correspondent aux variables d'état du système tandis que les transitions correspondent aux événements qui font passer le système d'un état à un autre. Les arcs orientés relient les places d'entrée aux transitions et les transitions aux places de sortie.

Dans le cas d'un réseau de Petri marqué, le marquage est défini comme le nombre de jetons ou de marques (la valeur d'une variable d'état) dans la place concernée. Un arc orienté ayant pour origine une transition désigne la libération d'un ou plusieurs jetons suite à la survenance d'un événement [Silva, 2013].
L'état global du système est représenté par l'ensemble des marquages de toutes les places. Il est aussi représenté sous forme d'un vecteur de marquages.

Pour la présentation graphique :

- Les places sont indiquées par des cercles vides.
- Les transitions sont indiquées par des barres perpendiculaires aux arcs.
- Les arcs sont indiqués par des flèches précisant le sens de circulation des jetons.
- Les jetons sont indiqués par des cercles pleins placés à l'intérieur des places.

Les arcs peuvent être pondérés en leur affectant un poids, indiqué au milieu de l'arc, ce qui représente soit le nombre de jetons libérés suite à l'occurrence d'un évènement soit le nombre

de jetons nécessaire pour la réalisation d'un événement. Dans un tel cas, le PN est dit généralisé. Le PN est dit ordinaire si le poids est par défaut unitaire pour tous les arcs du graphe. La Fig. 4. 2 représente un réseau de Petri généralisé.

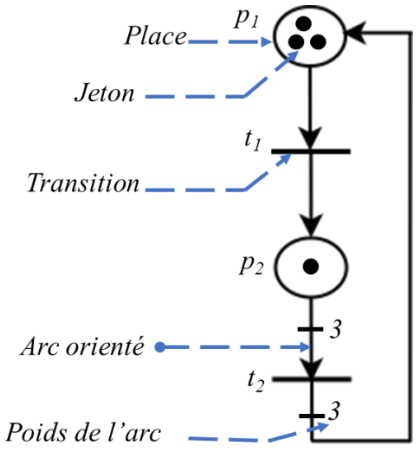


Figure 4. 2 : Réseau de Petri généralisé

Formellement, la structure de réseau de Petri est définie comme un quadruplet $PN = <P, T, W_{PR}, W_{PO}>$ [Cassandras & Lafortune, 2008 ; Murata, 1989] où :

- $P = \{p_1, p_2 \ldots p_n\}$ est un ensemble fini de n places ;
- $T = \{t_1, t_2, \ldots, t_p\}$ est un ensemble fini de p transitions ;
- $W_{PR} \in (N)^{n \times p}$ est la matrice de pré incidence du PN qui contient les poids des arcs allant des places aux transitions (N est l'ensemble des nombres entiers non négatifs) ;
- $W_{PO} \in (N)^{n \times p}$ est la matrice de post-incidence du PN qui contient le poids des arcs des transitions vers les places.

La matrice d'incidence d'un réseau de Petri, notée W est définie par $W = W_{PO} - W_{PR}$. $^\circ p_i$ (Preset) contient les transitions en entrée de la place p_i (c'est-à-dire, $^\circ p_i = \{t_j \in T \mid W_{PR}(p_i,t_j) > 0\}$ avec $W_{PR}(p_i,t_i)$ l'élément de W_{PR} à la ligne i et à la colonne j). De même, p_i° (Postset) contient les transitions de sortie de la place p_i (c'est-à-dire $p_i^\circ = \{t_j \in T \mid W_{PO}(p_i,t_j) > 0\}$). Le Preset et le Postset des transitions sont définis de manière similaire.

Chaque place du PN contient un nombre entier positif ou nul de jetons. Le marquage du PN est une application $M : P \rightarrow N^n$ qui attribue à chaque place un nombre de jetons. $M(p)$ désigne le nombre de jetons dans la place p.

$A = (PN, M_I)$ est une structure de réseau de Petri avec un marquage initial M_I.

Le franchissement d'une transition donnée conduit à la production et à la consommation de jetons dans les ensembles Preset et Postset de la transition qui est franchie. Le marquage *M'* résultant du franchissement de la transition *t* à partir du marquage *M* est calculé par : $M' = M + W(:,t)$ où $W(:, t)$ représente la colonne *t* de *W*.

$M [t \rangle$ est utilisé pour indiquer que la transition *t* est validée par *M* et $M [t \rangle M'$ est utilisé pour indiquer que le marquage *M'* est obtenu à partir de *M* en franchissant la transition *t*. Pour une séquence de transitions donnée : $Seq = t_1 t_2 \ldots t_k$ qui sont franchies successivement à partir du marquage *M*, *Y* est introduit comme le vecteur caractéristique, aussi appelé le vecteur de Parikh, de *Seq* et le marquage *M'* obtenu à partir du franchissement de *Seq* est défini par $M' = M + W.Y$.

4.3. Hypothèses et objectifs

Le problème considéré dans ce chapitre concerne la configuration et l'optimisation d'un ensemble de robots mobiles afin d'assurer la surveillance d'une zone donnée. L'objectif est d'optimiser les coûts d'investissement et d'exploitation des agents mobiles nécessaires à la réalisation de différentes mesures en garantissant un ensemble d'exigences de sécurité. Les patrouilles de surveillance sont effectuées par les robots équipés de capteurs permettant de réaliser les mesures souhaitées.

Rappelons que nous considérons un environnement constitué de *N* sites identifiés par un ensemble d'adresses $A = \{a_1, a_2, \ldots, a_N\}$. Cet environnement est repéré par un maillage où chaque cellule représente un site. Il comprend des bâtiments et des obstacles, un coût $c(a_i,a_j) \geq 0$ est défini pour chaque déplacement entre deux sites adjacents a_i et a_j. Le site d'adresse a_1 jouera le rôle particulier de site de départ et aussi de site d'arrivée des patrouilles.

En se basant sur la stratégie de modélisation proposée dans le deuxième chapitre, notre objectif est de définir la patrouille pour surveiller une séquence σ de plusieurs sites qui définit un ordre de précédence total sur les sites. Une ou plusieurs mesures doivent être effectuées sur chaque site de la séquence σ. Chaque mesure est supposée être effectuée par un capteur spécifique. Il existe différents types de robots mobiles. Chaque type de robot *r* est équipé par un sous-ensemble spécifique de capteurs *Sensor(r)* qui effectuent certaines mesures. Nous désignons par *Sensor* l'ensemble global des *q* capteurs utilisés pour réaliser la patrouille σ.

Afin de résoudre le problème d'optimisation du point de vue du nombre d'agents de chaque type *r* et de la patrouille de chaque agent de type *r*, la notion de configuration est introduite. Nous appelons configuration, une flotte où les types d'agents ont été entièrement définis [Gam et al.,

2020 ; Gam et al., 2021a]. Elle est considérée comme une relation qui associe les mesures à réaliser par les capteurs dans *Sensor* aux types de robots. X représente l'ensemble des configurations possibles et R_x représente l'ensemble des types de robots dans la configuration $x \in X$. En outre, $Sensor_x(r) \subseteq Sensor$ représente l'ensemble des capteurs installés sur les robots de type $r \in R_x$ dans la configuration x. Il faut noter qu'une configuration ne définit pas le nombre d'agents de chaque type requis par la patrouille. Lorsqu'une seule configuration est considérée, $Sensor_x(r)$ et R_x sont écrits comme $Sensor(r)$ et R pour plus de simplicité.

Un coût est également défini pour chaque capteur ainsi que pour le robot mobile qui supporte les capteurs. Ainsi, chaque type de robot r a un coût particulier $c(r)$ et par conséquent une patrouille donnée a un coût global $C(\sigma,x)$ qui dépend de la configuration x. $C(\sigma,x)$ peut être calculé comme la somme du coût d'investissement qui dépend uniquement de la configuration utilisée pour réaliser la patrouille et du coût d'exploitation qui dépend également des trajectoires des robots dans cette configuration.

Notre objectif est de réaliser les tâches de surveillance définies par la patrouille σ avec un coût minimal $C(\sigma,x)$. En particulier, nous devons déterminer

(i) La configuration x^* de coût minimal ;
(ii) Le nombre de robots de chaque type dans cette configuration ;
(iii) Les trajectoires de chaque robot.

La réalisation des tâches d'inspection est résolue sous les hypothèses suivantes.
Concernant les robots,

(a) chaque robot possède un nombre maximal de capteurs (et ne possède qu'un seul exemplaire de chaque type de capteur) ;

(b) chaque type de capteur effectue une mesure donnée et n'est associé qu'à un seul type de robots dans une configuration donnée. En conséquence, les ensembles $Sensor_x(r)$, $r \in R_x$ forment des partitions de *Sensor*.

Concernant les trajectoires,

(c) chaque robot peut se déplacer du site a_i au site a_j si $c(a_i,a_j)$ est défini, i.e., satisfait $c(a_i,a_j) < \infty$. $c(a_i,a_j)$ n'est défini que pour les sites adjacents et $c(a_i,a_j) = \infty$ sera utilisé pour les sites non adjacents et pour représenter un obstacle entre deux sites adjacents a_i et a_j.

Il n'y a pas d'autre hypothèse sur les trajectoires des robots. En particulier, chaque site de la patrouille peut être visité par plusieurs robots en même temps. De plus, chaque site de l'environnement peut être visité plusieurs fois par le même robot et les robots peuvent effectuer plusieurs mesures sur un même site en même temps si nécessaire.

4.4. Approche SED pour la planification des patrouilles de surveillance

4.4.1. Modélisation du problème

Ce problème est modélisé en représentant l'environnement à deux niveaux différents.

1) Dans une perspective bas niveau, un graphe d'états pondéré et non orienté est utilisé, où chaque état représente un site atomique de l'environnement qui peut être visité par les robots, les déplacements possibles entre les sites sont représentés par les transitions entre les nœuds du graphe.
2) Dans une perspective haut niveau, un modèle PN est utilisé dans lequel chaque place p_i correspond à un site où une ou plusieurs mesures doivent être prises. Une place particulière p_0 est également définie pour le site d'adresse a_1 pour le début et la fin de la mission.

L'ensemble des sites de surveillance d'une mission E est défini par :

$$E = \{(a_1, M(1)), (a_2, M(2)), \ldots, (a_{K-1}, M(K-1)), (a_K, M(K))\} \quad (1)$$

On définit pour l'ensemble E la notion d'ordre totale, on obtient alors la séquence d'inspection σ :

$$\sigma = (a_1, M(1))\,(a_2, M(2)) \ldots (a_{K-1}, M(k-1))\,(a_K, M(k))\,(a_1, M(1)) \quad (2)$$

avec $a_k \in A$, $k=1, \ldots, K$. $M(k) \subseteq Sensor$, $k=1, \ldots, K$ est l'ensemble des mesures à effectuer sur le site a_k et $M(1) = \varnothing$.

Exemple 4.1 : nous considérons l'environnement à surveiller dans la Fig. 4. 3 avec $N = 100$ sites : $A = \{a_0, \ldots, a_{99}\}$ est représenté par un graphe d'état (Fig.4. 3). Quatre mesures a, b, c, d doivent être effectuées dans certains sites. L'ensemble des mesures est $S = \{A, B, C, D\}$. Chaque robot est équipé par un ou plusieurs capteurs A, B, C, D qui sont utilisés pour collecter les mesures de type a, b, c, d.

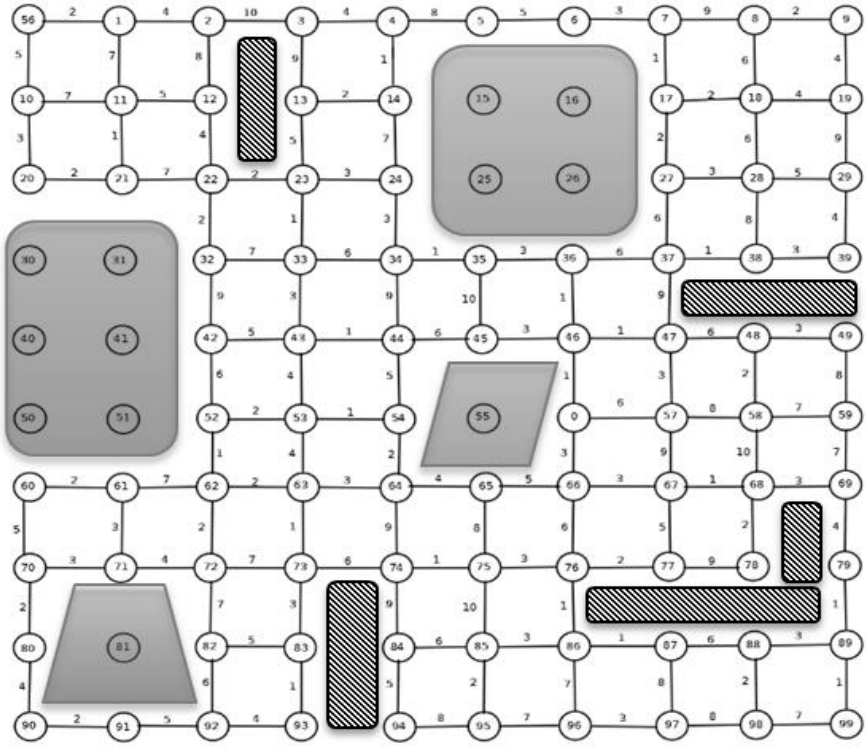


Figure 4. 3 : Un exemple d'environnement sous la forme d'un graphe d'états

Tableau 4. 1 : Coût de chaque dispositif

Agent	Coût
Plateforme Mobile	100
Un capteur de type *A*	20
Un capteur de type *B*	10
Un capteur de type *C*	10
Un capteur de type *D*	40

On considère l'ensemble des 12 sites à visiter :

$E = \{(a_0, \varnothing)(a_{22}, \{a, c, d\})\ (a_{74}, \{a, b\})\ (a_6, \{c, d\})\ (a_{39}, \{a, d\})\ (a_{52}, d)\ (a_{56}, A)\ (a_{87}, \{b, c\})\ (a_{69}, b)\ (a_{94}, \{b, c, d\})\ (a_{82}, \{a, b\})\ (a_{80}, \{c, d\})\}$.

Et la patrouille d'inspection suivante :

$\sigma = (a_0, \varnothing)(a_{22}, \{a, c, d\})\ (a_{74}, \{a, b\})(a_6, \{c, d\})(a_{39}, \{a, d\})(a_{52}, d)(a_{56}, a)\ (a_{87}, \{b, c\})(a_{69}, b)(a_{94}, \{b, c, d\})(a_{82}, \{a, b\})(a_{80}, \{c, d\})(a_0, \varnothing)$.

Dans le site a_0, il n'y a pas de mesure à effectuer, les robots commencent leur mission. Ensuite, sur le site a_{22}, les mesures *a*, *c* et *d* doivent être effectuées, par conséquent ce site doit être obligatoirement visité par un ou plusieurs robots équipés des capteurs *A*, *C* et *D* (les capteurs *A*, *C* et *D* peuvent être portés par le même robot ou par deux ou trois robots différents). Il en va de même pour les sites suivants de la patrouille et enfin les robots doivent retourner au site a_0.

4.4.2. Résolution du problème

Afin de résoudre le problème décrit dans la section 4.3, nous proposons d'utiliser une approche en 3 étapes, comme représenté dans la Fig. 4.4 ci-dessous.

1. Tout d'abord, en énumérant toutes les configurations possibles dans *X*, la configuration avec le coût minimal est calculée par énumération : le coût d'investissement de la patrouille σ est calculé pour chaque configuration *x*. L'environnement est décomposé à l'aide d'un diagramme de Voronoï où chaque centre correspond à un site de la séquence d'inspection. Un PN, associé à la décomposition de Voronoï, est alors construit et la patrouille de surveillance σ est décomposée en séquences élémentaires $\sigma_{r,x}$. Les robots de type $r \in R_x$ doivent visiter les sites correspondant à la séquence élémentaire $\sigma_{r,x}$. Le nombre minimal de robots de type *r* est calculé comme la solution d'un problème d'estimation du marquage initial minimal du PN. Enfin, le calcul du coût d'investissement dépend du nombre de différents types de robots et du coût de chaque dispositif.
2. La trajectoire optimale de chaque robot est calculée en prenant en compte les contraintes de précédence et les hypothèses liées à l'environnement. Les trajectoires doivent prendre en considération la nature de l'environnement spécifiée par les coûts $c(a_i,a_j)$ qui dépendent essentiellement du temps et de l'énergie nécessaires pour se déplacer du site a_i au site a_j. A cet effet, un ensemble de sous-séquences $\sigma^i_{r,x}$ est déterminé pour chaque séquence élémentaire $\sigma_{r,x}$. Chaque sous-séquence $\sigma^i_{r,x}$ sera effectuée par un robot donné de type *r*. Enfin, l'objectif est de déterminer la trajectoire optimale $(pw_{global}{}^i_{r,x})^*$ de coût minimal $(c_{pwglobal}{}^i_{r,x})^*$ pour $\sigma^i_{r,x}$.
3. L'optimisation globale de la patrouille de surveillance est obtenue en couplant les problèmes de configuration et de planification de trajectoire. La configuration optimale de l'ensemble des agents mobiles est déterminée en minimisant le coût global minimal qui résulte des coûts d'investissement et d'exploitation pour chaque configuration possible.

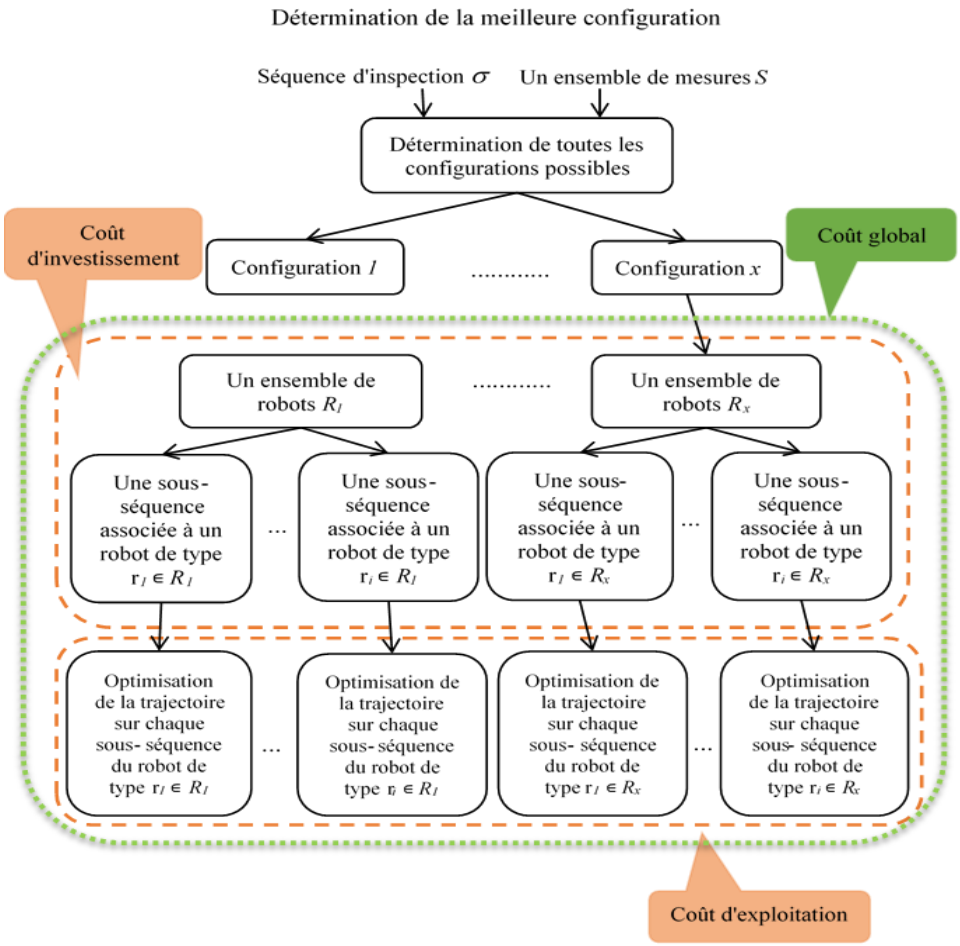


Figure 4. 4 : Une approche en 3 étapes pour optimiser la patrouille d'inspection

4.4.2.1. Configurations et calcul des coûts d'investissement

4.4.2.1.1. Détermination de l'ensemble des configurations

On détermine toutes les configurations possibles dans X en spécifiant les types de robots de chaque configuration, en partant de l'hypothèse que, pour une configuration donnée, le capteur utilisé pour effectuer une mesure donnée est associé à un seul type de robots. Toutes les configurations possibles sont déterminées en utilisant un algorithme récursif [Giua & Seatzu, 2002 ; Dijkstra, 1959]. Il construit la liste des partitions pour un ensemble de q éléments à partir de la liste des partitions pour un ensemble de q-1 éléments et d'un élément supplémentaire. Dans notre problème, les partitions sont calculées sur l'ensemble des capteurs.

Le nombre de partitions possibles pour un ensemble qui a exactement q éléments distincts est donné par le nombre de Bell B_q. La relation d'Aitken [Nobuhiro et al., 2000 ; Canfield, 1995] permet de calculer pas à pas B_q.

$B_{q+1} = \sum_{k=0}^{q} \binom{q}{k} B_k$ (3)

et $B_0 = 1$ (il y a exactement une partition de l'ensemble vide) et $\binom{q}{k} = \frac{k!}{(k-q)!\, k!}$.

- L'idée de l'algorithme constructif est de considérer chaque partition de q-1 éléments et d'ajouter un nouvel élément qui appartient soit à l'une des régions déjà existantes de cette partition, soit forme une nouvelle région [Gam et al., 2020].
- La construction des partitions commence avec k=1 et une seule partition, elle s'arrête avec k=q.

Exemple 4.2 : Considérons l'ensemble de 4 capteurs S = {A, B, C, D}. Il existe 15 configurations possibles qui sont énumérées ci-dessous :

x_1 = {[A], [B], [C], [D]}, x_2 = {[A C], [B D]}, x_3 = {[A C], [B], [D]},
x_4 = {[A D], [B C]}, x_5 = {[A], [B C], [D]}, x_6 = {[A], [B], [C D]},
x_7 = {[A D], [B], [C]}, x_8 = {[A], [B D], [C]}, x_9 = {[A B], [C D]},
x_{10} = {[A B], [C], [D]}, x_{11} = {[A C D], [B]}, x_{12} = {[A], [B C D]},
x_{13} = {[A B C], [D]}, x_{14} = {[A B D], [C]}, x_{15} = {[A B C D]}.

4.4.2.1.2. Décomposition de Voronoï

Un diagramme de Voronoï représente une partition d'une région en cellules (sous-régions adjacentes), déterminées par les distances au sein d'un ensemble de points prédéfinis (i.e., des centres) dans cette région [Voronoï, 1908 ; Aurenhammer, 1991]. Chaque sous-région est le polygone des points qui sont les plus proches d'un centre prédéfini donné (Fig. 4.5). Les cellules représentent d'une certaine manière la "zone d'influence" de ce centre. Plus formellement, soit S, un ensemble fini de n centres dans un espace à 2 dimensions E. La cellule de Voronoï associée à un centre donné $p \in S$, est définie comme suit :

$$Vor(p) = \{x \in E \,/\, \forall q \in S \; \|x - p\| \leq \|x - q\|\} \quad (4)$$

Où $\|.\|$ désigne une distance donnée dans E. Soit $H(p, q)$ le demi-plan contenant p délimité par la médiatrice du segment [p, q], on a :

$$H(p, q) = \{x \in E' \, \|x - p\| \leq \|x - q\|\} \quad (5)$$

et,

$$Vor(p) = \bigcap_{q \in S \setminus \{p\}} H(p, q) \quad (6)$$

Pour un ensemble de centres, le diagramme de Voronoï est donc construit en déterminant les médiatrices de chaque paire de centres. Un point d'une médiatrice appartient à une frontière de Voronoï s'il est équidistant d'au moins deux centres et s'il n'existe pas de distance inférieure entre ce point et un autre centre de l'ensemble *S*.

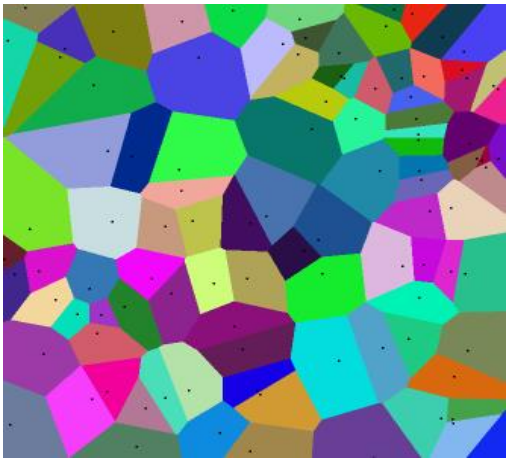


Figure 4. 5 : Un exemple de diagramme de Voronoï : chaque cellule (surface colorée) représente la zone d'influence d'un centre (points noirs)

Dans ce travail, les sites à visiter sont utilisés pour construire un diagramme de Voronoï de l'environnement considéré. La construction des cellules de Voronoï est basée sur la détermination du séparateur linéaire pour chaque paire de sites. De tels séparateurs sont équidistants des deux sites et il n'y a pas d'autre site avec une distance plus petite au séparateur. A partir de la décomposition de Voronoï, il devient possible de concevoir un modèle de réseau de Petri. Chaque place p_i du PN correspond à un site de la séquence. Une place particulière p_0 est associée au site d'adresse a_1. Pour chaque paire de places (p_{i1}, p_{i2}), on définit deux transitions t_{j1} and t_{j2} telles que $t_{j1} \in (p_{i1})°$, $t_{j1} \in °(p_{i2})$ et $t_{j2} \in (p_{i2})°$, $t_{j2} \in °(p_{i1})$ lorsqu'il y a un séparateur entre la paire de sites représenté par (p_{i1}, p_{i2}). Un ensemble de transitions supplémentaires est introduit pour connecter p_0 au reste du réseau :

- Les robots sortent du site associé à p_0 pour commencer la patrouille : chaque place p_i du modèle est connectée à la place p_0 par une transition t'_i telle que $t'_i \in °(p_0)$ et $t'_i \in (p_i)°$.

- Les robots retournent vers le site associé à p_0 lorsque la patrouille est effectuée : chaque place p_i du modèle est connectée à la place p_0 de fin de patrouille par une transition t''_i telle que $t''_i \in °(p_i)$ et $t''_i \in (p_0)°$.

Pour des raisons de lisibilité, les transitions t'_i et t''_i ne seront pas représentées dans les figures suivantes. Initialement, seule la place p_0 d'adresse a_1 correspondant au site du début de la patrouille est marqué et son marquage est inconnu.

Exemple 4.3 : Considérons le même ensemble *E* avec 12 sites à surveiller :

$e = \{(a_0, \varnothing)(a_{22}, \{a, c, d\})\ (a_{74}, \{a, b\})\ (a_6, \{c, d\})\ (a_{39}, \{a, d\})\ (a_{52}, d)\ (a_{56}, a)\ (a_{87}, \{b, c\})\ (a_{69}, b)\ (a_{94}, \{b, c, d\})\ (a_{82}, \{a, b\})\ (a_{80}, \{c, d\})\}$,

et la séquence d'inspection suivante :

$\sigma = (a_0, \varnothing)(a_{22}, \{a, c, d\})\ (a_{74}, \{a, b\})(a_6, \{c, d\})(a_{39}, \{a, d\})(a_{52}, d)(a_{56}, a)\ (a_{87}, \{b, c\})(a_{69}, b)(a_{94}, \{b, c, d\})(a_{82}, \{a, b\})(a_{80}, \{c, d\})(a_0, \varnothing)$.

L'environnement de la Fig. 4. 3 sera divisé en 12 cellules. La décomposition de Voronoï de l'environnement de la Fig. 4. 3 est représentée en rouge sur la Fig. 4. 6.

Le réseau de Petri associé à cette décomposition est représenté en couleur bleue dans la Fig. 4. 6 tel que $P = \{p_0, p_{22}, p_{74}, p_6, p_{39}, p_{52}, p_{56}, p_{87}, p_{69}, p_{94}, p_{82}, p_{80}\}$ représente l'ensemble des sites à visiter, $T = \{t_1, \ldots, t_{30}\}$ représente l'ensemble des déplacements possibles entre les sites à partir du site a_0, $T' = \{t'_{22}, t'_{74}, t'_6, t'_{39}, t'_{52}, t'_{56}, t'_{87}, t'_{69}, t'_{94}, t'_{82}, t'_{80}\}$ représente l'ensemble des déplacements possibles à partir du site a_0, et $T'' = \{t''_{22}, t''_{74}, t''_6, t''_{39}, t''_{52}, t''_{56}, t''_{87}, t''_{69}, t''_{94}, t''_{82}, t''_{80}\}$ représente l'ensemble des déplacements possibles vers le site a_0. La séquence σ est reformulée dans le modèle PN comme indiqué ci-dessous.

