Coordination d'une flotte hétérogène de robots pour la récolte d'information dans un environnement inconnu

Alvin Gandois

Normandie Université

UNIVERSITÉ CAEN NORMANDIE

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité **INFORMATIQUE**

Préparée au sein de l'**Université de Caen Normandie**

Coordination d'une flotte hétérogène de robots pour la récolte d'information dans un environnement inconnu

Présentée et soutenue par

GANDOIS ALVIN

Thèse soutenue le 11/12/2024

devant le jury composé de :

M. MOUADDIB ABDEL-ILLAH	Professeur des universités - Université de Caen Normandie (UCN)	Directeur de thèse
M. AL FALOU AYMAN	Directeur de recherche - INST SUP D'ELECTRONIQUE, DU NUMERIQUE	Co-directeur de thèse
M. SABBADIN REGIS	Directeur de recherche à l'INRA - INRA DE TOULOUSE	Président du jury
M. LE GLOANNEC SIMON	Docteur - INST SUP D'ELECTRONIQUE, DU NUMERIQUE	Membre du jury
MME BEYNIER AURÉLIE	Maître de conférences - UNIVERSITE PARIS 6 PIERRE ET MARIE CURIE	Rapporteur du jury
M. CHARPILLET FRANCOIS	Directeur de recherche à l'INRIA - UNIVERSITE NANCY 1 HENRI POINCARE	Rapporteur du jury

Thèse dirigée par **MOUADDIB ABDEL-ILLAH** (Groupe de recherche en informatique, image et instrumentation de Caen) et **AL FALOU AYMAN** (INST SUP D'ELECTRONIQUE, DU NUMERIQUE)

MIIS

GREYC
Laboratoire de recherche en sciences du numérique

Remerciements

Je tiens à exprimer toute ma gratitude aux personnes qui m'ont accompagné durant ces trois années, à commencer par Abdel-Illah Mouaddib, mon directeur de thèse. Ses conseils, sa patience, ses encouragements et son expertise m'ont permis de réaliser cette thèse dans les meilleures conditions. Un grand merci également à Simon Le Gloannec pour sa disponibilité et ses nombreux conseils tant d'un point de vue scientifique qu'organisationnel. Je remercie également Ayman Al Falou pour la confiance qu'il m'a accordée et qui a rendu cette thèse possible. Merci à Aurélie Beynier et François Charpillet d'avoir accepté d'être rapporteurs de cette thèse et pour la qualité de leur rapport malgré les contraintes de temps, ainsi qu'à Régis Sabbadin d'avoir accepté de faire partie de mon jury mais aussi d'avoir suivi mes travaux durant ces trois années à l'occasion de mon comité de suivi de thèse. Je remercie par la même occasion Yann Mathet d'avoir fait partie de mon comité de suivi de thèse ainsi que de m'avoir encadré lors de mon stage de 3$^{\text{ème}}$ année de licence, et qui a contribué à me donner envie de faire de la recherche en informatique. Merci au laboratoire GREYC et plus particulièrement à l'équipe MAD de m'avoir accueilli depuis mon stage de master et pour les discussions enrichissantes lors des groupes de travail. Merci aux enseignants d'informatique de l'Université de Caen pour leur excellente formation, de la licence jusqu'au master, et tout particulièrement à Bruno Zanuttini et Grégory Bonnet qui ont également contribué à mon intérêt pour l'intelligence artificielle au travers de leurs cours et projets tutorés. Merci aux différents co-bureaux que j'ai pu côtoyer durant mon parcours : Mathias, Mihail, Josselin, Paul et Geoffrey. Merci également à l'Agence de l'Innovation de Défense de m'avoir offert l'opportunité de réaliser cette thèse. Pour terminer, je remercie ma mère et mon beau-père, qui m'ont toujours fait confiance dans mes choix, et à qui je dédie cette thèse.

Notice

Les travaux réalisés durant cette thèse ont été financés par l'Agence de l'Innovation de Défense (AID) et le l@bISEN et sont le fruit d'une collaboration entre le laboratoire GREYC, l'Université de Caen Normandie, le l@bISEN et l'AID.

Table des matières

Table des figures

Introduction

1 Contexte

Les robots autonomes permettent d'assister l'être humain pour accomplir des tâches complexes, répétitives ou dangereuses. En particulier, le problème de la récolte d'information dans un environnement inconnu est le problème qui consiste à, étant donné un robot autonome, permettre à ce robot de prendre des décisions de manière à récolter de l'information sur l'environnement qui l'entoure, que ce soit un moyen de l'aider à accomplir sa mission, ou directement son but premier, comme cela peut être le cas dans les problèmes d'évaluation de situation après une catastrophe naturelle ou industrielle, pour rechercher d'éventuelles victimes ou analyser la situation courante.

Afin de gagner en efficacité, il est souvent intéressant d'avoir recours à plusieurs robots, et dans ce cas se pose la question de la coordination de cette flotte de robots, qui n'est pas toujours évidente car chaque robot doit prendre des décisions de manière à éviter les conflits avec les autres robots tout en accomplissant la mission commune de la manière la plus efficace possible. De plus, des contraintes liées à l'environnement comme les coupures de communication rendent parfois cette coordination très compliquée à atteindre en pratique et nécessitent généralement de sacrifier de l'efficacité pour la sécurité et la stabilité du système.

Les systèmes multi-agents hétérogènes permettent de résoudre une plus grande variété de problèmes car nous pouvons faire appel à des robots polyvalents et complémentaires, qui ont chacun leur spécificité permettant de résoudre différentes parties du problème ou d'assister les autres robots dans des tâches pour lesquelles ils ne sont pas adaptés. Cependant, la coordination d'une flotte hétérogène de robots est plus complexe car chaque robot peut avoir une stratégie qui lui est propre, et il est plus difficile pour un robot d'anticiper la stratégie d'un autre robot qui n'a pas les mêmes capacités que lui.

Les outils formels comme les processus décisionnels de Markov (MDP) et leurs extensions à l'observabilité partielle et aux systèmes multi-agents (DEC-POMDP), ainsi que la théorie de l'information, permettent de formaliser de tels problèmes. Malheureusement, les solutions existantes pour les résoudre sont difficiles à mettre en œuvre en pratique dû à leur forte complexité algorithmique.

L'objectif de nos travaux est alors de proposer des modèles et algorithmes permettant de formaliser et résoudre de tels problèmes, c'est-à-dire de coordonner une flotte hétérogène de robots dans le but de récolter de l'information dans un environnement inconnu.

2 Thèse

Afin de résoudre le problème de la récolte d'information dans un environnement inconnu par une flotte hétérogène de robots, nous avons proposé un premier modèle, le Meta-MDP, qui est un modèle hiérarchique basé sur les processus décisionnels de Markov (MDP) permettant de calculer une méta-politique pour choisir quelle politique un agent doit appliquer, et plus particulièrement, allouer des politiques aux agents en fonction de leurs propres capacités et de l'état actuel des connaissances des agents sur leur environnement. La particularité de ce modèle est la fonction de transition pour laquelle nous devons être capables de déterminer la distribution de probabilité des états atteignables après que chaque agent ait exécuté sa politique allouée. Nous avons proposé un algorithme permettant de calculer cette distribution avec une profondeur bornée et, ce processus étant coûteux, nous avons également proposé un algorithme d'approximation basé sur un élagage par seuil probabiliste permettant de réduire les temps de calcul. Nous avons évalué ce modèle expérimentalement et avons montré des résultats proches de la solution optimale sur un problème simple.

Cependant, ce premier modèle repose sur des hypothèses fortes sur les problèmes que l'on peut résoudre, et en particulier nécessite une connaissance a priori de la disposition de l'environnement et des emplacements des points d'intérêts pour lesquels on souhaite récolter de l'information. Nous avons donc étendu ce modèle au cas où nous n'avons aucune connaissance a priori sur l'environnement, et où cette fois-ci nous avons un seul agent embarquant plusieurs capteurs hétérogènes lui permettant de cartographier son environnement tout en récoltant de l'information sur les points d'intérêts qu'il aura détectés. La difficulté supplémentaire de cette approche est qu'il est très difficile en pratique de résoudre ce problème hors ligne, c'est-à-dire de calculer une stratégie en amont de la mission afin de l'exécuter directement sur place. Nous proposons alors une stratégie où l'agent calcule régulièrement une nouvelle politique sur place durant la mission, ce qui fait que le temps de calcul est un critère important à prendre en compte afin de ne pas perdre trop de temps de mission à calculer. Nous avons évalué ce modèle sur un problème concret d'exploration et de récolte d'information et avons montré de meilleurs résultats que des stratégies simples couramment employées, avec un très bon passage à l'échelle.

Les travaux menés durant cette thèse ont fait l'objet de deux publications dans des conférences internationales à comité de lecture :

— Gandois, A., Mouaddib, A.-I., Le Gloannec, S., and Alfalou, A. (2024). A meta-mdp approach for information gathering heterogeneous multi-agent systems. In International Conference on Robotics, Computer Vision and Intelligent Systems, pages 345–360. Springer. (**Best Paper Award**)

— Gandois, A., Mouaddib, A.-I., Le Gloannec, S., and Alfalou, A. (2025). Exploration and Active Recognition Strategy using Meta-MDP. (Symposium on Applied Computing, track Intelligent Robotics and Multi-Agent Systems). To appear

3 Sommaire

Ce manuscrit est organisé en trois parties. Dans la partie I, nous dressons un état de l'art en commençant par une brève introduction à l'intelligence artificielle pour ensuite introduire la planification sous incertitude par le biais des Processus Décisionnels de Markov, et enfin les problématiques liées à la perception active et les solutions existantes. Dans la partie II, nous présentons nos contributions : un modèle permettant de faire de la récolte d'information multi-agents hétérogènes dans un environnement partiellement observable, ainsi qu'une généralisation mono-agent au cas où l'environnement est topologiquement inconnu. Dans la partie III, nous présentons les résultats expérimentaux que nous avons obtenus. Nous terminons par une conclusion et les perspectives nouvelles apportées par nos travaux.

Partie I : État de l'art

Dans cette partie, nous présentons les outils formels nécessaires à la bonne compréhension de ce document, ainsi que les solutions existantes liées à notre problématique de recherche.

— **Chapitre 1.** Ce chapitre introduit les concepts fondamentaux en intelligence artificielle : les agents et leur environnement ainsi que les systèmes multi-agents. Nous présentons également le problème de la planification qui est un problème au cœur de notre étude.

— **Chapitre 2.** Ce chapitre aborde plus en détail la problématique de la planification avec le cas général de la planification sous incertitude. Nous introduisons différents modèles de processus décisionnels de Markov allant du simple modèle mono-agent à observabilité parfaite aux modèles plus complexes multi-agents à observabilité partielle. Nous présen-

tons également les techniques couramment employées pour résoudre de tels modèles ainsi que leurs limites.

— **Chapitre 3.** Ce chapitre se concentre sur le problème de la perception active en définissant plus formellement la notion d'incertitude et comment la mesurer à l'aide de la théorie de l'information. Nous présentons également des approches existantes de perception active pour la récolte d'information et les difficultés liées aux systèmes multi-agents hétérogènes.

Partie II : Contributions

Dans cette partie, nous présentons les contributions principales de cette thèse.

— **Chapitre 4.** Nous présentons ici un modèle de récolte d'information multi-agents hétérogènes dans un environnement inconnu. Ce modèle, que nous avons nommé Meta-MDP, se base sur les processus décisionnels de Markov et fonctionne en deux parties : une partie récolte d'information sur un point d'intérêt, et une partie allocation de points d'intérêts aux agents. Nous présentons également un algorithme pour calculer la fonction de transition du Meta-MDP qui est plus compliquée que pour un MDP classique, ainsi qu'une méthode d'approximation utilisant un élagage par seuil probabiliste permettant de réduire les coûts en calcul.

— **Chapitre 5.** Dans ce chapitre, nous proposons une extension de notre modèle précédent au cas où l'environnement est topologiquement inconnu et les emplacements des points d'intérêts ne sont pas connus à l'avance. Ce nouveau modèle permet à un seul agent embarquant plusieurs capteurs hétérogènes de construire une carte de son environnement tout en récoltant de l'information sur les éventuels points d'intérêts détectés.

Partie III : Évaluation expérimentale

Nous terminons par une évaluation expérimentale des modèles que nous avons proposés durant cette thèse.

— **Chapitre 6.** Dans ce chapitre, nous évaluons notre modèle de récolte d'information multi-agents hétérogènes dans un environnement simple topologiquement connu et avec des points d'intérêts fixés à l'avance. Nous comparons le Meta-MDP à une approche optimale et une approche myope d'un point de vue de la qualité de la solution obtenue et nous proposons une étude plus approfondie de notre modèle en termes de temps de calcul.

— **Chapitre 7.** Nous nous intéressons dans ce chapitre à un problème plus concret de récolte d'information mono-agent avec des capteurs hétérogènes dans un environnement topologiquement inconnu dans le but de construire une carte de l'environnement tout en récoltant de l'information sur d'éventuels objets présents dans cet environnement. Nous utilisons pour cela le modèle présenté dans le chapitre 5 et le comparons à des approches classiques de perception passive et perception complète, et nous montrons les meilleures performances de notre modèle par rapport à ces approches.

Pour terminer, nous concluons par un résumé des travaux menés durant cette thèse, et proposons quelques perspectives de recherche possibles en lien avec ces travaux.

Première partie

État de l'art

Chapitre 1

Intelligence artificielle

Sommaire

1.1 Introduction

L'intelligence artificielle est une discipline qui vise à créer des systèmes capables de réaliser des tâches qui nécessitent normalement l'intelligence humaine. À l'origine une branche des mathématiques, l'intelligence artificielle est aujourd'hui en lien avec une multitude de domaines, allant de l'informatique aux sciences humaines et sociales en passant par la philosophie et les neurosciences. Dans cette section, nous allons présenter un bref historique de l'intelligence artificielle depuis sa création dans les années 1950 à nos jours, et les domaines de recherche et applications qui en ont découlé.

1.1.1 Histoire de l'Intelligence Artificielle

L'intelligence artificielle (IA) a parcouru un long chemin depuis ses débuts dans les années 1950. Cette discipline, qui vise à créer des machines capables de simuler l'intelligence humaine, a évolué à travers différentes phases de développement, chacune marquée par des avancées technologiques significatives et des défis toujours plus ambitieux.

L'histoire de l'intelligence artificielle voit ses débuts en 1950, avec la publication du fameux article *Computing Machinery and Intelligence* par Alan Turing [Turing, 1950]. Dans cet article, Turing propose le célèbre "Test de Turing", un critère qui permet de déterminer si une machine est capable d'imiter l'intelligence humaine.

Cependant, ce n'est qu'en 1956 que la discipline portant le nom d'« Intelligence Artificielle » verra officiellement le jour, lors d'une conférence organisée au Dartmouth College aux États-Unis par le professeur John McCarthy et ses collègues [McCarthy et al., 2006], et où les premiers travaux de recherche en intelligence artificielle seront développés. À cette occasion, Newell et Simon ont présenté ce qui est considéré aujourd'hui comme le premier programme d'intelligence artificielle, Logic Theorist [Newell and Simon, 1956], un programme de démonstration de théorèmes logiques.

Les années suivantes voient l'émergence de programmes d'intelligence artificielle ayant pour ambition la résolution universelle de problèmes, par exemple le General Problem Solver (GPS) [Newell and Simon, 1961]. Cependant, malgré des avancées notables comme le programme de compréhension du langage naturel ELIZA [Weizenbaum, 1966], l'enthousiasme initial est rapidement freiné par les limitations technologiques et théoriques de l'époque. La promesse d'une intelligence artificielle généraliste capable de rivaliser avec l'intelligence humaine s'avère prématurée.

Les années 1970 et 1980 sont marquées par les "hivers de l'IA", des périodes de désillusion et de réduction des financements en raison des attentes non satisfaites. Néanmoins, des progrès sont réalisés dans des domaines spécifiques, notamment les systèmes experts, qui utilisent des règles codifiées pour imiter le jugement d'un humain dans un domaine particulier. MYCIN, un système expert pour le diagnostic médical, est un exemple notable de cette période [Nilsson, 2009].

À partir des années 1990-2000, l'intelligence artificielle connaît une renaissance grâce à l'augmentation des capacités de calcul et à l'accumulation de grandes quantités de données. L'apprentissage automatique (*machine learning*), en particulier, devient un domaine de recherche actif. Des algorithmes comme les réseaux de neurones artificiels, initialement conceptualisés dans les années 1950, trouvent de nouvelles applications grâce à des techniques améliorées comme l'apprentissage profond (*deep learning*).

En 1997, le superordinateur Deep Blue d'IBM bat le champion du monde d'échecs Garry Kasparov, marquant une étape historique pour l'intelligence artificielle.

Depuis 2010, l'intelligence artificielle a fait des avancées spectaculaires, transformant de nombreux secteurs. L'apprentissage profond [Goodfellow et al., 2016], en particulier, a permis des progrès majeurs dans la reconnaissance d'images, le traitement du langage naturel et les jeux. En 2012, le réseau de neurones convolutifs AlexNet remporte la compétition annuelle de reconnaissance d'images ImageNet, démontrant la puissance des techniques d'apprentissage profond. Des systèmes comme AlphaGo de DeepMind, qui a battu le champion du monde de Go en 2016, illustrent la capacité des IA modernes à maîtriser des tâches complexes.

Au début des années 2020, l'intelligence artificielle a connu une exposition au grand public sans précédent avec l'émergence de l'IA générative, qui a montré des capacités impressionnantes dans la génération de texte, la traduction et d'autres tâches liées au langage [Brown et al., 2020], ce qui a ouvert de nouvelles possibilités pour l'automatisation et l'assistance intelligente.

Aujourd'hui, l'intelligence artificielle est intégrée dans des applications quotidiennes allant des assistants virtuels aux systèmes de recommandation, en passant par les voitures autonomes. Les recherches en IA continuent de progresser, explorant des domaines comme l'éthique de l'IA, la transparence des algorithmes et les impacts sociaux de l'automatisation. Nous allons maintenant présenter des grands domaines de recherche en intelligence artificielle et leurs applications.

1.1.2 Domaines de recherche et applications

Nombreux sont les domaines de recherche en lien avec l'intelligence artificielle, avec un large spectre d'applications possibles. Nous allons en présenter quelques-uns.

— **Les jeux** : Les jeux sont un cadre idéal pour étudier les systèmes d'intelligence artificielle car il existe une multitude de jeu avec des règles très diverses et variées, et car ils permettent de se confronter directement à l'intelligence humaine. Le succès le plus marquant a été la victoire de l'ordinateur Deep Blue d'IBM contre le champion d'échecs Garry Kasparov en 1997. Cet exploit a été réitéré en 2016, lorsque le programme AlphaGo de DeepMind a battu Lee Sedol, champion de Go, un jeu dont la combinatoire est beaucoup plus importante que celle des échecs et où il est beaucoup plus difficile d'évaluer une position. L'intelligence artificielle trouve également des applications dans les jeux vidéos dans le but de remplacer des personnages non joueurs scriptés afin de les rendre adaptables à une plus grande variété de situations.

— **Les robots autonomes** : L'intelligence artificielle est utilisée en robotique pour rendre les robots autonomes. Un robot autonome embarque des capteurs et des instruments de calcul afin de prendre des décisions dans le but d'accomplir une mission donnée. Les robots autonomes peuvent s'avérer utiles lorsqu'il s'agit d'accomplir des tâches répétitives, complexes ou trop dangereuses pour les humains, mais aussi pour assister les humains dans la vie de tous les jours. Un exemple célèbre de robot autonome est le robot Curiosity, développé par la NASA, et qui a été déposé sur la planète Mars en 2012 dans le but de faire de l'exploration spatiale et de récolter des données. Dix ans plus tard, cette mission est toujours en cours. On peut également trouver des robots autonomes pour des missions de reconnaissance et de surveillance, des robots guides dans les centres commerciaux, etc. Un exemple récent est le développement de voitures autonomes capables de se déplacer toutes seules dans la circulation sans mettre en danger les autres usagers de la route.