$\sigma = (p_0, \varnothing)(p_{22}, \{a, c, d\})\ (p_{74}, \{a, b\})\ (p_6, \{c, d\})\ (p_{39}, \{a, d\})(p_{52}, \{d\})(p_{56}, \{a\})\ (p_{87}, \{b, c\})(p_{69}, \{b\})(p_{94}, \{b, c, d\})(p_{82}, \{a, b\})(p_{80}, \{c, d\})\ (p_0, \varnothing)$. (7)

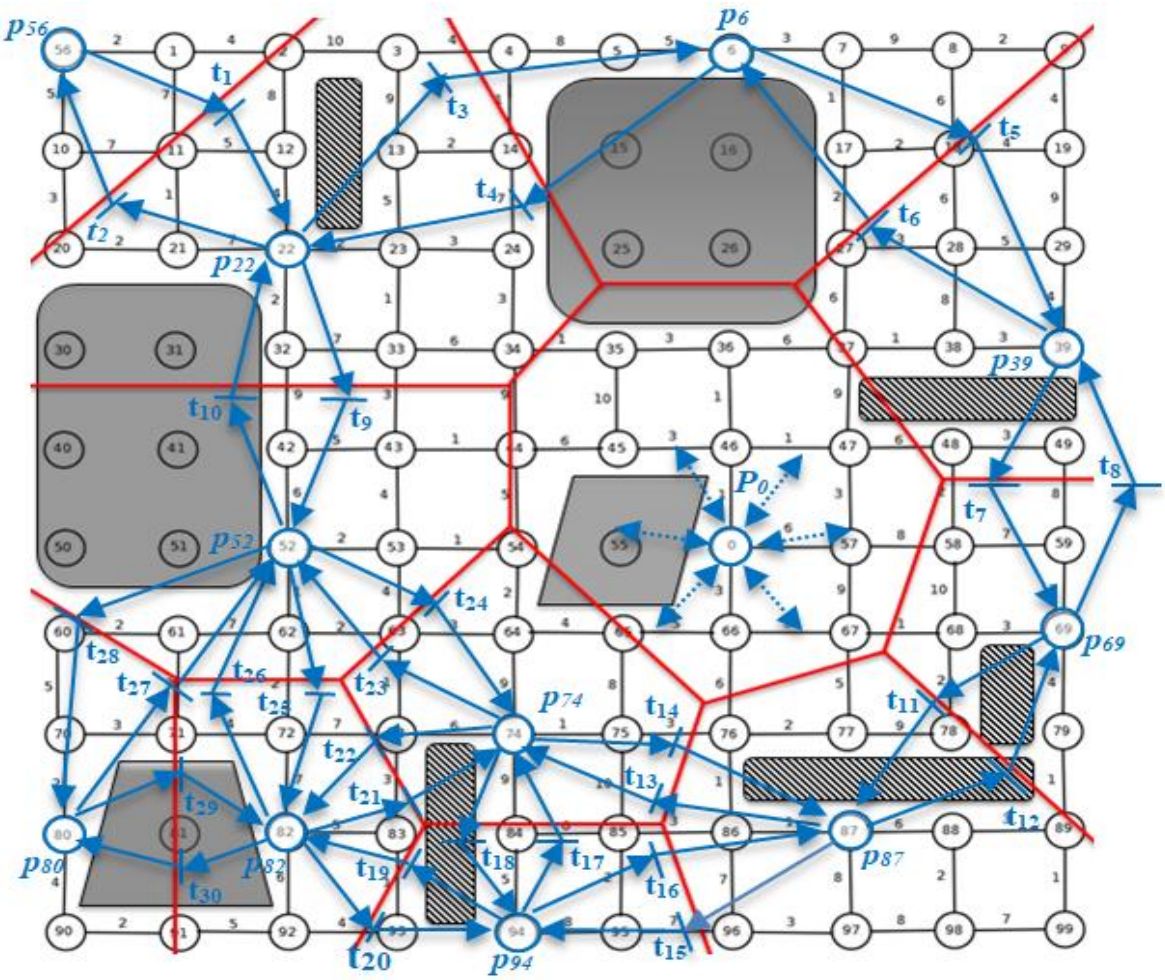


Figure 4. 6 : Décomposition de Voronoï et extraction du modèle par réseau de Petri

4.4.2.1.3. Calcul des coûts d'investissement

Dans la suite de cette section, nous désignons chaque site a_k de la séquence σ par la place *p(k)* du modèle PN. On peut donc réécrire σ :

$$\sigma = (p(1), \varnothing)\ (p(2), M(2))\ \ldots\ (p(k-1), M(k-1))\ (p(k), \varnothing) \qquad (8)$$

Tout d'abord, on décompose la séquence d'inspection σ en séquences élémentaires $\sigma_{r,x}$, chacune correspond aux sites qui doivent être visités par un robot de type $r \in R_x$: un site *p(k)* avec des mesures *M(k)* appartient à la séquence élémentaire $\sigma_{r,x}$ si au moins une mesure requise en *p(k)* est réalisée par un capteur d'un robot de type *r* dans la configuration *x*. Tant que les différents types de robots mettent en œuvre une partition de l'ensemble des capteurs, cette mesure ne peut pas être réalisée par un robot d'un autre type et le site *p(k)* doit être visité par un robot de type *r*. Pour chaque ensemble de mesures *M*, on définit :

$M' = M \cap Sensor_x(r)$. (9)

Pour $\sigma = (p, M)\ \sigma'$, on introduit le filtre $F_{r,x}$ qui est défini récursivement par

$F_{r,x}(\sigma) = F_{r,x}((p, M)\ \sigma') = (p, M')\ F_{r,x}(\sigma')$ if $M' \neq \varnothing$,
$F_{r,x}(\sigma) = F_{r,x}((p, M)\ \sigma') = F_{r,x}(\sigma')$ autrement,
$F_{r,x}(\varepsilon) = \varepsilon$ (10)

où ε représente la séquence vide. $F_{r,x}$ calcule récursivement la séquence élémentaire $\sigma_{r,x}$ à partir de σ.

$\sigma_{r,x} = (p_0, \varnothing)\ F_{r,x}(\sigma)\ (p_0, \varnothing)$ (11)

Ensuite, chaque séquence élémentaire $\sigma_{r,x}$ de k' places est transformée en une séquence de tir $\sigma'_{r,x}$ avec k'-1 transitions à franchir dans le modèle PN et le marquage initial minimal $M_I(\sigma'_{r,x})$ nécessaire pour franchir la séquence $\sigma'_{r,x}$ est calculé de manière itérative [Wang et al., 2011 ; Chao & Hongxia, 2010].

$M_I(j+1) = \max\{M_I(j) + W.Y_{j-1}, W_{PR}(:,t_{i(j)})\} - W.Y_{j-1},\ j=1,\ldots,p$ (12)

où le vecteur de Parikh Y_{j-1} correspond à la séquence de tir $\sigma'(j-1) = t_{i(1)} .. t_{i(j-1)}$, $M_I(1)$ est le vecteur nul de dimension n, et Y_0 est le vecteur nul de dimension p qui correspond à la séquence vide ε. Le vecteur $M_I(j)$ est le marquage initial minimal nécessaire pour franchir la séquence $\sigma'(j-1)$. Les auteurs dans [Wang et al., 2011 ; Chao & Hongxia, 2010] ont prouvé que l'estimation résultante $M_I(p+1)$ est minimale (c'est-à-dire qu'il n'existe aucun marquage $M < M_I(p+1)$ qui valide σ'). Par conséquent, la proposition 1 peut être formulée.

Proposition 1 : Le nombre minimal $m_{r,x}$ de robots de type $r \in R_x$ dans une configuration x donnée, nécessaire pour effectuer la séquence σ est donné par l'équation (13)

$m_{r,x}(\sigma) = M_I(\sigma'_{r,x}).(\mathbf{1})_n$ (13)

où $(\mathbf{1})_n$ est un vecteur de dimension n dont toutes les entrées sont égales à 1 et le marquage initial minimal nécessaire pour franchir $\sigma'_{r,x}$ est $M_I(\sigma'_{r,x})$, qui est calculé avec (12).

Preuve : Selon les hypothèses (a) et (b), la séquence σ est décomposée en un ensemble de séquences élémentaires $\sigma_{r,x}$. Cette décomposition est unique. Le marquage initial minimal est

calculé comme $M_I(\sigma'_{r,x}) = M_I(k'+2)$ pour chaque séquence élémentaire de longueur $k'+1$ de la forme :

$$\sigma_{r,x} = (p'(1), \varnothing)\ (p'(2), M'(2))\ \ldots\ (p'(k'+1), \varnothing) \qquad (14)$$

Par conséquent, le nombre minimal de robots de type r dans la configuration x est donné par (13).

Enfin, le coût d'investissement $C_{invest}(\sigma,x)$ pour la configuration x est la somme du nombre minimal de robots de chaque type pondéré par le coût des différents types de robots.

$$C_{invest}(\sigma,x) = \sum_{r \in R_x} \left(m_{r,x}(\sigma).c(r) \right) \qquad (15)$$

Exemple 4.4 : on considère à nouveau la séquence (7) de l'exemple 4.3 et la configuration $x_5 = \{[A], [B\ C], [D]\}$ qui correspond à 3 types de robots r_1, r_2 et r_3 tels que $S_{x5}(r_1) = \{A\}$, $S_{x5}(r_2) = \{B, C\}$, $S_{x5}(r_3) = \{D\}$. La séquence considérée dans l'exemple 4.1 est décomposée en 3 séquences élémentaires :

$\sigma_{r1, x5} = (p_0, \varnothing)\ F_{r1, x5}(\sigma)\ (p_0, \varnothing)$
$= (p_0, \varnothing)\ (p_{22}, a)\ (p_{74}, a)\ (p_{39}, a)\ (p_{56}, a)\ (p_{82}, a)\ (p_0, \varnothing).$
$\sigma_{r2, x5} = (p_0, \varnothing)\ F_{r2, x5}(\sigma)\ (p_0, \varnothing)$
$= (p_0, \varnothing)\ (p_{22}, c)\ (p_{74}, b)\ (p_6, c)\ (p_{87}, b, c)\ (p_{69}, b)\ (p_{94}, b, c)\ (p_{82}, b)\ (p_{80}, c)\ (p_0, \varnothing).$
$\sigma_{r3, x5} = (p_0, \varnothing)\ F_{r3, x5}(\sigma)\ (p_0, \varnothing)$
$= (p_0, \varnothing)\ (p_{22}, d)\ (p_6, d)\ (p_{39}, d)\ (p_{52}, d)\ (p_{94}, d)\ (p_{80}, d)\ (p_0, \varnothing).$

Les séquences élémentaires précédentes sont transformées en séquences de tirs :

$\sigma'_{r1,x5} = t'_{22}\ t'_{74}\ t'_{39}\ t_2\ t_{22}\ t''_{82}\ t''_{39}\ t''_{56}$

$\sigma'_{r2,x5} = t'_{22}\ t'_{74}\ t_3\ t_{14}\ t'_{69}\ t_{15}\ t_{19}\ t_{30}\ t''_{80}\ t''_6\ t''_{69}$

$\sigma'_{r3,x5} = t'_{22}\ t_3\ t_5\ t'_{52}\ t'_{94}\ t_{28}\ t''_{80}\ t''_{94}\ t''_{39}$

D'après la proposition 1, $m_{r1, x5}(\sigma) = M_I\ (\sigma'_{r1, x5}).\ (\mathbf{1})_{100} = 3$. Trois robots de type r_1 sont nécessaires pour effectuer la mesure A. Il en est de même pour les robots de types r_2 et r_3 : $m_{r2, x5}(\sigma) = M_I\ (\sigma'_{r2, x5}).\ (\mathbf{1})_{100} = 3$ et $m_{r3, x5}(\sigma) = M_I\ (\sigma'_{r3, x5}).\ (\mathbf{1})_{100} = 3$.

D'après (15), on peut calculer le coût d'investissement pour la configuration x_5 : on propose que $c(r_1) = 120$, $c(r_2) = 120$, $c(r_3) = 140$, alors :

$$C_{invest}(\sigma,x) = \sum_{r \in R_x} \left(m_{r,x}(\sigma).c(r) \right) = (m_{r1, x5}(\sigma) \times c(r_1)) + (m_{r2, x5}(\sigma) \times c(r_2)) + (m_{r3, x5}(\sigma) \times c(r_3)) = 1140.$$

4.4.2.2. Planification des trajectoires et calcul du coût global

4.4.2.2.1. Planification des trajectoires

On détermine d'abord les sous-séquences $\sigma^i_{r,x}$ (avec i = 1, ..., $m_{r,x}(\sigma)$) de chaque séquence élémentaire $\sigma_{r,x}$. Chaque sous-séquence sera effectuée par un robot de type $r \in R_x$. Nous réécrivons $\sigma^i_{r,x}$ comme suit :

$$\sigma^i_{r,x} = (p^i_{r,x}(1), \varnothing)\ (p^i_{r,x}(2), M^i_{r,x}(2))\ \ldots\ (p^i_{r,x}(k_i\text{-}1), M^i_{r,x}(k_i\text{-}1))\ (p^i_{r,x}(k_i), \varnothing) \quad (16)$$

La trajectoire admissible $pw_{global}{}^i_{r,x}$ de coût $c_{pwglobal}{}^i_{r,x}$ est définie alors pour chaque robot i qui participe à la séquence élémentaire $\sigma_{r,x}$ en respectant l'ordre de visite imposé par $\sigma^i_{r,x}$. Le problème est résolu en décomposant $\sigma^i_{r,x}$ en une succession de paires de sites consécutives $\sigma^i_{r,x}$ (*1,2*), $\sigma^i_{r,x}$ (*2,3*),… $\sigma^i_{r,x}$ (*k-1,k*), avec $\sigma^i_{r,x}$ *(j,j+1) = p*^{i}*(j) p*^{i}*(j+1), j =1,…,k-1* et *p*^{i}*(j), p*^{i}*(j+1)*$\subseteq P$. On désigne toute trajectoire de *p*^{i}*(j)* à *p*^{i}*(j+1)* par $pw^i_{r,x}(j,j+1)$:

$$pw^i_{r,x}(j,j+1) = p^i_{r,x}(j)\ a^i_{r,x}(j,1)\ a^i_{r,x}(j,2)\ldots\ a^i_{r,x}(j,k_i\text{-}1)\ p^i_{r,x}(j+1) \quad (17)$$

Avec :

- $a^i_{r,x}(h) \in A$, $h=1, \ldots, k_i$
- $p^i_{r,x}(j)$ et $p^i_{r,x}(j+1)$ sont adjacents pour $j=1, \ldots, k_i\text{-}1$
- $p^i_{r,x}(j) = a(j) = a^i_{r,x}(h\text{-}1)$, et $p^i_{r,x}(j+1) = a(j+1) = a^i_{r,x}(h,k_i)$

Le coût $c_{pw}{}^i_{r,x}(j,j+1)$ de la trajectoire $pw^i_{r,x}(j,j+1)$ est la somme des coûts des déplacements élémentaires $c(a_i(h),a_i(h+1))$ pour aller de $a_i(h)$ à $a_i(h+1)$:

$$c_{pw}{}^i_{r,x}(j,j+1) = \sum_{h=1}^{k_i-1} c(a_i(h), a_i(h+1)) \quad (18)$$

La trajectoire globale $pw_{global}{}^i_{r,x}$ de chaque robot i qui participe à la séquence élémentaire $\sigma_{r,x}$ est réalisée en assemblant par concaténation les k-1 sous-trajectoires $pw^i_{r,x}(j,j+1)$ obtenues précédemment et en supprimant les doublons aux extrémités des sous-trajectoires :

$$pw_{global}{}^i_{r,x} = p^i_{r,x}(1)\ a^i_{r,x}(1,1)\ldots\ a^i_{r,x}(1,k_1\text{-}1)\ p^i_{r,x}(2)\ a^i_{r,x}(2,1)\ldots\ a^i_{r,x}(2,k_2\text{-}1)\ \ldots\ p^i_{r,x}(k\text{-}1)\,a^i_{r,x}(k\text{-}1,1)\ldots\ a^i_{r,x}(k\text{-}1,k_{k-1}\text{-}1)\ p^i_{r,x}(k) \quad \text{avec } a^i_{r,x}(k_h\text{-}1,j) \in A,\ j=1, \ldots, k_h\text{-}1,\ h = 1,\ldots,k \quad (19)$$

Le coût d'exploitation $c_{pwglobal}{}^i_{r,x}$ de la trajectoire $pw_{global}{}^i_{r,x}$ est la somme des coûts des sous-trajectoires obtenues précédemment :

$$c_{pwglobal}{}^i_{r,x} = \sum_{i=1}^{k-1} c_{pw_{r,x}}{}^i(j, j+1) \quad (20)$$

L'algorithme de Dijkstra est utilisé pour résoudre le problème de la recherche de la trajectoire optimale (c'est-à-dire la trajectoire de coût minimal) $(pw^{i}_{r,x})^{*}(j,j+1)$ entre les nœuds $p^{i}_{r,x}(j)$ et $p^{i}_{r,x}(j+1)$, $j = 1,\ldots,k_i$, [Dijkstra, 1959 ; Wang et al., 2011 ; Chao & Hongxia, 2010]. Nous désignons le coût d'exploitation pour la trajectoire optimale $(pw^{i}_{r,x})^{*}(j,j+1)$ par $(c_{pw^{i}_{r,x}})^{*}(j,j+1)$.

Enfin, le coût d'exploitation $C_{op}(\sigma,x)$ de la patrouille pour la configuration x est défini par (21):

$$C_{op}(\sigma,x) = \sum_{r \in R_x} \sum_{i=1}^{m_{r,x}(\sigma)} (c_{pwglobal^{i}_{r,x}})^{*} \qquad (21)$$

Exemple 4.5 : On considère à nouveau la séquence (7) de l'exemple 4.3 et la configuration x_5. La détermination des sous-séquences de chaque robot qui participe aux séquences élémentaires est montrée ci-dessous :

- Pour $\sigma_{r1,x5} = (p_0, \varnothing)(p_{22}, a)\ (p_{74}, a)\ (p_{39}, a)\ (p_{56}, a)\ (p_{82}, a)\ (p_0, \varnothing)$, les sous- séquences $\sigma^{i}_{r1,x5}$, $i = 1, \ldots, 3$ sont :

  $\sigma^{1}_{r1,x5} = (p_0, \varnothing)(p_{22}, a)\ (p_{56}, a)\ (p_0, \varnothing)$
  $\sigma^{2}_{r1,x5} = (p_0, \varnothing)(p_{74}, a)\ (p_{82}, a)\ (p_0, \varnothing)$
  $\sigma^{3}_{r1,x5} = (p_0, \varnothing)(p_{39}, a)\ (p_0, \varnothing)$

- Pour $\sigma_{r2,x5} = (p_0, \varnothing)(p_{22}, c)\ (p_{74}, b)\ (p_6, c)\ (p_{87}, b, c)\ (p_{69}, b)\ (p_{94}, b, c)\ (p_{82}, b)\ (p_{80}, c)\ (p_0, \varnothing)$, les sous- séquences $\sigma^{i}_{r2,x5}$, $i = 1, \ldots, 3$ sont :

  $\sigma^{1}_{r2,x5} = (p_0, \varnothing)(p_{22}, c)\ (p_6, c)\ (p_0, \varnothing)$
  $\sigma^{2}_{r2,x5} = (p_0, \varnothing)(p_{74}, b)\ (p_{87}, b, c)\ (p_{69}, b)\ (p_0, \varnothing)$
  $\sigma^{3}_{r2,x5} = (p_0, \varnothing)(\ p_{94}, b, c)\ (p_{82}, b)\ (p_{80}, c)\ (p_0, \varnothing)$

- Pour $\sigma_{r3,x5} = (p_0, \varnothing)(p_{22}, d)\ (p_6, d)\ (p_{39}, d)\ (p_{52}, d)\ (p_{94}, d)\ (p_{80}, d)\ (p_0, \varnothing)$, les sous-séquences $\sigma^{i}_{r3,x5}$, $i = 1, \ldots, 3$ sont :

  $\sigma^{1}_{r3,x5} = (p_0, \varnothing)(p_{22}, d)\ (p_6, d)\ (p_{39}, d)\ (p_0, \varnothing)$
  $\sigma^{2}_{r3,x5} = (p_0, \varnothing)(p_{52}, d)\ (p_{80}, d)\ (p_0, \varnothing)$
  $\sigma^{3}_{r3,x5} = (p_0, \varnothing)(\ p_{94}, d)\ (p_0, \varnothing)$

L'algorithme de Dijkstra est ensuite appliqué pour chaque paire de sites successifs de $\sigma^{i}_{r,x}(j,j+1)$. Par exemple, pour les 3 robots de type $r_1 \in R_{x5}$ qui participent à la patrouille, nous détaillons les trajectoires optimales et les coûts d'exploitation dans le tableau 4. 2.

Tableau 4. 2 : Trajectoires et coûts optimaux de la sous-séquence $\sigma^{1}_{r1,\,x5}$

$\sigma^{1}_{r1,x5}(i,i+1)$	$(pw^{1}_{\ r1,x5})^{*}(i,i+1)$	$c_{(pw1r1,x5)*}(i,i+1)$
$p_0\ p_{22}$	$p_0\ a_{46}\ a_{36}\ a_{35}\ a_{34}\ a_{24}\ a_{23}\ p_{22}$	14
$p_{22}\ p_{56}$	$p_{22}\ a_{21}\ a_{11}\ a_1\ p_{56}$	17
$P_{56}\ p_0$	$p_{56}\ a_{10}\ a_{20}\ a_{21}\ a_{22}\ a_{23}\ a_{24}\ a_{34}\ a_{35}\ a_{36}\ a_{46}\ p_0$	31

Par conséquent :

$(pw_{global}{}^{1}{}_{r1,x5})^{*} = p_0\ a_{46}\ a_{36}\ a_{35}\ a_{34}\ a_{24}\ a_{23}\ p_{22}\ a_{21}\ a_{11}\ a_1\ p_{56}\ a_{10}\ a_{20}\ a_{21}\ a_{22}\ a_{23}\ a_{24}\ a_{34}\ a_{35}\ a_{36}\ a_{46}\ p_0$

Le coût minimal $(c_{pwglobal}{}^{1}{}_{r1,x5})^{*}$ de la trajectoire globale $(pw_{global}{}^{1}{}_{r1,x5})^{*}$ est 62. Les trajectoires optimales sont résumées dans le tableau 4. 3.

Tableau 4. 3 : Trajectoires et coûts élémentaires optimaux

Trajectoire	Détail	Coût
$\sigma^{1}_{r1,x5}(i,i+1)$	p_0 a46 a36 a35 a34 a24 a23 p_{22} a21 a11 a1 p_{56} a10 a20 a21 a22 a23 a24 a34 a35 a36 a46 p_0	62
$\sigma^{2}_{r1,x5}(i,i+1)$	p_0 a66 a76 a75 p_{74} a73 a83 p_{82} a83 a73 a63 a64 a65 a66 p_0	51
$\sigma^{3}_{r1,x5}(i,i+1)$	p_0 a46 a36 a37 a38 p_{39} a38 a37 a36 a46 p_0	24
Type r_1		137
$\sigma^{1}_{r2,x5}(i,i+1)$	p_0 a46 a36 a35 a34 a24 a23 p_{22} a23 a13 a14 a4 a5 p_6 a7 a17 a27 a37 a36 a46 p_0	57
$\sigma^{2}_{r2,x5}(i,i+1)$	p_0 a66 a76 a75 p_{74} a75 a76 a86 p_{87} a86 a76 a77 a67 a68 p_{69} a68 a67 a66 p_0	42
$\sigma^{3}_{r2,x5}(i,i+1)$	p_0 a66 a76 a86 a85 a95 p_{94} a84 a74 a73 a83 p_{82} a72 a71 a70 p_{80} a70 a71 a72 a62 a63 a64 a65 a66 p_0	95
Type r_2		194
$\sigma^{1}_{r3,x5}(i,i+1)$	p_0 a46 a36 a35 a34 a24 a23 p_{22} a23 a13 a14 a4 a5 p_6 a7 a17 a27 a37 a38 p_{39} a38 a37 a36 a46 p_0	65
$\sigma^{2}_{r3,x5}(i,i+1)$	p_0 a46 a45 a44 a43 a53 p_{52} a62 a72 a71 a70 p_{80} a70 a71 a72 a62 a63 a64 a65 a66 p_0	57
$\sigma^{3}_{r3,x5}(i,i+1)$	p_0 a66 a76 a86 a85 a95 p_{94} a95 a85 a86 a76 a66 p_0	46
Type r_3		168

Selon (21), on peut calculer le coût d'exploitation des robots de la configuration x_5 pour exécuter la patrouille :

$$C_{op}(\sigma,x_5) = \sum_{r \in R_{x_5}} \sum_{i=1}^{m_{r,x_5}(\sigma)} (c_{pwglobal^{i}_{r,x_5}})^{*} = 499$$

4.4.2.2.2. Coût global et configuration optimale

Le coût global résultant $C(\sigma,x)$ pour chaque configuration x dépend du nombre minimal de robots mobiles calculé précédemment, du coût de chaque type de robot et du coût de chaque trajectoire des sites à visiter par chaque robot. Ce coût est calculé en additionnant le coût d'investissement du matériel et le coût d'exploitation des robots.

$$C(\sigma,x) = C_{invest}(\sigma,x) + C_{op}(\sigma,x)$$
$$= \sum_{r \in R_x}\left(m_{r,x}(\sigma).c(r)\right) + \sum_{r \in R_x}\sum_{i=1}^{m_{r,x}(\sigma)}(c_{pwglobal_{r,x}^{i}})^{*} \quad (22)$$

Afin de déterminer la meilleure configuration x^* et en même temps le nombre de robots de chaque type dans cette configuration, on évalue le coût de chaque configuration. Par conséquent, x^* est obtenu comme suit :

$$x^* = \underset{x \in X}{\operatorname{argmin}}\{C(\sigma,x)\} \quad (23)$$

Exemple 4.6 : Selon (22), le coût global $C(\sigma,x_5)$ de la configuration x_5 est détaillé ci-dessous

$$C(\sigma,x_5) = C_{invest}(\sigma,x_5) + C_{op}(\sigma,x_5)$$
$$= 1140 + 499$$
$$= 1639.$$

Les résultats obtenus pour chaque configuration sont indiqués dans le tableau 4. 4. La meilleure configuration avec un coût minimal qui effectue la patrouille de surveillance est $x_{15} = \{A\ B\ C\ D\}$ avec $C\ (\sigma,x_{15}) = 992$. Cette configuration correspond à 4 robots de même type, chacun d'entre eux portant tous les capteurs. Si nous supposons que le nombre maximal de capteurs à transporter par chaque plate-forme mobile est limité à 3 ou 2 capteurs, la meilleure configuration deviendra $x_9 = \{[A\ B], [C\ D]\}$ avec $C(\sigma, x_9) = 1128$. Cette configuration correspond à 6 robots répartis selon 2 types (il y a deux capteurs sur chaque robot dans cette configuration). Le tableau 4. 4 indique également le nombre maximal de capteurs par robot (N_S) ainsi que le nombre total de robots nécessaires pour effectuer la patrouille (N_R).

Tableau 4. 4 : Coût de chaque configuration

N_S	Configurations x	$C_{invest}(\sigma,x)$	$C_{op}(\sigma,x)$	$C(\sigma,x)$	N_R
1	x_1 = {[*A*], [*B*], [*C*], *D*]}	1330	583	1913	11
2	x_2 = {[*A C*], [*B D*]}	840	442	1282	6
	x_3 = {[*A C*], [*B*], [*D*]}	1030	517	1547	8
	x_4 = {[*A D*], [*B C*]}	1000	444	1444	7
	x_5 = {[*A*], [*B C*], [*D*]}	1140	499	1639	9
	x_6 = {[*A*], [*B*], [*C D*]}	1030	424	1454	8
	x_7 = {[*A D*], [*B*], [*C*]}	1190	528	1718	9
	x_8 = {[*A*], [*B D*], [*C*]}	1140	508	1648	9
	x_9 = {[*A B*], [*C D*]}	840	298	1138	6
	x_{10} = {[*A B*], [*C*], [*D*]}	1140	457	1597	9
3	x_{11} = {[*A C D*], [*B*]}	900	369	1269	6
	x_{12} = {[*A*], [*B C D*]}	840	347	1187	6
	x_{13} = {[*A B C*], [*D*]}	840	420	1260	6
	x_{14} = {[*A B D*], [*C*]}	1010	433	1443	7
4	x_{15} = {*A B C D*}	720	272	992	4

4.5. Approche SED simplifiée pour la configuration des patrouilles de surveillance

Cette section présente une simplification de la méthode présentée dans la section précédente. En particulier, nous avons considéré les deux simplifications suivantes. L'environnement est directement représenté par un réseau de Petri sans passer par un graphe orienté. Le sous problème de planification des trajectoires n'est donc pas discuté dans cette partie. De plus, le coût global de la configuration dépend, ici, seulement du coût d'exploitation des agents, et le coût d'investissement des robots et capteurs n'est pas pris en compte. D'autres simplifications mineures, telles que l'omission du site initial dans l'écriture des séquences, sont aussi appliquées.

4.5.1. Modélisation du problème

Ce problème est modélisé en représentant l'environnement par un PN où chaque place représente un site. Les déplacements entre les sites sont représentés par des transitions et les robots sont modélisés par des jetons. Une place supplémentaire p_{EP} est utilisé pour montrer que les tâches d'inspection sont effectuées et chaque place p_k du modèle est connecté à p_{EP} par une

transition t'_i. Dans la suite de cette section, nous désignons chaque place p_k de la séquence σ par la place $p(k)$ du modèle PN.