— **La planification logistique** : La planification logistique a pour objectif de gérer la mise en place d'une chaîne logistique ou l'ordonnancement de tâches qui sont interdépendantes. En 1991, l'outil d'analyse et de replanification dynamique DART a automatisé la programmation des transports de 50000 véhicules, bateaux et soldats. En 4 ans, DART a permis de rentabiliser l'ensemble des sommes investies par la DARPA dans la recherche en intelligence artificielle les 30 dernières années.

— **L'aide à la décision** : L'aide à la décision consiste en un ensemble de méthodes dans le but d'aider l'humain dans sa prise de décision. Elle a des impacts dans de nombreux domaines : la finance, la médecine, l'économie, etc. Par exemple les systèmes de recommandations d'une boutique en ligne peuvent se servir de l'activité passée d'un client pour lui suggérer de nouveaux articles. Dans l'économie, des systèmes d'aide à la décision sont utilisés en bourse pour aider les investisseurs à prendre des décisions d'achat et vente d'actions dans le but de maximiser leur profit ou minimiser le risque.

— **L'éthique et l'explicabilité** : Avec l'émergence des modèles dits « boîte noire » comme l'apprentissage profond, les décisions prises par les IA peuvent parfois être incomprises par les humains. Cependant, pour garantir un lien de confiance entre les IA et l'humain, il est nécessaire que l'on comprenne les décisions prises par les IA. L'IA explicable (XAI [1]) est le domaine de recherche qui étudie à rendre de tels systèmes compréhensibles, en développant des IA qui sont capables d'expliquer leurs décisions, par exemple les critères pris en comptes, les effets positifs d'une décision par rapport à une autre, etc. Plus généralement, des recherches sont menées en éthique de l'intelligence artificielle afin d'étudier les

1. eXplainable Artificial Intelligence

biais des algorithmes, la transparence des algorithmes, l'équité, la protection des données personnelles, l'imputabilité et le respect des réglementations.

En résumé, l'intelligence artificielle est un domaine de recherche en pleine expansion avec régulièrement de nouvelles avancées techniques, et qui trouve des applications dans une grande variété de domaines. Dans cette thèse, nous nous intéressons particulièrement à des problématiques de planification sous incertitude dans un système multi-agents hétérogènes pour la récolte d'information. Dans la suite de ce chapitre, nous allons introduire les principaux concepts de l'intelligence artificielle utiles à la bonne compréhension de ce manuscrit.

1.2 Définitions générales

Nous allons commencer par définir les notions de base lorsque l'on parle d'intelligence artificielle : les agents et leur environnement. Nous allons pour cela principalement nous appuyer sur l'un des ouvrages de référence en la matière, *Artificial Intelligence : A Modern Approach*, de Russell et Norvig [Russell and Norvig, 2020], originalement paru en 1995 et dont la 4$^{\text{ème}}$ édition a été publiée en 2020.

1.2.1 Agents

Il existe une multitude de définitions pour le concept d'agent, qui peuvent être adaptées selon le contexte dans lequel on se place et le problème que l'on cherche à résoudre. Nous concernant, nous utiliserons la définition proposée par Russell et Norvig :

Définition 1.2.1. Un agent est une entité capable d'interagir avec son environnement. L'interaction peut s'effectuer de deux manières différentes : **percevoir** l'environnement grâce à des capteurs, ou **agir** sur l'environnement grâce à des actionneurs.

Le processus d'interaction entre un agent et son environnement est illustré dans la figure 1.1. Cette définition permet d'englober toute sorte d'agents : un être humain est un agent qui évolue dans le monde réel, qui perçoit l'environnement notamment grâce à ses yeux et ses oreilles, et qui peut agir sur son environnement grâce à ses mains ou ses jambes. Un véhicule autonome est un agent qui évolue également dans le monde réel, qui perçoit son environnement grâce à des caméras embarquées, et qui peut agir sur l'environnement en actionnant ses moteurs pour faire tourner ses roues. Il existe également des agents artificiels qui évoluent dans un environnement virtuel, c'est le cas par exemple des robots d'indexation du web, qui perçoivent leur environnement en lisant le code source d'une page web, et qui agissent sur l'environnement en naviguant sur les hyperliens des pages déjà indexées.

Dans cette thèse, nous nous intéresserons au processus de décision des agents autonomes. En particulier, comment un agent peut prendre des décisions de manière à résoudre efficacement un problème, accomplir une tâche plus ou moins complexe, ou explorer son environnement à la recherche de connaissances. Par conséquent, nous parlerons d'agents dits « rationnels » qui optimisent une fonction objective ou d'utilité. L'utilité d'un agent est une fonction de l'ensemble des facteurs à prendre en compte au moment de la prise de décision dans le but de choisir la meilleure action qui lui permettra d'accomplir sa tâche.

1.2.2 Rationalité

Russell et Norvig définissent la rationalité de la manière suivante :

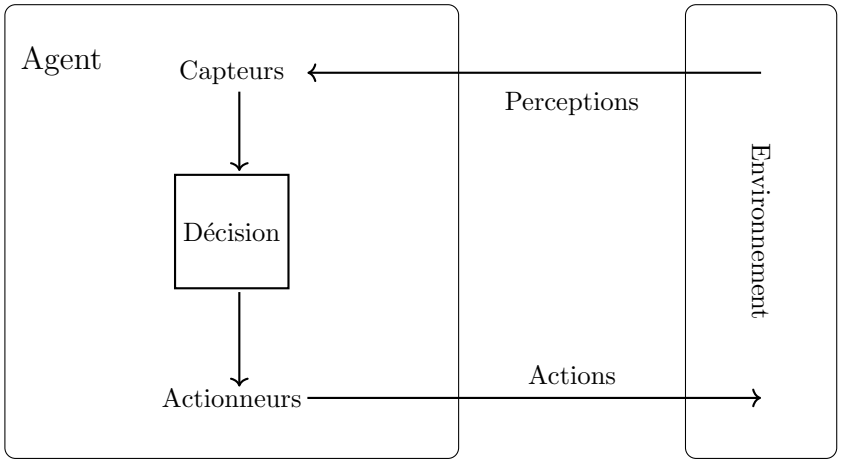


FIGURE 1.1 – Illustration du processus d'interaction entre un agent et son environnement.

Définition 1.2.2. Un agent rationnel est un agent qui prendra toujours une décision qui maximise son critère de performance, en fonction de ses connaissances actuelles sur l'environnement.

Par exemple, un agent dont le but est d'explorer son environnement, s'il est rationnel, évitera autant que possible de retourner à des endroits qu'il a déjà explorés, car il préférera se rendre dans des zones qu'il ne connaît pas encore afin de maximiser son utilité. Comme la définition le suggère, un agent peut être rationnel de son point de vue, mais pas forcément d'un point de vue extérieur. En effet, lorsqu'un agent décide d'une action à effectuer, il est rationnel selon ses croyances personnelles sur l'environnement. Cependant, d'un point de vue omniscient, la décision l'agent ne sera pas forcément rationnelle. Le critère de performance dépend de l'objectif donné à l'agent. Si l'on souhaite que l'agent accomplisse une certaine tâche, son critère de performance sera alors la réussite de cette tâche, et éventuellement d'autres critères tels que la rapidité d'exécution de la tâche. Si le but de l'agent est simplement d'explorer son environnement, on pourra mesurer la proportion d'environnement exploré.

Un agent peut adopter deux types de comportements : un comportement **réactif** ou un comportement **pro-actif**. Un agent réactif prendra des décisions sur le terrain en fonction de ses observations, tandis qu'un agent pro-actif planifiera à l'avance son exécution en anticipant les événements qui peuvent se produire. De ce fait, un agent pro-actif sera souvent plus performant et rationnel qu'un agent réactif, mais nécessitera une plus longue phase de préparation en amont de la mission.

Nous avons beaucoup abordé l'environnement d'un agent sans réellement définir de quoi il s'agit. Nous allons donc désormais définir concrètement ce qu'est un environnement et ses différentes propriétés.

1.2.3 Environnement

L'environnement correspond au monde dans lequel l'agent évolue ainsi que l'agent lui-même. Lorsque l'on souhaite modéliser un problème de prise de décision par un agent dans un environnement, on décrit généralement l'environnement comme l'ensemble des paramètres (variables d'état) utiles à la prise de décision de l'agent. Par exemple, un agent dont le but est de rejoindre une cible dans une zone n'aura besoin de connaître que sa propre position et la position de la cible, et un paramètre tel que la température ambiante n'est pas utile pour raisonner de manière rationnelle.

Un **état** de l'environnement est une instance des variables de l'environnement à un instant donné. L'état est généralement décomposé en deux parties : l'état interne et l'état externe. L'état interne représente l'état de l'agent lui-même (par exemple sa position), tandis que l'état externe représente l'état des variables de l'environnement (par exemple la présence ou non d'un obstacle). Lorsque l'agent est équipé de capteurs lui permettant d'observer l'ensemble des variables de l'environnement à chaque instant et de manière exacte, on dira que l'environnement est **complètement observable** et l'agent connaît donc l'état. Dans le cas contraire, si l'agent embarque des capteurs imprécis ou qui ne permettent pas d'observer l'environnement de manière globale, on dira que l'environnement est **partiellement observable** et l'agent ne connaît pas l'état mais il a une croyance sur l'état (distribution de probabilité sur l'ensemble des états possibles). Il est généralement plus difficile de travailler avec des environnements partiellement observables, car cela implique que l'agent garde en mémoire sa propre représentation de l'environnement à partir des observations qu'il a perçues.

Lorsque l'agent exécute une action dans l'environnement, l'état change. On parle de **transition** d'état. Si le nouvel état peut être complètement déterminé à partir de l'état courant et de l'action appliquée, on dira que l'environnement est **déterministe**. Sinon, l'environnement est **stochastique** et les transitions sont régies par une distribution de probabilité à partir de l'état courant et l'action courante. Par ailleurs, lorsque l'état de l'environnement n'évolue que lorsque l'agent effectue une action, on dit que l'environnement est **statique**. En revanche, si l'état de l'environnement peut évoluer au cours du temps sans aucune intervention de l'agent (par exemple, un feu qui se propage), alors on dit que l'environnement est **dynamique**. Un environnement statique est plus facile à gérer car l'agent a le temps de calculer une stratégie et l'exécuter car il sait que cette stratégie restera valide quoi qu'il arrive, ce qui n'est pas forcément le cas dans les environnements dynamiques.

Plusieurs agents peuvent évoluer dans le même environnement, dans ce cas on dira que l'environnement est **multi-agents**, par opposition à **mono-agent**. Dans la section suivante, nous allons détailler les spécificités de tels systèmes.

1.3 Systèmes multi-agents

Un environnement multi-agents, plus communément appelé **système multi-agents** (SMA) [Wooldridge, 2009], est un environnement dans lequel plusieurs agents évoluent simultanément. Dans un système multi-agent, les agents adoptent un comportement selon une **fonction d'utilité collective** (CUF) [Nicholson and Snyder, 1997]. Chaque agent possède sa propre fonction d'utilité collective, pouvant aller de la coopération totale (l'agent agit de manière coopérative avec les autres agents du système dans le but de réaliser une tâche commune), à la compétition totale (l'agent agit uniquement dans le but de nuire aux autres agents du système). Dans le cadre de cette thèse, nous considérons des agents coopératifs qui ont pour objectif d'accomplir une tâche commune. Nous allons désormais introduire différents types de systèmes multi-agents.

1.3.1 Architectures multi-agents

Il existe différents types de systèmes multi-agent qui permettent de résoudre différents types de problèmes, mais qui sont également plus ou moins complexes à mettre en œuvre. Nous allons ici décrire les plus communs, du plus simple au plus évolué, avec leurs avantages et inconvénients.

Système multi-agents centralisé

Dans un système multi-agents **centralisé**, il existe une entité centrale, qu'elle soit ou non un des agents du système, qui est responsable de coordonner l'ensemble des agents. Cette entité agit comme un « chef d'orchestre » dans le système. Les agents évoluent dans l'environnement et communiquent leurs observations à l'entité centrale qui est alors chargée de prendre une décision et d'assigner une tâche à chaque agent. De tels systèmes sont relativement simples à mettre en œuvre car ils se rapprochent le plus d'un système mono-agent : un seul agent collecte l'ensemble des observations sur l'environnement, calcule une stratégie, mais plusieurs agents l'exécutent. Cependant, tous les agents sont dépendants de cette entité centrale, ce qui les rend moins autonomes individuellement, mais surtout, en cas de panne de l'entité centrale ou de défaillance de communication, tout le système devient inopérant.

Système multi-agents distribué

Dans un système multi-agents **distribué** [Moulin and Chaib-Draa, 1996], le processus de prise de décision est distribué sur tous les agents du système sans forcément nécessiter d'entité centrale. Chaque agent communique ses connaissances à l'ensemble des agents du système, puis l'ensemble des agents prend une décision collective avant de l'exécuter. De tels systèmes sont plus robustes que des systèmes centralisés car si un agent tombe en panne, il n'affecte pas le bon fonctionnement du système. De plus, le processus de prise de décision étant réparti sur tous les agents, on peut profiter des ressources de calcul de tous les agents en effectuant les calculs en parallèle. Il peut toutefois exister une entité centrale qui est chargée de récolter les informations de tous les agents et de mettre en commun le résultat des calculs afin d'assigner une tâche aux agents (broadcast), ou bien chaque agent peut avoir la possibilité de communiquer avec les autres agents afin de mettre en commun les observations et les résultats des calculs. Dans les deux cas, la communication est un aspect primordial au bon fonctionnement du système. De ce fait, lorsque la communication est faible, voire impossible, de tels systèmes peuvent s'avérer très inefficaces.

Système multi-agents décentralisé

Dans un système multi-agents **décentralisé** [Brooks, 1986], il n'existe plus aucune centralisation. Chaque agent évolue de son propre chef, et n'a de connaissances uniquement sur ce qu'il a lui-même observé. De ce fait, chaque agent doit prendre sa propre décision à partir de ses connaissances locales sur l'environnement et sans savoir ce que les autres agents ont prévu. Parfois, la communication est permise par le biais d'une action de l'agent (par exemple, l'agent décide de communiquer une information à un ou plusieurs agents), mais cette communication a un coût [Liu et al., 2014]. Les systèmes multi-agents décentralisés sont donc insensibles aux pertes de communication, en revanche, les agents n'ayant que des connaissances locales sur leur environnement, la stratégie globale de résolution du problème est souvent moins efficace que dans un système centralisé.

1.3.2 Agents hétérogènes

Lorsque tous les agents du système ne partagent pas les mêmes capacités (capteurs différents, ressources de calcul différentes, actionneurs différents, ...), on parle de **système multi-agents hétérogène**. Les systèmes multi-agents hétérogènes [Zheng et al., 2011] permettent de résoudre une plus grande variété de problèmes que les systèmes multi-agents classiques du fait de la capacité de chaque agent à résoudre des sous-problèmes différents. Cependant, de tels systèmes

sont plus difficiles à contrôler. En effet, dans un système multi-agents **homogène** (donc où tous les agents sont identiques), un agent peut généralement raisonner par **empathie** afin d'anticiper la stratégie des autres agents. De la même manière, un plan calculé pour un agent du système sera également valide pour un autre agent. Dans un système hétérogène, il est plus difficile d'anticiper le comportement d'un autre agent, ce qui rend le processus de décision plus difficile.

1.3.3 Conclusion

Nous avons vu dans cette section différentes propriétés d'un système multi-agents et les architectures couramment employées pour les mettre en œuvre. L'architecture utilisée dépendra généralement du problème que l'on cherche à résoudre et particulièrement des contraintes de communication, et le choix de l'architecture aura une influence sur le comportement des agents. En effet, un agent évoluant dans un système décentralisé prendra des décisions de manière locale du fait de son manque d'information sur l'ensemble du système, ce qui fait que la stratégie de résolution globale pourra être moins efficace, tandis qu'un agent évoluant dans un système centralisé aura un comportement optimal selon les connaissances globales de tous les agents, mais sera sensible aux perturbations de communication et aux défaillances de l'entité centrale. Si l'on travaille avec des agents hétérogènes, alors le processus de prise de décision sera plus complexe, car il sera plus difficile d'anticiper le comportement des autres agents.

Nous allons désormais nous intéresser au processus de prise de décision d'un agent et plus particulièrement dans le cadre de la planification.

1.4 Planification

La planification en intelligence artificielle consiste à rechercher une séquence d'actions (un **plan**) dans le but d'accomplir une tâche précise. La planification caractérise le comportement proactif d'un agent : il va d'abord réfléchir à une solution pour résoudre le problème en anticipant les éventuelles complications, pour ensuite appliquer concrètement cette solution.

On distingue généralement deux types de planification :

— La planification **en ligne** qui consiste à planifier sur place durant la mission, par exemple lorsque l'agent reçoit de nouvelles informations, il peut mettre à jour son plan en conséquence. L'avantage de la planification en ligne est que l'on peut tenir compte des informations que l'on dispose au temps présent, ce qui réduit l'espace de recherche. Cependant, il faut régulièrement calculer une nouvelle stratégie, ce qui peut être coûteux en temps de calcul. Dans le cadre de la planification robotique par exemple, les robots ont des ressources limitées (batterie, puissance de calcul) et il est souvent plus intéressant de prévoir la stratégie à l'avance.

— La planification **hors ligne** qui consiste à prévoir à l'avance le plan pour l'exécution de toute la mission. L'avantage de la planification hors ligne est que le plan est calculé à l'avance donc l'agent ne perd pas de temps ni de ressources pour calculer durant sa mission. En revanche, étant donné que l'on ne connaît pas l'état actuel du système, il faut anticiper toutes les éventualités possibles, ce qui est plus coûteux.

Concrètement, le choix entre la planification en ligne ou la planification hors ligne dépendra du type de problème à traiter, des ressources à disposition de l'agent durant l'exécution de sa mission, et de la complexité de la tâche à réaliser.

La planification est un domaine très vaste de l'intelligence artificielle, et l'idée de cette section n'est pas d'être exhaustive, mais simplement d'aborder les notions principales qui permettront

par la suite de comprendre ce qui nous intéresse réellement dans cette thèse : la planification sous incertitude.

1.4.1 Planification classique

La planification classique est le cadre formel de planification le plus simple, où l'environnement est totalement observable, statique, déterministe. On dit que c'est un modèle restreint au sens de [Ghallab et al., 2004].