La séquence des sites à surveiller et les mesures à effectuer dans chaque site sont représentées par σ :

$$\sigma = (p(1), M(1))\ (p(2), M(2)) \cdots (p(K), M(K)) \qquad \text{avec } M(k) \subseteq Sensor,\ k = 1,\ldots,K. \qquad (24)$$

Exemple 4.7 : on considère un environnement avec 9 sites de la Fig. 4. 7 comme un exemple. Quatre mesures a, b, c, d doivent être effectuées dans les différents sites. L'ensemble des mesures est $S = \{a, b, c, d\}$.

7	8	9
4	5	6
1	2	3

Figure 4. 7 : Un environnement avec 9 sites

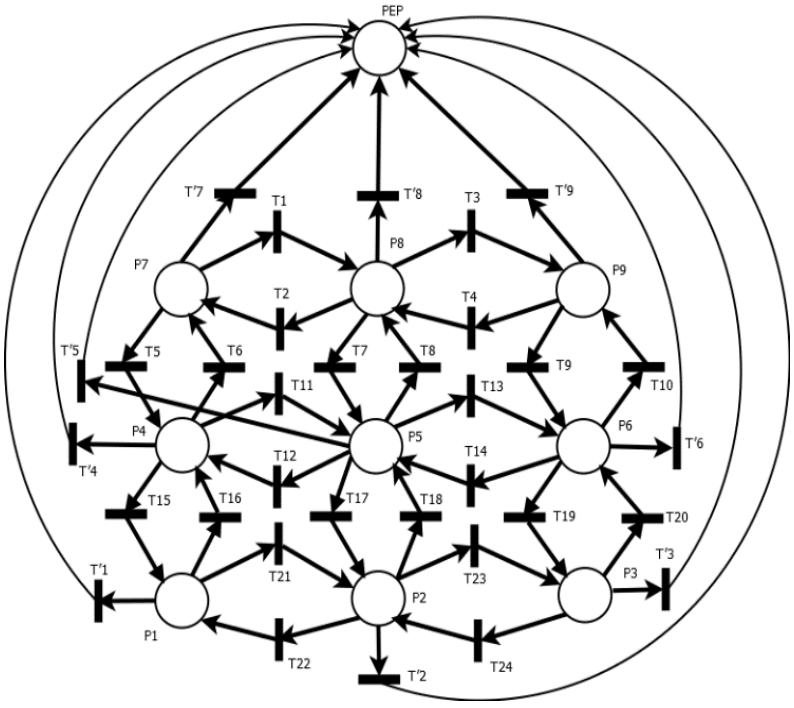


Figure 4. 8 : Modèle PN de l'environnement de la Fig. 4. 7

Tableau 4. 5 : Coûts des dispositifs

Agent	**Coût**
Plateforme mobile	100
Un capteur de type *A*	20
Un capteur de type *B*	10
Un capteur de type *C*	10
Un capteur de type *D*	40

Le modèle PN de cet environnement est présenté dans la Fig. 4. 8 où $P = \{p_1, p_2, \ldots, p_9\}$ représente les sites à visiter et $T = \{t_1, t_2, \ldots, t_{24}\}$ représente les déplacements possibles entre les sites. Les coûts des robots et des capteurs sont définis dans le tableau 4. 5. On considère la séquence σ avec 7 sites successifs à surveiller :

$\sigma = (p_7, \{a, c\})(p_4, \{b, d\})(p_8, \{a, c\})(p_9, \{a, c, d\})(p_1, \{a, b, d\})(p_6, \{a, b\})(p_2, \{a, b\})$.

Dans les sites p_7 et p_8, les mesures a et c doivent être effectuées ; puis, dans le site p_4, les mesures b et d doivent être effectuées ; dans le site p_9, les mesures a, c et d doivent être effectuées ; dans le site p_1, les mesures a, b et d ; dans le site p_6, les mesures a et b doivent être effectuées ; enfin, dans le site p_2, les mesures a et b doivent être effectuées.

4.5.2. Résolution du problème

D'abord, toutes les configurations possibles dans X sont listées. La configuration avec le coût minimal est calculée en calculant le coût de la séquence σ pour chaque configuration x.

À cet effet, σ est décomposé en séquence d'inspection élémentaires. Chaque séquence d'inspection élémentaire $\sigma_{r,x}$ correspond aux sites qui doivent être visités par un robot de type $r \in R_x$. Le problème qui vise à trouver le nombre minimal $m_{r,x}$ de robots de type r nécessaires pour exécuter les tâches de surveillance définies par $\sigma_{r,x}$, est reformulé comme un problème de marquage initial minimal dans le modèle PN. Afin de résoudre ce problème, nous utilisons l'approche présentée dans la section 4.4.2.1.1 pour déterminer l'ensemble des configurations possibles.

4.5.2.1. Décomposition de la séquence σ

En se basant sur l'approche de calcul des coûts d'investissements citée dans la section 4.4.2.1.3, la séquence d'inspection σ est décomposé en séquences d'inspection élémentaires $\sigma_{r,x}$ qui correspondent chacune aux sites qui doivent être visités par un robot de type $r \in R_x$.

Dans cette partie, nous modélisons la séquence d'inspection élémentaire par :

$$\sigma_{r,x} = F_{r,x}(\sigma)\,(p_{EP}, \varnothing) \qquad (25)$$

où $F_{r,x}$ calcule récursivement $\sigma_{r,x}$ à partir de σ et le site final de la patrouille p_{EP} est ajouté comme site final pour chaque ronde d'inspection élémentaire $\sigma_{r,x}$.

Exemple 4.8 : Considérons la configuration x_{10} = {[A B], [C], [D]} qui correspond à 3 types de robots r_1, r_2 et r_3 tels que $S_{x10}(r_1)$ = {A B}, $S_{x10}(r_2)$ = {C}, $S_{x10}(r_3)$ = {D}. Selon le tableau 4. 5, on peut calculer le coût de chaque type de robot : $c(r_1)$ = 130, $c(r_2)$ = 110, $c(r_3)$ = 140. La séquence σ considérée dans l'exemple 4.7 est décomposée en 3 séquences d'inspection élémentaires :

$\sigma_{r1,x10} = F_{r1,x10}(\sigma)(p_{EP}, \varnothing)$

$=(p_7, a)(p_4, b)(p_8, a)(p_9, a)(p_1, a, b)(p_6, a, b)(p_2, a, b)(p_{EP}, \varnothing).$

$\sigma_{r2,x10} = F_{r2,x10}(\sigma)(p_{EP}, \varnothing) = (p_7, c)(p_8, c)(p_9, c)(p_{EP}, \varnothing)$

$\sigma_{r3,x10} = F_{r3,x10}(\sigma)(p_{EP}, \varnothing) = (p_4, d)(p_9, d)(p_1, d)(p_{EP}, \varnothing)$

4.5.2.2. Détermination des séquences de transitions élémentaires

Chaque séquence d'inspection élémentaire $\sigma_{r,x}$ est transformée en une séquence de tir $\sigma'_{r,x}$ à exécuter dans le modèle PN avec l'algorithme 1. Considérons une telle séquence d'inspection élémentaire $\sigma_{r,x}$ sous sa forme générale

$$\sigma_{r,x} = (p'(1), M'(1))\,(p'(2), M'(2)) \cdots (p'(k), M'(k'))(p_{EP}, \varnothing) \quad (26)$$

Nous présentons ci-dessous le pseudo-code de l'algorithme 1 qui permet de calculer la séquence de tir $\sigma'_{r,x}$. Pour chaque site $p'(i)$, l'algorithme 1 recherche le site le plus proche $p'(j)$, $j = i+1,\ldots,k'$ dans la séquence $\sigma_{r,x}$ avec une transition dans le postset de $p'(i)$ ainsi que dans le preset de $p'(j)$. Si aucune place $p'(j)$ existe, l'algorithme 1 trouve nécessairement la transition unique t'_i dans le postset de $p'(i)$ ainsi que dans le preset de p_{EP}.

Algorithme 1: Firing sequence design
(Entrées : $\sigma_{r,x}$, PN; Sorties : $\sigma'_{r,x}$)
1. $\sigma'_{r,x} = \varnothing$
2. Pour chaque i allant de 1 jusqu'à k'
3. Extraire $p(i)$ de $\sigma_{r,x}$
4. $j = i+1$,
5. $t = p^\circ(i) \cap {}^\circ p(j)$
6. Tant que ($t = \varnothing$) ET ($j \leq k'$)
7. $j = j + 1$,
8. $t = p^\circ(i) \cap {}^\circ p(j)$
9. Fin tant que
10. $\sigma'_{r,x} = \sigma'_{r,x}\ t$
11. Fin pour

Exemple 4.9 : les séquences d'inspection élémentaires de l'exemple 4.8 sont transformées en des séquences de tir avec l'algorithme 1 comme suit :

$\sigma'_{r1,x10}=t_1\ t_{15}\ t_3\ t_9\ t_{21}\ t'_2\ t'_6$,

$\sigma'_{r2,x10}=t_1\ t_3\ t'_9$,

$\sigma'_{r3,x10}=t_{15}\ t'_9\ t'_1$.

4.5.2.3. Calcul du marquage initial minimal

Pour effectuer la séquence σ , nous nous basons sur l'approche de calcul des coûts d'investissements citée dans la section 4.4.2.1.3. En effet, chaque séquence d'inspection élémentaire s'exécute avec le nombre minimal de robots mobiles calculé avec (13) et le coût $C_{invest}(\sigma,x)$ de la séquence σ par rapport à la configuration x est donné par (15).

Exemple 4.10 : le nombre minimal $m_{r,x10}$ de robots de type $r \in R_{x10}$ nécessaire pour franchir chaque séquence de tir $\sigma'_{r,x}$ de l'exemple 4.9 est indiqué ci-dessous :

$m_{r1,x10}(\sigma) = M_I(\sigma'_{r1,x10}).(\mathbf{1})_9 = (1.\ p_7 + 1.\ p_8).(\mathbf{1})_9 = 2$

$m_{r2,x10}(\sigma) = M_I(\sigma'_{r2,x10}).(\mathbf{1})_9 = (1.\ p_7).(\mathbf{1})_9 = 1$

$m_{r3,x10}(\sigma) = M_I(\sigma'_{r3,x10}).(\mathbf{1})_9 = (1.\ p_4 + 1.\ p_9\ ).(\mathbf{1})_9 = 2$

Deux robots de type r_1 ayant comme positions initiales les sites p_7 et p_8, sont nécessaires pour effectuer les mesures *A* et *B*. Un seul robot de type r_2 avec la position initiale p_7 est nécessaire pour effectuer la mesure *C*. Enfin, deux robots de type r_3 avec les positions initiales p_4 et p_9, sont nécessaires pour effectuer la mesure *D*. Le coût résultant $c(\sigma,x_{10})$ pour la configuration x_{10} dépend du nombre minimal de robots mobiles calculé précédemment et du coût de chaque type de robot comme illustré ci-dessous

$$c(\sigma, x_{10}) = \sum_{R_{x10}}\left(m_{r,x10}(\sigma).c(r)\right)$$
$$= m_{r1,x10}(\sigma).c(r_1) + m_{r2,x10}(\sigma).c(r_2) + m_{r3,x10}(\sigma).c(r_3)$$
$$= 650.$$

4.5.2.4. Détermination de la meilleure configuration

Dans cette section, la détermination de la meilleure configuration x^*, ainsi le nombre de robots de chaque type dans cette configuration s'effectue selon l'approche du coût global et configuration optimale citée dans la section 4.4.2.2.2.

Exemple 4.11 : comme présenté dans le tableau 4.6, le coût des tâches d'inspection est calculé pour chaque configuration *x*.

Tableau 4. 6 : Calcul du coût de chaque configuration

Nombre maximal de capteurs/robot	Configurations *x*	Coût $c(\sigma, x)$	Nombre de robots / *x*
Max 1 capteur/robot	x_1 = {[*A*], [*B*], [*C*], [*D*]}	850	7
Max 2 capteurs / robot	x_2 = {[*A C*], [*B D*]}	560	4
	x_3 = {[*A C*], [*B*], [*D*]}	760	6
	x_4 = {[*A D*], [*B C*]}	560	4
	x_5 = {[*A*], [*B C*], [*D*]}	760	6
	x_6 = {[*A*], [*B*], [*C D*]}	760	6
	x_7 = {[*A D*], [*B*], [*C*]}	650	5
	x_8 = {[*A*], [*B D*], [*C*]}	650	5
	x_9 = {[*A B*], [*C D*]}	560	4
	x_{10} = {[*A B*], [*C*], [*D*]}	650	5
Max 3 capteurs / robot	x_{11} = {[*A C D*], [*B*]}	560	4
	x_{12} = {[*A*], [*B C D*]}	560	4
	x_{13} = {[*A B C*], [*D*]}	560	4
	x_{14} = {[*A B D*], [*C*]}	450	3
Max4 capteurs /robot	x_{15} = {*A B C D*}	360	2

4.5.2.5. Applications à la surveillance du port du Havre

On considère une partie de la zone industrielle et portuaire de la ville du Havre (Fig. 4. 9) comme une étude de cas. Cette zone est utilisée par de nombreuses entreprises chimiques et pétrochimiques dont les activités présentent un niveau de risque élevé. L'utilisation d'une technologie de surveillance est indispensable dans une telle zone pour limiter le risque. Notre objectif est de proposer un système automatique avec des véhicules sans pilote pour effectuer les tâches de surveillance. La zone décrite dans l'étude de cas comprend trois bâtiments industriels (Fig. 4. 9) et leur voisinage à surveiller. Elle a été divisée en 150 sites à surveiller. La séquence σ comportant 34 sites à surveiller successivement (également représentée par des carrés jaunes en pointillés sur la Fig. 4. 9) :

$\sigma = (p_{121}, \{a, b\})(p_{111}, \{a\})(p_{112}, \{a, b, c\})(p_{113}, \{c, d\})(p_{103}, \{b, c\})(p_{102}, \{a, b\})(p_{92}, \{a, b, c\})(p_{93}, \{a, b, c\})(p_{83}, \{b, c\})(p_{82}, \{a, b\})(p_{72}, \{a, b, c\})(p_{73}, \{c\})(p_{74}, \{a, c\})(p_{64}, \{a, c\})(p_{65}, \{a, b, c\})(p_{55}, \{b, c, d\})(p_{45}, \{b, c , d\})(p_{35}, \{b, c, d\})(p_{36}, \{b, c\})(p_{37}, \{b, c, d\})(p_{38}, \{b, c, d\})(p_{48}, \{b, c, d\})(p_{58}, \{b, c, d\})(p_{104}, \{a, b, c\})\ (p_{105}, \{a, b, c\})(p_{106}, \{a, b, c, d\})(p_{107}, \{c, d\})(p_{97}, \{c, d\})(p_{98}, \{d\})(p_{88}, \{c, d\})(p_{78}, \{c, d\})(p_{68}, \{c, d\})(p_{67}, \{b, c\})(p_{66}, \{b, c\}).$

Plusieurs batiments sont présents sur la zone. Le premier bâtiment est surveillé en effectuant des mesures sur les sites : $\{p_{121}, p_{111}, p_{112}, p_{113}, p_{103}, p_{102}, p_{92}, p_{82}, p_{72}\}$, le deuxième bâtiment est surveillé en effectuant des mesures sur les sites : $\{p_{104}, p_{105}, p_{106}, p_{107}, p_{97}, p_{98}, p_{88}, p_{78}, p_{68}, p_{67}, p_{66}, p_{65}, p_{64}, p_{74}, p_{73}, p_{83}, p_{93}\}$ et le troisième bâtiment est surveillé en effectuant des mesures sur les sites : $\{p_{65}, p_{55}, p_{45}, p_{35}, p_{36}, p_{37}, p_{38}, p_{48}, p_{58}, p_{68}, p_{67}, p_{66}\}$;

Quatre mesures $S = \{a, b, c, d\}$ doivent être effectuées dans les différents sites de la séquence σ. Les mesures correspondent aux zones colorées de la Fig. 4.9 : la couleur rouge signifie la mesure *a*, la couleur bleue signifie la mesure *b*, la couleur grise signifie la mesure *c* et la couleur verte signifie la mesure *d*. Chaque mesure est supposée être effectuée par un capteur spécifique : la mesure *a* est supposée être effectuée par un capteur de proximité, *b* par un capteur de pression, *c* par un capteur de température et *d* par un capteur d'accéléromètres. Les tâches de la patrouille sont effectuées par un ensemble de robots mobiles de différents types qui satisfont les hypothèses précédentes. Les coûts des robots et des capteurs sont définis dans le tableau 4. 5. Pour cette raison, il est nécessaire de déterminer la meilleure configuration x^* de coût minimal. Les résultats sont présentés dans le tableau 4. 7 où les configurations sont classées en fonction du nombre maximal de capteurs pouvant être installés sur la plateforme mobile.

Figure 4. 9 : Un environnement avec 150 sites de la zone industrielle et portuaire de la ville du Havre

Tableau 4. 7 : Résultats des différentes configurations

Nombre maximal de capteurs/robot	Configurations x	Coût $c(\sigma, x)$	Nombre de robots / x
Max 1 capteur/robot	x_1 = {[*A*], [*B*], [*C*], [*D*]}	2140	18
Max 2 capteurs / robot	x_2 = {[*A C*], [*B D*]}	990	7
	x_3 = {[*A C*], [*B*], [*D*]}	1610	13
	x_4 = {[*A D*], [*B C*]}	1440	10
	x_5 = {[*A*], [*B C*], [*D*]}	1520	12
	x_6 = {[*A*], [*B*], [*C D*]}	1590	13
	x_7 = {[*A D*], [*B*], [*C*]}	2060	16
	x_8 = {[*A*], [*B D*], [*C*]}	1520	12
	x_9 = {[*A B*], [*C D*]}	1100	8
	x_{10} = {[*A B*], [*C*], [*D*]}	1650	13
Max 3 capteurs / robot	x_{11} = {[*A C D*], [*B*]}	1000	8
	x_{12} = {[*A*], [*B C D*]}	960	7
	x_{13} = {[*A B C*], [*D*]}	980	7
	x_{14} = {[*A B D*], [*C*]}	950	7
Max4 capteurs /robot	x_{15} = {[*A B C D*]}	360	2

Selon le tableau 4. 7, la meilleure configuration avec le coût minimal pour effectuer les tâches de la patrouille est $x^* = x_{15} = \{[A\ B\ C\ D]\}$ avec c (σ, x_{15}) = 360. Cette configuration correspond à 2 robots (il y a un type unique de robots dans cette configuration).
Si nous supposons que le nombre maximal de capteurs installés sur chaque plateforme mobile est limité à 3, la meilleure configuration devient $x^* = x_{14} = \{[A\ B\ D], [C]\}$ avec c (σ, x_{14}) = 950. Cette configuration correspond à 3 robots de type r_1 et 4 robots de type r_2. Les positions initiales des robots de type r_1 correspondent aux sites p_{121}, p_{74} et p_{104}, tandis que les positions initiales des robots de type r_2 correspondent aux sites p_{112}, p_{92}, p_{88} et p_{104}. La sous-séquence pour les robots de type r_1 est définie par :

$\sigma_{r1,x14} = (p_{121}, \{a, b\})(p_{111}, \{a\})(p_{112}, \{a, b\})(p_{113}, \{d\})(p_{103}, \{b\})(p_{102}, \{a, b\})(p_{92}, \{a, b\})(p_{93}, \{a, b\})(p_{83}, \{b\})(p_{82}, \{a, b\})(p_{72}, \{a, b\})(p_{74}, \{a\})(p_{64}, \{a\})(p_{65}, \{a, b\})(p_{55}, \{b, d\})(p_{45}, \{b, d\})(p_{35}, \{b, d\})(p_{36}, \{b\})(p_{37}, \{b, d\})(p_{38}, \{b, d\})(p_{48}, \{b, d\})(p_{58}, \{b, d\})(p_{104}, \{a, b\})\ (p_{105}, \{a, b\})(p_{106}, \{a, b, d\})(p_{107}, \{d\})(p_{97}, \{d\})(p_{98}, \{d\})(p_{88}, \{d\})(p_{78}, \{d\})(p_{68}, \{d\})(p_{67}, \{b\})(p_{66}, \{b\}).$

La sous-séquence pour les robots de type r_2 est définie par :

$\sigma_{r2,x14} = (p_{112}, \{c\})(p_{113}, \{c\})(p_{103}, \{c\})(p_{92}, \{c\})(p_{93}, \{c\})(p_{83}, \{c\})(p_{72}, \{c\})\ (p_{73}, \{c\})(p_{74}, \{c\})(p_{64}, \{c\})(p_{65}, \{c\})(p_{55}, \{c\})(p_{45}, \{c\})(p_{35}, \{c\})(p_{36}, \{c\})(p_{37}, \{c\})(p_{38}, \{c\})(p_{48}, \{c\})(p_{58}, \{c\})(p_{104}, \{c\})\ (p_{105}, \{c\})(p_{106}, \{c\})(p_{107}, \{c\})(p_{97}, \{c\})(p_{88}, \{c\})(p_{78}, \{c\})(p_{68}, \{c\})(p_{67}, \{c\})(p_{66}, \{c\}).$

De même, si nous supposons que le nombre maximal de capteurs installés sur chaque plateforme mobile est limité à 2, la meilleure configuration devient à nouveau $x^* = x_2 = \{[A\ C], [B\ D]\}$ avec $c\ (\sigma, x_2) = 990$.

4.6. Conclusion

Afin d'assurer la surveillance d'une zone donnée, une approche a été proposée pour configurer et optimiser une patrouille de robots mobiles associés à divers capteurs selon une séquence de tâches de surveillance. L'environnement à surveiller ainsi que les trajectoires des robots ont été modélisés par des réseaux de Petri en définissant un coût élémentaire pour chaque composant du système. Une optimisation globale des tâches de surveillance est obtenue en couplant les problèmes de configuration et de planification des trajectoires. En particulier, la configuration optimale de l'ensemble des agents mobiles et la trajectoire de chaque agent sont déterminées en respectant l'investissement atomique et les coûts d'opération de chaque équipement.

Dans le chapitre suivant, nous allons procéder à l'optimisation des missions de surveillance pour résoudre le problème d'allocation des tâches. Ensuite, la recherche en faisceau sera proposée afin de résoudre notre problème en contexte centralisé et hors ligne.

Chapitre 5 : Optimisation par recherche en faisceau

5.1. Introduction

Ce chapitre propose une approche différente basée sur la recherche en faisceau pour résoudre le problème d'affectation des patrouilles de surveillance. Le problème est formalisé et modélisé en considérant deux étapes (Fig. 5. 1) :

1) Au niveau bas, nous calculons un graphe d'état pondéré et dirigé [Elwood, 1948 ; Booth, 1967] où un algorithme de Dijkstra est utilisé pour trouver le chemin le plus court entre les sites [Arjun et al., 2015]. Une matrice de coûts de déplacement rassemble les coûts de déplacement minimaux résultant de l'application de l'algorithme de Dijkstra sur chacun de paires de sites à surveiller.

2) Au niveau haut, l'algorithme de la recherche en faisceau filtrée hybride (HFBS) est utilisé pour calculer les sites successifs à visiter et les différentes mesures à prendre par chaque agent. A cette fin, l'ensemble des solutions est encodé et partiellement exploré sous forme d'arbre. L'explosion combinatoire est évitée grâce à une méthode qui élague les candidats non prometteurs [Mejia & Lefebvre, 2020 ; Mejia & Nino, 2017 ; Lefebvre & Basile, 2021]. L'élagage résulte d'un double mécanisme de filtrage : un filtre local qui élague les nœuds issus d'un même parent et un filtre global qui restreint le nombre total de nœuds à explorer à chaque niveau de l'arbre.

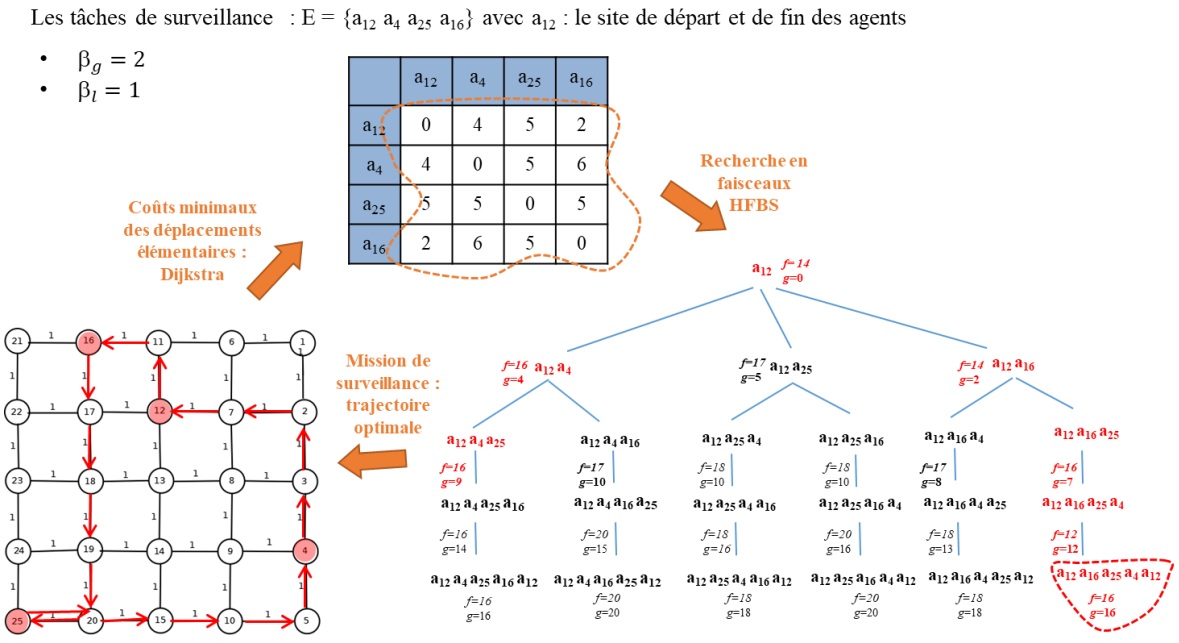


Figure 5. 1 : Graphe d'optimisation combinant les algorithmes HFBS et l'algorithme de Dijkstra.

Nous détaillons dans la deuxième section la modélisation de l'environnement par graphes orientés. La troisième section s'intéresse aux hypothèses et aux objectifs du problème. Sur la base de la stratégie de modélisation proposée dans le deuxième chapitre, la quatrième section présente une résolution par l'algorithme de recherche en faisceau filtrée hybride (HFBS). La cinquième section détaille et explique l'algorithme d'optimisation pour résoudre notre problème. La sixième section est consacrée à la discussion sur les aspects liés à la complexité. La septième section est une étude de cas pratique où les patrouilles sont optimisées pour une zone industrielle à Rouen, France. Enfin, la dernière section est réservée à la conclusion et aux perspectives.

5.2. Modélisation par graphe orienté

Les environnements industriels étant composés essentiellement de bâtiments, routes, couloirs, murs, et obstacles…, l'utilisation des graphes peut être utile pour les modéliser en assimilant les sites que les robots peuvent visiter à des sommets et en assimilant les espaces à parcourir par les robots entre les sommets à des arêtes.
Dans cette section, on présente quelques définitions des concepts utilisés avec les graphes orientés / non orientés, et notamment les composantes connexes et absorbantes.

5.2.1. Graphe orienté / non orienté

Dans la théorie des graphes, un graphe non orienté est un couple $G = (S, A)$ formé d'un ensemble fini S de nœuds ou sommets (Vertices en Anglais) et d'un ensemble fini E d'arêtes (Edges en Anglais) qui relient les sommets de l'ensemble S. Chaque arc peut être parcouru dans les deux sens.
Les caractéristiques d'un graphe non orienté sont :

- L'ordre du graphe qui est défini par le nombre de nœuds c'est-à-dire *card(S)*.
- La taille d'un graphe est le nombre de ses arêtes, c'est-à-dire *card(A)*.
- Deux sommets sont adjacents s'ils sont reliés entre eux par une arête.
- Le degré d'un nœud est défini comme le nombre d'arêtes issues de ce nœud.
- Un nœud est dit isolé s'il n'est adjacent à aucun autre nœud du graphe.
- Une boucle est définie comme une arête qui relie un nœud à lui-même.
- Un graphe est dit complet si deux nœuds quelconques distincts sont toujours adjacents. En d'autres mots, tous les nœuds sont reliés deux à deux par une arête.
- Un graphe est dit simple s'il ne contient pas de boucle, et s'il ne contient jamais plus d'une arête entre deux nœuds.

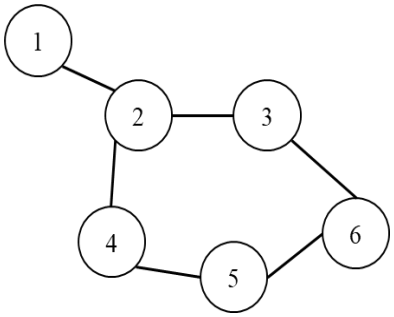


Figure 5. 2 : Un exemple de graphe simple non orienté

Un graphe orienté est un graphe dont les arêtes sont orientées, représentées par des flèches, et ne peuvent être parcourues que dans un seul sens.