Définition 1.4.1 (Problème de planification classique)**.** Formellement, un problème de planification classique [McDermott et al., 1998] est un tuple $\mathcal{P} = \langle \mathcal{M}, \mathcal{I}, \mathcal{G} \rangle$, où :

- $\mathcal{M} = \langle \mathcal{F}, \mathcal{A} \rangle$ est le modèle du domaine, où $\mathcal{F}$ est un ensemble fini de fluents, et $\mathcal{A}$ un ensemble d'actions
- Un état s de l'environnement est une affectation de tous les fluents de $\mathcal{F}$. $\mathcal{S}$ est l'ensemble des états possibles.
- $\mathcal{I} \in \mathcal{S}$ est l'état initial
- $\mathcal{G}$ est le but, c'est-à-dire une affectation d'un sous-ensemble de fluents de $\mathcal{F}$
- Chaque action $a \in \mathcal{A}$ est un tuple $\langle pre_a, add_a, del_a, c_a \rangle$, qui sont respectivement les préconditions de l'action, ses effets (ajouts et suppressions), et son coût.
- $\Gamma(\cdot)$ est la fonction de transition, telle que, $\Gamma(s, a) \models \bot$ si $s \not\models pre_a$, sinon $\Gamma(s, a) \models s \cup add_a \setminus del_a$

Solution d'un problème de planification classique

Une solution d'un problème de planification classique est un plan, ou une séquence d'actions, $\pi = \langle a_1, a_2, \ldots, a_n \rangle$ telle que, en partant de l'état initial, et en exécutant les actions du plan l'une après l'autre séquentiellement, l'agent atteint son but, c'est-à-dire $\Gamma(\mathcal{I}, \pi) \models \mathcal{G}$. Le coût d'un plan est donné par :

$$c(\pi) = \sum_{a \in \pi} c_a \tag{1.1}$$

Nous nous intéressons à des agents rationnels, qui cherchent donc à optimiser leur critère de performance. Par exemple, si l'agent cherche à se rendre d'un point A à un point B le plus rapidement possible, sa mesure de performance sera le temps (le coût d'une action est donc le temps d'exécution de cette action). De ce fait, résoudre un problème de planification classique de manière optimale consiste à trouver un plan π tel que $\Gamma(\mathcal{I}, \pi) \models \mathcal{G}$ et que, pour tout autre plan π' tel que $\Gamma(\mathcal{I}, \pi') \models \mathcal{G}$,

$$c(\pi) \leq c(\pi') \tag{1.2}$$

Méthodes de résolution

Le General Problem Solver (GPS) [Newell and Simon, 1961] était l'un des premiers planificateurs automatiques proposés. Il est capable de résoudre des problèmes simples comme les Tours de Hanoi, illustré en figure 1.2, mais est très limité pour résoudre des problèmes plus complexes dû à l'explosion combinatoire. En effet, GPS résout le problème en utilisant une **analyse des fins et des moyens** (*means-end analysis*) : pour un état donné, on cherche un moyen de se rapprocher des buts, ce qui génère de nouveaux sous-buts à résoudre, et ainsi de suite jusqu'à avoir résolu tous les buts.

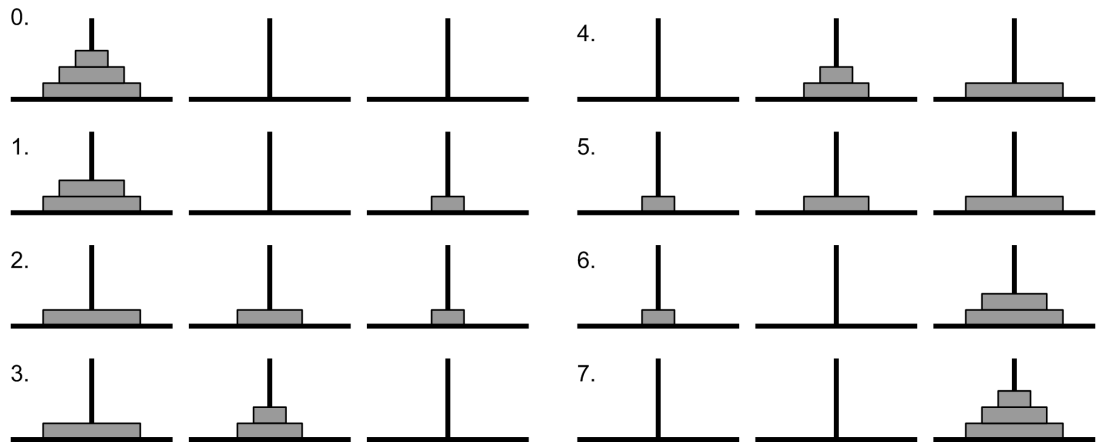


FIGURE 1.2 – Exemple du problème des Tours de Hanoi avec 3 tours. On cherche un plan pour atteindre l'état but (en bas à droite) à partir de l'état initial (en haut à gauche). Le joueur ne peut déplacer que un disque à la fois, un disque ne peut être placé que sur un disque plus grand que lui ou sur un emplacement vide. Les étapes 1 à 6 illustrent un exemple d'états successifs pour atteindre le but.

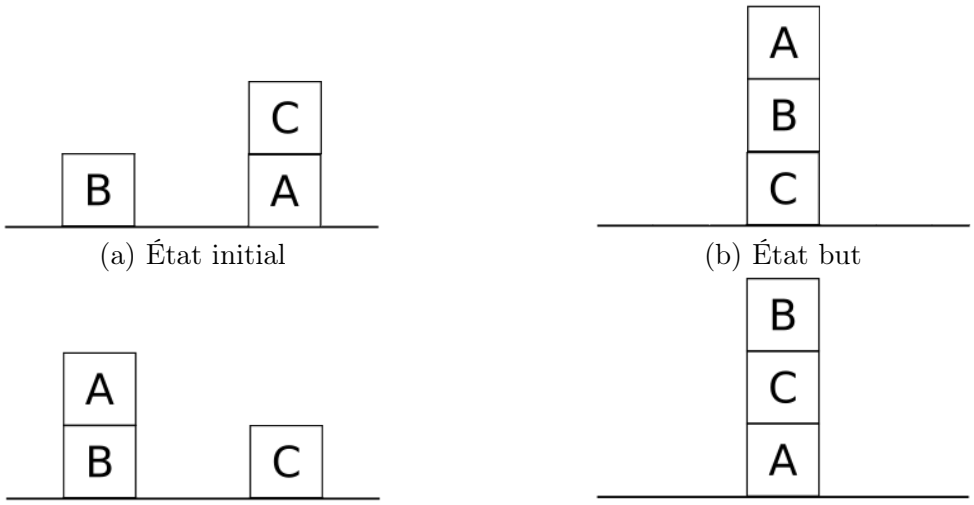
Le planificateur STRIPS [Fikes and Nilsson, 1971] (STanford Research Institute Problem Solver) est également l'un des premiers planificateurs développés, qui utilise également l'analyse des fins et des moyens. Cet algorithme garde en mémoire une représentation du monde qui peut évoluer durant le déroulement de l'algorithme, par exemple en effectuant une action qui modifie l'état du monde. STRIPS, tout comme GPS, est un algorithme de planification **linéaire**, ce qui signifie qu'il existe un ordre sur les buts (représenté par une **pile** en mémoire) et que les buts sont traités dans l'ordre les uns après les autres. De ce fait, les sous-buts doivent être indépendants, et le planificateur ne peut pas modifier le plan calculé pour un but particulier. Sussman a montré dans [Sussman, 1973] qu'il existe de simples problèmes qui ne peuvent être résolus par de la planification linéaire. Prenons l'exemple de la figure 1.3, qui illustre le domaine de planification bien connu « blocks world ». Dans ce domaine, un agent doit déplacer des blocks d'un état initial, illustré par la figure 1.3a vers un état but, illustré par la figure 1.3b. L'agent ne peut déplacer qu'un seul block à la fois. Un planificateur linéaire décomposera ce but en sous-buts suivants :

1. Déplacer le block A sur le block B
2. Déplacer le block B sur le block C

Si le planificateur commence par résoudre le premier sous-but, il va mettre le block C de côté, puis prendre le block A et le déplacer sur le block B, comme illustré dans la figure 1.3c. Cependant, une fois le premier sous-but accompli, l'agent ne peut plus résoudre le deuxième sous-but, car cela impliquerait de défaire le premier sous-but. À l'inverse, si le planificateur commence par résoudre le deuxième sous-but, dans ce cas il déplace le block B directement sur le block C, comme illustré dans la figure 1.3d. Mais cette fois, il ne peut plus résoudre le premier sous-but, car cela nécessiterait de défaire le deuxième sous-but.

Pour résoudre de tels problèmes, la planification linéaire n'est pas suffisante. Des techniques comme le *least commitment planning* [Weld, 1994] permettent de surpasser ces limitations en retardant au maximum l'exécution d'un but, de ce fait la probabilité de validité d'une action est maximale au moment de l'application.

La planification non-linéaire [Peot and Smith, 1992], quant à elle, voit les buts comme un "ensemble de buts" à résoudre sans ordre précis : ce qui compte est que tous les buts soient



(a) État initial (b) État but

(c) Résolution du premier sous-but (d) Résolution du deuxième sous-but

FIGURE 1.3 – Anomalie de Sussman dans le domaine Blocks World

résolus à la fin. De ce fait, la planification non-linéaire est insensible aux anomalies de Sussman. En revanche, la planification non-linéaire est plus coûteuse car l'espace de recherche inclut tous les ordres possibles des sous-buts. Des techniques comme le chaînage avant (*forward search*, c'est-à-dire partir de l'état initial et progresser jusqu'à l'état but) ou arrière (*backward search*, c'est-à-dire partir de l'état but et régresser jusqu'à l'état initial) peuvent être utilisées pour résoudre des problèmes de planification non-linéaire. On peut également voir le problème de planification comme un problème de recherche de chemin dans un graphe, le graphe étant induit par l'espace d'états (nœuds) et la fonction de transition (arcs), et utiliser des algorithmes de parcours de graphe pour trouver un plan pour atteindre l'état but à partir de l'état initial.

Nous avons donc introduit la planification classique et les difficultés que cela implique. En planification classique, l'environnement est complètement observable et déterministe, ce qui restreint le champ d'applications possibles. Dans la suite, nous allons nous intéresser à la planification sous incertitude, c'est-à-dire la planification lorsque l'environnement est stochastique et/ou l'observabilité est partielle.

1.5 Conclusion

Dans ce chapitre, nous avons tout d'abord commencé par un bref historique de l'intelligence artificielle ainsi que les domaines de recherche et leurs applications. Ensuite, nous avons introduit les concepts clés nécessaires à la bonne compréhension de ce manuscrit. Nous avons défini les notions d'agent et d'environnement et comment un agent raisonne de manière rationnelle. Nous avons présenté les propriétés qui caractérisent un environnement, comment un agent perçoit son environnement de manière complète ou partielle, l'influence du hasard sur les transitions du système qui peut être déterministe ou stochastique, et la manière dont le système évolue sans l'influence de l'agent, qui peut être statique ou dynamique. Nous avons introduit les systèmes multi-agents qui peuvent être centralisés, distribués ou décentralisés et contenir des agents homogènes ou hétérogènes. Enfin, nous avons présenté la problématique de la planification en intelligence artificielle.

Dans cette thèse, nous nous intéressons à la planification dans un environnement stochastique,

partiellement observable, avec des agents hétérogènes. Dans le chapitre suivant, nous allons introduire les cadres formels permettant de formaliser de tels problèmes et les méthodes de résolution couramment employées ainsi que les limites de ces approches.

Chapitre 2

Planification sous incertitude

Sommaire

2.1 Introduction

La planification sous incertitude, ou planification probabiliste, est un cas général de planification où l'environnement est stochastique, c'est-à-dire régi par le hasard. L'incertitude peut apparaître sous différentes formes : dans les actions des agents qui peuvent échouer, par exemple un robot qui se déplace sur un terrain glissant, ce qui induit de l'incertitude sur les transitions de l'agent ; dans l'observabilité des agents qui peut être imparfaite, par exemple un robot embarquant des capteurs imprécis, ce qui induit de l'incertitude sur l'état réel de l'environnement. Dans cette thèse, nous abordons des problématiques de planification dans un environnement inconnu, donc sous incertitude. Nous allons pour cela utiliser le cadre théorique des Processus Décisionnels de Markov.

2.2 Processus décisionnels de Markov

Les Processus Décisionnels de Markov [Puterman, 1994] (MDP [2]) sont un cadre formel largement utilisé pour modéliser des problèmes de prise de décision séquentielle sous incertitude. Leur fonctionnement est étroitement lié à celui des chaînes de Markov, que nous allons présenter dans la section suivante.

2.2.1 Les chaînes de Markov

Les Processus Décisionnels de Markov généralisent le concept des **chaînes de Markov** [Markov, 1906]. Les chaînes de Markov sont un outil mathématique introduit dans le début du XX$^{\text{ème}}$ siècle par le mathématicien russe **Andreï Markov** et permettent de modéliser des processus stochastiques à temps discret.

Définition 2.2.1 (Chaîne de Markov)**.** Soit un système qui peut être dans un ensemble S d'états. L'état du système évolue régulièrement par pas de temps discret selon une matrice de transition donnée. Une chaîne de Markov est une séquence de variables aléatoires $X_0, X_1, \ldots, X_n$, où X_i est une distribution de probabilité sur S au temps i.

L'évolution d'une chaîne de Markov respecte une règle appelée Propriété de Markov et qui est définie comme suit :

Définition 2.2.2 (Propriété de Markov)**.** La propriété de Markov suppose que la probabilité de transition d'un état s_t à un état s_{t+1} ne dépend que de l'état courant s_t et non pas de l'historique d'états. Formellement,

$$P(s_{t+1} \mid s_0, s_1, \ldots, s_t) = P(s_{t+1} \mid s_t) \tag{2.1}$$

Nous avons introduit le principe des chaînes de Markov. Dans la suite, nous allons étendre ce concept aux processus décisionnels de Markov.

2.2.2 Formalisme

Les chaînes de Markov permettent de modéliser l'évolution d'un système stochastique. Cependant, nous n'avons aucun contrôle sur l'évolution du système : nous sommes seulement observateurs du système, il n'y a pas de prise de décision possible. Les Processus Décisionnels de Markov étendent ce principe en ajoutant des **actions** qui vont permettre de prendre des **décisions** dans le but d'agir, à chaque pas de temps, sur la manière dont le système va évoluer. Plus particulièrement, nous allons pouvoir modéliser le comportement d'un agent prenant des décisions dans un environnement stochastique.

Définition 2.2.3 (Processus Décisionnel de Markov)**.** Un Processus Décisionnel de Markov (MDP) est un tuple $\langle S, A, T, R \rangle$ où :

- S est un ensemble fini des états de l'environnement
- A est un ensemble fini des actions applicables par l'agent
- $T : S \times A \times S \to [0, 1]$ est la fonction de transition de l'environnement telle que $T(s, a, s') = P(s' \mid s, a)$. Cela correspond à la probabilité de transiter dans l'état $s' \in S$ lorsque l'agent exécute l'action $a \in A$ dans l'état $s \in S$.

2. Markov Decision Process

— $R : S \times A \to \mathbb{R}$ est la fonction de récompense, où R(s, a) est la récompense obtenue par l'agent lorsqu'il exécute l'action $a \in A$ dans l'état $s \in S$.

Un Processus Décisionnel de Markov est un **processus à temps discret**, c'est-à-dire que, à chaque pas de temps, l'agent se trouve dans un état $s \in S$, il exécute une action $a \in A$, puis transite dans un nouvel état $s' \in S$ selon la distribution de probabilité de $T(s, a)$, et reçoit la récompense $R(s, a)$ correspondante. Ce processus est répété pour un nombre de pas de temps donné, noté H et que nous appellerons l'**horizon**, qui peut être fini ou infini.

Les Processus Décisionnels de Markov possèdent également la propriété de Markov, c'est-à-dire que la probabilité de transiter d'un état $s \in S$ à un état $s' \in S$ après avoir exécuté l'action $a \in A$ ne dépend que de l'état courant et de l'action courante, et non pas de l'historique d'états ou d'actions. Formellement,

$$P(s^{t+1} \mid s^0, a^0, s^1, a^1, \ldots, s^t, a^t) = P(s^{t+1} \mid s^t, a^t) \tag{2.2}$$

La fonction de récompense de notre MDP va être ce qui guide l'agent dans sa prise de décision. Elle peut soit représenter un gain lorsque l'agent se trouve dans une situation favorable, soit un coût lorsqu'il se trouve dans une situation défavorable. Nous avons précédemment défini la fonction de récompense comme étant la récompense $R(s, a)$ lorsque l'agent exécute une action a dans un état s, mais cette définition est bien souvent ajustée en fonction du problème que l'on cherche à résoudre. On pourra notamment trouver régulièrement les définitions suivantes :

— $R : S \times A \times S \to \mathbb{R}$: récompense l'agent pour transiter dans un état s' à partir d'un état s en effectuant l'action a

— $R : S \to \mathbb{R}$: récompense l'agent pour se trouver dans un état s

Afin de prendre la meilleure décision dans un état donné, il ne suffit pas seulement de choisir l'action qui maximise la récompense courante espérée. En effet, il s'agirait d'une approche myope, et adopter un tel comportement risquerait à l'agent de se retrouver dans un état très néfaste qu'il aurait pu éviter. Il faut donc considérer la récompense espérée sur tout l'ensemble de l'horizon d'exécution. Nous aborderons cette notion plus en détail lorsque nous parlerons de la résolution des MDP. Pour le moment, nous allons synthétiser toutes les notions abordées jusqu'à maintenant à l'aide d'un exemple.

2.2.3 Exemple

Un robot navigue dans un environnement dont il connaît parfaitement la disposition tant il l'a déjà parcouru plusieurs fois : il y a des murs à certains endroits, et il sait que certaines zones sont inondées car il a plu récemment. Il doit se déplacer d'un point A à un point B le plus rapidement possible pour économiser ses batteries. Le problème, c'est que ses roues ne sont pas adaptées à la navigation sur l'eau, ce qui risque de le ralentir fortement : lorsqu'il entre dans une flaque d'eau, il a une chance de rester bloqué pendant un certain temps. Cette fois, il se décide alors à trouver une stratégie optimale pour arriver à bon port dans les meilleurs délais. La figure 2.1 illustre cette situation.

Notre robot peut se déplacer dans les quatre directions cardinales : HAUT, DROITE, BAS, GAUCHE, que nous noterons par la suite $\{\uparrow, \rightarrow, \downarrow, \leftarrow\}$, respectivement. Dans une situation normale, le chemin le plus rapide pour se rendre au point d'arrivée serait tout simplement de longer la bordure gauche ou droite, puis de tourner en direction du point d'arrivée. Cependant, ces chemins sont couverts d'eau, ce qui risque de fortement ralentir notre robot. Nous savons que notre robot a 1 chance sur 2 de rester bloqué à sa position dans l'état suivant s'il se trouve dans

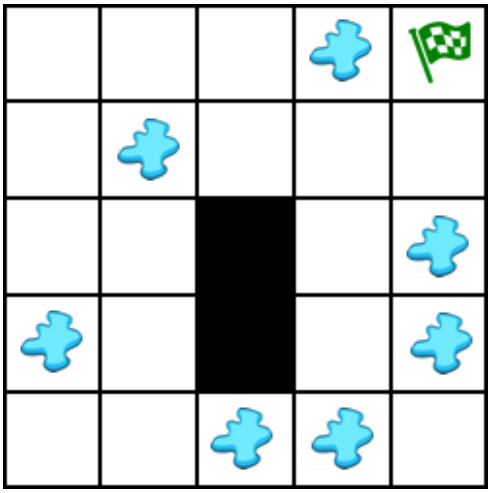


FIGURE 2.1 – Un environnement avec un état but (drapeau vert), des murs (cases noires), des flaques d'eau (bleues)

de l'eau. Il va falloir trouver une stratégie pour atteindre l'arrivée en perdant le moins de temps possible. Nous pouvons formaliser ce problème à l'aide d'un Processus Décisionnel de Markov !

Comme vu précédemment, un MDP est formulé par un tuple $\langle S, A, T, R \rangle$. Pour notre exemple, nous avons alors :

- $S = \{(x, y) \mid x, y \in [1, 5]\}$
- $A = \{\uparrow, \rightarrow, \downarrow, \leftarrow\}$
- $T(\square, *, \square) = 1$
- $T(\square, *, \text{🧩}) = 1$
- $T(\text{🧩}, *, \square) = \frac{1}{2}$
- $T(\text{🧩}, *, \text{🧩}) = \frac{1}{2}$
- $T(\square, *, \text{🏁}) = 1$
- $T(\text{🧩}, *, \text{🏁}) = \frac{1}{2}$
- $T(\text{🏁}, *, \text{🏁}) = 1$
- $R(\text{🏁}) = 1$

Par souci de simplicité, nous n'avons pas spécifié les actions pour les transitions. Évidemment, une action n'est applicable uniquement sur une case adjacente. Aussi, les transitions non spécifiées ont toutes une probabilité de 0, et un processus de normalisation est appliqué pour obtenir une distribution de probabilité valide, de la manière suivante :

$$T(s, a, s') = \frac{T(s, a, s')}{\sum_{s'' \in S} T(s, a, s'')} \tag{2.3}$$

Nous pouvons noter que notre fonction de transition respecte bien la propriété de Markov, à savoir l'état suivant ne dépend que de l'état courant. La modélisation précédente laisse supposer que nous définissons notre fonction de récompense de la manière $R : S \rightarrow \mathbb{R}$ avec une récompense seulement lorsque le robot atteint l'état but.

Maintenant que nous avons formalisé notre problème, nous allons nous intéresser à sa résolution. La résolution d'un MDP consiste à calculer une **politique** qui dicte l'action à effectuer dans un état. Plus particulièrement, nous aborderons l'**optimalité** des politiques et comment calculer une politique d'action optimale pour notre robot.

2.2.4 Politiques

Dans un environnement déterministe, on calcule un plan, c'est-à-dire une séquence d'actions, que l'agent doit exécuter dans l'ordre afin d'accomplir sa mission. Les choses se compliquent lorsque l'on passe dans le domaine de l'incertain. En effet, les transitions d'états étant stochastiques, il est difficile de se fixer une séquence d'actions qui restera valide quels que soient les imprévus rencontrés. Nous allons pour cela introduire la notion de **politique**. Dans un MDP, la politique de l'agent lui **dicte**, pour chaque état de l'environnement, quelle action il doit exécuter. Cette formulation rend ainsi insensible aux perturbations stochastiques.

Définition 2.2.4 (Politique)**.** Une politique $\pi : S \to A$ est une fonction qui associe, à chaque état de l'environnement, l'action que l'agent doit exécuter.

La définition précédente est valable pour les politiques **déterministes**. Il existe également les politiques **stochastiques** qui attribuent une distribution d'actions à chaque état de l'environnement. Dans le cadre de cette thèse, nous nous intéresserons seulement aux politiques déterministes.

Vous pourrez remarquer qu'une telle politique possède également la propriété de Markov : l'action à exécuter ne dépend que de l'état courant et ne tient pas compte de l'historique. Par ailleurs, une telle politique est dite **stationnaire** : elle ne dépend pas de l'instant. C'est-à-dire, pour un horizon H,

$$\forall i, j \in \{0, 1, \ldots, H-1\}, \pi^i(s) = \pi^j(s), \tag{2.4}$$

où π^t est la politique lorsqu'elle est appliquée à l'instant t. Dans le cas contraire, une politique non stationnaire pourrait, pour un même état, prescrire des actions différentes selon l'instant t. Dans cette thèse, nous nous intéresserons uniquement aux politiques stationnaires. De manière générale, le terme « politique » fait référence à une politique déterministe stationnaire.

Nous avons posé les bases de ce qu'est une politique dans le cadre des processus décisionnels de Markov. Nous allons maintenant tenter de mesurer l'efficacité d'une politique pour un critère de performance donné, à savoir la fonction de récompense.

Pour mesurer l'efficacité d'une politique, nous allons d'abord introduire la notion de **fonction de valeur** d'une politique.

Définition 2.2.5 (Fonction de valeur)**.** La fonction de valeur V^π d'une politique π est une fonction $V^\pi : S \to \mathbb{R}$ qui associe, à chaque état $s \in S$ de l'environnement, la récompense espérée lorsque l'on exécute l'action $\pi(s)$ dans l'état s.

La récompense espérée ne dépend pas seulement de l'action à effectuer dans l'état courant. En effet, comme nous l'avons mentionné précédemment, récompenser l'agent seulement pour choisir la meilleure action dans l'état courant risquerait de le conduire vers des états très défavorables dans le futur. Au lieu de ça, nous allons déterminer la récompense espérée sur le long terme, c'est-à-dire en évaluant les conséquences à long terme du choix de cette action dans cet état, afin de choisir l'action qui maximise la récompense espérée. Formellement, la fonction de valeur pour une politique π, à horizon H, est définie comme :

$$\forall s \in S, V^\pi(s) = E\left(\sum_{t=0}^{H-1} \gamma^t r_t \mid s^t = s\right) \tag{2.5}$$

où $r_t = R(s^t, \pi(s^t))$ est la récompense directe perçue par l'agent lorsqu'il exécute la politique π dans l'état s^t à l'instant t, et $0 < \gamma \leq 1$ est le **facteur d'atténuation** [3].

3. *discount factor* en anglais

Le facteur d'atténuation est une constante fixée à l'avance selon le problème à traiter. Un faible γ permet de donner moins d'importance aux récompenses plus éloignées dans le temps, afin d'inciter l'agent à récolter les récompenses qui sont proches de lui plus rapidement. À l'inverse, un γ de 1 accordera autant d'importance aux récompenses passées qu'aux récompenses directes.

Une telle fonction de valeur peut être calculée en utilisant la définition récursive suivante :

$$V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s') V^{\pi}(s') \tag{2.6}$$

Cette formulation a été proposée dans les années 1950 par le mathématicien américain **Richard Bellman** dans son ouvrage fondateur en programmation dynamique, *Dynamic Programming* [Bellman, 1957]. Cette équation porte aujourd'hui son nom : l'**équation de Bellman**.

Nous avons introduit les notions permettant de calculer la valeur d'une politique. Nous allons désormais nous intéresser à la résolution des MDP, à savoir, calculer une politique optimale.

2.2.5 Résolution

Résoudre un Processus Décisionnel de Markov consiste à trouver une politique π^* telle que, pour tout état $s \in S$ et pour toute politique $\pi \neq \pi^*$,

$$V^{\pi^*}(s) \geq V^{\pi}(s) \tag{2.7}$$

Une telle politique est appelée **politique optimale**, et sa fonction de valeur V^{π^*} est régulièrement notée simplement V^*. Nous allons maintenant nous intéresser aux algorithmes pour calculer une telle politique.

De manière générale, une telle politique peut être calculée en utilisant des algorithmes de programmation dynamique. Parmi eux, les deux plus populaires fonctionnent sur un principe d'**itération** :

— Itération sur la fonction de valeur avec l'algorithme **Value Iteration** [Bellman, 1957]
— Itération sur la politique avec l'algorithme **Policy Iteration** [Howard, 1960]

L'algorithme Value Iteration

L'algorithme Value Iteration, introduit par Bellman [Bellman, 1957], est un des algorithmes les plus utilisés pour résoudre des processus décisionnels de Markov. Il commence par fixer une fonction de valeur V_0 nulle telle que, $\forall s \in S, V_0(s) = 0$, puis l'améliore de manière itérative en mettant à jour, à chaque itération, la valeur de chaque état $s \in S$ de la manière suivante :

$$V_{t+1}(s) = \max_{a \in A} \left(R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V_t(s') \right) \tag{2.8}$$

Ce processus est répété de manière itérative jusqu'à un horizon H fixé, ou jusqu'à convergence dans le cas d'un horizon infini. La politique correspondante à cette fonction de valeur peut être construite durant ce processus, ou après, à partir de la fonction de valeur obtenue, en sélectionnant l'action qui maximise dans chaque état la fonction de valeur optimale :

$$\pi^*(s) = \arg\max_{a \in A} \left(R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^*(s') \right) \tag{2.9}$$

L'algorithme 1 présente un pseudo-code pour le cas général, c'est-à-dire pour un horizon infini.

Algorithme 1 : Value Iteration

Données : Un MDP $\langle S, A, T, R \rangle$, un facteur d'atténuation γ, un critère de convergence ϵ
Résultat : Une fonction de valeur optimale V^*

$V_0(s) \leftarrow 0, \forall s \in S$
$\Delta \leftarrow +\infty$
$t \leftarrow 0$
tant que $\Delta \geq \epsilon$ **faire**
  $\Delta \leftarrow 0$
  $t \leftarrow t + 1$
  pour chaque $s \in S$ **faire**
    $V_t(s) \leftarrow \max_{a \in A}(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V_{t-1}(s'))$
    $\Delta \leftarrow \max_{a \in A}(\Delta, |V_{t-1}(s) - V_t(s)|)$
  fin
fin
retourner V_t

Cet algorithme est le cas général de l'algorithme Value Iteration à horizon infini. Pour un horizon H fini, on peut arrêter l'algorithme prématurément lorsque $t > H$.

Le paramètre $\epsilon \in \mathbb{R}$ est un critère de convergence qui permet à l'algorithme de s'arrêter lorsque plus aucune amélioration significative n'est trouvée dans la fonction de valeur. Pour un ϵ donné, l'erreur d'approximation est bornée à $\frac{\epsilon}{1-\gamma}$, avec γ le facteur d'atténuation. À horizon infini, γ doit forcément être strictement inférieur à 1, dans le cas contraire les récompenses passées s'additionnent à la récompense directe avec la même importance à chaque itération et l'algorithme ne converge jamais.

La complexité de cet algorithme pour une itération est de $O(|S|^2|A|)$, ce qui met en évidence une des principales limitations dans la résolution des MDP, qui est la taille de l'espace d'état. Sutton et Barto [Sutton and Barto, 2018] ont montré que des MDP possédant jusqu'à un million d'états peuvent être résolus en un temps raisonnable avec cet algorithme, mais au-delà la résolution devient de plus en plus difficile.

Nous allons maintenant présenter le second algorithme couramment utilisé pour résoudre les MDP, Policy Iteration.

L'algorithme Policy Iteration

L'algorithme Policy Iteration, introduit par Howard [Howard, 1960], a un fonctionnement différent de l'algorithme Value Iteration. En effet, au lieu de calculer une politique à partir de la fonction de valeur optimale, Policy Iteration va directement construire une politique et l'améliorer par itérations successives. Il commence par fixer une politique arbitraire π_0, puis chaque itération i de l'algorithme se déroule de la manière suivante :

1. **Évaluation** On calcule la fonction de valeur V^{π_i} de la politique π_i en utilisant l'équation de Bellman. Ce processus est généralement coûteux.
2. **Amélioration** On améliore la politique à partir de la fonction de valeur V^{π_i} en utilisant l'équation 2.9.

Ce processus est répété jusqu'à convergence, c'est-à-dire lorsqu'aucune amélioration n'a été trouvée pour la politique durant une itération. On obtient alors une politique optimale. Cet algorithme permet de converger vers une politique optimale en moins d'itérations que Value

Iteration. En revanche, la complexité est également de $O(|S|^2|A|)$ pour une itération, et les étapes d'évaluation et d'amélioration de la politique sont coûteuses. D'après [Littman, 1996], le choix entre Value Iteration et Policy Iteration dépendrait en réalité de la structure du MDP à résoudre, mais dans le cas général, personne n'a encore réussi à les départager. Cet algorithme est décrit dans l'algorithme 2. Les lignes 4 à 6 représentent l'évaluation de la politique courante, et les lignes 7 à 13 représentent l'amélioration de la politique à partir de la fonction de valeur.

Algorithme 2 : Policy Iteration

Données : Un MDP $\langle S, A, T, R\rangle$, un facteur d'atténuation γ
Résultat : Une politique optimale π^*

```
1  Initialiser π(s) arbitrairement ∀s ∈ S
2  répéter
3  |  π' = π
4  |  pour chaque s ∈ S faire
5  |  |  V^π(s) = R(s, π(s)) + γ Σ_{s'∈S} T(s, π(s), s') V^π(s')
6  |  fin
7  |  pour chaque s ∈ S faire
8  |  |  si ∃a ∈ A, R(s, a) + γ Σ_{s'∈S} T(s, a, s') V^π(s') > V^π(s) alors
9  |  |  |  π'(s) = a
10 |  |  sinon
11 |  |  |  π'(s) = π(s)
12 |  |  fin
13 |  fin
14 jusqu'à π' = π
15 retourner π
```

Nous avons présenté ici les deux algorithmes les plus populaires pour résoudre des MDP. Cependant, pour des MDP trop complexes à résoudre, souvent en raison de leur espace d'état trop grand, des techniques ont été développées pour tirer parti de leur structure ou de la structure du problème à résoudre, ou bien des techniques approchées, de manière à faciliter le processus de résolution. Nous allons en énumérer quelques-unes.

Factored MDP

Les Factored Markov Decision Process [Boutilier et al., 1995] (FMDP) – MDP factorisés – exploitent le fait que certaines variables composant un état de l'environnement sont indépendantes, et donc il n'est pas toujours nécessaire de travailler sur l'espace d'état complet. De ce fait, la fonction de transition d'un FMDP est représentée comme un Réseau Bayésien Dynamique (DBN) [Dean and Kanazawa, 1989], ce qui permet une représentation compacte des dépendances entre les variables. On profite également de cette indépendance en définissant une fonction de récompense sur des sous-ensembles de variables dépendantes entre elles. Des algorithmes ont été développés pour résoudre des FMDP de manière efficace, en particulier Structured Value Iteration et Structured Policy Iteration [Boutilier et al., 2000]. Ces algorithmes utilisent des arbres de décision pour représenter les différentes fonctions du modèle (fonction de transition, fonction de récompense, fonction de valeur, et la politique).

MDP hiérarchiques

Les MDP hiérarchiques [Hauskrecht et al., 2013] décomposent le problème à résoudre en un ensemble de problèmes plus petits, qui sont résolus de manière indépendante pour former des macro-actions. Une macro-action est une politique locale, qui agit sur une région de l'espace d'état, et qui se termine lorsque cette région est quittée. L'inconvénient de cette formulation est qu'il faut calculer plusieurs politiques au lieu d'une seule, ce qui induit un coût supplémentaire, mais ce coût est compensé par le fait que le problème général est plus facile à résoudre. Par ailleurs, les macro-actions peuvent être réutilisées pour résoudre d'autres MDP similaires, ce qui est avantageux dans le cas de prise de décision en ligne, lorsqu'il faut régulièrement calculer une nouvelle politique d'action.

Apprentissage par renforcement

De très nombreuses méthodes à base d'apprentissage par renforcement ont été développées dans le but d'approximer une politique optimale pour un MDP. L'apprentissage par renforcement [Sutton and Barto, 2018] consiste à apprendre, à partir de traces d'exécution, une politique optimale. À chaque exécution, la politique est mise à jour en fonction des récompenses que l'agent a reçues. En particulier, les algorithmes de Q-learning [Watkins and Dayan, 1992] permettent d'apprendre une politique sans connaître le modèle de l'environnement, ce qui est intéressant lorsque ce modèle est totalement inconnu ou trop complexe à représenter en mémoire. L'inconvénient de ces méthodes est que l'apprentissage peut nécessiter un grand nombre d'épisodes sans réelle garantie de converger vers une politique optimale.

2.3 Observabilité partielle

Nous avons précédemment décrit le formalisme des Processus Décisionnels de Markov. Cependant, ce formalisme repose sur l'hypothèse forte que l'environnement est complètement observable. Malheureusement, cette hypothèse n'est pas vérifiée dans beaucoup d'applications, en particulier dans les problèmes de récolte d'information où, par définition, l'état de l'environnement n'est pas connu et doit être exploré par l'agent. Les Processus Décisionnels de Markov Partiellement Observables [Smallwood and Sondik, 1973] sont une extension des MDPs pour tenir compte de l'observabilité partielle de l'environnement.

Définition 2.3.1 (Processus Décisionnel de Markov Partiellement Observable)**.** Un Processus Décisionnel de Markov Partiellement Observable (POMDP) est un tuple $\langle S, A, T, R, \Omega, O \rangle$ où :

- S est un ensemble fini des états de l'environnement
- A est un ensemble fini des actions applicables par l'agent
- $T : S \times A \times S \to [0, 1]$ est la fonction de transition de l'environnement telle que $T(s, a, s') = P(s' \mid s, a)$. Cela correspond à la probabilité de transiter dans l'état $s' \in S$ lorsque l'agent exécute l'action $a \in A$ dans l'état $s \in S$.
- $R : S \times A \to \mathbb{R}$ est la fonction de récompense, où R(s, a) est la récompense obtenue par l'agent lorsqu'il exécute l'action $a \in A$ dans l'état $s \in S$.
- Ω est un ensemble fini d'observations possibles par l'agent
- $O : A \times S \times \Omega \to [0, 1]$ est la fonction d'observation telle que $O(a, s, o) = P(o \mid s, a)$. Cela correspond à la probabilité que l'agent reçoive l'observation $o \in \Omega$ lorsque le système a transité dans l'état $s' \in S$ après avoir exécuté l'action $a \in A$ dans l'état $s \in S$.

Dans un POMDP, l'agent possède des capteurs qui ne permettent pas de déterminer l'état de l'environnement de manière exacte. De ce fait, l'incertitude sur l'environnement réside dans le modèle de transition qui est stochastique, mais également dans les observations de l'agent qui peuvent être imprécises. Reprenons notre exemple de la section 2.2.3. Imaginons que le robot n'embarque pas de capteur lui permettant d'observer le sol, ou de détecter la présence d'eau ou non. Dans ce cas, si le robot roule sur de l'eau, à l'état suivant, il ne saura pas s'il se trouve sur de l'eau ou non.

Afin de calculer une politique d'un POMDP, on pourrait être tenté de construire un MDP dont l'espace d'états est l'ensemble des observations possibles, et considérer que l'état de l'environnement correspond à la dernière observation reçue. Cependant, il a été montré [Littman, 1994] qu'une politique optimale pour un tel MDP donne de très mauvais résultats.

De ce fait, nous allons devoir raisonner sur l'historique des observations perçues par l'agent. Cependant, la propriété de Markov impose que l'on ne doit pas tenir compte de l'historique dans le processus de transition du système. De ce fait, un tel modèle ne serait pas markovien. De plus, l'ensemble d'historiques d'observations possibles croît exponentiellement avec l'horizon, ce qui devient vite intraitable en horizon fini, et impossible en horizon infini, car il existe une infinité d'historiques d'observations possibles. En revanche, Bertsekas [Bertsekas, 1995] a montré que l'historique d'observation peut être résumé dans un **état de croyance** (*belief state*).

2.3.1 État de croyance

L'état de croyance est une statistique suffisante pour raisonner de manière optimale dans un POMDP. Il s'agit d'une distribution de probabilité sur l'espace d'état. Nous notons $\mathcal{B}$ l'ensemble des états de croyance possibles. Lorsque l'agent évolue dans son environnement, il met à jour son état de croyance b en fonction de l'action choisie, l'état de croyance initial, et l'observation reçue après avoir exécuté cette action. Ce processus de mise à jour de croyances est réalisé en utilisant un estimateur d'état (*State Estimator (SE)*) appliquant directement le théorème de Bayes [Bayes, 1958] pour calculer la nouvelle probabilité $P(s' \mid a, o, b)$ d'un état, avec $s' \in S$ un état, $o \in \Omega$ l'observation reçue, et $b \in \mathcal{B}$ l'état de croyance initial [Cassandra et al., 1994] :

$$\begin{aligned} SE_{s'}(b, a, o) &= P(s' \mid a, o, b) \\ &= \frac{P(o \mid s', a, b) P(s' \mid a, b)}{P(o \mid a, b)} \\ &= \frac{O(a, s', o) \sum_{s \in S} T(s, a, s') b(s)}{P(o \mid a, b)} \end{aligned} \tag{2.10}$$

où $P(o \mid a, b)$ est un facteur de normalisation défini tel que :

$$P(o \mid a, b) = \sum_{s' \in S} O(a, s', o) \sum_{s \in S} T(s, a, s') b(s) \tag{2.11}$$

L'état de croyance initial du POMDP, noté b^0, est généralement une donnée du modèle. Il peut être initialisé de manière uniforme si l'on n'a aucune connaissance a priori sur l'environnement. Ensuite, à chaque fois que l'agent reçoit une observation, cet état de croyance est mis à jour selon l'équation 2.10 pour tout état $s \in S$. Grâce aux états de croyance, nous pouvons traiter notre POMDP comme un MDP dont l'espace d'états est $\mathcal{B}$. Le MDP résultant est appelé un **Belief MDP** (MDP de croyances).