Dans un graphe orienté :

- Une boucle est définie comme un arc dont l'origine et l'extrémité sont identiques (c'est-à-dire un arc qui relie un sommet à lui-même) ;
- Un chemin est une succession d'arêtes dont l'extrémité de chacune est l'origine de la suivante sauf la dernière ;
- Le nombre d'arêtes qui composent un chemin est nommé la longueur du chemin ;
- Un chemin dont l'origine et l'extrémité coïncident est appelé un chemin fermé ;
- Un circuit est défini comme un chemin fermé dont les arcs sont tous distincts ;
- Un graphe orienté qui ne contient pas de boucle est nommé élémentaire.

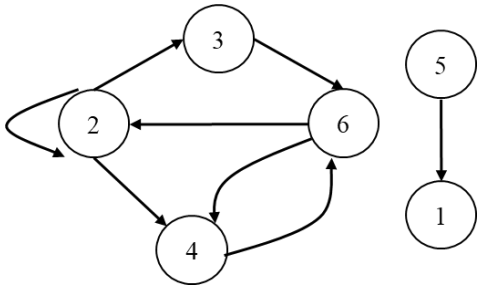


Figure 5. 3 : Un exemple de graphe orienté non élémentaire

Un graphe étiqueté est un graphe orienté ou non dont les arêtes entre les nœuds sont affectées d'étiquettes (symbole, lettre, mot, etc.…).

De plus, un graphe étiqueté peut être pondéré si toutes les étiquettes sont des nombres réels positifs ou nuls. Ces nombres sont les poids des arêtes entre les nœuds.

Le poids d'un chemin est la somme des poids des arcs qui le composent et un plus court chemin entre 2 nœuds est celui qui a le poids minimum.

5.2.2. Composante connexe et composante absorbante

Plusieurs endroits dans un environnement peuvent être des zones non accessibles ou des zones dont les robots ne peuvent pas s'échapper. Ces zones sont représentées par des composantes connexes, fortement connexes ou absorbantes du graphe.
Un graphe non orienté $G = (S, A)$ est connexe s'il existe une chaîne reliant n'importe quelle paire de nœuds distincts du graphe. Dans un espace topologique, un sous graphe connexe maximal de G est une composante connexe de ce graphe.

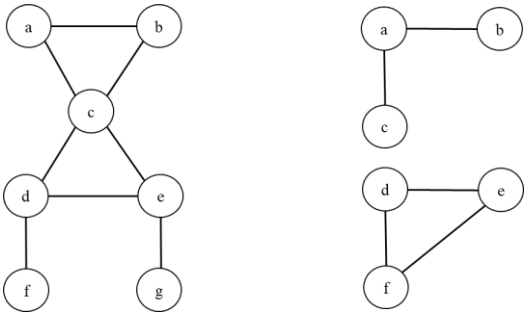


Figure 5. 4 : Exemples de graphe connexe (à gauche) et non connexe (à droite)

Dans la Fig. 5. 4, le graphe à gauche est dit connexe puisqu'il y a une chaîne qui relie n'importe quelle paire de nœuds distincts. Le graphe à droite n'est pas connexe puisqu'il existe deux composantes connexes disjointes, il n'y a pas de chemin entre les nœuds de premier sous graphe et le deuxième.

Une composante absorbante dans un graphe orienté est un sous ensemble de nœuds connectés et accessibles depuis le reste du graphe mais pour lesquels il n'existe aucun chemin allant vers le reste du graphe.

5.3. Hypothèses et objectifs

L'approche développée, dans ce chapitre, fait partie de la classe des problèmes ST-MR-TA, i.e., les agents peuvent effectuer plusieurs mesures dans une cellule donnée si cela est nécessaire et cela sera considéré comme une seule tâche (système ST) et chaque tâche peut nécessiter un ou plusieurs agents pour être entièrement réalisée (système MR). Les agents retourneront en a_1 lorsqu'ils auront terminé les tâches qui leur ont été assignées, ce qui signifie que nous considérons une planification future et que l'affectation est donc TA. Il convient toutefois de noter qu'avec une perspective légèrement différente, où une tâche est définie comme une mesure

spécifique à collecter à un endroit donné, le problème considéré peut également être examiné comme un problème MT-SR-TA.

La modélisation des différents éléments du problème à savoir l'environnement, les agents mobiles et les tâches de surveillance sont ceux présentés dans le chapitre 2 et la notion de configuration est la même que celle présentée dans le chapitre 4.

Dans ce contexte, deux problèmes sont présentés pour configurer les robots mobiles avec des capteurs appropriés et pour affecter à chaque robot un ensemble de mesures à collecter. Les robots doivent effectuer toutes les mesures à un coût minimal. Contrairement aux contributions proposées dans le chapitre 4, nous considérons ici un ensemble de sites et non pas une séquence de sites et nous levons l'hypothèse qui supposait une partition des capteurs sur les différents types de robots.

Prenons un exemple illustratif présenté dans la Fig. 5. 7 pour illustrer les problèmes considérés. Un environnement industriel est représenté avec certains bâtiments où les robots ne peuvent pas circuler (zones grises). Quatre sites particuliers sont spécifiés par leurs positions géographiques a_i, i = 2,3,4,5 et par les mesures à prendre en a_i : a, b, c.

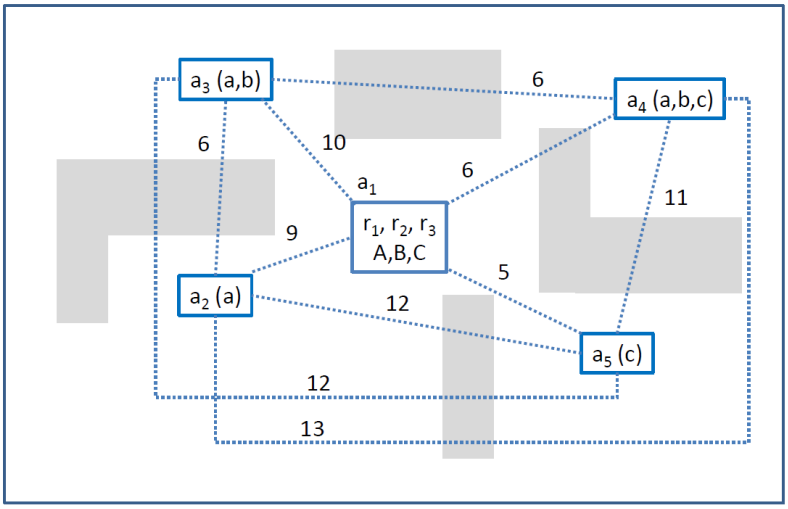


Figure 5. 5 : Un exemple illustratif d'une mission de surveillance

Un site particulier, à savoir a_1, est la station où les robots r_1, r_2 et r_3 commencent la patrouille et doivent revenir après avoir effectué leurs mesures. Avant d'affecter les mesures aux robots, il faut équiper chaque robot d'un ou de plusieurs capteurs A, B, C qui sont utilisés pour collecter les mesures de type a, b, c. Nous supposons connaître les coûts nécessaires pour se déplacer entre n'importe quelle paire de sites et, par souci de simplicité, nous supposons ici que ces coûts sont symétriques : par exemple, le coût pour se déplacer de a_1 à a_3 ou de a_3 à a_1 est 10.

L'approche proposée vise d'abord à aider l'utilisateur à configurer la patrouille, c'est-à-dire à décider combien de robots sont nécessaires et quels sont les capteurs portés par chaque robot, et ensuite à affecter à chaque robot une séquence de mesures de telle façon que toutes les mesures aux 4 sites seront collectées avec un coût minimal. La solution retournée doit prendre en compte certaines contraintes qui seront détaillées dans la suite. Ici, une solution possible (si chaque robot ne peut pas porter plus de 2 capteurs) est d'équiper r_1 avec les capteurs *A* et *B* et r_2 avec *B* et *C*. Ensuite, l'algorithme proposé calcule les séquences de tâches effectuées par chaque agent : r_1 visite d'abord a_4 et prend les mesures *a* et *b* puis il visite a_3 et prend à nouveau *a* et *b*, et enfin il visite a_2 et prend *a*. De même pour r_2 qui recueille successivement *c* en a_5 et en a_4. Les deux robots doivent retourner en a_1 après avoir effectué leurs mesures. On suppose qu'un seul robot ne peut pas effectuer toute la patrouille et que les robots doivent collaborer pour prendre toutes les mesures en a_4.

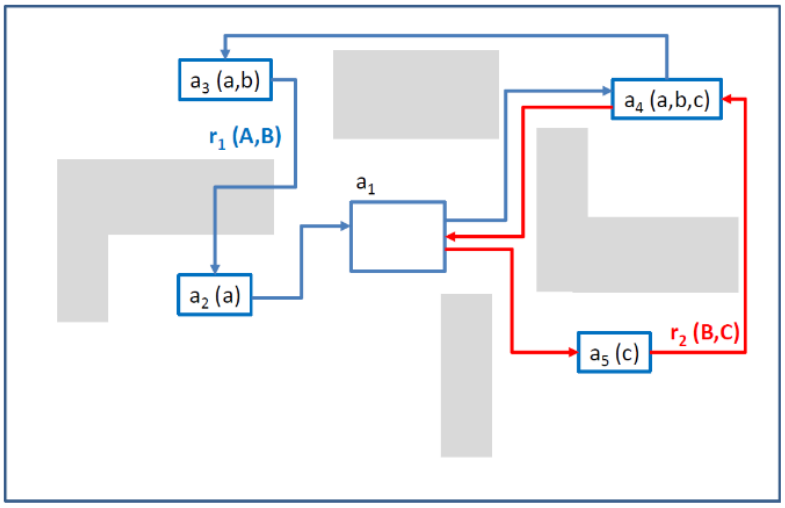


Figure 5. 6 : Une solution possible pour l'exemple illustratif

L'objectif de ce chapitre est de résoudre des problèmes similaires de plus grande dimension en utilisant une approche flexible. Les principales contributions sont d'abord, de spécifier la patrouille, en particulier, le nombre d'agents et les capteurs portés par chaque agent, et ensuite, de diriger les agents afin de minimiser le coût de la patrouille par rapport aux objectifs de surveillance. Ces deux aspects sont combinés dans un problème d'optimisation couplé. Pour être applicable à des situations pratiques, ce problème d'optimisation prend en compte diverses contraintes telles que l'autonomie énergétique de chaque agent mais aussi d'éventuelles contraintes de précédence entre les tâches ou d'autres contraintes fonctionnelles.

Une approche de recherche en faisceau est utilisée pour résoudre de tels problèmes. À cette fin, l'ensemble des solutions est codé et partiellement exploré sous forme d'arbre. L'explosion

combinatoire est évitée grâce à une méthode qui élague les candidats non prometteurs avec la variante hybride filtrée de BS (HFBS) détaillée dans [Lefebvre & Basile, 2021 ; Mejía & Lefebvre, 2020 ; Mejía & Niño, 2017]. L'élagage résulte d'un double mécanisme de filtrage : un filtre local qui élague les nœuds issus d'un même nœud parent et un filtre global qui restreint le nombre total de nœuds à explorer à chaque niveau de l'arbre.

5.4. Méthode d'optimisation de patrouilles

Une étape préliminaire consiste à calculer les trajectoires de moindre coût entre n'importe quelles paires de sites (a_i, a_k) dans E. Une matrice des coûts de déplacement $C(E, r)$, $r = 1, ..., Nr$, de dimension $K \times K$ est définie pour chaque type r d'agents. Cette matrice $C(E, r)$ fournit le coût de déplacement $c(a_i, a_k, r)$ pour chaque paire de sites $a_i, a_k \in E$. Le coût $c(a_i, a_k, r)$ est obtenu à partir des coûts élémentaires $c(a_j, a_h)$, $j, h = 1, ..., N$ en déterminant d'abord le chemin de moindre coût pour aller de a_i à a_k avec l'algorithme de Dijkstra [Gam et al., 2021a] ; ensuite en pondérant le coût $c(a_i, a_k)$ de ce chemin par le coût d'exploitation $c(r)$ des robots de type r : $c(a_i, a_k, r) = c(a_i, a_k) \times c(r)$.

5.4.1. Résolution par la méthode HFBS

Dans l'optimisation de la patrouille de surveillance, chaque nœud S représentera une solution candidate pour le problème considéré. Cette solution candidate est complète si elle satisfait TC, c'est-à-dire que

(i) Toutes les tâches ont été effectuées ;

(ii) Tous les agents sont retournés au site a_1.

Dans ce cas, on écrit $End(S) = 1$; sinon S est non complète et on écrit $End(S) = 0$.

Une solution candidate S est définie par les séquences de sites $Agent_x(S,r) = a_{k1} \ldots a_{k(r)}$, $r = 1,\ldots,N_r$ qui ont déjà été visités dans la configuration x. $Agent_x(S,r)$ est composé d'une ou plusieurs trajectoires. Une trajectoire commence et se termine par le site a_1. On suppose que plusieurs trajectoires peuvent être nécessaires pour effectuer la patrouille $Agent_x(S,r)$ en raison des contraintes énergétiques : lorsqu'un agent n'a pas assez d'énergie pour collecter toutes les mesures dans les différents sites, il retourne au site a_1 (pour recharger sa batterie) et un autre agent du même type est utilisé pour continuer la patrouille. Il convient de noter que, bien que les trajectoires soient définies comme des sous-séquences successives dans $Agent_x(S,r)$, tous les agents de type r peuvent commencer leurs trajectoires simultanément.

Tasks(S) définit les tâches de surveillance restantes à *S*. *Tasks(S)* est initialisé avec l'ensemble des tâches de surveillance *E*. Lorsqu'une mesure *m* est effectuée sur un site donné a_i, alors *m* est retiré de *M(i)* dans *Tasks(S)* et une fois que *M(i)* devient vide, alors le site a_i est également retiré de *Tasks(S)*.

A partir de chaque solution candidate *S*, il est également possible de calculer la liste de sites *Succ(S,r)* qui contient les successeurs possibles à visiter par un agent de type *r*. *Succ(S,r)* est calculé à partir de $Agent_x(S,r)$, *Sensor(r)* et *Tasks(S)* de telle sorte qu'un site a_i est ajouté dans *Succ(S,r)* si un agent de type *r* possède les capteurs appropriés pour effectuer une ou plusieurs mesures au site a_i, peut atteindre a_i et retourner à a_1 tout en satisfaisant les contraintes de précédence et d'énergie [Gam et al., 2022a].

Exemple 5.1 : Pour illustrer les spécifications citées précédemment, on considère à nouveau l'exemple décrit dans la section 5.3, L'ensemble des tâches est défini par $E = \{(a_1, \emptyset), (a_2,\{a\}), (a_3,\{a,b\}),(a_4,\{a,b,c\}),(a_5,\{c\})\}$ et il existe une configuration unique avec deux types d'agents r_1 et r_2 avec $Sensor(r_1) = \{A,B\}$ et $Sensor(r_2) = \{B,C\}$. Un exemple de solutions candidates non complètes est le nœud 2 dans la Fig. 5. 9 avec $Agent\ (2, r_1) = [a_1\ a_2]$ et $Agent\ (2, r_2) = [a_1]$. Les tâches de surveillance restantes du nœud 2 sont spécifiées par $Tasks(2) = \{(a_1, \emptyset), (a_3,\{a,b\}), (a_4,\{a,b,c\}), (a_5,\{c\})\}$ tandis que la liste des successeurs est définie par $Succ(2, r_1) = \{a_3,a_4,a_1\}$ et $Succ(2, r_2) = \{a_3,a_4,a_5\}$. Un autre exemple de solution complète pour les candidats S^* peut être visualisé dans la Fig. 5. 11 avec $Agent\ (S^*, r_1) = [a_1\ a_4\ a_3\ a_2\ a_1]$ et $Agent\ (S^*, r_2) = [a_1\ a_5\ a_4\ a_1]$. De toute évidence, $Tasks(S^*) = Succ(S^*, r_1) = Succ(S^*, r_2) = \emptyset$.

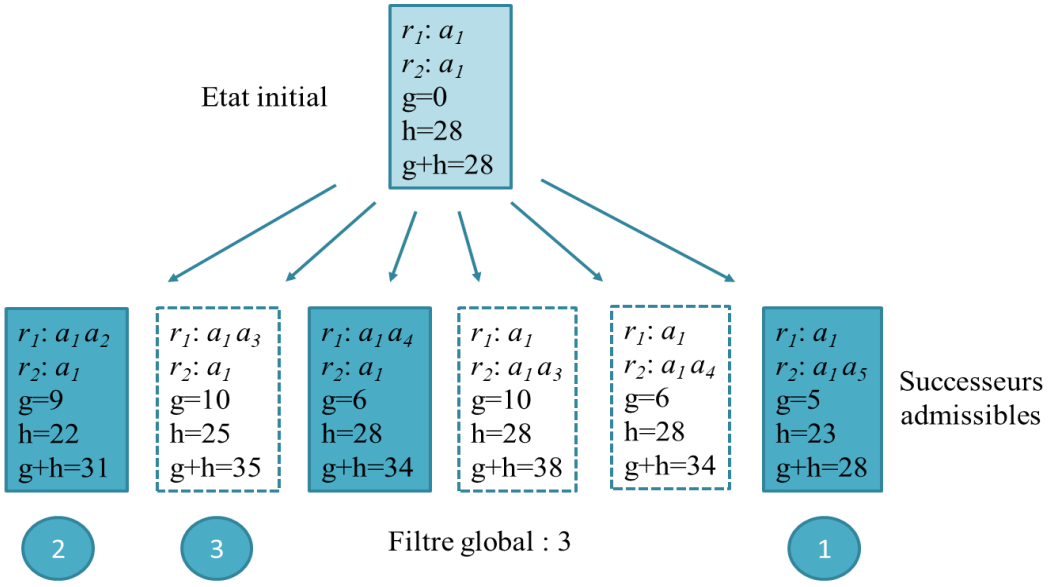


Figure 5. 7 : Méthode d'optimisation : première étape

La recherche A* nécessite un temps et une mémoire exponentiels [Russell & Norvig, 1995]. Pour cette raison, l'élagage de l'arbre de recherche est une pratique courante mais au détriment de l'optimalité. De nombreuses améliorations ont été développées dans ce but, et en particulier des approches BS ont été développées.

Exemple 5.2 : Les grands principes de HFBS sont expliqués à l'aide de l'exemple décrit dans la section 5.3. L'algorithme commence avec l'état initial S_0 où un agent de chaque type reste dans le site a_1 avec $Succ(S_0, r_1) = \{a_2,a_3,a_4\}$ et $Succ(S_0,r_2) = \{a_3,a_4, a_5\}$ (Fig. 5. 9).
Par conséquent, six nouvelles solutions candidates sont créées et triées par rapport à la fonction de coût $f(S) = g(S) + h(S)$ (les détails de g et h seront discutés plus tard).
On suppose qu'un filtre global avec le paramètre $\beta_g = 3$ soit utilisé : seuls les trois meilleurs candidats, à savoir 1, 2 et 3 sur la figure 5. 9, sont considérés pour les étapes suivantes. Le meilleur candidat (1) avec la valeur de coût la plus basse $f = 28$ est sélectionné et, à nouveau, ses successeurs possibles sont calculés (Fig. 5. 10).
Comme un filtre local avec le paramètre $\beta_l = 2$ est utilisé, seuls les deux meilleurs successeurs sont conservés et ajoutés à la liste des candidats qui augmente jusqu'à [1.1, 2, 1.2, 3] sur la figure 5. 10. Le filtre global est appliqué une autre fois pour restreindre la liste à [1.1, 2, 1.2]. L'exploration partielle de l'arbre se poursuit et s'arrête lorsqu'une solution candidate est trouvée et satisfait *TC* (Fig. 5. 11).

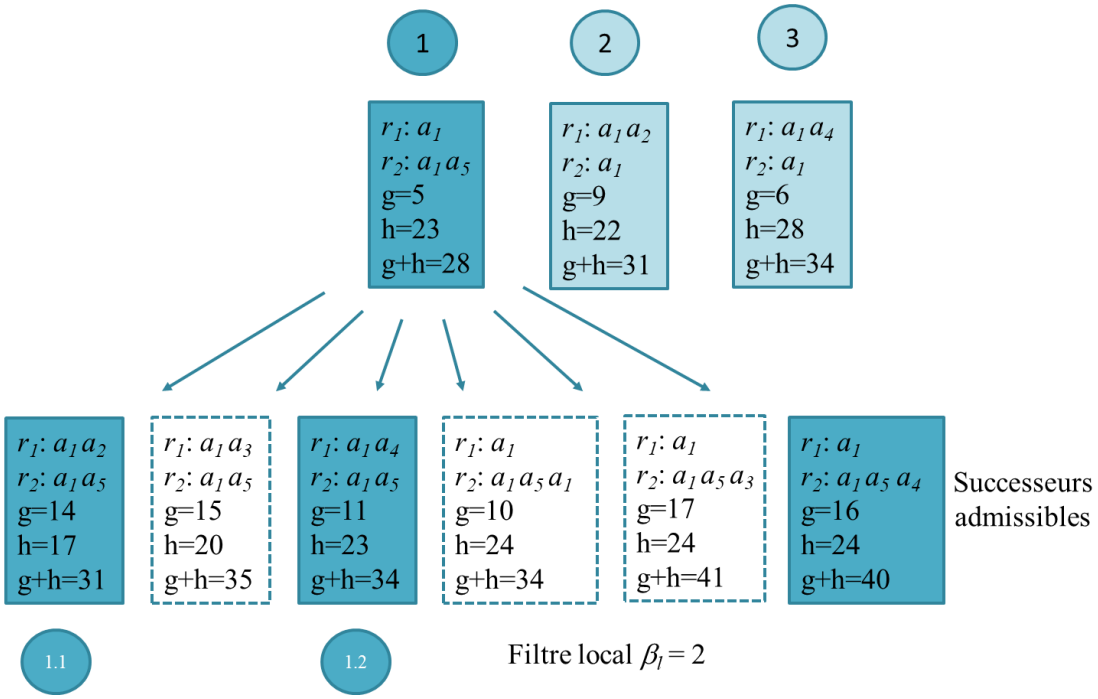


Figure 5. 8 : Méthode d'optimisation : deuxième étape

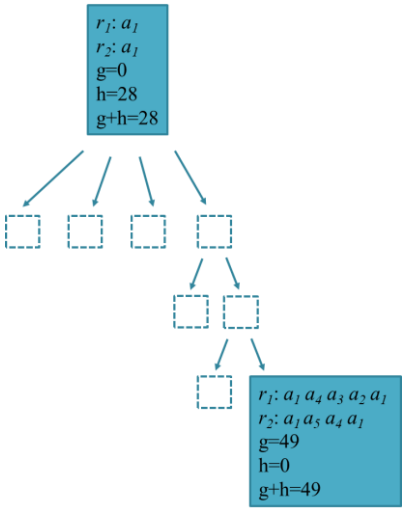


Figure 5. 9 : Méthode d'optimisation : vue d'ensemble

5.4.2. Fonction coût

La fonction $f(S) = g(S) + h(S)$ représente le coût global du nœud S. $g(S)$ est le coût réel des tâches déjà effectuées au nœud S et $h(S)$ est une estimation du coût restant pour terminer la mission. $g(S)$ est défini par l'équation (2).

$$g(S) = \sum_{r \in R_x} \left(\sum_{i=1...h-1} c(a_{k(i)}, a_{k(i+1)}, r) \right) \tag{2}$$

où $Agent_x(S,r) = a_{k(1)} \ldots a_{k(h)}$ définit la séquence des sites déjà visités par un agent de type r.

$h(S)$ est une estimation du coût restant pour atteindre TC. Cette estimation est calculée en ignorant la nature des différentes mesures à effectuer dans chaque site et en ignorant également les capteurs utilisés par chaque type d'agent. Pour calculer $h(S)$, on définit d'abord $L(S)$ comme l'ensemble des sites qui doivent encore être visités ($L(S)$ résulte directement de $Task(S)$ en ignorant les mesures), $a(S)$ comme la liste des positions actuelles des agents ($a(S)$ résulte du dernier élément dans les listes $Agent_x(S,r)$, $r = 1,\ldots,N_r$) et $L_1(S) = L(S) \cup \{a_1\}$, $L_a(S) = L(S) \cup a(S)$. L'estimation $h(S)$ est calculée comme suit

$$h(S) = max\left\{ \underbrace{\left(c^*(a(S), L_1(S)) + \sum_{a \in L(S)} c^*(a, L_1(S)) \right)}_{h_1(S)}, \underbrace{\sum_{a' \in L_1(S)} c^*(L_a(S), a')}_{h_2(S)} \right\} \tag{3}$$

où $c^*(a(S), L_1(S))\}$ est le coût minimal non-nul des positions actuelles des agents aux sites de $L_1(S)$, $c^*(a, L_1(S))$ est le coût minimal non-nul de la position actuelle a de l'un des agents aux

sites de $L_1(S)$, et $c^*(L_a(S), a')$ est le coût minimal non nul entre les sites de l'ensemble $L_a(S)$ et le site particulier a' [Gam et al., 2022a].

Exemple 5.3 : L'équation 3 est illustrée dans la Fig. 5. 12 pour l'exemple décrit dans la section 5.3. On considère un nœud particulier S avec $Agent\ (S, r_1) = [a_1\ a_2]$ et $Agent\ (S, r_2) = [a_1\ a_5]$. Nous avons $L(S) = \{a_3, a_4\}$ parce que toutes les mesures dans les sites a_2 et a_5 ont déjà été effectuées en S. De plus, $a(S) = \{a_2, a_5\}$ parce qu'un agent de type r_1 est actuellement en position a_2 et un agent de type r_2 reste en a_5. Enfin, $L_1(S) = \{a_3, a_4, a_1\}$ et $L_a(S) = \{a_2, a_5, a_3, a_4\}$ et les coûts entre tous les sites dans $L_1(S)$ et $L_a(S)$ sont reportés dans la Fig. 5.12. Fondamentalement, la première partie $h_1(S)$ de l'équation 3 correspond au coût minimal nécessaire pour quitter les positions actuelles des agents plus le coût minimal nécessaire pour quitter chaque site qui nécessite encore une visite. Cela correspond au calcul et $h_1(S) = 17$ (Fig. 5. 12). La deuxième partie $h_2(S)$ de l'équation 3 correspond au coût minimal nécessaire pour atteindre chaque site qui doit encore être visité plus le coût minimal nécessaire pour revenir à a_1. Cela correspond au calcul $h_2(S) = 17$ (Fig. 5. 12). L'estimation $h(S)$ est obtenue par $h(S) = \max\{h_1(S), h_2(S)\}$.

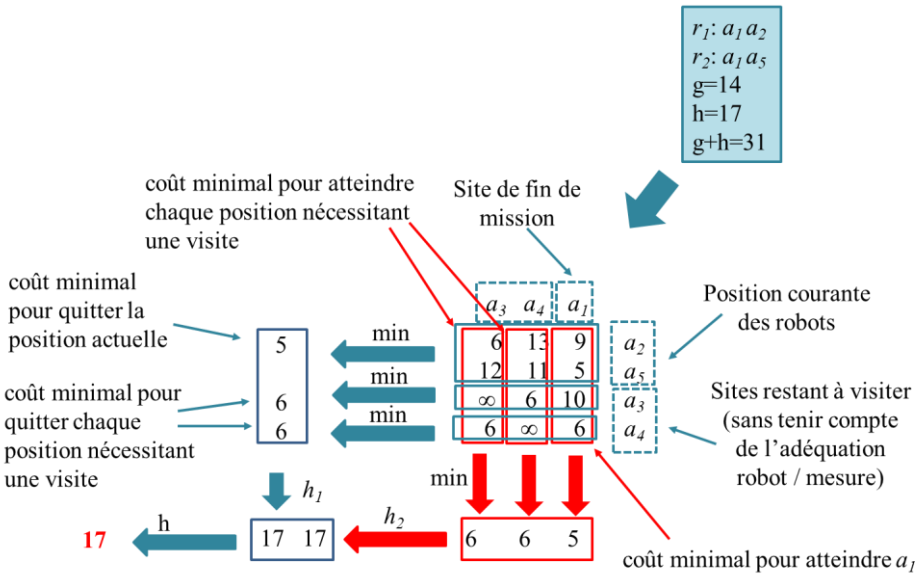


Figure 5. 10 : Détail de la fonction heuristique $h(S)$

Proposition 1 : La fonction heuristique $h(S)$ est inférieure au coût optimal $c^*(S, TC)$ nécessaire pour satisfaire la condition terminale TC à partir de l'état S.

Preuve : On observe d'abord que le nombre de déplacements élémentaires de site-à-site est au moins $|L(S) + 1|$, c'est-à-dire le nombre de sites restants à visiter plus le site terminal où les agents doivent retourner (le nombre de segments de déplacement de site-à-site est exactement

$|L(S) + 1|$ dans le cas où l'on considère un seul agent capable d'effectuer toutes les mesures). Par conséquent, $|L(S) + 1|$ est une borne inférieure du nombre de déplacements élémentaires de site à site lorsque tous les agents sont considérés.