2.3.2 Belief MDP

Un Belief MDP est un processus décisionnel de Markov généré à partir d'un POMDP, et dont l'espace d'état est l'ensemble des états de croyance possibles.

Définition 2.3.2 (Belief MDP)**.** Un Belief MDP pour un POMDP est un tuple $\langle \mathcal{B}, \mathcal{A}, \tau, \rho \rangle$ où :
- $\mathcal{B}$ est l'ensemble des états de croyance du POMDP
- $\mathcal{A}$ est l'ensemble d'actions du POMDP
- $\tau \times \mathcal{A} \times \tau \to [0, 1]$ est la fonction de transition de l'état de croyance
- $\rho : \mathcal{B} \times \mathcal{A} \to \mathbb{R}$ est la fonction de récompense de l'état de croyance

La fonction de transition d'état de croyance τ peut être dérivée à partir des fonctions de transition et d'observation du POMDP en utilisant l'estimateur d'état :

$$\tau(b, a, b') = \sum_{\{o \in \Omega | SE(b,a,o)=b'\}} P(o \mid a, b) \tag{2.12}$$

où $P(o \mid a, b)$ est défini dans l'équation 2.11.

De la même manière, on peut définir la fonction de récompense ρ à partir de la fonction de récompense du POMDP :

$$\rho(b, a) = \sum_{s \in S} b(s) R(s, a) \tag{2.13}$$

Nous pouvons également définir une politique $\pi : \mathcal{B} \to A$ pour un belief MDP, et sa fonction de valeur peut être reformulée de la manière suivante :

$$V^{\pi}(b) = \rho(b, \pi(b)) + \gamma \sum_{o \in \Omega} P(o \mid \pi(b), b) V^{\pi}(SE(b, \pi(b), o)) \tag{2.14}$$

où $P(o \mid \pi(b), b)$ est défini dans l'équation 2.11.

Comme pour un MDP classique, l'objectif est de calculer une politique optimale sur l'espace d'états, qui est donc ici un espace d'états de croyance sur les états du POMDP. Cependant, comme nous allons le voir, ce processus n'est pas aussi simple que pour un MDP à observabilité complète.

2.3.3 Résolution

Un belief MDP est donc équivalent à un MDP totalement observable. Cependant, nous ne pouvons pas utiliser les algorithmes de résolution de la même manière que pour un MDP classique. En effet, bien qu'il ait été montré qu'une politique optimale pour un belief MDP est équivalent à une politique optimale pour un POMDP [Åström, 1965], un belief MDP est un MDP à espace d'état **continu** car il y a en réalité une infinité d'états de croyance possibles (qui sont l'ensemble de toutes les distributions de probabilité), et les algorithmes Value Iteration et Policy Iteration fonctionnent par itérations successives sur l'espace d'état.

En revanche, la fonction de valeur de la politique optimale d'un POMDP a été prouvée, à horizon fini, comme étant **linéaire par partie** et **convexe** [Sondik, 1978] (PWLC [4]), et cette propriété peut être exploitée algorithmiquement pour partitionner l'espace d'états de croyance en hyperplans, comme illustré dans la figure 2.2. Un hyperplan est représenté par un vecteur de dimension $|S|$, que nous nommons α-vecteur, et qui contient les coefficients de l'équation de cet hyperplan. De ce fait, une fonction de valeur à horizon fini peut être représentée par un ensemble d'α-vecteurs. [Smallwood and Sondik, 1973] ont prouvé qu'il existe une action optimale pour chaque partition.

4. Piecewise Linear and Convex

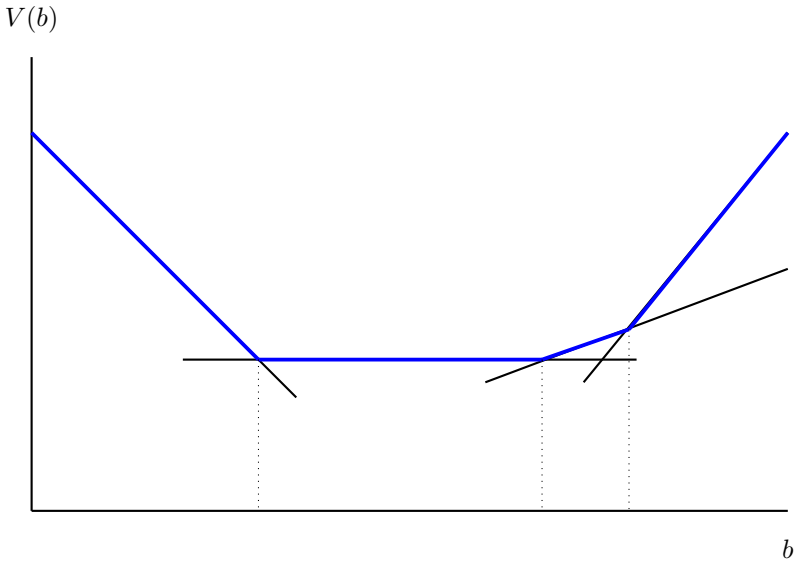


FIGURE 2.2 – Illustration de la propriété PWLC de la fonction de valeur d'un POMDP à deux états. Les segments bleus représentent la fonction de valeur optimale.

Des algorithmes exacts basés sur Value Iteration ont été développés pour exploiter la représentation en α-vecteur [Sondik, 1978, Cassandra et al., 1997]. En termes de complexité, le passage d'un environnement totalement observable à un environnement partiellement observable a des conséquences importantes sur la résolution du modèle. En effet, à horizon fini, la résolution d'un MDP est P-complet, tandis que la résolution d'un POMDP est PSPACE-complet [Papadimitriou and Tsitsiklis, 1987]. Pour un POMDP à horizon infini, le problème est indécidable [Madani et al., 1999]. La complexité importante de ce modèle a de ce fait entraîné beaucoup de recherches d'algorithmes, notamment les approches Point-based qui travaillent sur un sous-ensemble de l'état de croyance [Lovejoy, 1991]. Ces techniques ont montré d'impressionnants résultats et ont continué d'être améliorées par la suite. Hauskrecht [Hauskrecht, 2000] a proposé de calculer l'ensemble des états de croyance accessibles depuis l'état initial pour discrétiser l'espace d'états de croyance. Des algorithmes tels que PBVI (*Point-based Value Iteration*) [Pineau et al., 2003], PERSEUS [Spaan and Vlassis, 2005] et SARSOP [Kurniawati et al., 2009] ont ensuite étendu ces approches pour résoudre de manière approximative des POMDP. Des algorithmes à base de méthodes de Monte-Carlo ont également été proposés [Silver and Veness, 2010, Bai et al., 2011], ainsi que des approches heuristiques telles que HSVI (*Heuristic Search Value Iteration*) [Smith and Simmons, 2012]. Les POMDP n'ont pas non plus été épargnés par les techniques d'apprentissage par renforcement qui ont montré de bons résultats notamment sur des jeux vidéo où l'agent n'a accès qu'à une seule image par pas de temps [Hausknecht and Stone, 2015, Igl et al., 2018].

Lorsque l'état de l'environnement est composé d'un sous-ensemble de variables complètement observables, et un autre sous-ensemble de variables partiellement observables, on peut formuler le problème comme un processus décisionnel de Markov à **observabilité mixte** (*Mixed Observability Markov Decision Process* - MOMDP) [Ong et al., 2010]. Les algorithmes de résolution de POMDP peuvent être facilement étendus à la résolution des MOMDP et le processus de résolution est généralement plus simple dû au fait que seulement une partie de l'environnement est partiellement observable.

Pour résoudre des problèmes de récolte d'information, les ρPOMDP ont été introduits par [Araya et al., 2010]. Ce modèle permet de définir une fonction de récompense ρ dépendant de l'état de croyance, par exemple en utilisant l'entropie. Nous reviendrons plus en détail sur ces définitions dans le chapitre 3 sur la perception active. Malheureusement, les fonctions de mesure d'information comme l'entropie n'ont pas la propriété PWLC. On ne peut donc pas utiliser les algorithmes traditionnels pour les résoudre. Cependant, il a été montré que la fonction de valeur optimale d'un ρPOMDP à horizon fini est Lipschitz-continue [Fehr et al., 2018], et cette propriété a pu être exploitée pour étendre l'algorithme HSVI. Plus récemment, un algorithme basé sur les techniques d'échantillonnage de Monte-Carlo a également été proposé [Thomas et al., 2021].

Nous avons désormais présenté les formalismes permettant à un agent d'agir de manière optimale dans un environnement stochastique et partiellement observable. Cependant, ces modèles ne permettent de calculer la stratégie que pour un seul agent. Dans des applications réelles, il est souvent intéressant d'avoir recours à plusieurs agents afin de répartir la charge de travail et d'accomplir une mission de manière plus efficace. Pour cela, nous allons introduire les extensions multi-agents aux processus décisionnels de Markov.

2.4 Processus décisionnels de Markov multi-agents

Dans cette section, nous allons présenter différents modèles qui généralisent les processus décisionnels de Markov pour le cas multi-agents où plusieurs agents évoluent dans le même environnement stochastique et agissent dans un intérêt commun. Lorsque l'on traite de systèmes multi-agents, la communication entre les agents est un aspect important à prendre en compte. Il existe différents modèles basés sur les processus décisionnels de Markov pour tenir compte d'environnements multi-agents. En fonction des modèles, certaines hypothèses sont faites sur le partage d'information entre les agents et le niveau d'observabilité du système. Nous allons commencer par faire un aperçu de ces différents modèles.

2.4.1 L'observabilité et le partage d'information

Comme nous l'avons présenté dans la section sur les systèmes multi-agents, il existe différents types de SMA. Dans cette partie, nous allons nous intéresser aux systèmes multi-agents centralisés et décentralisés.

Système centralisé

Dans le cadre des MDP, tenir compte d'un environnement multi-agents centralisé n'est pas si différent que pour un environnement mono-agent. En effet, étant donné qu'il existe une entité centrale qui collecte les connaissances de chaque agent afin de centraliser le processus de décision, on peut définir un MDP dont l'espace d'états est l'ensemble des états joints de chaque agent, et l'espace d'actions est l'ensemble des actions jointes de chaque agent (et définir les fonctions de transition et de récompense en conséquence). Cette approche a été proposée sous le nom de MMDP (Multiagent Markov Decision Process) [Boutilier et al., 1999].

Définition 2.4.1 (Processus décisionnel de Markov multi-agents)**.** Un processus décisionnel de Markov multi-agents (MMDP) est un tuple $\langle S, A, T, R \rangle$ où :

- $S = S_1 \times \cdots \times S_n$ est l'espace d'états joints, où S_i est l'ensemble fini des états de l'agent i
- $A = A_1 \times \cdots \times A_n$ est l'espace d'actions jointes, où A_i est l'ensemble fini des actions applicables par l'agent i.

— $T : S \times A \times S \to [0, 1]$ est la fonction de transition de l'environnement telle que $T(s, a, s') = P(s' \mid s, a)$. Cela correspond à la probabilité de transiter dans l'état $s' \in S$ lorsque les agents exécutent l'action jointe $a \in A$ dans l'état $s \in S$.
— $R : S \times A \to \mathbb{R}$ est la fonction de récompense, où $R(s, a)$ est la récompense obtenue par les agents lorsqu'ils exécutent l'action jointe $a \in A$ dans l'état $s \in S$.

De la même manière, un MPOMDP [Messias et al., 2011] (*Multiagent Partially Observable Markov Decision Process*) est un POMDP multi-agent centralisé, c'est-à-dire un POMDP dont l'espace d'états est l'ensemble des états joints de chaque agent, l'espace d'actions est l'ensemble des actions jointes de chaque agent, et l'espace d'observations est l'ensemble d'observations jointes de chaque agent. Un cas particulier de MPOMDP est le cas où chaque agent a une observabilité partielle sur l'environnement, mais la combinaison des observations de chaque agent permet de déterminer de manière exacte l'état de l'environnement. Dans ce cas, on dit que l'environnement est **conjointement totalement observable** (*jointly fully observable*), et un tel MPOMDP est équivalent à un MMDP.

Bien que la complexité pour résoudre un MMDP (resp. MPOMDP) soit équivalente à celle pour résoudre un MDP (resp. POMDP) et que les algorithmes classiques peuvent être utilisés, calculer une politique d'actions jointes pour un MMDP ou un MPOMDP est en réalité beaucoup plus coûteux. En effet, étant donné que le nombre d'états joints (et d'actions jointes) est exponentiel en fonction du nombre d'agents, ces techniques sont difficiles à mettre en œuvre lorsqu'il y a un grand nombre d'agents.

Cependant, les systèmes centralisés impliquent un partage d'information constant avec une entité centrale, ce qui n'est pas toujours possible dans des applications réelles, que ce soit pour des raisons d'économie d'énergie, de réduction de coût (embarquer un module de communication sur un robot), ou de sécurité (il peut exister un risque que les communications soient interceptées). C'est pourquoi il a également été développé des MDP décentralisés.

Système décentralisé

Lorsque le contrôle est décentralisé, le manque de communication rend le processus de prise de décision plus difficile. Nous allons ici nous intéresser au cas où il n'y a pas de communication explicite possible entre les agents.

Le formalisme des POMDP peut être étendu à plusieurs agents décentralisés où chaque agent reçoit des observations locales sur l'environnement et prend des décisions en fonction de ces observations. En revanche, le processus de transition du système et les récompenses sont communes à tous les agents en fonction de l'action jointe effectuée. Nous nommons ce modèle un processus décisionnel de Markov décentralisé et partiellement observable (DEC-POMDP) [Bernstein et al., 2002].

Dans un tel système, l'observabilité est un critère important à prendre en compte. On émet généralement différentes hypothèses sur l'observabilité de l'environnement, de la plus à la moins restrictive [Pynadath and Tambe, 2002] :

1. **Observabilité complète** : chaque agent peut observer l'état complet de l'environnement à chaque instant de manière exacte
2. **Observabilité jointe** : la combinaison des observations des agents permet de déterminer l'état de l'environnement de manière exacte
3. **Observabilité partielle** : la combinaison des observations des agents ne permet pas de déterminer l'état de l'environnement

4. **Observabilité nulle** : les agents ne peuvent pas observer l'environnement

Le cas de l'observabilité nulle est un cas très particulier que nous n'aborderons pas dans le cadre de cette thèse. Lorsque l'observabilité est complète ou jointe, un DEC-POMDP peut être ramené à un DEC-MDP, c'est-à-dire un processus décisionnel de Markov décentralisé. Dans le cas contraire, un DEC-POMDP à observabilité partielle est l'extension naturelle des POMDP dans un environnement multi-agents décentralisé.

De ce fait, un DEC-POMDP peut être vu comme une généralisation des modèles de MDP, POMDP et DEC-MDP. La figure 2.3 illustre les relations entre ces différents modèles.

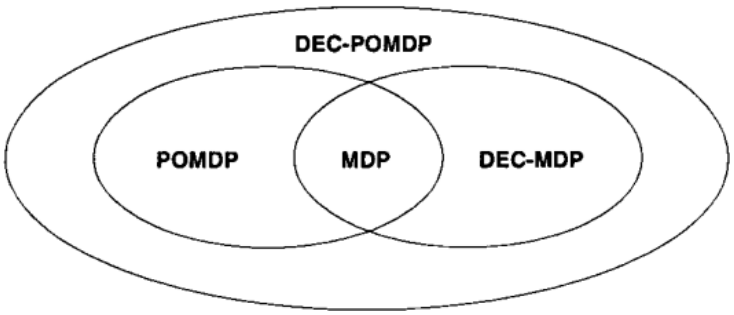


FIGURE 2.3 – Les relations entre les différents modèles de MDP (de [Bernstein et al., 2002]).

Nous allons désormais définir le modèle le plus générique, les processus décisionnels de Markov décentralisés et partiellement observables.

2.4.2 Le formalisme des DEC-POMDP

Les processus décisionnels de Markov décentralisés et partiellement observables (DEC-POMDP) [Bernstein et al., 2002] sont une extension des POMDP au cas multi-agents décentralisé, c'est-à-dire où chaque agent fait des observations locales sur l'environnement, et où il n'y a pas de communication explicite entre les agents.

Définition 2.4.2 (Processus décisionnel de Markov décentralisé et partiellement observable)**.** Un processus décisionnel de Markov décentralisé et partiellement observable (DEC-POMDP) est un tuple $\langle \mathcal{D}, \mathcal{S}, \mathcal{A}, T, R, \Omega, O, h, I \rangle$ où [Oliehoek, 2012] :

- $\mathcal{D} = \{1, \ldots, n\}$ est un ensemble de n agents
- $\mathcal{S}$ est un ensemble fini des états de l'environnement
- $\mathcal{A} = A_1 \times \cdots \times A_n$ est l'ensemble fini des actions jointes, où A_i est un ensemble fini des actions applicables par l'agent i
- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ est la fonction de transition de l'environnement telle que $T(s, a, s') = P(s' \mid s, a)$. Cela correspond à la probabilité de transiter dans l'état $s' \in \mathcal{S}$ lorsque les agents exécutent l'action jointe $a \in \mathcal{A}$ dans l'état $s \in \mathcal{S}$.
- $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ est la fonction de récompense, où $R(s, a)$ est la récompense obtenue par les agents lorsqu'ils exécutent l'action jointe $a \in \mathcal{A}$ dans l'état $s \in \mathcal{S}$.
- $\Omega = \Omega_1 \times \cdots \times \Omega_n$ est l'ensemble fini des observations jointes, où Ω_i est un ensemble fini des observations possibles par l'agent i
- $O : A \times \mathcal{S} \times \Omega \to [0, 1]$ est la fonction d'observation telle que $O(a, s, o) = P(o \mid s, a)$. Cela correspond à la probabilité que les agents reçoivent l'observation jointe $o \in \Omega$ lorsque le système a transité dans l'état $s' \in \mathcal{S}$ après avoir exécuté l'action jointe $a \in \mathcal{A}$ dans l'état $s \in \mathcal{S}$.

— h est l'horizon du problème
— $I : \mathcal{P}(\mathcal{S})$ est la distribution d'états initiale au temps $t = 0$

La difficulté des DEC-POMDP réside dans le manque d'information des agents par rapport aux observations des autres agents. En effet, dans un POMDP classique, nous avons introduit la notion d'état de croyance qui est une statistique suffisante pour raisonner de manière optimale. Malheureusement, dans un DEC-POMDP, nous ne pouvons pas maintenir un état de croyance sur l'espace d'états car cela nécessiterait de connaître les observations reçues par tous les agents. De ce fait, nous devons nous en remettre à l'utilisation des historiques d'observations.

Définition 2.4.3 (Historique d'observation [Oliehoek, 2012])**.** L'historique d'observation pour un agent i, noté $\bar{\omega}_i$, est défini comme la séquence d'observations qu'un agent a reçu. Au pas de temps t, l'historique d'observation est :

$$\bar{\omega}_i^t = (\omega_i^1, \ldots, \omega_i^t) \tag{2.15}$$

L'historique d'observation joint au temps t, noté $\bar{\omega}^t$, est l'historique d'observation pour tous les agents : $\bar{\omega}^t = \langle \bar{\omega}_1^t, \ldots, \bar{\omega}_n^t \rangle$. L'ensemble de tous les historiques d'observations pour un agent i au temps t est noté $\bar{\Omega}_i^t$, et l'ensemble de tous les historiques d'observations possibles est noté $\bar{\Omega}$.

De cette définition, nous pouvons définir une politique individuelle $\pi_i : \bar{\Omega}_i \to A_i$ qui est une fonction qui associe une action à un historique d'observation. Une politique jointe $\pi : \bar{\Omega} \to \mathcal{A}$ est l'ensemble des politiques de tous les agents : $\pi = (\pi_1, \ldots, \pi_n)$.

Comme pour les modèles de MDP classiques, nous définissons la fonction de valeur d'une politique comme la récompense totale espérée. Cependant, nous la définissons sur l'espace d'états et d'observations, et non plus seulement sur l'espace d'états. Formellement, la fonction de valeur au temps t est [Oliehoek, 2012] :

$$V^\pi(s^t, \bar{\omega}^t) = \begin{cases} R(s^t, \pi(\bar{\omega}^t)) & \text{si } t = h - 1 \\ R(s^t, \pi(\bar{\omega}^t)) + \sum_{s^{t+1} \in \mathcal{S}} \sum_{\omega \in \Omega} P(s^{t+1}, \omega \mid s^t, \pi(\bar{\omega}^t)) V^\pi(s^{t+1}, \bar{\omega}^{t+1}) & \text{sinon} \end{cases} \tag{2.16}$$

Calculer une politique optimale pour un DEC-POMDP est une tâche très complexe, en particulier à cause du nombre d'historiques d'observations qui croît de manière exponentielle avec l'horizon du problème. Au temps t, il y a $(|A_i| \times |\Omega_i|)^t$ historiques d'observations possibles pour l'agent i. [Bernstein et al., 2002] ont montré que résoudre un DEC-POMDP à horizon fini est un problème NEXP-complet lorsque $n \geq 2$. Cependant, ce n'est pas seulement l'observabilité partielle qui est en cause, car résoudre un DEC-MDP est également un problème NEXP-complet lorsque $n \geq 3$.

Dans la section suivante, nous allons présenter les principales solutions qui ont été proposées pour résoudre des DEC-POMDP ainsi que d'autres extensions des DEC-POMDP qui ont été introduites.

2.4.3 Algorithmes et autres formalismes

Bien que la complexité des DEC-POMDP soit importante, des algorithmes ont été développés pour les résoudre de manière exacte ou approchée, et d'autres formalismes ont émergé afin de simplifier la formalisation des problèmes ou de profiter d'hypothèses sur l'environnement pour faciliter la résolution.

[Szer et al., 2012] ont proposé MAA* (Multi-agent A*), une approche heuristique pour résoudre les DEC-POMDP à horizon fini de manière complète et optimale, en reprenant le principe de l'algorithme A*.

Les processus décisionnels de Markov sont étroitement liés à la théorie des jeux. En particulier, les jeux stochastiques partiellement observables (POSG) [Hansen et al., 2004] sont une généralisation des modèles de MDP que nous avons présentés dans ce chapitre. L'algorithme JESP (*Joint Equilibrium-based Search for Policies*) [Nair et al., 2003] est un algorithme reprenant certains concepts de la théorie des jeux. Cet algorithme fonctionne sur le principe de **best response** (meilleure réponse) : on commence par fixer une politique (par exemple aléatoire) pour tous les agents, puis on cherche la meilleure réponse pour l'agent i, puis la meilleure réponse pour l'agent j, et ainsi de suite jusqu'à convergence. L'inconvénient de cette méthode est que l'on n'est pas garanti de trouver une politique optimale. En effet, selon les politiques initiales choisies, on peut trouver une politique jointe qui est un optimum local.

Des approches ont également été développées afin d'exploiter l'indépendance des interactions entre les agents. Les ND-POMDP [Nair et al., 2005, Varakantham et al., 2007] (*Network distributed POMDP*) utilisent l'indépendance des transitions et des observations pour factoriser les modèles de DEC-MDP et DEC-POMDP. L'hypothèse de l'indépendance des transitions étant une hypothèse forte, d'autres modèles ont été proposés pour exploiter uniquement l'indépendance des observations [Oliehoek et al., 2008].

Récemment, un algorithme a été proposé dont le but est de transformer un DEC-POMDP en MDP à espace d'états continu [Dibangoye et al., 2016] et a montré un meilleur passage à l'échelle que les autres algorithmes de l'état de l'art.

Des bibliothèques logicielles ont été développées dans le but de proposer à la communauté scientifique des implémentations performantes des principaux algorithmes de l'état de l'art, notamment MADP Toolbox [Oliehoek et al., 2017] et AI-Toolbox [Bargiacchi et al., 2020].

[Beynier and Mouaddib, 2005] ont introduit les OC-DEC-MDP (*Opportunity Cost DEC-MDP*), un modèle qui permet de tenir compte de contraintes temporelles et de précédence dans le but d'accomplir un ensemble de tâches par plusieurs agents. Ils ont également montré que les OC-DEC-POMDP sont de complexité polynomiale et ont proposé un algorithme pour les résoudre.

Les MTDP (*Multiagent Team Decision Process*) [Pynadath and Tambe, 2002] sont un autre formalisme permettant de résoudre des problèmes de prise de décision sous incertitude dans les systèmes multi-agents décentralisés. Il a été montré que la classe de complexité de ce formalisme est équivalent à celle des DEC-POMDP [Seuken and Zilberstein, 2008].

Des modèles ont été développés pour permettre la communication explicite entre les agents par le biais de messages. Les DEC-POMDP-COM fournissent un cadre formel où les agents peuvent communiquer une information aux autres agents, que ce soit une observation ou tout autre type de message prédéfini dans le modèle. Dans un DEC-POMDP-COM, en plus de la politique d'action classique, on calcule une politique de communication, qui associe un historique d'observations et un historique de messages reçus à un message à transmettre. Les DEC-POMDP-COM introduisent également une fonction de coût qui correspond au coût de transmission d'un message. Il a été montré qu'il est possible de réduire un DEC-POMDP-COM à un DEC-POMDP équivalent [Goldman and Zilberstein, 2003].

De la même manière, les COM-MTDP [Pynadath and Tambe, 2002] introduisent la communication explicite au formalisme des MTDP, de manière assez similaire aux DEC-POMDP-COM, c'est-à-dire que l'on calcule une politique de communication en plus de la politique d'action. Comme pour les MTDP et les DEC-POMDP, il a été montré que les COM-MTDP et les DEC-POMDP-COM sont équivalents [Seuken and Zilberstein, 2008]. Étant donné qu'un DEC-POMDP-COM peut être réduit à un DEC-POMDP équivalent, tous ces modèles (DEC-POMDP,

DEC-POMDP-COM, MTDP, COM-MTDP) sont équivalents [Seuken and Zilberstein, 2008]. De ce fait, la complexité de tous ces modèles est NEXP-complet. Cependant, il peut parfois être intéressant de considérer la communication de manière explicite, par exemple pour raisonner sur une stratégie de communication indépendamment de la stratégie d'action.

Nous avons présenté succinctement les grandes extensions des modèles de DEC-PODMP. Pour une revue plus détaillée de ces différents modèles, le lecteur intéressé peut se référer à [Amato et al., 2013].

Les I-POMDP (*Interactive POMDP*) [Gmytrasiewicz and Doshi, 2005] étendent le modèle des POMDP en intégrant à l'état de croyance une croyance sur les autres agents du système. Plus spécifiquement, un agent raisonne sur ses croyances sur l'état des autres agents, qui eux raisonnent sur leurs croyances sur l'état des autres agents, et ce récursivement pour une profondeur de raisonnement donnée, potentiellement infinie. [Doshi et al., 2009] ont proposé une représentation graphique des I-POMDP, les Diagrammes d'Influence Interactifs (*Interactive Influence Diagram*, I-ID) et les Diagrammes d'Influence Interactifs Dynamiques (*Interactive Dynamic Influence Diagram*, I-DID), qui généralisent les représentations en Diagrammes d'Influence et Diagrammes d'Influence Dynamiques des POMDP [Boutilier and Poole, 1996] au cas des systèmes multi-agents.

2.5 Conclusion

Dans ce chapitre, nous avons présenté les modèles couramment utilisés pour formaliser et résoudre des problèmes de planification sous incertitude, et plus particulièrement les processus décisionnels de Markov. Nous avons commencé par un cadre simple où l'environnement est totalement observable avec un seul agent pour ensuite introduire l'observabilité partielle de l'environnement et les états de croyance. Enfin, nous avons présenté les extensions de ces modèles au cas des systèmes multi-agents avec les difficultés computationnelles occasionnées. La forte complexité de ces modèles est une des raisons qui ont mené à nos travaux. Dans cette thèse, nous nous intéressons à des problématiques de récolte d'information dans les systèmes multi-agents. Les modèles classiques de MDP ne peuvent être utilisés directement pour résoudre des problèmes de récolte d'information car les agents obtiennent des récompenses sur l'état du système et non pas sur leurs connaissances et, bien que des extensions des POMDP pour la récolte d'information ont été proposées, ces modèles ne tiennent compte que d'un seul agent. Dans le chapitre suivant, nous allons introduire la théorie sur la récolte d'information en intelligence artificielle par le biais de la perception active, qui consiste à prendre des décisions dans le but d'obtenir de l'information sur son environnement.

Chapitre 3

Perception active

Sommaire

3.1 Introduction

Initialement, la perception active (*active sensing* ou *active perception*) a été définie comme "le problème d'optimisation des stratégies appliquées au processus d'acquisition des données, qui dépendent de l'état actuel de l'interprétation des données et de l'objectif ou de la tâche du processus" [Bajcsy, 1988].

Mihaylova et ses collègues [Mihaylova et al., 2003] ont ensuite défini la perception active comme "le processus de déterminer les entrées en optimisant un critère de performance, qui est fonction à la fois des coûts et de l'utilité". Ils évoquent les raisons suivantes qui font que la perception active est une tâche difficile :

— Les modèles des agents et des capteurs sont non linéaires et, bien que certaines méthodes linéarisent les modèles, beaucoup de problèmes non-linéaires ne peuvent pas être résolus de cette manière.
— La résolution d'une tâche dépend d'un critère d'optimalité qui est une fonction multi-objective qui prend en compte à la fois le gain en information et d'autres utilités ou coûts. La complexité de tels systèmes est donc élevée, ce qui peut poser problème en particulier pour la planification en ligne.
— L'incertitude dans le modèle de l'agent, dans l'environnement et dans les capteurs utilisés doit être prise en compte.

— Souvent, les informations obtenues par une observation n'informent pas l'agent sur l'ensemble des variables de l'environnement, c'est-à-dire que le système est partiellement observable.

Dans le cadre des systèmes multi-agents, la perception active ajoute des nouvelles difficultés, en particulier pour les agents hétérogènes qui peuvent avoir des capteurs différents, ce qui fait qu'ils peuvent apporter des informations complémentaires mais également conflictuelles, voire incohérentes et moins précises, ce qui augmente encore la complexité des modèles [Chung et al., 2004].

Le problème de la perception active a massivement été étudié [Floreano and Mondada, 1994, Weyns et al., 2004, Bajcsy et al., 2018] et appliqué à différents domaines tels que la surveillance et l'exploration [Renoux, 2015].

Comme nous l'avons évoqué, un des problèmes majeurs de la perception active est l'incertitude sur différents aspects du système. Nous allons maintenant définir concrètement ce qu'est l'incertitude, ses origines, et comment la mesurer mathématiquement.

3.2 Information et incertitude

Nous définissons le problème de récolte d'information comme le problème de réduire l'incertitude d'un agent sur son environnement. De ce fait, nous devons d'abord déterminer ce qu'est l'incertitude, comment elle peut intervenir dans le monde réel ou virtuel des agents, et comment formaliser et mesurer l'incertitude.

3.2.1 L'origine de l'incertitude

L'incertitude peut apparaître sous différentes formes, mises en évidence par [Dubois, 2018] : l'incertitude de l'aléatoire, l'incertitude épistémique, et l'incertitude de l'incohérence. Ces incertitudes peuvent provenir d'origines différentes, comme illustré par la figure 3.1.

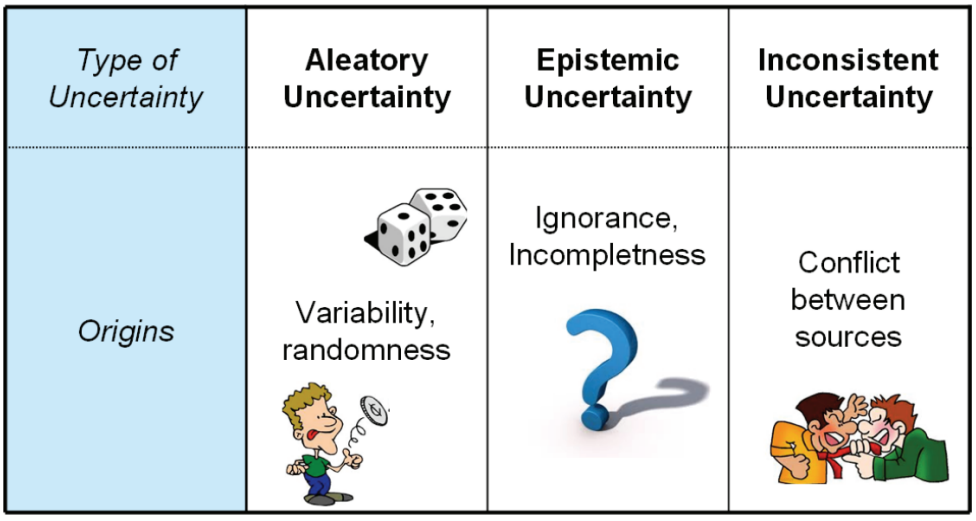


FIGURE 3.1 – L'origine de l'incertitude (de [Bellenger, 2013])

L'incertitude de l'aléatoire provient de la non-prédictibilité des expériences que nous menons. Par exemple, lorsque nous jetons un dé, il est impossible de prédire à l'avance sur quelle face il va tomber. En revanche, nous pouvons mesurer cette incertitude, car nous savons que le dé va

tomber sur l'une des 6 faces de manière équiprobable (dans le cas d'un dé à 6 faces non truqué). De la même manière, si nous utilisons un capteur imprécis, nous pouvons généralement mesurer le taux de bruit de ce capteur. L'incertitude épistémique, en revanche, ne peut pas être mesurée, car cela représente l'incertitude sur les connaissances qui peuvent être incomplètes. L'incertitude sur l'incohérence, quant à elle, correspond à l'incertitude des informations de multiples sources qui peuvent être incompatibles, et dans ce cas, la difficulté est de déterminer quelle source est la plus fiable.

Dans le cadre de ces travaux, nous nous intéressons particulièrement à l'incertitude de l'aléatoire, qui a beaucoup été étudiée et qui peut être mesurée, comme nous allons le présenter dans la section suivante.

3.2.2 La théorie de l'information et l'entropie de Shannon

La théorie de l'information a été originalement décrite dans les années 1940 par le mathématicien américain **Claude Shannon** [Shannon, 1948]. Historiquement, la théorie de l'information a été développée dans une optique de faciliter les télécommunications, en particulier de pouvoir transmettre des informations le plus rapidement possible en tenant compte du fait que les messages transmis peuvent être altérés durant la transmission. La théorie de l'information s'est ensuite étendue à d'autres domaines, notamment la physique, la biologie, le traitement du langage, et d'autres encore.

Un des principes fondamentaux de la théorie de l'information est l'entropie de Shannon, qui permet de mesurer l'incertitude sur une source d'information.

L'entropie de Shannon

L'entropie de Shannon, plus communément appelée entropie, est une mesure de l'incertitude (désordre) relative à une source d'information, comme par exemple un fichier texte, un message transmis entre un émetteur et un récepteur, ou un signal électrique. Dans notre cas, nous modéliserons une source d'information comme étant une variable aléatoire discrète. Le but est de déterminer quelle est l'incertitude sur cette variable.

Supposons une variable aléatoire discrète X à n valeurs $\{x_1, \ldots, x_n\}$. Formellement, l'entropie en base b de X, notée $H_b(X)$, est définie de la manière suivante :

$$H_b(X) = -\sum_{i=1}^{n} P(x_i) \log_b P(x_i) \tag{3.1}$$

où $P(x_i)$ est la probabilité que la variable aléatoire X soit à la valeur x_i, et $\log_b$ le logarithme en base b.

La base utilisée pour le logarithme dépend de l'application visée. Généralement, en informatique, on utilise un logarithme en base 2, car cela permet de mesurer l'information en nombre de **bits**. Dans le cadre de cette thèse, nous considérerons l'entropie en base 2, que nous noterons simplement $H(X)$.

La figure 3.2 illustre la valeur de l'entropie en base 2 pour une variable binaire (domaine $\{0, 1\}$). Nous notons $P(x) = P(x = 1)$ la probabilité que la variable X soit à la valeur 1, et $P(\bar{x}) = P(x = 0) = 1 - P(x)$ la probabilité que la variable X soit à la valeur 0. On remarque que lorsque $P(x) = 1$ ou $P(x) = 0$, l'entropie est minimale : on connaît exactement la valeur de la variable, il n'y a aucune incertitude. À l'inverse, lorsque $P(x) = 0.5$, l'entropie est maximale : on n'a aucune information sur la valeur de cette variable.

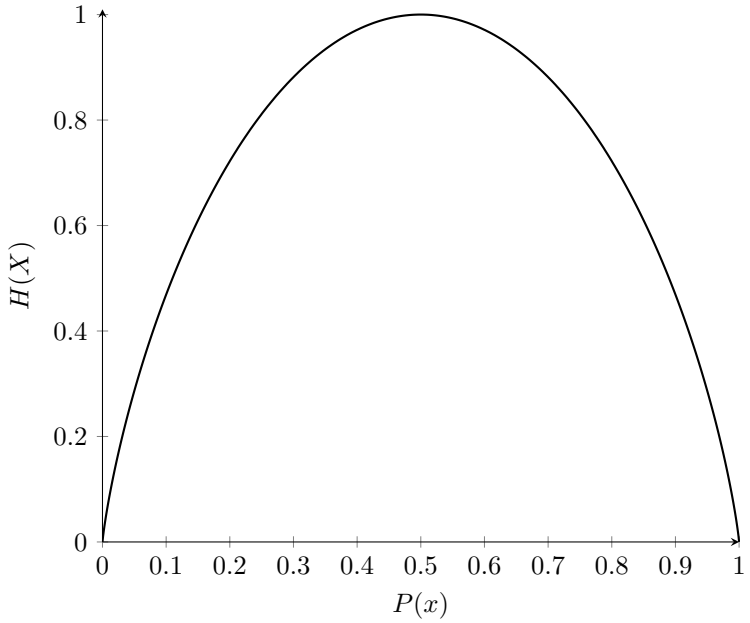


FIGURE 3.2 – Entropie en base 2 pour une variable binaire.