On calcule le plus petit coût de déplacement $c^*(a(S), L_1(S))$ des sites actuellement occupés par les agents aux sites dans $L_1(S)$. Ensuite, pour chacun des sites restants $a \in L(S)$ à visiter, on cherche les plus proches voisins dans $L_1(S)$ accessibles depuis a et $c^*(a, L_1(S))$ est le coût de trajet minimal correspondant. En conséquence,

$$c^*(S,TC) \geq c^*(a(S),L_1(S)) + \sum_{a \in L(S)} c^*(a,L_1(S)) \quad (4)$$

$c^*(a, L_1(S))$ est composé du coût des déplacements élémentaires de site à site, chacun étant une borne inférieure du coût requis pour atteindre un site dans $L_1(S)$ à partir de a.

Ensuite, un raisonnement similaire s'applique à la recherche du voisin le plus proche dans $L_a(S)$ pour atteindre un site donné $a' \in L_1(S)$. Pour chaque site $a' \in L_1(S)$, $c^*(L_a(S), a')$ est le coût minimal de déplacement minimal correspondant. Par conséquent,

$$c^*(S,TC) \geq \sum_{a' \in L_1(S)} c^*(L_a(S),a') \quad (5)$$

Par ailleurs, $h(S)$ est calculé comme le coût minimal des trajectoires requis par les agents pour atteindre successivement tous les sites non visités. Ce coût ne tient pas compte du fait que certains sites doivent être visités par deux ou plusieurs agents de types différents. De plus, il ne considère pas les contraintes d'énergie ou de précédence. Par conséquent, il s'agit d'une borne inférieure du coût optimal $c^*(S, TC)$ [Gam et al., 2022a].

5.5. Résolution du problème

Cette section détaille les pseudo-codes de la méthode HFBS qui résout le problème d'optimisation de la patrouille. Nous proposons de résoudre 3 problèmes de difficulté croissante

1. **Problème 1 :** Optimiser à la fois le nombre d'agents d'un type donné r et la patrouille de chaque agent de ce type,
2. **Problème 2 :** Optimiser à la fois le nombre d'agents d'une configuration donnée x et la patrouille de chaque agent (de chaque type) de la flotte,
3. **Problème 3 :** Optimiser d'abord la configuration de la flotte ; ensuite le nombre d'agents de chaque type r pour la configuration optimale x^* et la patrouille de chaque agent (de chaque type pour la configuration optimale) de la flotte.

5.5.1. Optimisation pour un seul type d'agents

L'algorithme 1 implémente la méthode HFBS pour résoudre le problème 1. Dans la mesure où ce problème implique un seul type d'agents, la notation *r* sera omise pour plus de simplicité. On suppose que chaque agent possède les capteurs nécessaires pour collecter toutes les mesures dans chaque site de la patrouille. Par conséquent, le problème 1 se réduit à trouver le meilleur chemin dans les sites à visiter en faisant abstraction de la répartition des tâches de surveillance entre plusieurs types d'agents.

L'algorithme 1 utilise la définition de l'ensemble des tâches de surveillance E, les contraintes de précédence $\sigma(E)$, les contraintes d'énergie T_{max}, la matrice des coûts de déplacement C et les paramètres β_g et β_l comme entrées et renvoie l'indicateur *success* = 1 s'il trouve une solution, sinon il renvoie *success* = 0. Dans le cas où *success* = 1, il renvoie également la solution S^* et son coût f^* [Gam et al., 2022a].

Algorithme 1 : Recherche en faisceau filtrée hybride pour un seul type d'agents

Entrées : E, $\sigma(E)$, T_{max}, C, β_g, β_l; Sorties : *success*, σ^*, f^*

1. $Agent(S_0) \leftarrow a_{k1}$, $Tasks(S_0) \leftarrow E$, $Succ(S_0) \leftarrow E$, $End(S_0) \leftarrow 0$,
2. $g(S_0) \leftarrow 0$, $h(S_0) \leftarrow h(S_0,TC)$, $OPEN \leftarrow \{S_0\}$, $S^* \leftarrow \emptyset$, $f^* \leftarrow \infty$, $success \leftarrow 0$
3. Tant que ($OPEN \neq \emptyset$) et (*success* = 0) faire
4. Supprimer le premier candidat S de $OPEN$
5. Si $End(S) = 1$
6. $success \leftarrow 1$, $S^* \leftarrow \sigma$, $f^* \leftarrow g(S)$
7. sinon si $Succ(S) \neq \emptyset$
8. $TEMP \leftarrow \emptyset$
9. Pour chaque site a dans $Succ(S)$
10. calculer $S(a)$ où a est le prochain site à visiter par les agents;
11. $TEMP \leftarrow TEMP \cup \{S(a)\}$
12. Fin pour
13. Trier $TEMP$ par ordre croissant de $g+h$
14. Ajouter les β_l premiers candidats de $TEMP$ dans $OPEN$
15. sinon
16. Trier $OPEN$ par ordre croissant de $g+h$
17. Sauvegarder les β_g premiers candidats de $OPEN$ et supprimer les autres candidats
18. Fin si
19. Fin Tant que

Exemple 5.4 : on considère l'environnement décrit dans la Fig. 5. 13 comme exemple pour illustrer l'algorithme 1. Cet environnement est composé d'une grille de dimension 6 × 6 sites d'adresses $\{a_1, \ldots, a_{36}\}$. Chaque site est représenté par son adresse. Pour simplifier, les coûts

élémentaires entre les sites adjacents sont supposés égaux à 1 ou égaux à ∞ lorsqu'un obstacle existe entre les deux sites.

On suppose que les tâches de surveillance sont définies par :

E={(a_{13}, ∅), (a_{21}, {a, b}), (a_2, {b}), (a_{29}, {a, b, c}), (a_{15}, {a}), (a_{23}, {a}), (a_{11}, {a, b, c}), (a_{31}, {a, c}), (a_{25}, {c}), (a_5, {a, c}), (a_{34}, {b, c})}

Les agents commencent et terminent la patrouille au site a_{13}, et doivent visiter les sites {a_{21}, a_2, a_{29}, a_{15}, a_{23}, a_{11}, a_{31}, a_{25}, a_5, a_{34}} pour effectuer les mesures a, b ou c et finalement retourner au site a_{13}. Les agents évitent implicitement certaines zones qui correspondent aux composantes absorbantes connectées {a_{35}, a_{36}}, {a_7}, {a_{30}} et {a_1} à partir desquelles les agents ne pourraient pas sortir.
Dans cet exemple, on considère un seul type d'agents et on suppose que chaque agent porte 3 capteurs nécessaires pour effectuer les 3 mesures a, b et c. Pour plus de simplicité, le coût des agents est ignoré, c'est-à-dire le coût d'exploitation de chaque agent est $c(r)$ = 1.

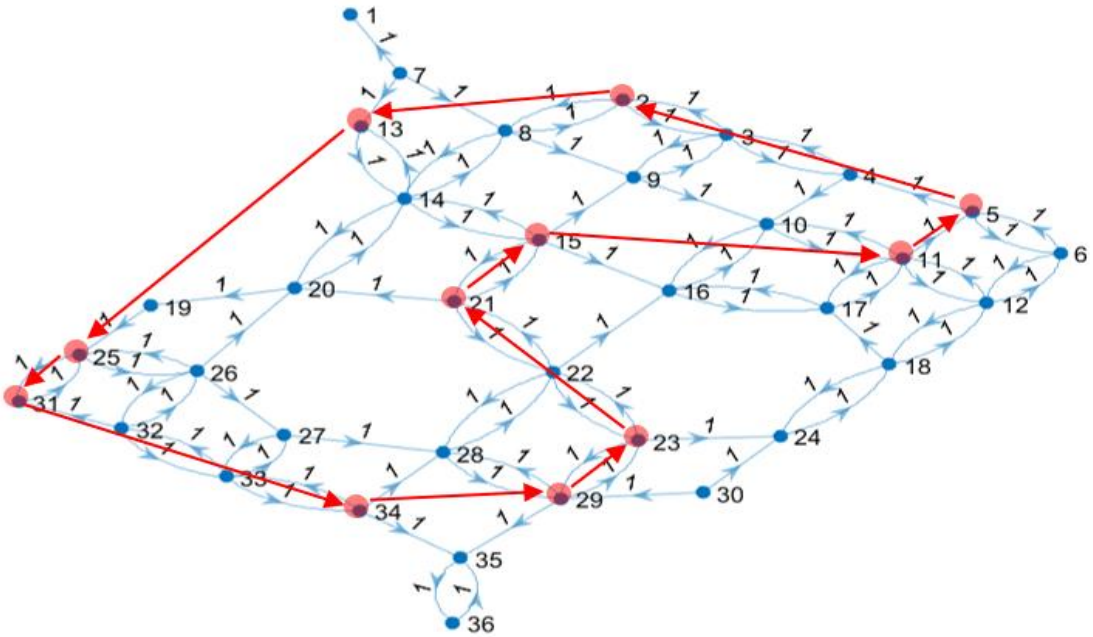


Figure 5. 11 : Solution optimale pour le problème 1, cas 1 dans l'exemple 5.4

Cas 1 : les contraintes de précédence et d'énergie ne sont pas prises en compte. Le tableau 5. 1 présente les coûts (mesurés en Unité de Temps (UTs)) des solutions trouvées par l'algorithme 1 et le nombre de nœuds explorés (c'est-à-dire la complexité numérique de la recherche) pour différentes valeurs des paramètres β_g et β_l. L'algorithme s'arrête dès qu'une solution est trouvée.

Tableau 5. 1 : Coût de la solution (UTs) et complexité pour le cas 1

$\beta_l \setminus \beta_g$	5	10	20	100
2	32 (402)	26* (652)	26* (912)	26* (1252)
5	30 (372)	30 (722)	26* (1102)	26* (1742)
10	-	30 (722)	26* (1102)	26* (3272)
20	-	-	26* (1102)	26* (3272)

" * " signifie que la solution est optimale et " - " que $\beta_l > \beta_g$.

On peut constater que le coût optimal $f^* = 26$ UTs est trouvé pour la plupart des valeurs des paramètres β_l et β_g. Ce coût correspond par exemple à la trajectoire *Agent (TC, r)* = $a_{13}\, a_{25}\, a_{31}\, a_{34}\, a_{29}\, a_{23}\, a_{21}\, a_{15}\, a_{11}\, a_5\, a_2\, a_{13}$ qui est reportée sur la Fig. 5. 13 et qui nécessite un seul agent. Mais d'autres trajectoires optimales comme *Agent' (TC, r)* = $a_{13}\, a_{31}\, a_{25}\, a_{34}\, a_{29}\, a_{23}\, a_{21}\, a_{15}\, a_{11}\, a_5\, a_2\, a_{13}$ existent également. On remarque que la complexité augmente en fonction des paramètres β_l et β_g.

Cas 2 : Les contraintes d'énergie ne sont pas prises en compte mais des contraintes de précédence partielle sont définies selon $\sigma(E) = a_{25} < a_{29} < a_{34} < a_{15}$. Les autres spécifications du problème restent inchangées. Le tableau 5. 2 présente les résultats.

Tableau 5. 2 : Coût de la solution (UTs) et complexité pour le cas 2

$\beta_l \setminus \beta_g$	5	10	20	100
2	44 (482)	36* (852)	36* (1672)	36* (5732)
5	44 (482)	44 (1072)	36* (1702)	42 (8832)
10	-	44 (1072)	36* (1672)	36* (8492)
20	-	-	36* (1672)	36* (8492)

Dans ce cas, le coût de la solution optimale augmente à $f^* = 36$ UTs à cause des contraintes de précédence partielle et une patrouille optimale est obtenue pour deux agents. La trajectoire du premier agent est défini par *Agent (TC, r_1)* = $a_{13}\, a_{25}\, a_{31}\, a_{34}\, a_{21}\, a_{15}\, a_{11}\, a_5\, a_2\, a_{13}$ alors que la trajectoire du second agent est défini par *Agent (TC, r_2)* = $a_{13}\, a_{23}\, a_{29}\, a_{13}$. Dans la mesure où les contraintes d'énergie ne sont pas considérées, les deux trajectoires peuvent être réalisées par le même agent ou par deux agents différents.

Cas 3 : Une contrainte d'énergie de 15 UTs correspondant à l'autonomie des agents est ajoutée aux contraintes de précédence partielle précédentes. Les autres spécifications du problème restent inchangées. Le tableau 5. 3 présente les résultats.

Tableau 5. 3 : Coût de la solution (UTs) et complexité pour le cas 3

$\beta_l \backslash \beta_g$	5	10	20	100
2	54 (376)	54 (660)	50 (1051)	50 (2559)
5	54 (385)	54 (614)	54 (1213)	48* (4003)
10	-	54 (614)	54 (1213)	48* (4201)
20	-	-	54 (1213)	48* (4201)

Dans ce cas, le coût global de la solution optimale augmente à $f^* = 48$ UT en raison des contraintes de la précédence partielle et des contraintes d'énergie. Une patrouille optimale est obtenue pour 4 agents : *Agent (TC,* r_1*)* $= a_{13}\, a_{11}\, a_5\, a_2\, a_{13}$, *Agent (TC,* r_2*)* $= a_{13}\, a_{34}\, a_{21}\, a_{15}\, a_{13}$, *Agent (TC,* r_3*)* $= a_{13}\, a_{25}\, a_{31}\, a_{13}$ et *Agent (TC,* r_4*)* $= a_{13}\, a_{23}\, a_{29}\, a_{13}$ sont les trajectoires des 4 agents. Pour conclure, une solution optimale a été trouvée pour le problème 1 quelle que soit la valeur des paramètres β_l et β_g. Augmenter la taille de l'environnement et compliquer la topologie avec des chemins non symétriques de poids différents rendra la recherche plus longue mais ne changera pas l'approche de résolution globale. On note que tester plusieurs valeurs pour les paramètres β_l et β_g est important pour trouver la meilleure solution. Dans ce qui suit, nous présentons un ensemble d'expérimentations numériques afin d'évaluer la complexité du calcul.

5.5.2. Optimisation pour plusieurs types d'agents dans une seule configuration

L'algorithme 2 implémente la méthode HFBS pour résoudre le problème 2. Par rapport à l'algorithme 1 et au problème 1, l'algorithme 2 manipule plusieurs types d'agents. Dans la mesure où ce problème implique une seule configuration, la notation x sera omise pour plus de simplicité. L'algorithme 2 utilise la définition des tâches de surveillance E, les contraintes de précédence $\sigma(E)$, la matrice des coûts de déplacement C, les contraintes d'énergie $T_{max}(r)$, l'ensemble des capteurs *Sensor(r)* disponibles pour les agents de type r, et les paramètres β_g et β_l et renvoie le flag *success* = 1 s'il trouve une solution, sinon il renvoie *success* = 0. Dans le cas où *success* = 1, il renvoie également la solution S^* et son coût f^* [Gam et al., 2022a].

Exemple 5.5 : on considère le même environnement et les mêmes tâches de surveillance que ceux décrits dans l'exemple 5.4 pour illustrer l'algorithme 2. Dans cet exemple, deux types d'agents r_1 et r_2 sont considérés : les agents de type r_1 portent les capteurs $\{A, B\}$ tandis que les agents de type r_2 portent les capteurs $\{A, C\}$. Ainsi, le principal défi, ici, est d'allouer les mesures A aux agents de type r_1 ou r_2.

Algorithme 2 : Recherche en faisceau filtrée hybride pour plusieurs types d'agents

Entrées : E, $\sigma(E)$, C, $T_{max}(r)$, $Sensor(r)$, β_g, β_l; Sorties : $success$, σ^*, f^*

1. $Agent(S_0,r) \leftarrow a_{k1}$, $Tasks(S_0,r) \leftarrow E$, $Succ(S_0,r) \leftarrow E$, $S^* = \emptyset$,
2. $End(S_0) \leftarrow 0$, $g(S_0) \leftarrow 0$, $h(S_0) \leftarrow h(S_0, TC)$, $OPEN \leftarrow \{S_0\}$, $f^* \leftarrow \infty$, $success \leftarrow 0$
3. Tant que ($OPEN \neq \emptyset$) ET ($success = 0$)
4. Supprimer le premier candidat S de $OPEN$
5. Si $End(S) = 1$
6. $success \leftarrow 1$, $S^* \leftarrow S$, $f^* \leftarrow g(S)$
7. Sinon si $Succ(S,r) \neq \emptyset$
8. $TEMP \leftarrow \emptyset$
9. Pour chaque type r d'agent
10. Pour chaque site a dans $Succ(S,r)$
11. calculer le noeud $S(a,r)$ où a est le prochain site à visiter par un agent de type r;
12. $TEMP \leftarrow TEMP \cup \{S(a,r)\}$
13. Fin pour
14. Fin pour
15. Trier $TEMP$ par ordre croissant de $g+h$
16. Ajouter les β_l premiers candidats de $TEMP$ dans $OPEN$
17. Sinon
18. Trier $OPEN$ par ordre croissant de $g+h$
19. Sauvegarder les β_g premiers candidats de $OPEN$ et supprimer les autres candidats
20. Fin si
21. Fin tant que

Cas 4 : les contraintes de précédence, les contraintes d'énergie et le coût des agents ($c(r_1) = c(r_2) = 1$) ne sont pas pris en compte. Le tableau 5. 4 présente le coût des solutions trouvées par l'algorithme 2 et le nombre de nœuds explorés pour différentes valeurs des paramètres β_g et β_l.

Tableau 5. 4 : Coût de la solution (UTs) et complexité pour le cas 4

$\beta_l \setminus \beta_g$	5	10	20	100
2	70 (852)	72 (1732)	58^* (3232)	58^* (14332)
5	68 (882)	74 (1922)	70 (3142)	58^* (16442)
10	-	74 (1922)	70 (3142)	62 (15812)
20	-	-	70 (3142)	62 (15812)

Le coût optimal $f^* = 58$ correspond, par exemple, aux trajectoires $Agent\ (TC, r_1) = a_{13}\, a_2\, a_{15}\, a_{11}\, a_5\, a_{13}\, a_{34}\, a_{29}\, a_{21}\, a_{13}$ et $Agent\ (TC, r_2) = a_{13}\, a_{25}\, a_{31}\, a_{34}\, a_{29}\, a_{23}\, a_{21}\, a_{11}\, a_5\, a_{13}$ qui impliquent 2 agents de type r_1 et 1 agent de type r_2. On observe que l'augmentation de la valeur des paramètres β_l ou β_g n'améliore pas nécessairement le coût de la solution retournée. La non-monotonie de la

performance par rapport à β_l ou β_g est un inconvénient de la méthode proposée qui ne peut être surmonté qu'en considérant plusieurs couples (β_l, β_g).

Cas 5 : Les contraintes de précédence, les contraintes d'énergie et le coût des agents ne sont pas pris en compte. Le coût d'exploitation de chaque type d'agents est supposé varier selon la première colonne du tableau 5. 5. Ce tableau présente les meilleures solutions trouvées pour différentes valeurs des paramètres β_g et β_l.

Tableau 5. 5 : Solutions pour le cas 5

$c(r_1)/\ c(r_2)$	f	Durée totale du trajet	Solution
1 / 1	58*	58	$Agent_1(TC, r_1) = a_{13}\ a_2\ a_{15}\ a_{11}\ a_5\ a_{13}$ $Agent_2(TC, r_1) = a_{13}\ a_{34}\ a_{29}\ a_{21}\ a_{13}$ $Agent\ (TC, r_2) = a_{13}\ a_{25}\ a_{31}\ a_{34}\ a_{29}\ a_{23}\ a_{21}\ a_{11}\ a_5\ a_{13}$
2 / 1	90*	58	$Agent\ (TC, r_1) = a_{13}\ a_{15}\ a_{21}\ a_{23}\ a_{29}\ a_{11}\ a_5\ a_2\ a_{34}\ a_{13}$ $Agent_1(TC, r_2) = a_{13}\ a_{11}\ a_5\ a_{13}$ $Agent_2(TC, r_2) = a_{13}\ a_{31}\ a_{25}\ a_{34}\ a_{29}\ a_{13}$
1 / 2	82*	58	$Agent_1(TC, r_1) = a_{13}\ a_2\ a_{15}\ a_{21}\ a_{11}\ a_5\ a_{13}$ $Agent_2(TC, r_1) = a_{13}\ a_{34}\ a_{29}\ a_{13}$ $Agent\ (TC, r_2) = a_{13}\ a_{31}\ a_{25}\ a_{34}\ a_{29}\ a_{23}\ a_{11}\ a_5\ a_{13}$

On observe que la patrouille de chaque type d'agents varie en fonction du coût des agents. Lorsque le coût d'exploitation de l'agent de type r_1 augmente par rapport au coût d'exploitation de l'agent de type r_2, l'algorithme 2 utilise plus d'agents de type r_2. Au contraire, il utilise plus d'agents de type r_1 lorsque $c(r_2)$ augmente par rapport à $c(r_1)$.

5.5.3. Optimisation des configurations

Dans cette section, nous résolvons le problème 3 qui optimise également la configuration de la flotte. La solution est obtenue en énumérant d'abord les configurations candidates. Ensuite, l'algorithme 2 est utilisé pour optimiser la patrouille pour chaque configuration. Par conséquent, le nombre d'agents de chaque type r et la patrouille de chaque agent de chaque type est obtenue pour chaque configuration, et on peut sélectionner la meilleure configuration.

Exemple 5.6 : Les exemples 5.4 et 5.5 sont poursuivis ici. Les contraintes de précédence et les contraintes d'énergie ne sont pas considérées. Le tableau 5. 6 compare plusieurs configurations et rapporte pour chaque configuration

(i) Le nombre d'agents requis par la patrouille ;

(ii) Le coût de la meilleure solution trouvée par l'algorithme 2 et le nombre de nœuds explorés ;

(iii) Le détail des trajectoires des différents agents pour la meilleure solution. Différentes valeurs des paramètres β_g et β_l sont utilisées.

En fonction du nombre maximal d'agents qui peuvent être utilisés et en fonction du nombre maximal de capteurs que chaque agent peut porter, on peut optimiser la patrouille de surveillance. Pour l'exemple considéré, Si l'on suppose qu'il n'y a pas de limitation du nombre de capteurs portés par chaque agent, la meilleure configuration est obtenue avec un seul agent de type *r* équipé de *A*, *B*, et *C*. Cette patrouille de coût f = 26 UT est reportée sur la Fig. 5. 13. Si l'on suppose que chaque agent peut porter au maximum deux capteurs, la meilleure configuration est obtenue avec un agent de type r_1 : {*B*, *C*} et un second de type r_2 : {*A*}. Le coût de cette patrouille passe à f = 52 UT. Enfin, si chaque agent porte un seul capteur, alors la configuration nécessite 3 types d'agents et le coût de la patrouille augmente à f = 80 UT.

Tableau 5. 6 : Solutions pour diverses configurations

Configuration	Nombre d'agents	f (nœuds exp.)	Solution
r: {*A*,*B*,*C*}	1	26 (652)	*Agent (TC, r_1) = a_{13} a_{25} a_{31} a_{34} a_{29} a_{23} a_{21} a_{15} a_{11} a_5 a_2 a_{13}*
r_1: {*A*,*B*} r_2: {*A*,*C*}	2 1	58 (3232)	*Agent$_1$ (TC, r_1) = a_{13} a_2 a_{15} a_{11} a_5 a_{13},* *Agent$_2$ (TC, r_1) = a_{13} a_{34} a_{29} a_{21} a_{13}* *Agent (TC, r_2) = a_{13} a_{25} a_{31} a_{34} a_{29} a_{23} a_{21} a_{11} a_5 a_{13}*
r_1: {*A*,*B*} r_2: {*B*,*C*} or r_2: {*C*}	1 1	54 (14212)	*Agent (TC, r_1) = a_{13} a_2 a_{15} a_{21} a_{23} a_{29} a_{11} a_5 a_{31} a_{13}* *Agent (TC, r_2) = a_{13} a_{31} a_{25} a_{34} a_{29} a_{11} a_5 a_{13}*
r_1: {*A*,*C*} r_2: {*B*,*C*}or r_2: {*B*}	1 1	56 (14342)	*Agent (TC, r_1) = a_{13} a_{15} a_{21} a_{25} a_{31} a_{34} a_{29} a_{23} a_{11} a_5 a_{13}* *Agent (TC, r_2) = a_{13} a_2 a_{11} a_5 a_{21} a_{34} a_{29} a_{13}*
r_1: {*B*,*C*} r_2: {*A*}	1 1	52 (14602)	*Agent (TC, r_1) = a_{13} a_{31} a_{25} a_{34} a_{29} a_{21} a_{11} a_5 a_2 a_{13}* *Agent (TC, r_2) = a_{13} a_{15} a_{21} a_{23} a_{29} a_{11} a_5 a_{31} a_{13}*
r_1: {*A*} r_2: {*B*} r_3: {*C*}	1 1 1	80 (22332)	*Agent (TC, r_1) = a_{13} a_{15} a_{21} a_{23} a_{29} a_{11} a_5 a_{31} a_{13}* *Agent (TC, r_2) = a_{13} a_2 a_{13} a_{34} a_{29} a_{21} a_{11} a_{13}* *Agent (TC, r_3) = a_{13} a_{25} a_{31} a_{34} a_{29} a_{11} a_5 a_{13}*

5.6. Analyse de la complexité

La performance et la complexité numérique de l'approche d'optimisation dépendent fortement des valeurs des paramètres β_g et β_l. En général, la performance, c'est-à-dire le coût de la solution

retournée par l'algorithme HFBS par rapport au coût de la solution optimale, diminue et la complexité numérique, c'est-à-dire le nombre total de solutions candidates explorées, augmente lorsque β_g et β_l augmentent. Un compromis doit donc être trouvé entre la performance et la complexité. L'objectif de cette section est de discuter en détail les aspects de la complexité [Gam et al., 2022a].

En premier lieu, pour 100 ensembles de tâches de surveillance de dimension k=50 générés aléatoirement dans un environnement de taille $N=50\times50$, la variabilité de la complexité est reportée dans la Fig. 5. 14. Dans ces expériences, on considère un seul type d'agents mobiles une seule mesure (q=1), et des valeurs des paramètres β_g =10, β_l = 5.

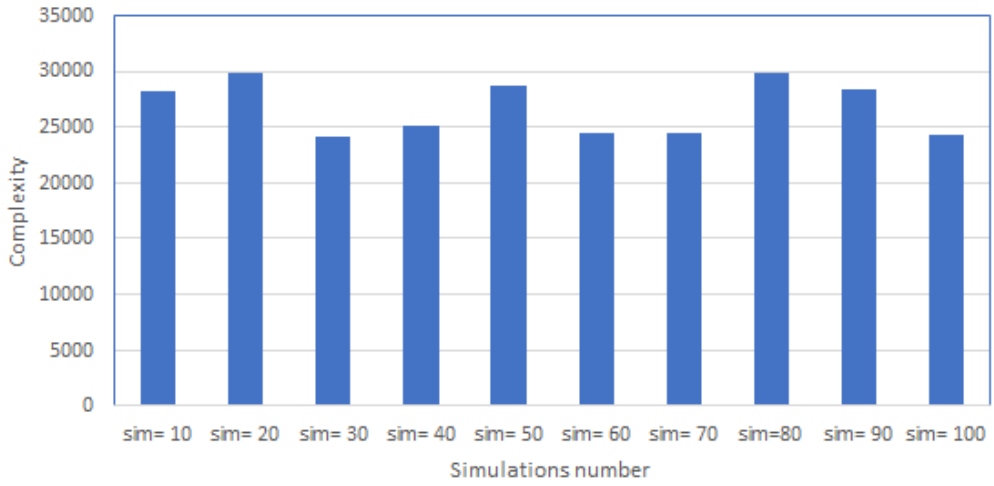


Figure 5. 12 : Variabilité de la complexité de calcul.

D'après la Fig. 5. 14, la complexité est presque constante, ce qui signifie que la distribution spatiale des sites nécessitant une visite a une faible influence sur la complexité numérique (dans la mesure où l'environnement et le nombre de sites sont supposés constants).

En deuxième lieu, pour analyser la variation de la complexité en fonction des valeurs des paramètres N et k, nous considérons 10 valeurs de la taille de l'environnement N, et 5 valeurs de la dimension des tâches de surveillance k. Dans ces expériences, un seul type d'agents mobiles, une seule mesure (q=1) et des valeurs β_g =10, β_l = 5 sont considérés. Les résultats sont présentés dans la Fig. 5. 15 où le nombre de solutions candidates explorées est indiqué en fonction de N et de k.

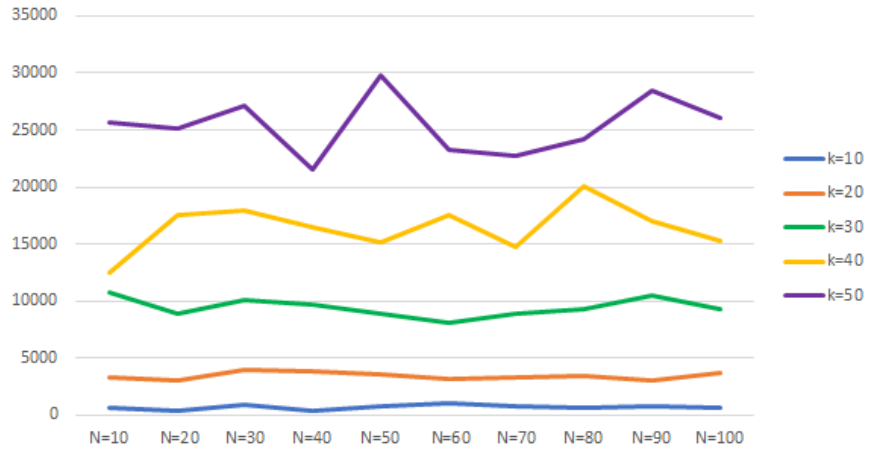


Figure 5. 13 : Analyse de la complexité pour différentes valeurs des paramètres *k* et de *N*.