Dans le cas général, c'est-à-dire pour des variables aléatoires à n valeurs, $H(X)$ est minimale lorsque l'une des valeurs a une probabilité de 1 :

$$\exists i \in [1..n], P(x_i) = 1 \tag{3.2}$$

Contrairement aux variables binaires, l'entropie n'est pas forcément minimale lorsque la probabilité d'une valeur du domaine est égale à 0. En effet, on peut très bien avoir $P(x_1) = 0$ et $P(x_2) = 0.5, P(x_3) = 0.5$. L'entropie est maximale lorsque la distribution est uniforme :

$$\forall i \in [1..n], P(x_i) = \frac{1}{n} \tag{3.3}$$

L'entropie maximale, notée $H_{max}(X)$, est définie de la manière suivante :

$$H_{max}(X) = \log_2(n) \tag{3.4}$$

L'entropie est additive pour le cas de variables indépendantes. C'est-à-dire que si l'on cherche à mesurer la quantité d'information contenue dans plusieurs sources d'informations indépendantes, on peut additionner les entropies de ces sources. Par exemple, pour deux variables indépendantes X et Y, l'entropie est définie telle que :

$$H(X, Y) = H(X) + H(Y) \tag{3.5}$$

Lorsque l'on cherche à résoudre des problèmes de récolte d'information, on préfère généralement utiliser l'entropie négative plutôt que l'entropie standard. L'entropie négative, notée $H^-(X)$, est définie de la manière suivante :

$$H^-(X) = H_{max}(X) - H(X) \tag{3.6}$$

L'entropie négative est maximale lorsque l'incertitude est nulle, et minimale lorsque l'incertitude est totale. On peut donc utiliser l'entropie négative pour récompenser un agent en fonction de la quantité d'information apportée par l'exécution d'une action.

Nous avons introduit les notions permettant de mesurer l'information. Dans les sections suivantes, nous allons présenter les problèmes de perception active qui nous intéressent dans le cadre de cette thèse, c'est-à-dire les problèmes de récolte d'information et l'application à l'exploration de carte par un ou plusieurs agents autonomes.

3.3 La récolte d'information

La récolte d'information, dans le cadre de la perception active, est le problème dans lequel un agent agit dans son environnement de manière à maximiser ses connaissances sur ce dernier. Dans cette thèse, nous nous intéresserons particulièrement au fait d'améliorer son état de croyance sur l'environnement, c'est-à-dire avoir un état de croyance qui minimise l'incertitude. Nous allons présenter ici différents modèles de décision qui ont été développés dans un but de récolte d'information. Cette section a été rédigée en partie avec l'aide des travaux de [Renoux, 2015].

3.3.1 Modèles de décision pour la récolte d'information

Beaucoup de travaux en récolte d'information reposent sur la classification d'éléments présents dans l'environnement. [Guo, 2003] proposent un modèle utilisant les POMDP, où les actions permettent de déployer un capteur dans le but de recevoir une classification. L'agent est récompensé lorsqu'il reçoit une classification correcte, sinon il est pénalisé. De ce fait, l'agent doit attendre d'avoir suffisamment d'information pour avoir la certitude que sa classification est correcte. Les récompenses sont fixées en fonction de l'état en lui-même et non en fonction de son état de croyance, de ce fait cette approche reste dans le cadre des POMDP classiques. [Ji and Carin, 2007] ont proposé une approche similaire, avec l'ajout d'un modèle de Markov caché (*Hidden Markov Model*) pour tenir compte de l'ordre précis de réception des observations. Par la suite, [Spaan et al., 2010b] ont appliqué ce processus de classification au cas multi-agents de manière à faire de la perception active coopérative en utilisant des actions de classification. La perception coopérative tient dans le modèle d'observation, dû au fait que les taux de faux positifs et de faux négatifs sont différents en fonction de la position et des caractéristiques des capteurs. Les agents sont récompensés la première fois qu'ils classifient correctement un événement intéressant, et pénalisés la première fois qu'ils ne le classifient pas correctement.

Cependant, tous les problèmes de perception active ne peuvent être définis en utilisant des actions de classification. Dans certaines applications, l'objectif doit être explicitement décrit comme étant la réduction de l'incertitude sur l'état de l'environnement. Dans ce cas, les POMDP ne sont pas adaptés, car la fonction de récompense doit être exprimée sur l'état de croyance au lieu de l'état réel de l'environnement. Dans [Spaan, 2008], les auteurs ont présenté la problématique des récompenses basées sur l'état de croyance et ont relevé des problèmes que ce soit pour la modélisation ou la résolution. Cette étude présente seulement le problème d'un point de vue théorique et ne propose pas de solution pratique pour le résoudre. Par ailleurs, il traite uniquement du cas mono-agent, et le cas multi-agents n'est pas pris en compte. [Eidenberger and Scharinger, 2010] présentent un modèle basé sur les POMDP pour la perception active, où un agent doit déployer ses capteurs de manière à reconnaître des objets dans l'environnement. Ici, la fonction de récompense est basée sur l'état de croyance attendu après avoir appliqué l'action, en tenant compte du coût de cette action. Cependant, les algorithmes classiques ne peuvent être utilisés en particulier dû au fait que la fonction de récompense n'a pas la propriété PWLC. Ensuite, [Araya et al., 2010] ont introduit les ρPOMDP, qui est une extension des POMDP où la fonction de récompense peut être définie sur l'espace d'états de croyance. [Satsangi et al., 2015] ont proposé une approche basée sur les POMDP ainsi qu'un algorithme pour sélectionner les capteurs de manière dynamique.

En utilisant les ρPOMDP et l'entropie négative comme fonction de récompense, ils ont montré la sous-modularité de la fonction de valeur. Cependant, ce travail ne considère qu'un seul agent pour choisir les capteurs. [Renoux et al., 2015] se sont intéressés au cas multi-agents et ont proposé un modèle basé sur les POMDP avec un état de croyance étendu où chaque agent possède sa propre croyance sur l'environnement mais également des croyances sur les autres agents. Cette approche autorise également la communication par le biais d'actions et d'une politique de communication calculée conjointement à la politique d'action, et met en place des mécanismes de pertinence de l'information afin de déterminer si l'information reçue par un agent est pertinente ou non. Malheureusement, ce modèle souffre d'une forte complexité.

Nous avons présenté ici des modèles de décision permettant de faire de la récolte d'information, et en particulier de réduire l'incertitude sur l'état de croyance d'un agent. Nous allons désormais nous intéresser au problème de récolte d'information dans le but de l'exploration, c'est-à-dire l'exploration d'une carte par un ou plusieurs agents autonomes.

3.4 Le problème de l'exploration de carte

L'exploration de carte par un agent autonome est un problème de récolte d'information dans lequel un agent explore son environnement dans le but de construire une carte de ce dernier (cartographie). Lorsque l'on n'a aucune information au préalable sur l'environnement, la position initiale de l'agent dans l'environnement ne peut pas être connue. De ce fait, ce problème est généralement traité conjointement avec le problème de localisation dans l'espace, sous le nom Cartographie et Localisation Simultanées [Durrant-Whyte and Bailey, 2006] (*Simultaneous Localization And Mapping*, SLAM). Dans cette thèse, nous considérerons que la position initiale de l'agent dans l'environnement est connue. De ce fait, le problème de la localisation ne sera pas traité ici. Les lecteurs intéressés peuvent néanmoins se référer à [Aulinas et al., 2008, Placed et al., 2023] pour des revues approfondies sur les principaux travaux existants en SLAM.

Concrètement, un robot mobile embarque un capteur capable de détecter les obstacles et leurs distances, typiquement un capteur de type *Laser rangefinder* (télémètre laser), qui envoie des rayons lasers dans une ou plusieurs directions qui sont ensuite réfléchis au contact d'un obstacle. Ensuite, en connaissant le temps écoulé entre l'envoi du rayon et le retour (*time of flight*), on peut mesurer la distance à l'obstacle avec une certaine précision (et si le rayon ne revient pas, on sait alors qu'il n'y a aucun obstacle dans cette direction). Cette technique est très efficace et permet de construire des cartes, que ce soit dans des environnements intérieurs ou extérieurs. La figure 3.3 montre un exemple de carte construite à partir de cette technique. Les zones grises correspondent aux zones pas encore explorées, les zones blanches correspondent aux zones libres, et les zones noires correspondent aux obstacles.

En mémoire, les cartes sont représentées sous la forme d'une grille d'occupation ou d'une grille d'évidences. Une grille d'occupation est une grille dont chaque cellule est soit libre, soit occupée, soit inconnue. Durant l'exploration, la grille est mise à jour selon les informations recueillies par l'agent. Les grilles d'évidences fonctionnent sur le même principe mais en incluant des probabilités (probabilité que la cellule soit occupée) [Moravec and Elfes, 1985]. Une grille d'évidences peut être ramenée à une grille d'occupation après l'application d'un seuil pour délimiter à partir de quelle probabilité on considère qu'une cellule est vraiment occupée.

La question qui se pose est donc la suivante : comment explorer l'environnement de manière efficace ? En effet, un robot pourrait simplement se déplacer de manière aléatoire dans l'environnement tout en évitant les obstacles, seulement ce ne serait pas très efficace. Au lieu de ça, nous allons utiliser des techniques de perception active afin que l'agent se déplace dans

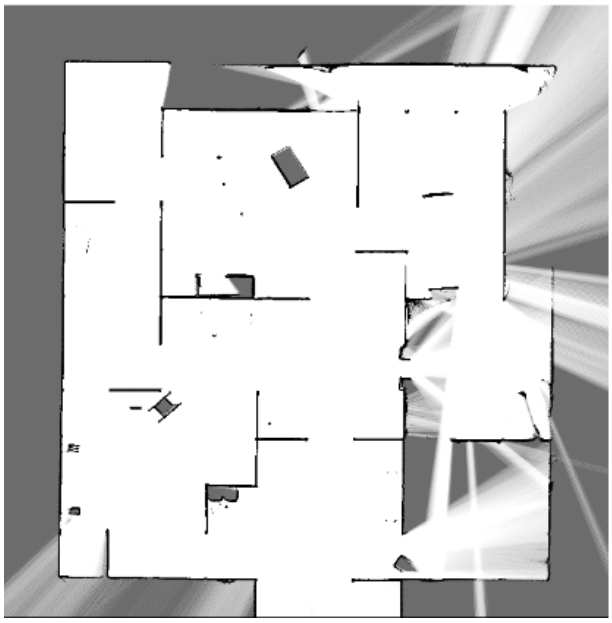


FIGURE 3.3 – Exemple de carte construite par un robot autonome

l'environnement avec le but précis d'explorer les endroits inconnus.

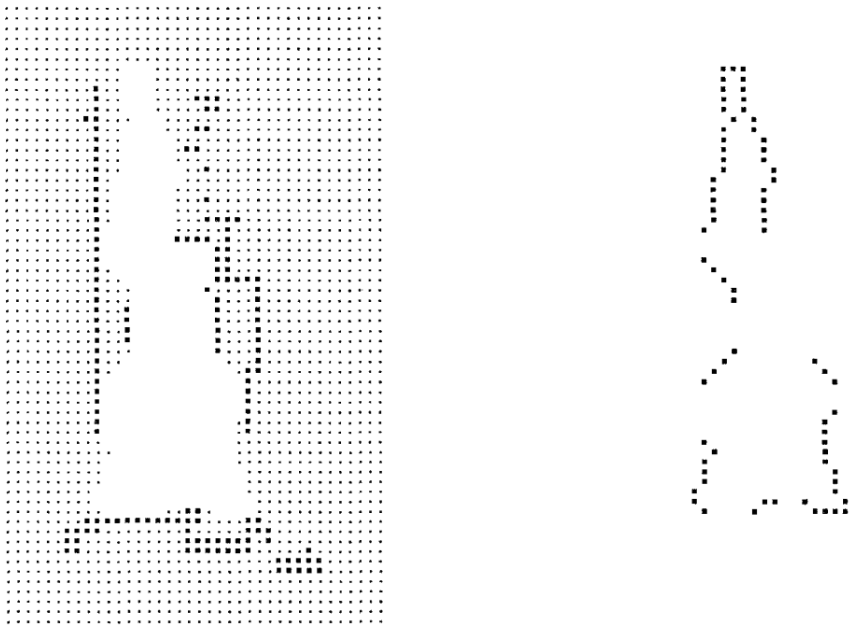
3.4.1 Exploration à base de frontières

Dans un premier temps, nous allons considérer le problème d'exploration de carte dans un environnement mono-agent. La technique la plus populaire pour résoudre de tels problèmes est l'**exploration à base de frontières**, introduite par [Yamauchi, 1997] (*frontier-based exploration*). Le principe est de détecter les frontières entre les zones non explorées et les zones déjà explorées, car ce sont les endroits où le potentiel gain d'information est le plus élevé. Ensuite, l'utilisation d'un algorithme de recherche de chemin permet de calculer un chemin pour que le robot se déplace près d'une frontière. La figure 3.4.1 illustre cette méthode.

Le problème du choix de la frontière est important car cela va déterminer la capacité de l'agent à explorer efficacement l'environnement. Généralement, on appelle ce problème la recherche du *Best View Point* (meilleur point de vue). Dans la publication originale, [Yamauchi, 1997] propose de choisir la frontière la plus proche qui est accessible et qui n'a pas encore été visitée. Ensuite, d'autres critères de sélection ont été proposés. [Bachrach et al., 2009] proposent de choisir la frontière qui est à la fois la plus susceptible d'apporter beaucoup d'information, mais également qui permette d'améliorer le processus de localisation de l'agent. Des solutions utilisent la théorie de l'information pour estimer le potentiel gain en information apporté par une frontière [Amigoni and Caglioti, 2010, Charrow et al., 2015]. Le processus de détection des frontières pouvant être coûteux, des améliorations ont été proposées pour détecter les frontières uniquement à partir des nouvelles données reçues [Keidar and Kaminka, 2014].

D'autres techniques d'exploration ont été proposées. Le *greedy mapping* (exploration gloutonne), qui consiste à toujours se déplacer en direction de l'endroit inconnu le plus proche, a été étudié [Koenig et al., 2001]. Dans [Lauri and Ritala, 2016], les auteurs utilisent une combinaison de l'exploration à base de frontière et de POMDP pour choisir en fonction de la situation quelle stratégie d'exploration adopter.

Le problème de l'exploration de carte a très largement été étudié au cours de ce siècle et



(a) Grille d'occupation (b) Frontières

FIGURE 3.4 – Exploration à base de frontières (de [Yamauchi, 1997])

nous n'avons ici présenté que quelques approches communément utilisées. Pour une étude approfondie du sujet, [Thrun et al., 2002] et plus récemment [Lluvia et al., 2021] passent en revue les différentes solutions existantes.

Cependant, les méthodes d'exploration que nous avons présentées tiennent compte d'un environnement déterministe, c'est-à-dire un environnement où les transitions sont parfaitement prévisibles. Nous allons désormais présenter une approche qui permet de faire de la cartographie dans un environnement stochastique.

3.4.2 Exploration sous incertitude

Nous avons longuement abordé la planification sous incertitude dans le chapitre 2 au travers de différents modèles basés sur les processus décisionnels de Markov, qui permettent de formaliser des problèmes de prise de décision sous incertitude. Dans cette section, nous allons principalement nous appuyer sur les travaux de [Le Gloannec et al., 2010], où les auteurs considèrent un problème d'exploration mono-agent dans un environnement inconnu et stochastique et proposent une approche utilisant les MDP pour faire de l'exploration. Le problème de la localisation dans l'espace n'est pas considéré ici, c'est-à-dire que l'agent sait à chaque instant où il se situe.

L'environnement est représenté comme une grille d'occupation hexagonale où chaque cellule peut être soit libre, soit occupée, soit inconnue (non explorée). Plus précisément, le modèle est divisé en trois couches, comme illustré dans la figure 3.5. L'environnement réel est représenté dans la première couche, qui est l'environnement dans lequel l'agent évolue. La carte produite par l'agent est stockée dans la couche de pixels, qui se présente sous la forme d'une image. La dernière couche est la couche de décision, qui est une abstraction de la couche de pixels, utilisée comme environnement pour le calcul du MDP. L'utilisation d'hexagones permet à l'agent de se déplacer de manière fluide dans 6 directions tout en gardant une matrice de transition facile à définir.

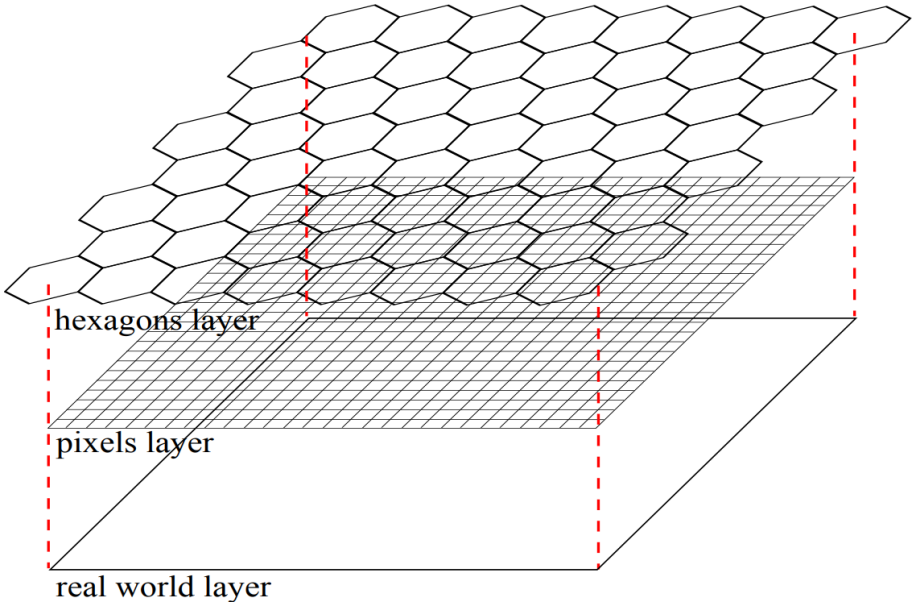


FIGURE 3.5 – Grille d'occupation hexagonale. Les caractéristiques du monde réel sont projetées simultanément sur une grille de pixels et sur une carte de décision hexagonale (de [Le Gloannec et al., 2010])

Le problème de l'exploration de carte étant partiellement observable, les POMDP pourraient être utilisés pour le résoudre. Malheureusement, une des difficultés principales est la taille de la grille. En effet, généralement les politiques de (PO)MDP sont calculées hors-ligne avant l'exécution de la mission. Cependant, cela nécessiterait de considérer un espace d'états qui contient l'ensemble des grilles possibles. Autrement dit, pour une grille de taille $n \times m$, il y aurait 3^{nm} états de grille possibles : chaque cellule de la grille peut être soit libre, soit occupée, soit inconnue. De ce fait, il est impossible de calculer une politique hors-ligne même pour des grilles de taille raisonnable. La solution proposée par [Le Gloannec et al., 2010] est de calculer la politique en ligne en considérant l'état actuel de la grille comme une connaissance de l'agent et non comme faisant partie de l'état du système. Concrètement, le processus d'exploration se déroule de la manière suivante :

1. acquérir des données de l'environnement, par exemple avec un capteur de type Laser rangefinder
2. déterminer les endroits intéressants à explorer
3. calculer une politique pour atteindre un ou plusieurs de ces endroits
4. exécuter la politique
5. répéter

Étant donné que la grille ne fait plus partie de l'état, et que nous considérons l'observabilité totale sur la localisation de l'agent, nous pouvons utiliser un MDP pour résoudre ce problème. En revanche, l'inconvénient de cette méthode est que l'agent devra régulièrement calculer une nouvelle politique durant son exploration, ce qui peut être coûteux en temps et en ressources. Mais cet inconvénient est déjà présent dans l'exploration à base de frontières où il faut régulièrement calculer un nouveau chemin pour atteindre une frontière.