D'après la Fig. 5. 15, on peut observer que la variation de la complexité est presque constante par rapport à *N*, surtout lorsque la valeur de *k* est petite. La raison est que l'algorithme HFBS utilise la matrice des coûts de déplacements entre les sites et non directement la carte de l'environnement. Cette matrice est calculée hors ligne avant l'exécution de l'algorithme HFBS. Par conséquent, l'environnement a peu d'influence sur la complexité.

En troisième lieu, pour différentes valeurs des paramètres β_g et β_l, le calcul de la complexité est présenté dans la Fig. 5. 16. Dans ces expériences, nous utilisons N=10 ×10, k=10, un seul type d'agents mobiles et une seule mesure (q=1).

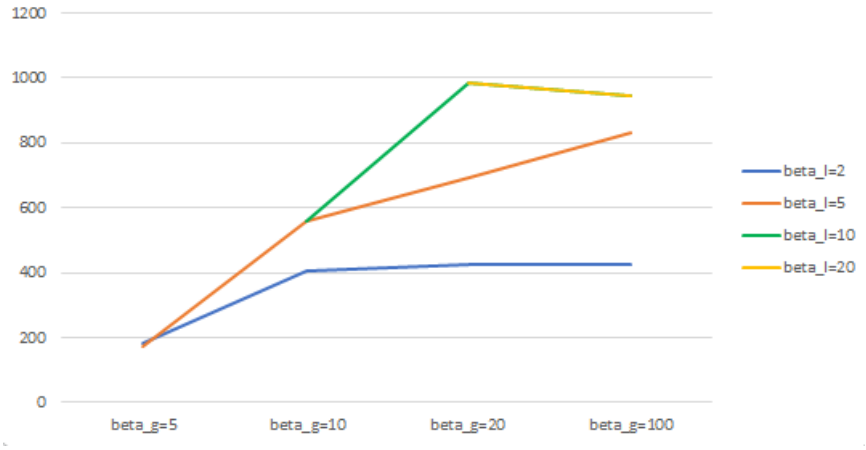


Figure 5. 14 : Analyse de la complexité pour différentes valeurs des paramètres β_g et β_l.

D'après la Fig. 5. 16, on peut observer que la complexité augmente avec β_g et β_l. On peut remarquer que pour de petites valeurs de β_g, l'augmentation est rapide, et pour de plus grandes valeurs de β_g, une saturation apparaît. On peut également conclure que l'augmentation de β_l influence la variation de la complexité et par la suite le nombre de nœuds explorés.

Finalement, nous considérons plusieurs types de mesure et étudions la complexité numérique par rapport aux nombres q de types de mesure et N_r de types d'agents. Dans ces expériences, la taille de l'environnement est N=10×10, la dimension des tâches de surveillance est k=11, β_g =10 et β_l = 5.

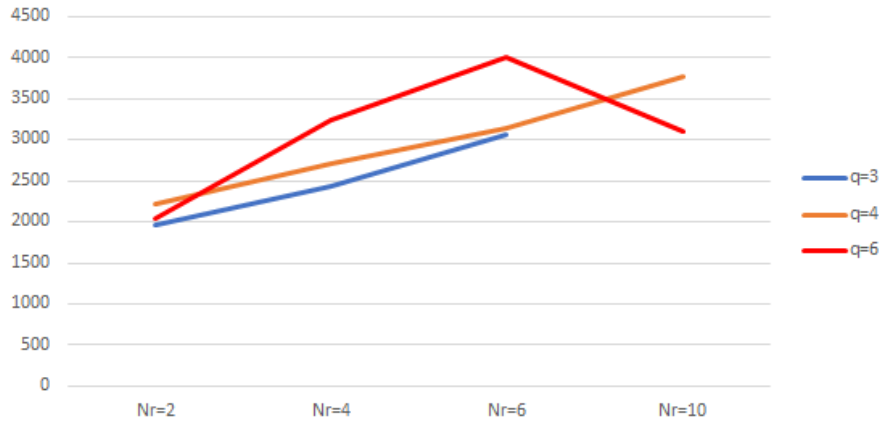


Figure 5. 15 : Analyse de la complexité pour différentes valeurs des paramètres q et N_r.

D'après la Fig. 5. 17, on observe que la complexité augmente en fonction des paramètres q et N_r mais les variations sont faiblement dépendantes de ces paramètres. On peut également remarquer que lorsque la valeur de q est faible, la complexité est plus ou moins constante. On peut conclure que le paramètre q ne peut influencer l'évolution de la complexité que pour un grand nombre de types de mesures [Gam et al., 2022a].

5.7. Application à la surveillance du site de Lubrizol

On considère l'usine Lubrizol de la ville de Rouen (Fig. 5. 18) comme un cas d'étude. Cette usine est classée Seveso seuil haut et génère des activités à haut niveau de risque. L'usine synthétise et stocke des produits chimiques (phosphore et composés organosoufrés) destinés à être utilisés comme additifs pour les lubrifiants. Malheureusement, un incendie s'est déclaré dans la nuit du 26 septembre 2019. La dégradation de 5 253 tonnes de produits tels que des hydrocarbures, des huiles et des additifs, a conduit à l'émission de produits toxiques et à des dégazages dans l'atmosphère qui ont entraîné une immense colonne de fumée noire dirigée vers le nord-est pendant environ 24 heures.

Le gouvernement a mentionné que la violence de l'incendie a complètement détruit plusieurs bâtiments ainsi qu'un entrepôt industriel de la Société Commerciale d'Entreposage et de Transport mais heureusement l'incendie n'a pas causé de victimes. Il a également annoncé que

le risque principal est celui d'une propagation de la nappe enflammée aux installations "sensibles" de l'usine, par contre le risque de pollution de la Seine reste secondaire par rapport à ce risque chimique [Huard et al., 2020 ; Negre, 2021].

En plus des études d'urbanisation et de gestion des risques (PPRT) déjà existantes, l'utilisation d'une technologie de surveillance avancée devient nécessaire pour mettre en œuvre toutes les mesures de sécurité possibles afin d'atteindre un niveau de risque aussi bas que possible. Un système avec des véhicules à guidage automatique (AGVs) est proposé à cet effet. La zone décrite sur la Fig. 5. 18 comprend deux grands bâtiments industriels de l'usine Lubrizol entourés de nombreux bâtiments qui doivent tous être surveillés [Gam et al., 2022a].

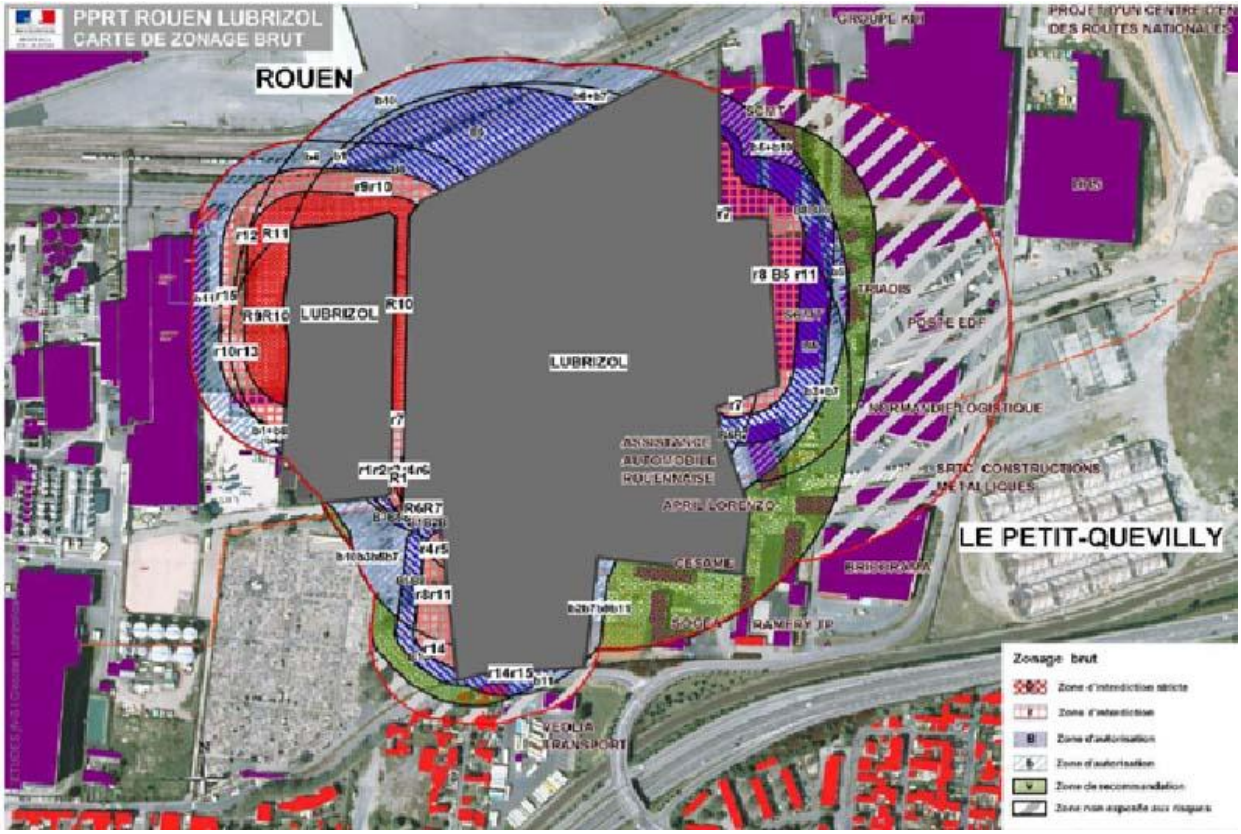


Figure 5. 16 : Usine Lubrizol de la ville de Rouen.

Cet environnement a été représenté par une grille de dimension 43 × 29 (C = 43 et L =29) et divisé en 1247 cellules d'adresses $\{a_1, \ldots, a_{1247}\}$ (voir Fig. 5. 19). Les cellules noires représentent des bâtiments ou des obstacles que les AGVs ne peuvent pas visiter, tandis que les cellules blanches représentent des zones ouvertes où les robots mobiles peuvent se déplacer et prendre des mesures. Le coût $c\ (a_i, a_j)$ entre les cellules adjacentes a_i et a_j est supposé être égal à 1 si les AGV peuvent se déplacer de a_i à a_j, sinon $c\ (a_i, a_j) = \infty$ lorsque a_j est un obstacle. Chaque cellule est représentée par son adresse et un système de coordonnées cartésiennes est

également affecté à la Fig. 5. 19 pour identifier chaque cellule d'adresse a_i par ses coordonnées (x_i, y_i). Pour des raisons de lisibilité, les adresses des cellules ne sont pas représentées dans les figures suivantes mais on peut calculer l'adresse a_i de chaque cellule à partir de ses coordonnées et vice versa, en utilisant les équations suivantes :

$$a_i = (y_i\text{-}1) \times C + x_i$$
$$x = a_i - (y_i - 1) \times C \qquad (6)$$
$$y_i = \lceil a_i / C \rceil$$

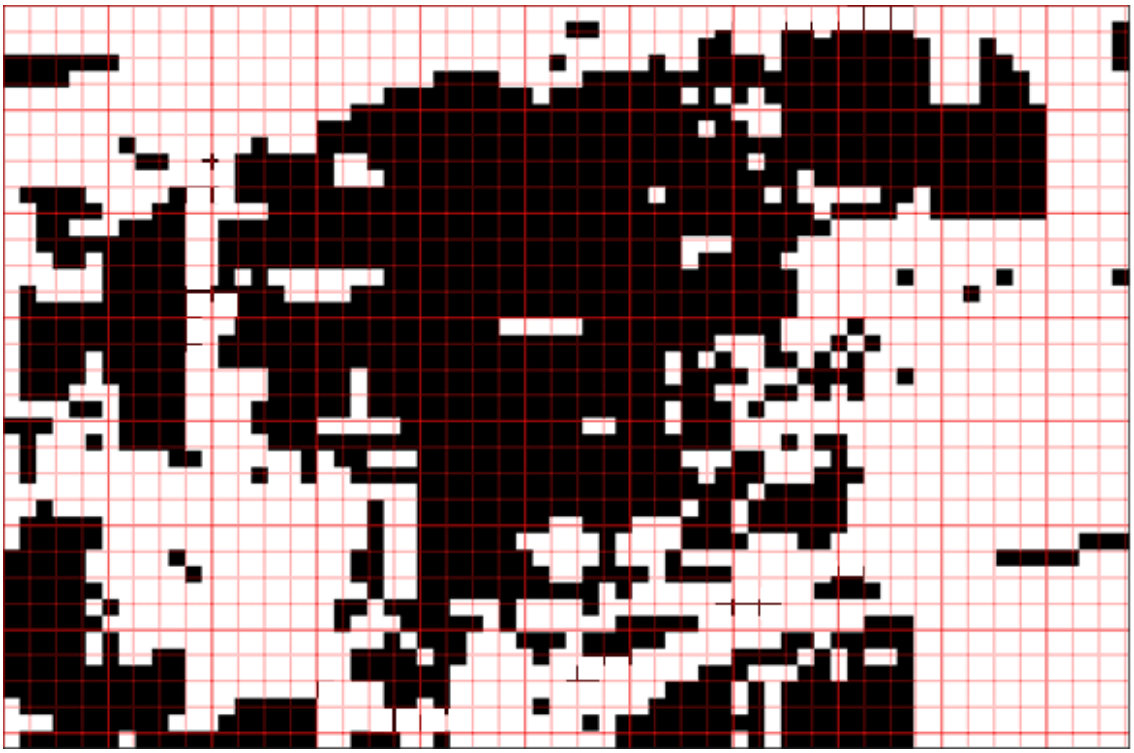


Figure 5. 17 : L'environnement avec 1247 cellules pour l'usine Lubrizol de la ville de Rouen.

Deux exemples de patrouilles de surveillance illustrent notre travail. La première patrouille est un exemple simple avec un petit nombre de sites à inspecter :

$E_1 = \{(a_{1226}, \varnothing), (a_{1088}, \{a\}), (a_{483}, \{a\}), (a_{148}, \{a\}), (a_{545}, \{a\}), (a_{806}, \{a\})\}$

Les agents démarrent la patrouille depuis le site a_{1226}, ils visitent successivement les sites $\{a_{1088}, a_{483}, a_{148}, a_{545}, a_{806}\}$ pour effectuer une seule mesure a et reviennent finalement au site a_{1226}. Dans cet exemple, un seul type d'agents est considéré avec le capteur nécessaire pour effectuer la mesure a. Les coûts d'exploitation des agents sont supposés être identiques : $c(r) = 1$, $r = 1, \ldots, Nr$. Dans la mesure où les contraintes de précédence et d'énergie ne sont pas prises en

compte, l'algorithme affecte un seul agent à la patrouille et calcule sa trajectoire $\sigma = a_{1226}\ a_{1088}\ a_{483}\ a_{148}\ a_{545}\ a_{806}\ a_{1226}$ comme indiqué sur la Fig. 5. 20. Le coût optimal $f^* = 136$ UTs est trouvé pour $\beta_l = 5$ et $\beta_g = 7$.

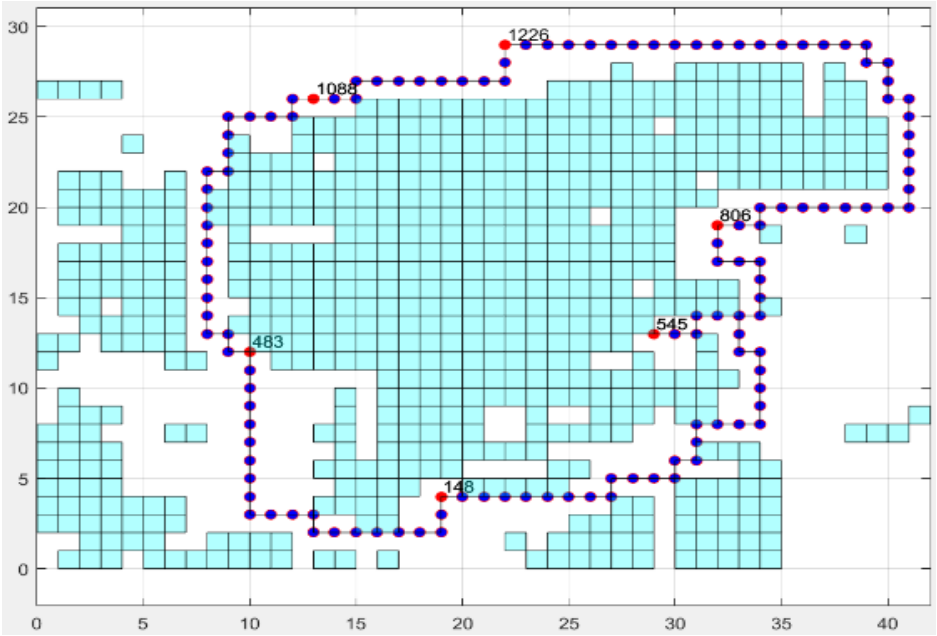


Figure 5. 18 : Solution optimale trouvée pour la patrouille E_1 sans aucune contrainte de précédence et d'énergie

Ensuite, une contrainte d'énergie de 109 UTs qui correspond à l'autonomie de l'agent est ajoutée au cas précédent. Les autres spécifications du problème restent inchangées. L'algorithme affecte deux agents (qui peuvent opérer simultanément) à la patrouille et calcule leurs trajectoires $Agent_1\ (TC, r_1) = a_{1226}\ a_{1088}\ a_{483}\ a_{148}\ a_{1226}$ et $Agent_2\ (TC, r_1) = a_{1226}\ a_{806}\ a_{545}\ a_{1226}$ comme indiqué sur la Fig. 5. 21. Dans ce cas, le coût global de la solution optimale augmente et atteint $f^* = 206$ UT, trouvé pour $\beta_l = 1$ et $\beta_g = 2$. Cet exemple illustre comment notre algorithme adapte le nombre d'agents aux contraintes d'énergie.

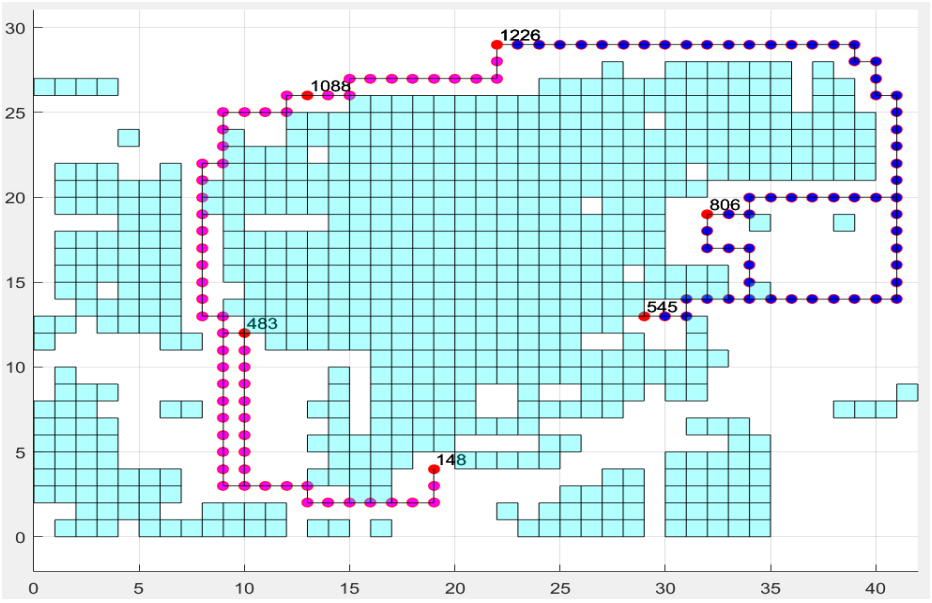


Figure 5. 19 : Solution optimale trouvée pour la patrouille E_1 avec les contraintes énergétiques

La deuxième patrouille est un exemple de plus grande taille qui comprend la surveillance de 26 sites et 4 types de mesures

E_2 = {(a_{1226}, ∅), (a_{1184}, {a, b, c, d}), (a_{1079}, {a, b}), (a_{1247}, {c, d}), (a_{1134}, {b, c}), (a_{1155}, {a, b, c, d}), (a_{951}, {a, d}) , (a_{1030}, {a}) , (a_{776}, {b, c}) , (a_{849}, {b, c, d}) , (a_{782}, {a, d}), (a_{812}, {b}) , (a_{612}, {a, b, c, d}), (a_{774}, {d}) , (a_{636}, {b, c}) , (a_{476}, {c}) , (a_{443}, {a, b}) , (a_{549}, {c, d}) , (a_{429}, {a, c, d}) , (a_{310}, {a, b, d}) , (a_{178}, {a, b, c, d}) , (a_{142}, {b, c, d}), (a_{287}, {a, b}) , (a_{165}, {c, d}) , (a_{111}, {b, d}) , (a_{61}, {b, d})}

Les agents commencent la patrouille dans le site a_{1226}, puis ils visitent les sites {a_{1184}, a_{1079}, a_{1247}, a_{1134}, a_{1155}, a_{951}, a_{1030}, a_{776}, a_{849}, a_{782}, a_{812}, a_{612}, a_{774}, a_{636}, a_{476}, a_{443}, a_{549}, a_{429}, a_{310}, a_{178}, a_{142}, a_{287}, a_{165}, a_{111}, a_{61}} pour effectuer dans chaque site des mesures a, b, c ou d et finalement retournent au site a_{1226}. Dans cet exemple, trois types d'agents r_1, r_2 et r_3 sont considérés : les agents de type r_1 ont des capteurs pour les mesures a et c tandis que les agents de types r_2 et r_3 ont des capteurs pour mesurer respectivement b et d ou a et b. Le coût d'exploitation de chaque type d'agents est supposé être le même : $c(r_1) = c(r_2) = c(r_3) = 1$.

Dans la mesure où les contraintes de précédence et d'énergie ne sont pas prises en compte, l'algorithme proposé affecte 2 agents à la patrouille : 1 agent de type r_1 (Fig. 5. 22 haut-gauche), 1 agent de type r_2 (Fig. 5. 22 milieu-gauche). Aucun agent de type r_3 n'est requis car les agents

de type r_3 sont équipés de capteurs *A* et *B* qui sont déjà portés par les agents de types r_1 et r_2 (Fig. 5. 22 bas-gauche).

Agent (TC, r_1) = $a_{1226}\ a_{1184}\ a_{1134}\ a_{1079}\ a_{951}\ a_{776}\ a_{782}\ a_{612}\ a_{443}\ a_{310}\ a_{476}\ a_{178}\ a_{142}\ a_{287}\ a_{165}\ a_{549}\ a_{849}\ a_{636}\ a_{429}\ a_{1030}\ a_{1247}\ a_{1155}\ a_{1226}$

Agent (TC, r_2) = $a_{1226}\ a_{1155}\ a_{812}\ a_{774}\ a_{429}\ a_{165}\ a_{549}\ a_{636}\ a_{849}\ a_{287}\ a_{111}\ a_{61}\ a_{142}\ a_{178}\ a_{443}\ a_{310}\ a_{612}\ a_{782}\ a_{776}\ a_{951}\ a_{1079}\ a_{1134}\ a_{1184}\ a_{1247}\ a_{1226}$

Le coût de cette patrouille est $f^* = 488$ UT (obtenu pour $\beta_l = 30$ et $\beta_g = 50$ avec un arbre de 57127 nœuds).

Ensuite, le cas précédent est reconsidéré avec une contrainte d'énergie de 120 UTs. Les autres spécifications du problème restent inchangées. Dans un tel cas, l'algorithme affecte 11 agents à la patrouille : 4 agents de type r_1 (Fig. 5. 22 haut-droit), 4 agents de type r_2 (Fig. 5. 22 milieu-droit) et 3 agents de type r_3 (Fig. 5. 22 bas-droit).

Agent$_1$ (TC, r_1) = $a_{1226}\ a_{1184}\ a_{1226}$

Agent$_2$ (TC, r_1) = $a_{1226}\ a_{1155}\ a_{1247}\ a_{1030}\ a_{849}\ a_{636}\ a_{549}\ a_{165}\ a_{429}\ a_{1226}$

Agent$_3$ (TC, r_1) = $a_{1226}\ a_{1134}\ a_{1079}\ a_{776}\ a_{951}\ a_{142}\ a_{782}\ a_{1226}$

Agent$_4$ (TC, r_1) = $a_{1226}\ a_{612}\ a_{310}\ a_{476}\ a_{443}\ a_{178}\ a_{1226}$

Agent$_1$ (TC, r_2) = $a_{1226}\ a_{1184}\ a_{1134}\ a_{951}\ a_{1079}\ a_{782}\ a_{612}\ a_{178}\ a_{310}\ a_{443}\ a_{1226}$

Agent$_2$ (TC, r_2) = $a_{1226}\ a_{1155}\ a_{1247}\ a_{744}\ a_{812}\ a_{849}\ a_{636}\ a_{549}\ a_{429}\ a_{1226}$

Agent$_3$ (TC, r_2) = $a_{1226}\ a_{142}\ a_{61}\ a_{111}\ a_{287}\ a_{1226}$

Agent$_4$ (TC, r_2) = $a_{1226}\ a_{165}\ a_{1226}$

Agent$_1$ (TC, r_3) = $a_{1226}\ a_{1134}\ a_{1184}\ a_{1226}$

Agent$_2$ (TC, r_3) = $a_{1226}\ a_{951}\ a_{1079}\ a_{776}\ a_{782}\ a_{612}\ a_{310}\ a_{443}\ a_{1226}$

Agent$_3$ (TC, r_3) = $a_{1226}\ a_{287}\ a_{1030}\ a_{1226}$

Le coût global de cette patrouille est $f^* = 1024$ UT (obtenu pour $\beta_l = 20$ et $\beta_g = 30$ avec un arbre de 78082 nœuds).

Ce dernier exemple montre que notre travail peut être appliqué à des situations réelles avec un environnement étendu et des tâches de surveillance compliquées [Gam et al., 2022a].

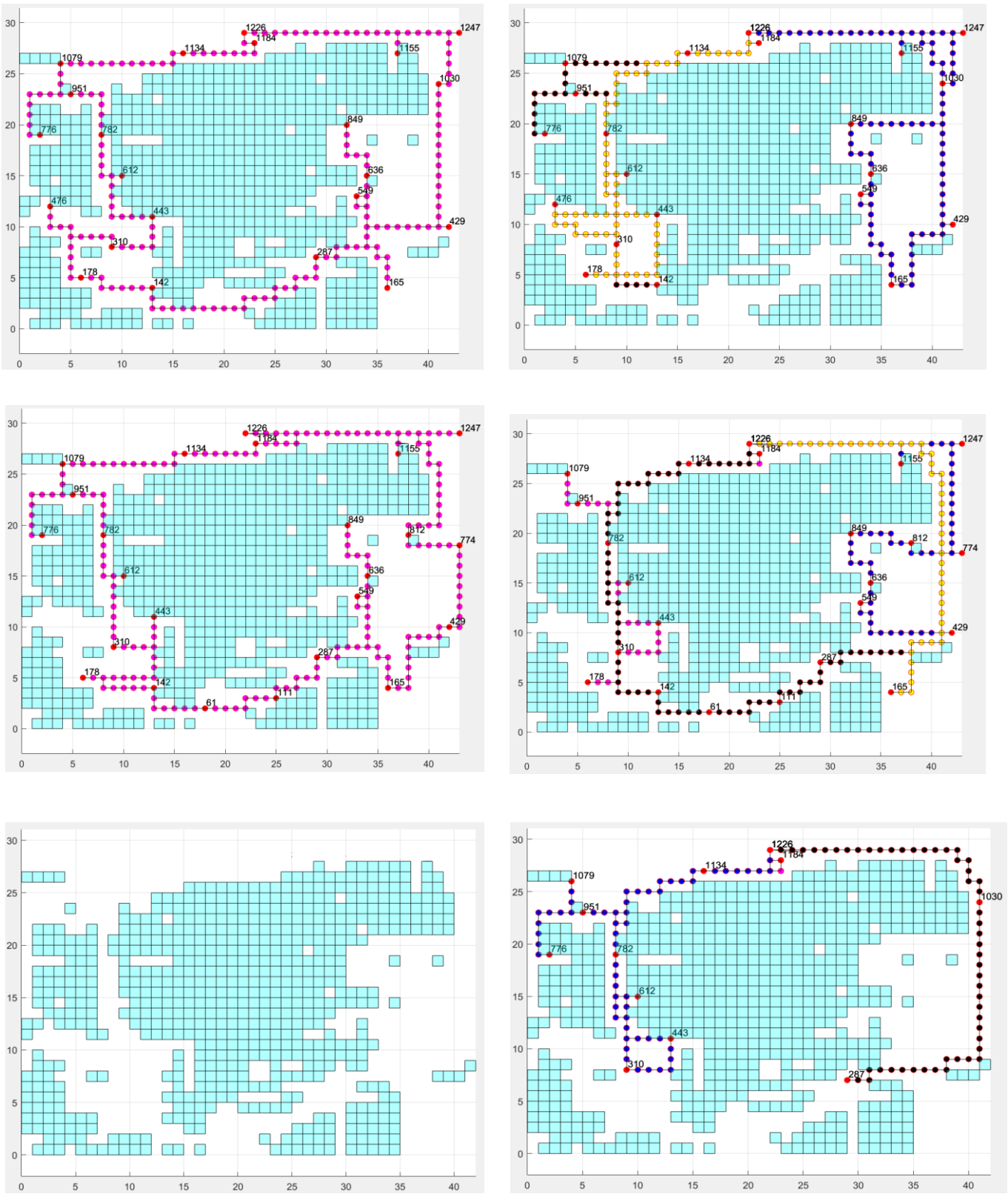


Figure 5. 20 : Solutions optimales trouvées pour la patrouille E_2 : trajectoires des agents de type r_1 (en haut) ; trajectoires des agents de type r_2 (au milieu) ; trajectoires des agents de type r_3 (en bas) pour une patrouille sans contraintes (à gauche) et avec contraintes (à droite).

5.8. Conclusion

Afin de réaliser les tâches de surveillance d'une zone industrielle donnée exposée à un risque technologique ou à un danger naturel, une approche a été proposée qui optimise le nombre d'agents de chaque type, les tâches de surveillance allouées à chaque agent et leurs trajectoires. Le cœur de l'approche est l'algorithme de recherche en faisceau filtrée hybride qui est utilisé pour explorer partiellement l'ensemble des solutions possibles. En ajoutant d'éventuelles contraintes de précédence et d'énergie, la méthode est applicable à une grande variété de situations.

Chapitre 6 : Conclusion générale

6.1. Conclusions

Depuis plusieurs décennies, la surveillance est devenue nécessaire pour éviter l'apparition de pannes catastrophiques dans les systèmes industriels et améliorer la sécurité des systèmes avec un niveau de risque élevé. Pour cette raison, les systèmes de contrôle automatisés sont exploités comme des outils de collecte de données. Ils sont utilisés efficacement afin de doter les robots d'intelligences artificielles et d'algorithmes de prise de décision dans le but d'effectuer les tâches de surveillance distribuées sur les sites industriels. Dans cette thèse, les conclusions principales que nous avons obtenues sont les suivantes :

Premièrement, les approches SED permettent de configurer et d'optimiser une séquence de tâches de surveillance. A cet effet, l'environnement à surveiller ainsi les trajectoires des robots ont été modélisés par les réseaux de Petri. Une optimisation globale des tâches de surveillance est obtenue en couplant les problèmes de configuration et de planification des trajectoires. En particulier, la configuration optimale de l'ensemble des agents mobiles et la trajectoire de chaque agent sont déterminées en fonction de coûts d'investissement des équipements et des coûts d'exploitation de chaque équipement.

Deuxièmement, les méthodes d'optimisation combinatoire ont été utilisées pour résoudre le même type de problèmes. Une approche a été proposée pour optimiser le nombre d'agents de chaque type, les tâches de surveillance allouées à chaque agent et les trajectoires des agents afin de minimiser le coût global de la mission. Le cœur de cette méthode réside dans la formalisation du problème par la théorie des graphes et dans sa résolution par des techniques de parcours de graphe notamment la recherche en faisceau filtrée hybride (HFBS). Le HFBS est utilisé pour explorer partiellement l'ensemble des solutions possibles. En ajoutant des contraintes de précédence et d'autonomie, la méthode est applicable à une grande variété de situations pour traiter des situations réelles.

6.2. Perspectives

6.2.1. A court terme

Au terme de ces travaux de recherche, plusieurs perspectives intéressantes se dégagent dans le but d'étendre les contributions et d'améliorer les résultats obtenus :

- La résolution du problème est actuellement réalisée avant le début de la mission c'est-à-dire hors-ligne et n'inclut pas de reconfiguration en ligne. Une adaptation de la méthode

d'optimisation par HFBS peut être envisagée dans nos travaux futurs pour étudier la reconfiguration en ligne des patrouilles i.e., la reconfiguration de la solution en cours de mission. En effet, le problème peut être contraint à des situations qui nécessitent une intervention immédiate sur l'environnement ou sur le site d'action. Cette intervention permettra notamment d'améliorer et d'adapter la mission pour prendre en considération des tâches qui arrivent à la volée et aussi pour redistribuer les tâches restantes en cas de panne des agents.

- En outre, le calcul de certains indicateurs moyens peut évaluer la fiabilité et la disponibilité de la patrouille et la durée moyenne de la surveillance en prenant en compte la disponibilité des ressources et les défaillances futures de l'un des agents et/ou des capteurs.

- Dans ce travail de thèse, nous avons considéré une approche centralisée i.e., une unité centrale prend les décisions et qui les impose aux autres agents. Les robots exécutent simplement les trajectoires générées par cette unité. Par contre, dans les approches distribuées, chaque robot est aussi une unité de prise de décision et communique avec les autres robots pour coordonner leurs actions. L'allocation centralisée a la possibilité de produire des solutions optimales ou quasi-optimales, car toutes les informations pertinentes sont rassemblées au point de décision unique. En revanche, les approches distribuées ont tendance à produire des solutions approchées car les décideurs individuels ne travaillent généralement qu'avec des informations locales. Par contre, ces solutions sont généralement plus robustes car moins sensibles aux pannes des agents.

- Dans cette thèse, le coût de la mission est basé sur le coût d'exploitation des robots i.e., le coût de déplacement pour aller d'un site à un autre et le coût d'investissement du matériel. Cependant, le coût de la prise de mesure n'est pas considéré dans ce travail. L'idée est de mettre en œuvre une extension de la modélisation du problème en considérant non seulement le coût de déplacement, mais aussi le coût de prise de mesure.

- Parmi les méthodes d'optimisation combinatoire, l'algorithme de la recherche en faisceau filtrée hybride (HFBS) a été choisi dans notre travail comme une méthode heuristique afin de trouver la meilleure solution. Nous savons que la performance d'une méthode heuristique dépend tout d'abord de la pertinence de la fonction objectif et surtout de la partie estimée de cette fonction. Une meilleure estimation pourrait être étudiée. Une autre

amélioration consisterait à remplacer HFBS par une métaheuristique telle que les algorithmes génétiques, les algorithmes de colonies de fourmis, ou la recherche tabou.

6.2.2. A long terme

Enfin, les travaux de recherches développés dans cette thèse permettent de dégager des perspectives de recherche à plus long terme :

- Nos travaux futurs se concentreraient d'abord sur l'étude des patrouilles de sauvetage pour lesquelles les agents doivent transporter non seulement des capteurs mais aussi des ressources qui doivent être transportées et distribuées depuis des sites d'origine vers des sites de destination.
- Une autre perspective à long terme serait d'utiliser des systèmes logiciels pour contrôler à distance et en temps réel les missions de surveillance. En effet, les systèmes logiciels peuvent servir de planificateur centralisé pour gérer à distance les flottes de robots en partageant des informations entre les robots et en assurant ainsi leur coopération. L'objectif est de fournir en temps réel un traitement des données de l'environnement dynamique où de nouveaux événements peuvent survenir pendant la mission des robots.

Les travaux de recherche présentés dans cette thèse ont abouti à plusieurs publications et ont été conclus par la publication d'un article de revue [Gam et al., 2021a], la soumission d'un deuxième article de revue [Gam et al., 2022a] actuellement en révision, ainsi que la publication des conférences internationales [Gam et al., 2020 ; Gam et al., 2021b ; Gam et al., 2022b].

Bibliographie

[Aggoune, 2002] Aggoune, R., (2002), « Ordonnancement d'ateliers sous contraintes de disponibilité des machines », Thèse de Doctorat, Metz.

[Ahmadi & Stone, 2006] Ahmadi, M., Stone, P., (2006), « Keeping in touch: Maintaining biconnected structure by homogeneous robots », Proceedings of the National Conference on Artificial Intelligence and the Innovative Applications of Artificial Intelligence Conference, Boston, Massachusetts, USA, pp. 580-585.

[Ahmadzadeh et al., 2006] Ahmadzadeh, A., Jadbabaie, A., Kumar, V., Pappas, G. J., (2006), « Multi-UAV Cooperative Surveillance with Spatio-Temporal Specifications », Proceedings of the 45th IEEE Conference on Decision and Control, San Diego, CA, USA, pp. 1191-2216.

[Akli, 2007] Akli, I., (2007), « Elaboration d'une stratégie de coordination de mouvements pour un manipulateur mobile redondant », Thèse de Doctorat, Université des Sciences et Technologies Houari Boumediene.

[Alitappeh & Jeddisaravi, 2022] Alitappeh, R. J., Jeddisaravi, K., (2022), « Multi-robot exploration in task allocation problem », *Applied Intelligence*, Vol. 52, No. 2, pp. 2189–2211.

[Al-Yafi et al., 2009] Al-Yafi, K., Lee, H., Mansouri, A., (2009), « Mtap-masim:a multi-agent simulator for the mobile task allocation problem », IEEE International Workshop on Enabling Technologies: Infrastructures for Collaborative Enterprises, Groningen, Netherlands, pp. 25–27.

[Angeli & Chatzinikolaou, 2004] Angeli, C., Chatzinikolaou, A., (2004), « On-line fault detection techniques for technical systems: a survey », *International Journal of Computer Science & Applications*, Vol. I, No. 1, pp. 12 - 30.

[Anjomshoaa et al., 2018] Anjomshoaa, A., Duarte, F., Rennings, D., Matarazzo, T., deSouza, P., Ratti, C., (2018), « City Scanner: Building and Scheduling a Mobile Sensing Platform for Smart City Services », *IEEE Internet of Things Journal*, Vol. 5, No. 6, pp. 4567 – 4579.

[Arjun et al., 2015] Arjun, R. K., Reddy, P., Shama, Yamuna, M., (2015), « Research on the Optimization of Dijkstra's Algorithm and Its Applications », *International Journal of Science, Technology & Management*, Vol. 4, No.1, pp. 304-309.

[Atay & Bayazit, 2006] Atay, N., Bayazit, B., (2006), « Mixed-Integer Linear Programming Solution to MRTA Problem », All Computer Science and Engineering Research, https://openscholarship. wustl.edu/cse_research/205.

[Aurenhammer, 1991] Aurenhammer, (1991), « Voronoi diagrams: a survey of a fundamental geometric data structure », *ACM Computing Surveys*, Vol. 23, No. 3, pp. 345–405.

[Bacco et al., 2017] Bacco, M., Delmastro, F., Ferro, E., Gotta, A., (2017), « Environmental Monitoring for Smart Cities », *IEEE Sensors Journal*, Vol. 17, No. 23, pp. 7767 - 7774.

[Bae et al., 2019] Bae, J., Lee, J., Chung, W., (2019), « A Heuristic for Task Allocation and Routing of Heterogeneous Robots while Minimizing Maximum Travel Cost », International Conference on Robotics and Automation, Montreal, Canada, DOI: 10.1109/ICRA.2019.8794257.

[Bahi et al., 2019] Bahi, J., Elghazel, W., Guyeux, C., Hakem, M., Medjaher, K., Zerhouni, N., (2019), « Reliable diagnostics using wireless sensor networks », *Computers in Industry*, Vol. 104, pp.103–115.

[Balas et al., 2008] Balas, E., Simonetti, N., Vazacopoulos, A., (2008), « Job shop scheduling with setup times, deadlines and precedence constraints » *Journal of Scheduling*, Vol. 11, No. 4, pp. 253–262.

[Baptiste & Le Pape, 1996] Baptiste, P., Le Pape, C., (1996), « A constraint-Based Branch and Bound Algorithm for Preemptive Job-Shop Scheduling », 5th IEEE International Symposium on Assembly and Task Planning, Belgium, pp. 263–287.

[Barichard, 2003] Barichard, V., (2003), « Approches hybrides pour les problèmes ` multiobjectifs », Thèse de Doctorat, Ecole Doctorale d'Angers.

[Battiti & Tecchiolli, 1994] Battiti, R., Tecchiolli, T., (1994), « The Reactive Tabu Search », *ORSA Journal on Computing*, Vol. 6, No. 2, pp. 126-140.

[Baylie, 2008] BAYLE, B., (2008), « Robotique mobile », Cours, Université de Strasbourg, bernard.bayle@unistra.fr.

[Bektas, 2006] Bektas, T., (2006), « The multiple traveling salesman problem: an overview of formulations and solution procedures », Omega, Vol. 34, No. 3, pp. 209–219.

[Belhoul, 2014] Belhoul, L., (2014), « Résolution de problèmes d'optimisation combinatoire mono et multi-objectifs par énumération ordonnée », Thèse de doctorat, Université Paris Dauphine - Paris IX.

[Bellman, 1986] Bellman, R. E., (1986), « The Bellman continuum », Editions Robert S. Roth, Philadelphia, USA.

[Bertsekas, 1990] Bertsekas, D. P., (1990), « The Auction Algorithm for Assignment and Other Network Flow Problems: A Tutorial ». *Informs Journal on Applied Analytics*, Vol. 20, No 4, pp. 133-149.

[Bertsekas, 1992] Bertsekas, D. P., (1992), « Auction Algorithms for Network Flow Problems: A Tutorial Introduction ». *Computational Optimization and Applications*, Vol. 1, No. 1, pp. 7-66.

[Biçen et al., 2014] Biçen, Y., Aras, F., Kirkici, H., (2014) « Lifetime estimation and monitoring of power transformer considering annual load factors », *IEEE Transactions on Dielectrics and Electrical Insulation*, Vol. 21, No. 3, pp. 1360 – 1367.

[Booth, 1967] Booth, T. L., (1967), « Sequential Machines and Automata Theory », *The Computer Journal*, Vol. 11, No. 2, p. 176.

[Bourgeois & Lassalle, 1971] Bourgeois, F., Lassalle, J. C., (1971) « An extension of the Munkres algorithm for the assignment problem to rectangular matrices ». *Communications of the ACM*, Vol. 14, No 12, pp. 802-804.

[Braysy & Gendreau, 2005] Braysy, O., Gendreau, M. (2005), « Vehicle routing problem with time windows, Part I: Route construction and local search algorithms », *Transportation Science* Vol. 39, No. 1, pp. 104-118.

[Brumitt & Stentz, 1998] Brumitt, B.L., Stentz, A., (1998), « GRAMMPS: a generalized mission planner for multiple mobile robots in unstructured environments », IEEE International Conference on Robotics and Automation (ICRA), Leuven, Belgium, pp. 1564–1571.

[Brusco & Stahl, 2005] Brusco, M. J., Stahl, S., (2005), « Branch-and-Bound Applications in Combinatorial Data Analysis », book of Statistics and Computing , Springer.

[Burkard & Eranda, 1999] Burkard, R. E., Eranda, Ç., (1999), « Linear Assignment Problems and Extensions ». *Handbook of Combinatorial Optimization*, Vol. A, pp. 75-149.

[Canfield, 1995] Canfield, E. R., (1995), « Engel's inequality for Bell numbers », *Journal of Combinatorial Theory*, Vol. 72, No. 1, pp. 184–187.

[Carlier & Chrétienne, 1982] Carlier, J., Chrétienne, P., (1982), « Un domaine très ouvert : les problèmes d'ordonnancement », *Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle*, Vol. 16, No. 3, pp. 175-217.

[Carlier & Chrétienne, 1988] Carlier, J., Chrétienne, P., (1988), « Problèmes d'ordonnancement, Modélisation, Complexité, Algorithmes », Edition Masson, Paris.

[Cassandras & Lafortune, 2008] Cassandras, C.G., Lafortune, S., (2008), « Introduction to Discrete Event Systems », Livre Springer, New York, NY, USA, pp. 617–740.

[Cerný, 1985] Cerný, V., (1985), « A thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm », *Journal of Optimization Theory and* Applications, Vol. 45, No. 1, pp. 41–51.

[Chabot, 2015] Chabot, T., (2015), « Résolution des problèmes de tournées de véhicules pour le transport des échantillons biomédicaux au Québec », Mémoire de maîtrise, Université Laval.

[Chao et al., 1996] Chao, I-M., Golden, B. L., Wasil, E. A., (1996), « The team orienteering problem », *European Journal of Operational Research*, Vol. 88, No. 3, pp. 464–474.

[Chao & Hongxia, 2010] Chao, Y., Hongxia, W., (2010), « Developed Dijkstra Shortest Path Search Algorithm and Simulation », International Conference On Computer Design And Appliations (ICCDA), Qinhuangdao, China, pp. 116-119.

[Chen et al., 2014] Chen, Y. Z., Shen, S-f., Chen, T., and Yang, R., (2014), « Path optimization study for vehicles evacuation based on Dijkstra algorithm », *Procedia Engineering*, Vol. 71, pp. 159-165.

[Cherif et al., 2019] Cherif, G., Leclercq, E., Lefebvre, D., (2019), « Generation Filtered Beam Search algorithm for the scheduling of hybrid FMS using T-TPN », 18th European Control Conference (ECC), Napoli, Italy, pp. 3225-3230.

[Cherif, 2021] Cherif, G., (2021), « Ordonnancement dans les ateliers hybrides en environnement incertain », Université Le Havre, Normandie.

[Claude Elwood, 1948] Claude Elwood, S., (1948), « A Mathematical Theory of Communication », *Bell System Technical Journal*, Vol. 27, No.4, pp. 623–666.

[Colorni et al., 1991] Colorni, A., Dorigo, M., Maniezzo, V. (1991), « Distributed Optimization by Ant Colonies », Proceedings of the European Conference on Artificial Life (ECAL'91), Amsterdam, pp. 134-142.

[Colorni et al., 1992] Colorni, A., Dorigo, M., Maniezzo, V. (1992), « An Investigation of some Properties of an Ant Algorithm », In Reinhard Männer & Bernard Manderick, pp. 515-526

[Coltin & Veloso, 2010] Coltin, B., Veloso, M., « Mobile robot task allocation in hybrid wireless sensor networks », IEEE International Conference on Intelligent Robots and Systems (IROS), Taipei, Taiwan, pp. 2932–2937.

[Cook, 1971] Cook, S.A., (1971), « The complexity of theorem proving procedures », Association of Computing Machinery, Proceedings of the third annual ACM Symposium on the Theory of Computing, United States, pp. 151-158.

[Cordeau et al., 2001] Cordeau, J., Laporte, G., Mercier, A. (2001), « A unified tabu search heuristic for vehicle routing problems with time windows », *The Journal of the Operational Research Society*, Vol. 52, No. 8, pp.928-936.

[Cordeau et al., 2002] Cordeau, J., Desaulniers, G., Desrosiers, J., Solomon, M., Soumis, F. (2002), « VRP with Time Windows », *The Vehicle Routing Problem*, pp. 157-193.

[Cordeau et al., 2005] Cordeau, J. F., Laporte, G., Savelsbergh, M. W.P., Vigo, D., (2005) « Chapter 6 Vehicle Routing », *Handbooks in Operations Research and Management Science*, Vol. 14, Elsevier, 2007, pp. 367-428.

[Croce et al., 2004] Croce, F. D., Ghirardi, M., Tadei, R., (2004), « Recovering Beam Search: Enhancing the Beam Search Approach for Combinatorial Optimization Problems », *Journal of Heuristics*, Vol. 10, No. 1, pp. 89-104.

[Dantzig et al., 1954] Dantzig, G. B., Fulkerson, R., Johnson, S., (1954), « Solution of a Large-Scale Traveling-Salesman Problem », *Journal of the Operations Research Society of America*, Vol. 2, No 4, pp. 393-410.

[Dantzig & Ramser, 1959] Dantzig, G. B., Ramser, J. H., (1959), « The Truck Dispatching Problem », *Management Science*, Vol. 6, No 1, pp. 80-91.

[Delvosalle et al., 1998] Delvosalle, C., Fievez, C., Benjelloun, F., (1998), « Development of a methodology for the identification of potential domino effects in « Seveso » industries », Seminar on" Sortware Tools Relevant to the Seveso in Directive", Second Meeting of the Committee of Competent Authorities Responsible for the Implementation of Directive, Vol. 96, p. 82.

[De Ryck et al., 2020] De Ryck, M., Versteyhe, M., Debrouwere, F., (2020), « Automated guided vehicle systems, state-of-the-art control algorithms and techniques », *Journal of Manufacturing Systems*, Vol. 54, pp. 152–173.

[Dijkstra, 1959] Dijkstra, E.W., (1959), « A note on two problems in connexion with graphs », *Numerische Mathematik 1*, pp. 269–271.

[Djellal, 2016] Djellal, A., (2016), « Modélisation et Analyse des comportements d'une société d'agents mobiles à l'aide des réseaux de Petri », Thèse de doctorat, Université de Annaba.

[Dou et al. (2017)] Doua, L., Song, C., Wang, X., Liu, L., Feng, G., (2017), « Non uniform coverage control for heterogeneous mobile sensor networks on the line », *Automatica*, Vol. 81, pp. 464–470.

[Dunbabin & Marques, 2012] Dunbabin, M., Marques, L., (2012), « Robotics for Environmental Monitoring », *IEEE Robotics & Automation Magazine*, Vol. 19, No. 1, pp. 24-39.

[Elmogy, 2010] Elmogy, A. M., Khamis. A. M., Karray. F., (2010), « Market-based framework for mobile surveillance systems », International Conference on Autonomous and Intelligent Systems, Berlin, Heidelberg, vol. 7326, pp 69–78.

[Elwood, 1948] Elwood, S. C, (1948), « A Mathematical Theory of Communication », *Bell System Technical Journal*, Vol. 27, No. 4, pp. 623–666.

[Espina et al., 2011] Espina, M.V., Grech, R., De Jager, D., Remagnino, P., Iocchi, L., Marchetti, L., King, C., (2011), « Multi-robot teams for environmental monitoring », *Innovations in Defence Support Systems*, Vol. 336, pp. 183–209.

[Esquirol & Lopez, 1999] Esquirol, P., Lopez, P., (1999), « L'ordonnancement », Economica, Paris.

[Filliat, 2011] Filliat, D., (2011), « Robotique Mobile. Engineering school. Robotique Mobile », ENSTA ParisTech, 2011, pp.175.

[Gam et al., 2020] Gam, M., Lefebvre, D., Telmoudi, A J., Nabli, L., (2020), « Configuration of surveillance patrols with Petri nets for safety issues », Conference on Control, Decision and Information Technologies (CODIT), Prague, pp. 427-432.

[Gam et al., 2021a] Gam, M., Lefebvre, D., Nabli, L., Telmoudi, A. (2021), « A Petri Nets Based Approach for the Optimization of Surveillance Patrols », *International Journal of Sensor Networks*, Vol. 36, No. 4, pp. 181 - 193.

[Gam et al., 2021b] Gam, M., Lefebvre, D., Nabli, L., Telmoudi, A. (2021), « Optimization of maintenance patrols planning », Mediterranean Conference on Control and Automation (MED), Puglia, Italy, pp. 1299 - 1304.

[Gam et al., 2022a] Gam, M., Telmoudi, A.J., Lefebvre, D., (2022), « Hybrid Filtered Beam Search algorithm for the optimization of monitoring patrols », *Journal of Intelligent & Robotic Systems*, Soumis, en révision.

[Gam et al., 2022b] Gam, M., Hayane, O., Lefebvre, D., (2022), « Optimization of a two-stage logistic system using Hybrid Filtered Beam Search algorithm », Industrial Simulation Conference (ISC'2022), Dublin, Ireland, accepté.

[Gargouri, 2003] Gargouri, E., (2003), « Ordonnancement coopératif en industries agroalimentaires », Thèse de Doctorat, Université des Sciences et Technologies de Lille 1.

[Gerkey & Mataric, 2003] Gerkey, B., Mataric, M., (2003), « A framework for studying multi-robot task allocation », Proceedings of the NRL Workshop on Multi-Robot Systems, Washington, USA.

[Gerkey & Matarié, 2004] Gerkey, B. P., Matarié, M. J., (2004), « A formal analysis and taxonomy of task allocation in multi-robot systems », *International Journal of Robotics Research*, Vol. 23, No. 9, pp. 939-954.

[Ghoseiri & Ghannadpour, 2010] Ghoseiri, K., Ghannadpour, S. F. (2010), « Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm », *Applied Soft Computing Journal*, Vol. 10, No. 4, pp. 1096–1107.

[Giordani et al., 2010] Giordani, S., Lujak, M., Martinelli, F., (2010), « A distributed algorithm for the multi-robot task allocation problem », International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Berlin, Heidelberg, pp. 721–730.

[Giua, 1997] Giua, A., (1997), « Petri net state estimators based on event observation », IEEE Conference on Decision and Control, San Diego, CA, USA, pp. 4086–4091.

[Giua & Seatzu, 2002] Giua, A., Seatzu, C., (2002), « Observability of place/transition nets », *IEEE Transactions on Automatic Control*, Vol. 47, No. 9, pp. 1424 –1437.

[Giua et al., 2003] Giua, A., Julvez, J., Seatzu, C., (2003), « Marking estimation of Petri nets based on partial observation », Proceedings of the American Control Conference, Denver, CO, USA, pp. 7803-7896.

[Giua et al., 2007] Giua, A., Seatzu, C., Corona, D., (2007), « Marking estimation of Petri nets with silent transitions », *IEEE Transactions on Automatic Control*, Vol. 52, No. 9, pp. 1695–1699.

[Gockenbach & Borsi, 2008] Gockenbach, E., Borsi, H., (2008), « Condition monitoring and diagnosis of power transformers », International Conference on Condition Monitoring and Diagnosis, Beijing, China, DOI:10.1109/CMD.2008.4580427.

[Guiochet et al., 2017] Guiochet, J., Machin, M., Waeselynck, H., (2017), « Safety-critical advanced robots: A survey », *Robotics and Autonomous Systems*, Vol. 94, pp. 43–52.

[Han-Lim et al., 2009] Han-Lim, C., Brunet, L., How, J., (2009), « Consensus-based decentralized auctions for robust task allocation », *IEEE Transactions on Robotics*, Vol. 25, No. 4, pp. 912–926.

[He et al., 2009] He, H., Zhu, Z., Makinen, E., (2009), « A Neural Network Model to Minimize the Connected Dominating Set for Self-Configuration of Wireless Sensor Networks », *IEEE Transactions on Neural Networks*, Vol. 20, No. 6, pp. 973–982.

[Holland, 1975] Holland, J.H., (1975), « Adaptation in natural and artificial systems », Thèse de Doctorat, Thesis Michigan Press University, Ann Arbor, Michigan.

[Horling & Lesser, 2004] Horling, B., Lesser, V., (2004), « A survey of multi-agent organizational paradigms », *The Knowledge Engineering Review*, Vol. 19 , No. 4, pp. 281-316.

[Higuera & Dudek, 2013] Higuera, J., Dudek, G., (2013), « Fair subdivision of multi-robot tasks », IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, pp. 3014–3019.

[Huard et al., 2020] Huard, J., Gueudry, J., Leroy, J.-P., Joly, L.-M., Muraine, M., (2020), « Impact de l'incendie de l'usine Lubrizol à Rouen le 26 septembre 2019 sur la fréquentation des urgences ophtalmologiques », *Journal Français d'Ophtalmologie*, Vol. 44, No. 8, pp. 1121-1128.

[Jawhar et al. 2014] Jawhar, I., Mohamed, N., Al-Jaroodi, J., Zhang, S., (2014) « A Framework for Using Unmanned Aerial Vehicles for Data Collection in Linear Wireless Sensor Networks », *Journal of Intelligent & Robotic Systems volume*, Vol. 74, No. 1, pp. 437–453.

[Johnsen, 2012] Johnsen, S.O., (2012), « Resilience at interfaces: improvement of safety and security in distributed control systems by web of influence », *Information Management & Computer Security*, Vol. 20, No. 2, pp. 71-87.

[Jose & Pratihar, 2016] Jose, K., Pratihar, D. K., (2016), « Task allocation and collision-free path planning of centralized multi-robots system for industrial plant inspection using heuristic methods », *Robotics and Autonomous Systems*, Vol. 80, pp. 34–42.

[Kammarti et al., 2005] Kammarti, R., Hammadi, S., Borne, P., Ksouri, M., (2005), « Special Tabu Search in an Hybrid Evolutionary Approach for the PDPTW », IEEE International Conference, Orlando (USA), pp. 682-687.

[Kancir, 2018] Kancir, P., (2018), « Méthodologie de conception de système multi-robots : de la simulation à la démonstration », thèse de doctorat, Université de Bretagne Sud.

[Karakatič & Podgorelec, 2015] Karakatič, S., Podgorelec, V., (2015), « A survey of genetic algorithms for solving multi depot vehicle routing problem », *Applied Soft Computing Journal*, Vol. 27, pp. 519–532.

[Karray et al., 2008] Karray, A., Laabidi, K., Ksouri, M., (2008), « Métaheuristiques pour la commande des systèmes non linéaires », CIFA Conférence International Francophone d'Automatique, Roumanie.

[Kato et Wong, 2011] Kato, S., Wong, K. W., (2011), « Intelligent Automated Guided Vehicle Controller with Reverse Strategy », *Journal of Advanced Computational Intelligence and Intelligent Informatics*, Vol.15, No.3, pp. 304-312.

[Kefi, 2008] Kefi, K., (2008), « Optimisation Heuristique Distribuée du Problème de Stockage de Conteneurs dans un Port », Thèse de Doctorat, Ecole Centrale de Lille.

[Khamis et al., 2011] Khamis, A. M., Elmogy, A. M., Karray, F. O., (2011), « Complex Task Allocation in Mobile Surveillance Systems », *Journal of Intelligent & Robotic Systems*, Vol. 64, No. 1, pp. 33–55.

[Khamis & ElGindy, 2012] Khamis, A., ElGindy, A., (2012), « Minefield mapping using cooperative multirobot systems », *Journal of Robotics*, Vol. 2012, No. 2012, pp. 1-17.

[Khan, 2020] Khan, M. A., (2020) « Design and control of a robotic system based on mobile robots and manipulator arms for picking in logistics warehouses », thèse de doctorat, Université de la Normandie.

[Khosiawan et al., 2018] Khosiawan, Y., Khalfay, A., Nielsen, I., (2018), « Scheduling unmanned aerial vehicle and automated guided vehicle operations in an indoor manufacturing environment using differential evolution-fused particles warm optimization », *International Journal of Advanced Robotic Systems*, Vol. 15, No. 1, p. 172988141775414.

[Kirkpatrick et al., 1983] Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., (1983), « Optimization by simulated annealing », *Science*, Vol. 220, No. 4598, pp. 671-680.

[Korsah et al., 2013] Korsah, G.A., Stentz, A., Dias, M.B., (2013), « A comprehensive taxonomy for multi-robot task allocation », *The International Journal of Robotics Research*, Vol. 32, No. 12, pp. 1495–1513.

[Kriaa et al., 2015] Kriaa, S., Cambacedes, L. P., Bouissou M., Halgand Y., (2015), « A survey of approaches combining safety and security for industrial control systems », *Reliability Engineering and System Safety*, Vol. 139, No. 2015, pp. 156–178.

[Kuhn, 1955] Kuhn, H. W., (1955), « The Hungarian Method for the Assignment Problem ». *Naval Research Logistics Quarterly*, Vol. 2, No. 1-2, pp. 83-97.

[Kuipers et Byun, 1991] Kuipers, B. J., Byun, Y. T., (1991), « A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations », *Robotics and Autonomous Systems*, Vol. 8, No. 1-2, pp. 47–63.

[Labbassen, 2010] Labbassen, D., (2010), « Dissémination des données dans les réseaux de capteurs sans fil », Thèse de doctorat, Université des Sciences et de la Technologie Houari Boumediene.

[Lagoudakis et al., 2005] Lagoudakis, M. G., Markakis, E., Kempe, D., Keskinocak, P., Koenig, S., Kleywegt, A., Tovey, C., Meyerson, A., Jain, S., (2005), « Auction-based multi-robot routing », Proceedings of Robotics: Science and Systems Conference, Cambridge, MA, USA.

[Lai et al., 2020] Lai, J., Ren, Z., Wu, Z., Liu, Y., Xie, S., (2020), « Deep Neural Network-Based Real-time Trajectory Planning for an Automatic Guided Vehicle with Obstacles », Automation Congress (CAC), Shanghai, Chinese, pp. 6311-6316.

[Lampe, 2006] Lampe, A., (2006), « Méthodologie d'évaluation du degré d'autonomie d'un robot mobile terrestre », Thèse de doctorat, Institut National Polytechnique de Toulouse - INPT.

[Lamine et al., (2020)] A. Lamine, Mguis, F., Snoussi, H., Ghedira, K., (2020), « Linear models for total coverage problem with connectivity constraints using multiple unmanned aerial vehicles », *International Journal of sensors Networks*, Vol. 34, No.1, pp.15 – 25.

[Laporte, 1992a] Laporte, G., (1992), « The Travelling Salesman Problem », *European Journal of Operational Research*, Vol. 59, No. 2, pp. 231–247.

[Laporte, 1992b] Laporte, G., (1992), « The vehicle routing problem: An overview of exact and approximate algorithms », *European journal of operational research*, Vol. 59, No. 3, pp. 345-358.

[Lawler, 1966] Lawler, E. L., Wood, D. E., (1966), « Branch-and-Bound Methods: A Survey », *Operations Research*, Vol. 14, No 4, pp. 699-719.

[Lefebvre & Basile, 2021] Lefebvre, D., Basile, F., (2021), « An approach based on timed Petri nets and tree encoding to implement search algorithms for a class of scheduling problems », *Information Sciences*, Vol. 559, pp. 314-335.

[Lefebvre & Mejía, 2018] Lefebvre, D., Mejía, G. (2018), « Robust scheduling in uncertain environment with Petri nets and beam search », IFAC-Papers On Line, Vol 51, No. 11, pp 1077-1082.

[Le Pape, 1995] Le Pape, C., (1995), « Three mechanisms for managing resource constraints in the library for constraint-based scheduling », INRIA/IEEE Conference on Emerging Technologies and Factory Automation, Paris, France, pp. 980-995.

[Li et al., 2013] Li, L., Hasjicostis, C. N., (2013), « Minimum Initial Marking Estimation in Labeled Petri Nets », *IEEE Transactions on Automatic Control*, Vol. 58, No. 1, pp.198 - 203.

[Li et Kara., 2017] Li, W., and Kara, S., (2017), « Methodology for Monitoring Manufacturing Environment by Using Wireless Sensor Networks (WSN) and the Internet of Things (IoT) », *Procedia CIRP*, Vol. 61, pp. 323 – 328.

[Liao et Su, 2011] Liao, Y.L., et Su, K.L., (2011), « Multi-robot-based intelligent security system », Int. Symposium on Artif Life and Robotics, Oita, Japan, Vol. 16, No. 2, pp. 137-141.

[Liouane, 1998] Liouane, N., (1998), « Contribution à l'élaboration d'un outil d'aide à la décision pour l'ordonnancement de production en environnement incertain », Thèse de Doctorat, Université des Sciences et Technologies de Lille1.

[Liu et al., 2019] Liu, Y., Liu, Z., Shi, J., Wu, G., and Chen, C., (2019), « Optimization of Base Location and Patrol Routes for Unmanned Aerial Vehicles in Border Intelligence, Surveillance, and Reconnaissance », *Journal of Advanced Transportation*, Vol. 2019, p. 1-13.

[Liu et al., 2021] Liu, J., Liu, Z., Zhang, H., Yuan, H., Manokaran, K. B., Maheshwari, M., (2021), « Multi-sensor information fusion for IoT in automated guided vehicle in smart city », Soft Computing, Vol. 25, No. 18, pp.12017–12029.

[Liu & Kroll, 2012] Liu, C., & Kroll, A., (2012), « A centralized multi-robot task allocation for industrial plant inspection by using A* and genetic algorithms », International Conference on Artificial Intelligence and Soft Computing, Berlin Heidelberg, Vol. 7268, pp.466–474.

[Lopez & Esquirol, 1999] Lopez, P., Esquirol, P., (1999), « L'ordonnancement », Éditions Économia.

[López et al., (2013)] López, J., Pérez, D., Paz, E., Santana, A., (2013), « A building maintenance and surveillance system based on autonomous robots », *Robotics and Autonomous Systems*, Vol. 61, No. 12, pp. 1559–1571.

[Louis et al., 1999] Louis. S. J., Yin, X., Yuan, Z. U., (1999), « Multiple Vehicle Routing With Time Windows Using Genetic Algorithms », Proceedings of the 1999 Congress on Evolutionary Computation-CEC99, Washington, DC, USA, DOI: 10.1109/CEC.1999.785493.

[Lowerre, 1976] Lowerre, B., (1976), « The Harpy Speech Recognition System », Thèse de Doctorat, Carnegie Mellon University.

[Luo et al., 2015] Luo, J.C., Xing, K., Zhou, M.C., Li, X.L., Wang, X.N., (2015), « Deadlock-Free Scheduling of Automated Manufacturing Systems Using Petri Nets and Hybrid Heuristic Search », *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Vol 45, No. 3, pp. 530-541.

[Maha & Ahmed, 2020] Maha, A., Ahmed, Z., (2020), « Genetic algorithms for the multiple travelling salesman problem », *International Journal of Advanced Computer Science and Applications*, Vol. 11, No. 7, pp. 553–560.

[Mancel, 2004] Mancel, C., (2004), « Modélisation et résolution de problèmes d'optimisation combinatoire issus d'applications spatiales », INSA de Toulouse.

[Marino et al., 2013] Marino, A., Parker, L.E., Antonelli, G., Caccavale, F., (2013), « A decentralized architecture for multirobot systems based on the null-space-behavioral control with application to multi-robot border patrolling », Journal of Intelligent & Robotic Systems, vol. 71, pp. 423–444.

[Mejía & Lefebvre, 2019] Mejía, G., Lefebvre, D., (2019), « Robust scheduling of flexible manufacturing systems with unreliable operations and resources », *International Journal of Production Research*, Vol. 58, No. 21, pp. 6474-6492.

[Mejía & Lefebvre, 2020] Mejía, G., Lefebvre, D., (2020), « Robust scheduling of flexible manufacturing systems with unreliable operations and resources », *International Journal of Production Research*, Vol. 58, No.21, pp. 6474–6492.

[Mejía & Montoya, 2009] Mejía, G., Montoya, C., (2009), « Scheduling manufacturing systems with blocking: A Petri net approach », *International Journal of Production Research*, Vol. 47, No.22, pp. 6261-6277.

[Mejía & Niño, 2017] Mejía, G., Niño, K., (2017), « A new Hybrid Filtered Beam Search algorithm for deadlock-free scheduling of flexible manufacturing systems using Petri Nets », *Computers & Industrial Engineering*, Vol 108, pp. 165-176.

[Mejía & Odrey, 2005] Mejía, G., Odrey, N. G., (2005), « An approach using Petri nets and improved heuristic search for manufacturing system scheduling », *Journal of Manufacturing Systems*, Vol. 24, No. 2, pp. 462- 476.

[Mellouli et al., 2004] Mellouli, K., El Kamel, A., Borne, P., (2004), « Programmation linéaire et applications. Eléments de courset exercices résolus », Editions Technip.

[Mesghouni, 1999] K. Mesghouni, (1999), « Application des algorithmes évolutionnistes dans les problèmes d'optimisation en ordonnancement de la production », Thèse de Doctorat, Université des Sciences et Technologies de Lille 1.

[Metropolis et al., 1953] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., Teller, E., (1953), « Equation of state calculations by fast computing machines », *The journal of chemical physics*, Vol. 21, No. 6, pp. 1087-1092.

[Mohamad et al., 2018] Mohamad, N. R., Fauadi, M. H. F. M., Zainudin, S. F., Noor, A. Z. M., Jafar, F. A., Ali, M. M., (2018), « Optimization of Material Transportation Assignment for Automated Guided Vehicle (AGV) System », *International Journal of Engineering &Technology*, Vol. 7, No. 3.20, pp. 334-338.

[Munkres, 2004] Munkres, J., (2004), « Algorithms for the Assignment and Transportation Problems », *Journal of the Society for Industrial and Applied Mathematics*, Vol. 5, No. 1, pp. 32-38.

[Murata, 1989] Murata, T., (1989), « Petri nets: Properties, analysis and applications », *Proceedings of the IEEE*, Vol. 77, No. 4, pp. 541_580.

[Nagar & Tawfik, 2007] Nagar, A., Tawfik, H., (2007), « A Multi-Criteria Based Approach to Prototyping Urban Road Networks », *Issues in Informing Science & Information Technology*, Vol. 4, pp. 749-756.

[Nagatani et al., 2011] Nagatani, K., Okada, Y., Tokunaga, N., Kiribayashi, S., Yoshida, K., Ohno, K., Takeuchi, E., Tadokoro, S., Akiyama, H., Noda, I., Yoshida, T., Koyanagi, E., (2011), « Multirobot Exploration for Search and Rescue Missions : A Report on Map Building in RoboCup Rescue 2009 », *Journal of Field Robotics*, Vol. 28, No. 3, pp. 373–387.

[Nallusamy et al., 2009] Nallusamy, R., Duraiswamy, K., Dhanalaksmi, R., Parthiban, P. (2009), « Optimization of multiple vehicle routing problems using approximation algorithms », *International Journal of Engineering Science and Technology*, Vol. 1, No. 3, pp. 129-135.

[Negre, 2021] Negre, E., (2021), « Crisis management and distrust: Study of an industrial accident in France », Proceedings of the 54th Hawaii International Conference on System Sciences (HICSS), Hawaii, pp. 2235 – 2244.

[Nishi et Tanaka, 2012] Nishi, T., et Tanaka, Y., (2012), « Petri Net Decomposition Approach for Dispatching and Conflict-Free Routing of Bidirectional Automated Guided Vehicle Systems », *IEEE Transactions on systems, man, and cybernetics*, Vol. 42, No. 5, pp. 1230-1243.

[Nishikawa et al., 2016] Nishikawa, N., Suzuki, R., Arita, T., (2016), « Coordination control design of heterogeneous swarm robots by means of task-oriented optimization », *Artificial Life and Robotics*, Vol. 21, pp. 57–68.

[Nobuhiro et al., 2000] Nobuhiro, A., Izumi, K., Hui-Hsiung, K., (2000), « Bell numbers, log-concavity, and log-convexity », *Acta Applicandae Mathematica*, Vol. 63, No. (1–3), pp. 79–87.

[Nunes et al., 2017] Nunes, E., Manner, M., Mitiche, H., Gini, M., (2017), « A taxonomy for task allocation problems with temporal and ordering constraints », *Robotics and Autonomous Systems*, Vol. 90, pp. 55–70.

[Ow & Morton, 1988] Ow, P. S., Morton, T. E., (1988), « Filtered beam search in scheduling », *International Journal of Production Research*, Vol. 26, No. 1, pp. 35–62.

[Öztürk & Kuzucuoğlu, 2015] Öztürk, S., Kuzucuoğlu, A. E., (2015), « Optimal bid valuation using path finding for multi-robot task allocation », *Journal of Intelligent Manufacturing*, Vol. 26, No. 5, pp. 1049–1062.

[Paltrinieri et Renier, 2017] Paltrinieri, N., et Renier, G., (2017), "Dynamic risk analysis for Seveso sites", *Journal of Loss Prevention in the Process Industries*, Vol. 49, pp. 111-119.

[Papadimitriou, 1994] Papadimitriou. C., (1994), « Computational Complexity », AddisonWesley.

[Pentico, 2007] Pentico, D. W., (2007), « Assignment Problems: A Golden Anniversary Survey ». *European Journal of Operational Research*, Vol. 176, No. 2, pp. 774-793.

[Ping-an et al., 2009] Ping-an, G., Zi-xing, C., Ling-li, Y., (2009), « Evolutionary computation approach to decentralized multirobot task allocation », International Conference on Natural Computation (ICNC), Tianjian, China, pp. 415–419.

[Pruul et al., 1988] Pruul, E. A., Nemhauser, G.L., Rushmeier, R. A., (1988), « Branch-and-Bound and Parallel Computation: A Historical Note », *Operations Research Letters*, Vol. 7, No 2, pp. 65-69.

[Quarta et al., 2017] Quarta, D., Pogliani, M., Polino, M., Maggi, F., Zanchettin, A. M., Zanero S., (2017), « An Experimental Security Analysis of an Industrial Robot Controller », IEEE Symposium on Security and Privacy, San Jose, CA, USA, pp. 268-286.

[Rabin, 2000] Rabin, S., (2000), « A* Speed Optimizations », In Mark Deloura Game Programming Gems, pp. 272-287.

[Rachid, 2010] Rachid, M. H., (2010) « Les problèmes de tournées de véhicules en planification industrielle : classification et comparaison d'opérateurs évolutionnaires », Thèse de doctorat, Université d'Alep.

[Ralphs, 2003] Ralphs, T. K., (2003) « Parallel Branch and Cut for Capacitated Vehicle Routing », *Parallel Computing*, Vol. 29, No. 5, pp. 607-629.

[Ren et al., 2019] Ren, H., Zhao, Y., Xiao, W., Hu, Z., (2019), « A review of UAV monitoring in mining areas: current status and future perspectives », International Journal of Coal Science & Technology, vol. 6, No. 3, pp. 320–333.

[Ruangpayoongsak et al., 2005] Ruangpayoongsak, N., Roth, H., Chudoba, J., (2005) « Mobile Robots for Search and Rescue », International Workshop on Safety, Security and Rescue Robotics, Kobe, Japan, pp. 212-217.

[Russell & Norvig, 1995] Russell, S., Norvig, P. (1995), « Artificial Intelligence: A Modern Approach », Prentice Hall.

[Sabuncuoglu & Bayiz, 1999] Sabuncuoglu, I., Bayiz, M., (1999), « Job shop scheduling with beam search », *European Journal of Operational Research*, Vol. 118, No.2, pp. 390–412.

[Saeedvand et al., 2019] Saeedvand, S., Aghdasi, H. S., Baltes, J., (2019), « Robust multi-objective multi-humanoid robots task allocation based on novel hybrid metaheuristic algorithm », *Applied Intelligence*, Vol. 49, No. 12, pp. 4097–4127.

[Sakarovitch, 1984] Sakarovitch, M., (1984), « Graphes et Programmation Linéaire », Edition Hermann, Paris.

[Sanan and al., 2013] Sanan, S., Jain, L., Kappor, B., (2013), « Shortest path Algorithm », *International Journal of Application or Innovation in Engineering & Management*, Vol. 2, No. 7, pp. 316-320.

[Sarin et al., 2005] Sarin, S. C., Sherali, H. D., Bhootra, A., (2005), « New tighter polynomial length formulations for the asymmetric traveling salesman problem with and without precedence constraints », *Operations Research Letters*, Vol. 33, No. 1, pp. 62–70.

[Scheepers & Engelbrecht, 2016] Scheepers, C., Engelbrecht, A.P., (2016), « Training multi-agent teams from zero knowledge with the competitive coevolutionary team-based particle swarm optimiser », *Soft Computing*, Vol. 20, No. 2, pp. 607–620.

[Schneider, 2018] Schneider, E., (2018), « Mechanism Selection for Multi-Robot Task Allocation », thèse de doctorat, University of Liverpool.

[Shelkamy et al., 2020] Shelkamy, M., Elias, C. M., Mahfouz, D. M., Shehata, O. M., (2020), « Comparative Analysis of Various Optimization Techniques for Solving Multi-Robot Task Allocation Problem », Novel Intelligent and Leading Emerging Sciences Conf, Giza, Egypt, pp. 538–543.

[Shi et al., 2022] Shi, W., He, Z., Tang, W., Liu, W., Ma, Z., (2022), « Path Planning of Multi-Robot Systems With Boolean Specifications Based on Simulated Annealing ». *IEEE Robotics and Automation Letters*, Vol. 7, No. 3, pp. 6091-6098.

[Shiomi et al., 2013] Shiomi, M., Kamei, K., Kondo, T., Miyashita, T., Hagita, N., (2013), « Robotic service coordination for elderly people and caregivers with ubiquitous network robot platform », IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO), Tokyo, Japan, pp. 57-62.

[Shkurti et al., 2012] Shkurti, F., Xu, A., Meghjani, M., GamboaHiguera, J.C., Girdhar, Y., Giguere, P., Dudek, G., (2012), « Multi-domain monitoring of marine environments using a heterogeneous robot team », IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, pp. 1747–1753.

[Silva, 2013] Silva, M., (2013), « Half a century after Carl adam Petri's Phd thesis: A perspective on the field », *Annual Review in Controls*, Vol. 37, No. 2, pp. 191-192.

[Simmons et al., 2000] Simmons, R., Apfelbaum, D., Fox, D., Goldman, R. P., Haigh, K. Z., Musliner, D. J., Pelican, M., Thrun, S., (2000), « Coordinated deployment of multiple, heterogeneous robots », IEEE International Conference on Intelligent Robots and Systems (IROS), Takamatsu, Japan, Vol. 3, pp. 2254-2260 .

[Solnon, 2008] Solnon, C., (2008), « Optimisation par colonies de fourmis », Hermès science publications-Lavoisier.

[Soltani et al., 2002] Soltani, A. R., Tawfik, H., Fernando, T., (2002), "A multi-criteria based path finding application for construction site layouts", in Information Visualisation, Proceedings. Sixth International Conference, pp. 779-784.

[Sundar & Rathinam, 2015] Sundar, K., Rathinam, S., (2015), « An exact algorithm for a heterogeneous, multiple depot, multiple traveling salesman problem », International Conference on Unmanned Aircraft Systems, Denver, Colorado, USA, DOI: 10.1109/ICUAS.2015.7152311.

[Stelzenmuller et al., 2013] Stelzenmuller, V., Lee, J., South, A., Foden, J., Rogers, S. I., (2012), « Practical tools to support marine spatial planning : A review and some prototype tools », *Marine Policy*, Vol. 38, pp. 214-227.

[Taillard et al., 2001] Taillard, E. D., Gambardella, L. M., Gendreau, M., Potvin, J. Y., (2001), « Adaptive memory programming: a unified view of metaheuristics », *European Journal of Operational Research*, Vol. 135, No. 1, pp. 1-16.

[Tan et al., 2006] Tan, K. C., Chew, Y. H., Lee, L. H. (2006), « A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows », *Computational Optimization and Applications*, Vol. 34, No. 1, pp. 115–151.

[Tan et al., 2001] Tan, K. C., Lee, L. H., Ou, K. (2001), « Artificial intelligence heuristics in solving vehicle routing problems with time window constraints », *Engineering Applications of Artificial Intelligence*, Vol. 14, No. 6, pp. 825-837.

[Tanner, 2017] Tanner, H.G., (2007), « Switched UAV-UGV Cooperation Scheme for Target Detection », IEEE Int. Conference on Robotics and Automation, Rome, Italy, pp. 3457-3462.

[Taylor, 1967] Taylor, B. L., (1967), « Sequential Machines and Automata Theory », John Wiley and Sons, New York, pp. 592.

[Thrun, 1999] Thrun, S., (1999), « Learning metric-topological maps for indoor mobile robot navigation », *Artificial Intelligence*, Vol. 99, No. 1, pp. 21–71.

[Tillmann & Ney, 2018] Tillmann, C., Ney, H., (2018), « Word Reordering and a Dynamic Programming Beam Search Algorithm for Statistical Machine Translation », Computational Linguistics, Vol. 29, No. 1, pp. 97-133.

[Tixier et al., 2002] Tixier, J., (2002), « Méthodologie d'évaluation du niveau de risque d'un site industriel de type Seveso, basée sur la gravité des accidents majeurs et la vulnérabilité de l'environnement », Thèse de doctorat, Université Aix Marseille.

[Toth & Vigo, 2002] Toth, P., Vigo, D. (2002), « The vehicle routing problem », SIAM Monographs on Discrete Mathematics and Applications.

[Turner et al., 2018] Turner, J., Meng, Q., Schaefer, G., Whitbrook, A., Soltoggio, A. (2018), « Distributed Task Rescheduling with Time Constraints for the Optimization of Total Task Allocations in a Multirobot System », *IEEE Transactions on Cybernetics*, Vol. 48, No. 9, pp. 2583–2597.

[Usmani et al., 2010] Usmani, Z. U. H., Alghamdi, F. A., Tariq, A., Puri, T. N., (2010), « Relative ranking - A biased rating », Innovations and Advances in Computer Sciences and Engineering, No. 5, pp. 25–29.

[Voronoï, 1908] Voronoï, G (1908), « New applications of continuous parameters to the theory of quadratic forms. First thesis. On some properties of perfect positive quadratic forms », *Journal für die Reine und Angewandte Mathematik*, Vol. 1908, No. 133, pp. 134–198.

[Wang et al., 2011] Wang, H., Yu, Y., Yuan, Q., (2011), « Application of Dijkstra algorithm in robot path-planning », international conference on mechanic automation and control engineering, Hohhot, pp. 1067-1069.

[Wang et al., 2016] Wang, J., Zhou, Y., Wang, Y., Zhang, J., Chen, C. L. P., Zheng, Z., (2016), « Multiobjective Vehicle Routing Problems with Simultaneous Delivery and Pickup and Time Windows: Formulation, Instances, and Algorithms », *IEEE Transactions on Cybernetics*, Vol. 46, No. 3, pp. 582–594.

[Wu et al., 2011] Wu, k. C., Ting, c. J., Lan, w. C., (2011), « A Beam Search Heuristic for the Traveling Salesman Problem with Time Windows », *Journal of the Eastern Asia Society for Transportation Studies*, Vol. 9, pp. 702–712.

[Wu et al., 2020] Wu, P. Xiao, F., Sha, C., Huang, H., Sun, L., (2020), « Trajectory Optimization for UAVs' Efficient Charging in Wireless Rechargeable Sensor Networks », *IEEE Transactions on Vehicular Technology*, vol. 69, No. 4, pp. 4207-4220.

[Wurm et al., 2013] Wurm, K. M., Dornhege, C., Nebel, B., Burgard, W., Stachniss, C., (2013), « Coordinating heterogeneous teams of robots using temporal symbolic planning », *Autonomous Robots*, Vol. 34, No. 4, pp. 277–294.

[Yakovlev et al., 2020] Yakovlev, K., Andreychuk, A., Rybecky, T., Kulich, M., (2020), « On the Application of Safe-Interval Path Planning to a Variant of the Pickup and Delivery Problem », Int. Conf on Informatics in Control, Automation and Robotics, pp. 521-528.

[Yao et al., 2018] Yao, F., Keller, A., Ahmad, M., Ahmad, B., Harrison, R., Colombo, A. W., (2018), « Optimizing the Scheduling of Autonomous Guided Vehicle in a Manufacturing Process », IEEE International Conf on Industrial Informatics, Porto, Portugal, pp. 264-269.

[Yi-Lin & Kuo-Lan ,2011] Yi-Lin, L., Kuo-Lan, S., (2011), « Multi-robot-based intelligent security system », *Artificial Life and Robotics*, Vol. 16, No. 2, pp. 137-141.

[Zhang & Smith, 2020] Zhang, Y., Smith, W., (2020), « Achieving Multi-Tasking Robots in Multi-Robot Tasks », IEEE International Conference on Robotics and Automation, Xi'an, China, pp. 8948-8954.

[Zhong et al., 2020] Zhong, M., Yang, Y., Sun, S., Zhou, Y., (2020), Postolache, O., Ge, Y. E., (2020), « Priority-based speed control strategy for automated guided vehicle path planning in automated container terminals », *Transactions of the Institute of Measurement and Control*, Vol. 42, No. 16, pp. 3079-3090.

Optimisation de la surveillance des sites industriels à risques par des robots mobiles

Résumé. Les accidents industriels ont de nombreuses conséquences sur le plan économique mais aussi humain et environnemental. Pour cette raison, la surveillance et les systèmes de contrôle automatisés, sont devenus nécessaires pour limiter les risques, améliorer la sécurité et éviter l'apparition des pannes les plus graves. Cette thèse vise à proposer une approche de surveillance systématique pour la configuration et la planification des missions de surveillance réalisées par des agents automatisés (AGV ou UAV) dans des applications variées. Elle étudie particulièrement le problème d'allocation des tâches de levée de doute en différents points de mesure, dans un contexte centralisé et hors-ligne. Différents types de robots mobiles, équipés de capteurs, sont utilisés comme des outils de collecte des mesures. L'optimisation de la mission doit être réalisée en respectant les contraintes opérationnelles liées à l'autonomie des agents mais aussi fonctionnelles, par exemple, dans certains cas, l'ordre des points de mesures à visiter. Pour résoudre ces problèmes, nous proposons une modélisation du problème et une méthodologie de résolution issues des systèmes à événements discrets (SED). Cette approche permet de coupler les problèmes de configuration et de planification des trajectoires en faisant apparaître la complexité exponentielle du problème. Pour réduire cette complexité et étudier des situations réelles, nous proposons une méthode d'optimisation combinatoire basée sur la recherche en faisceau filtrée hybride et sur l'algorithme de Dijkstra qui minimise le coût global de la mission. Le résultat de cette recherche permet d'optimiser le nombre de robots de chaque type, l'ensemble des mesures affectées à chaque agent et les trajectoires des robots.

Mots clés : Sécurité, mission de surveillance, réseau de Petri, théorie des graphes, optimisation, allocation des tâches, planification des trajectoires, recherche en faisceau, robots mobiles.

Optimization of the monitoring for industrial sites by means of mobile robots

Abstract. Industrial accidents have many economic, human and environmental consequences. For this reason, monitoring and automated control systems have become necessary to limit risks, improve safety and avoid the occurrence of the most serious events. This thesis aims to propose a systematic monitoring approach for the configuration and planning of surveillance patrols carried out by automated agents (AGV or UAV) in various applications. Our research particularly focuses on the problem of allocating tasks at different measurement points in a centralized and off-line context. Different types of mobile robots, equipped with sensors, are used to collect measurement. The optimization of the mission must be carried out while respecting the operational constraints mainly due to the autonomy of the agents but also functional constraints, for example in certain cases, the order of the measurement points to be visited. To solve these problems, we propose to model and solve the problem in the perspective of discrete event systems. This approach makes it possible to couple the problems of configuration and planning of trajectories by revealing the exponential complexity of the problem. To reduce this complexity and study real situations, we propose a combinatorial optimization method based on hybrid filtered beam search and Dijkstra's algorithm which minimizes the overall cost of the patrol. The result of this research makes it possible to optimize the number of robots of each type, the set of measurements assigned to each agent and the trajectories of the robots.

Keywords: Safety, surveillance patrol, Petri net, graph theory, optimization, task allocation, trajectory planning, beam search, mobile robots.