Formellement, le problème est formalisé comme un MDP $\langle S, A, T, R \rangle$ où :

— S est l'ensemble fini des états internes de l'agent. Par la suite nous considérerons un état $s \in S$ comme étant la localisation de l'agent et son orientation : $s = (x, y, \theta)$
— A est l'ensemble fini des actions applicables par l'agent, typiquement des actions de déplacement
— T est la fonction de transition du système telle que définie dans un MDP classique
— $R : S \to \mathbb{R}$ est la fonction de récompense

La fonction de récompense est ce qui va être déterminant dans la qualité de l'exploration de l'environnement par l'agent. Le but est de récompenser l'agent lorsqu'il atteint des endroits qui sont encore inconnus tout en évitant les obstacles. Formellement, la fonction de récompense est définie de la manière suivante :

$$\begin{aligned} R(s) &= R(x, y, \theta) = R(x, y) \\ &= \begin{cases} r > 0 & \text{si } (x, y) \text{ est libre } \textbf{et } (x, y) \text{ est proche d'une cellule inconnue} \\ 0 & \text{sinon} \end{cases} \end{aligned} \tag{3.7}$$

En principe, pour simplifier le processus de calcul d'une politique, on peut fixer une récompense positive sur les cellules inconnues, puis propager ces récompenses dans un certain rayon autour de ces cellules. De ce fait, ce qui rend cette approche intéressante, c'est que nous n'avons pas besoin de choisir de meilleur point de vue (*Best View Point*) comme cela peut être le cas dans l'exploration à base de frontières. En effet, les algorithmes de résolution de MDP considèrent l'ensemble des cellules intéressantes et choisissent la meilleure comme meilleur point de vue.

Le MDP obtenu peut être résolu avec les algorithmes classiques de résolution de MDP, avec la particularité du choix de l'horizon. En effet, généralement l'horizon de planification est spécifié à l'avance dans le modèle. Cependant, étant donné que les politiques sont calculées en ligne et qu'il est important de réduire les temps de calcul, il faut choisir un horizon le plus faible possible mais qui permette quand même d'atteindre les cellules contenant des récompenses. De ce fait, l'horizon est ajustable, c'est-à-dire qu'il peut être ajusté au moment du calcul de la politique pour, lorsqu'il n'y a aucune récompense suffisamment proche, utiliser un horizon de calcul plus grand afin d'atteindre les récompenses plus éloignées, puis à nouveau réduire l'horizon lorsque l'agent se rapproche des cellules contenant des récompenses.

Cette approche a montré de bonnes performances pour l'exploration d'environnements réels avec un robot autonome. Le problème de l'exploration de carte est un problème de planification dont la tâche est d'explorer efficacement son environnement. Nous avons présenté ici des approches mono-agent pour résoudre ce problème, mais comme tout problème de résolution de tâche, il peut souvent être intéressant d'avoir recours à plusieurs agents. Dans la suite, nous allons présenter des extensions multi-agent au problème de l'exploration de carte, que ce soit dans un environnement déterministe ou stochastique.

3.4.3 Exploration multi-agents

L'exploration multi-agent est intéressante car chaque agent peut explorer une région distincte de l'environnement, ce qui fait que l'exploration globale est plus efficace. Cependant, la coordination de plusieurs agents est plus coûteuse et entraîne de nouvelles difficultés. En effet, afin que l'exploration soit vraiment efficace, il faut mettre en place des mécanismes pour éviter les conflits entre les agents de manière à ce qu'ils explorent des régions distinctes. La communication est également un élément important car il faut régulièrement mettre en commun les cartes générées par chaque agent afin de ne pas explorer inutilement des zones déjà explorées par un autre agent. Comme c'est souvent le cas dans les systèmes multi-agents, on distingue les systèmes centralisés des systèmes décentralisés.

Les premiers travaux en exploration multi-agents continuent dans l'idée de l'exploration à base de frontières. [Yamauchi, 1998] propose une exploration multi-agents à base de frontières où le système est centralisé et chaque agent partage sa carte et ses frontières avec un planificateur central. Les agents choisissent ensuite la frontière qui leur est la plus proche pour continuer leur exploration. L'inconvénient de cette approche est qu'il y a un fort risque que plusieurs agents choisissent la même frontière. Cette approche a donc été améliorée [Bautin et al., 2011] pour tenir compte du nombre d'agents proches d'une même frontière et choisir celle où il y a le moins d'agents, ce qui diminue le risque de conflits. Cependant, il existe toujours un risque que plusieurs agents choisissent la même frontière, ce qui rend l'exploration moins efficace. Afin de pallier à ce problème, il est nécessaire de mettre en place une coordination explicite entre les agents, qu'elle soit centralisée ou non. Une coordination centralisée signifie que le planificateur central va affecter une frontière distincte à chaque agent sur la base d'un certain critère d'utilité, tandis qu'une coordination décentralisée implique une communication entre les agents pour décider de comment se partager les frontières.

Lorsque la coordination est centralisée, l'allocation des frontières aux agents est généralement faite de manière à minimiser le coût pour atteindre cette frontière et maximiser l'utilité de cette frontière, c'est-à-dire le potentiel gain en information. Dans [Simmons et al., 2000], les auteurs proposent une approche basée sur les systèmes d'enchères où les agents placent une enchère auprès du planificateur central qui est fonction du coût pour atteindre les différentes frontières et leur estimation de gain en information. Ensuite, le planificateur central alloue les frontières aux agents de manière à maximiser la récompense totale des agents, c'est-à-dire l'utilité moins le coût. D'autres méthodes pour calculer l'utilité ont été proposées. [Burgard et al., 2003] considèrent la probabilité qu'une frontière soit visible depuis le point de vue attribué à l'agent. Dans [Spaan et al., 2010a], les auteurs calculent des politiques de POMDP pour exécuter une tâche, qui peut être par exemple atteindre une frontière, puis utilisent les fonctions de valeur de ces politiques pour placer des enchères auprès du planificateur central.

D'autres approches ont été proposées pour résoudre le problème de l'exploration, en particulier [Wurm et al., 2008] proposent de partitionner l'environnement en un ensemble de segments (par exemple une salle entière à explorer), puis ces segments sont alloués aux agents sur la base d'un critère de performance.

Dans ces travaux, les agents communiquent leurs cartes individuelles au planificateur central qui est chargé de les fusionner. La fusion des cartes n'est pas toujours évidente, en particulier lorsque la position des agents est partiellement connue, car les agents construisent des cartes relatives à leur propre position. Différentes approches pour le problème de la fusion de cartes ont été proposées. [Zhou and Roumeliotis, 2006, Wang et al., 2007].

Bien que les approches centralisées soient efficaces pour coordonner les agents, elles posent également plusieurs problèmes. Tout d'abord, le coût en temps de calcul croît fortement avec le nombre d'agents, ce qui devient un problème lorsqu'il y a beaucoup d'agents. Le problème de la communication avec le planificateur central se pose également, car les agents doivent être à portée de ce dernier pour pouvoir communiquer avec lui.

De ce fait, des approches décentralisées ont également été développées. [Zlot et al., 2002] proposent une approche basée sur les enchères et les négociations, où les agents négocient directement avec les autres agents qui sont à portée de communication pour choisir quelle frontière ils doivent explorer. [Burgard et al., 2005] ont également proposé une approche, basée sur leur précédente approche centralisée [Burgard et al., 2003], en appliquant leur algorithme d'allocation de frontières uniquement dans des sous-ensembles d'agents qui peuvent communiquer entre eux. [Capitan et al., 2013] ont également proposé une approche basée sur les enchères et généralisant les travaux de [Spaan et al., 2010a] avec une faible nécessité de communication.

[Matignon et al., 2012b] ont proposé une approche basée sur les DEC-MDP et utilisant les fonctions de valeur distribuées [Schneider et al., 1999], où chaque agent calcule sa propre politique d'exploration qui évite les interactions avec les autres agents tout en maximisant l'exploration de la carte. Ces travaux ont ensuite été étendus au cas où la communication est difficile par moments voire impossible [Matignon et al., 2012a]. Par la suite, ce travail a été utilisé pour faire de l'exploration et de la récolte d'information (reconnaissance d'objets) dans un environnement inconnu [Matignon et al., 2015].

3.5 Conclusion

Dans ce chapitre, nous avons abordé les problématiques de la perception active pour la récolte d'information dans un environnement inconnu et partiellement observable. Nous avons d'abord introduit les notions de perception active qui consistent à prendre des décisions dans le but de percevoir son environnement, puis nous avons défini l'incertitude et d'où elle peut provenir. Nous avons ensuite présenté une manière communément employée de formaliser l'incertitude afin de pouvoir l'utiliser dans le processus de décision de nos agents. Nous avons également introduit la problématique de la récolte d'information, en particulier dans le cadre de l'amélioration de l'état de croyance d'un agent, dans le cadre des processus décisionnels de Markov. Enfin, nous avons présenté le problème de l'exploration de carte par des agents autonomes.

Bilan de la partie I

Dans cette partie, nous avons introduit les notions importantes à la compréhension de ce manuscrit et à la mise en contexte de nos travaux. Nous avons commencé par présenter les notions génériques relatives aux problèmes d'intelligence artificielle et des agents autonomes, pour ensuite introduire le problème de la planification sous incertitude et les processus de décisions permettant de formaliser et résoudre de tels problèmes. Enfin, nous avons présenté les problématiques qui nous intéressent particulièrement dans le cadre de nos travaux, qui sont la perception active pour la récolte d'information et l'exploration de cartes.

Les limites des solutions existantes se trouvent sous différentes formes. Tout d'abord, cette thèse porte sur la prise de décision multi-agents sous incertitude, en particulier dans un but d'améliorer l'état de croyance des agents sur leur environnement. Nous avons présenté la planification sous incertitude et particulièrement le formalisme des DEC-POMDP qui permet de résoudre de tels problèmes. Malheureusement, la complexité des DEC-POMDP est telle qu'il est difficile de les utiliser dans des applications concrètes, en particulier lorsqu'il y a un grand nombre d'agents. Par ailleurs, nous nous intéressons à des problématiques de récolte d'information. Les outils de récolte d'information pour l'amélioration de l'état de croyance que nous avons présentés ciblent principalement les applications mono-agent ou souffrent également d'une forte complexité. Enfin, nous nous intéressons à des systèmes multi-agents hétérogènes, c'est-à-dire des systèmes où évoluent plusieurs agents qui peuvent avoir des caractéristiques différentes. Bien que les DEC-POMDP permettent naturellement de traiter de tels problèmes car chaque agent possède son propre modèle d'action et d'observation, de tels problèmes sont encore plus compliqués à résoudre efficacement, car les techniques basées sur l'empathie pour estimer les prochaines actions d'un agent ne peuvent pas être utilisées. Dans cette thèse, nous proposons un modèle hiérarchique basé sur les POMDP afin de décomposer le problème de récolte d'information en deux couches dans le but de réduire la complexité du problème. La partie suivante sera donc consacrée à la description formelle de ce modèle ainsi que l'application dans un problème concret de récolte d'information dans un environnement topologiquement inconnu.

Deuxième partie

Contributions

Chapitre 4

Un modèle hiérarchique pour la récolte d'information multi-agents hétérogènes

Sommaire

4.1 Introduction

Nous avons évoqué dans les chapitres précédents les difficultés des approches classiques pour la résolution de problèmes de récolte d'information multi-agents. Nous proposons dans ce chapitre une approche hiérarchique centralisée basée sur les processus décisionnels de Markov. Nous considérons un problème d'évaluation de situation avec des robots hétérogènes (robots ayant différentes capacités d'observation, de navigation, d'embarquement) dans un environnement stochastique et partiellement observable (par exemple, l'évaluation d'une situation après une catastrophe naturelle ou industrielle). Les robots doivent coopérer afin d'évaluer efficacement la situation sans intervention humaine. L'objectif est de recueillir des informations sur un ensemble de cibles dans l'environnement dans un temps le plus court possible. La stochasticité de l'environnement se manifeste de plusieurs manières : (1) les actions des robots peuvent être non déterministes, par exemple un robot peut glisser et légèrement dévier de sa position prévue ; (2) le plan de l'environnement peut différer du plan attendu, par exemple lors de l'évaluation d'une situation après une catastrophe, certains chemins peuvent ne plus être accessibles ou être trop dangereux ; (3) la précision des capteurs des robots, qui peut ne pas être parfaite. Chaque cible dans l'environnement est un point d'intérêt pour lequel nous souhaitons récolter de l'information. Nous divisons le problème de récolte d'information sur les points d'intérêts en deux couches : pour chaque agent et chaque point d'intérêt, nous proposons un modèle de récolte d'information

pour ce point et cet agent. Ensuite, nous proposons un modèle d'allocation de points d'intérêts aux agents de manière optimale sur le long terme en utilisant les politiques de récolte d'information ainsi obtenues. Nous avons nommé cette approche un Meta-MDP et nous allons la décrire formellement dans les sections suivantes.

Les travaux présentés dans ce chapitre ont fait l'objet d'une publication à la *4th International Conference on Robotics, Computer Vision and Intelligent Systems* (ROBOVIS 2024), ont été présentés publiquement à Rome, Italie [Gandois et al., 2024], et ont été récompensés du *Best Paper Award* (prix du meilleur article).

4.2 Idée générale

L'idée derrière notre modèle est de réduire la complexité des DEC-POMDP au niveau de l'espace de recherche. En effet, calculer une politique pour un DEC-POMDP requiert de parcourir l'espace de recherche qui croît de manière exponentielle avec l'horizon, comme illustré par la figure 4.1.

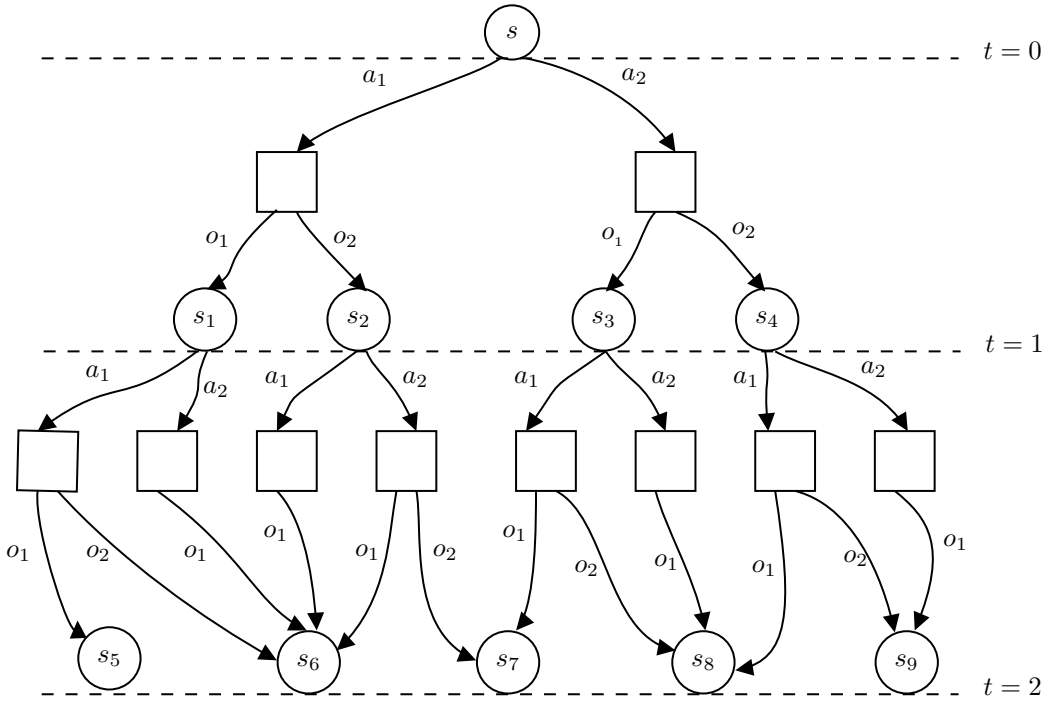


FIGURE 4.1 – Exemple de graphe d'un POMDP à horizon 2

Avec le Meta-MDP, nous calculons des politiques individuelles, que nous nommerons politiques d'exploration, qui seront ensuite utilisées comme actions. Lorsque l'agent applique une politique d'exploration, nous considérons l'exécution de cette politique comme un seul pas de temps au niveau méta, mais d pas de temps au niveau global (l'horizon de la politique), ce qui permet de réduire l'espace de recherche de la méta-politique. En effet, les politiques utilisées comme actions étant déjà optimales, certaines branches de l'arbre de recherche qui sont inintéressantes ne seront pas explorées au niveau méta. Dans le cas où nous avons k agents et n points d'intérêt, le nombre d'actions $|A|$ du Méta-MDP correspond au nombre d'arrangements

de k parmi n possibles :

$$|A| = A_n^k = \frac{n!}{(n-k)!} \tag{4.1}$$

Par exemple, pour un Meta-MDP à deux agents et deux points d'intérêts, nous avons les méta-actions suivantes : $A = \langle(\pi_{1,1}, \pi_{2,2}), (\pi_{1,2}, \pi_{2,1})\rangle$, où $\pi_{i,j}$ est la politique de l'agent i pour le point d'intérêt j. La figure 4.2 illustre un exemple de graphe de Meta-MDP.

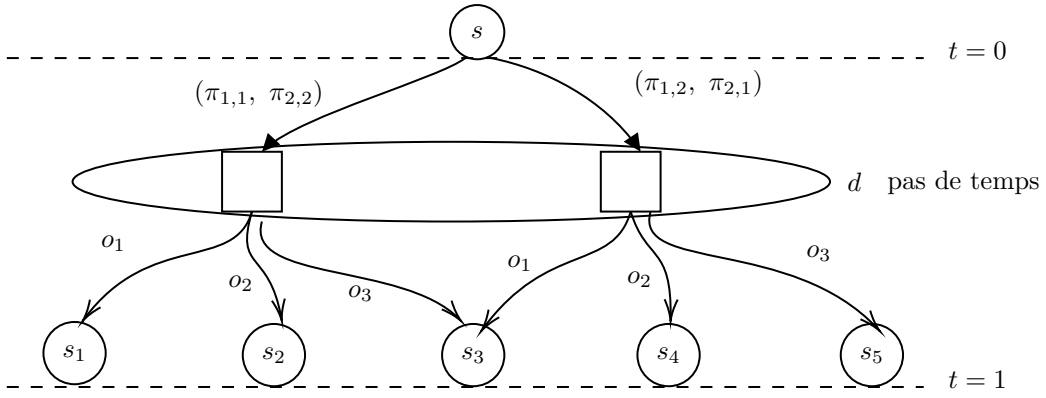


FIGURE 4.2 – Exemple de graphe d'un Meta-MDP à horizon 1

De ce fait, nous espérons gagner en temps de calcul au niveau de la politique du Meta-MDP, malgré un coût supplémentaire pour calculer les politiques individuelles. Une autre difficulté est la fonction de transition du Méta-MDP faisant intervenir des politiques à suivre au lieu de simples actions, ce qui fait qu'il est plus coûteux de la calculer. Par ailleurs, étant donné que chaque agent calcule sa propre politique individuelle de récolte d'information sur un point d'intérêt, le Méta-MDP permettra d'allouer les points d'intérêt de manière optimale en considérant les capacités de chacun selon leurs politiques, ce qui permet de tenir compte sans problème de l'hétérogénéité des agents.

Nous allons désormais, dans les sections suivantes, définir formellement le modèle et les algorithmes qui l'accompagnent.

4.3 Définition du modèle

Nous considérons un ensemble de points d'intérêt, noté X, où chaque point d'intérêt $x \in X$ est représenté par une variable aléatoire et prend une valeur dans son domaine $DOM(x)$.

Nous distinguons l'état joint des agents de l'état de l'environnement. L'état de l'environnement, noté ϵ, est décrit comme une affectation d'une valeur à chaque point d'intérêt. Nous notons $\mathcal{E}$ l'ensemble de tous les états possibles de l'environnement, c'est-à-dire toutes les affectations de valeurs possibles aux points d'intérêt. Formellement,

$$\mathcal{E} = DOM(x_1) \times \cdots \times DOM(x_n) \tag{4.2}$$

Notre objectif est de récolter de l'information à propos de l'état réel de l'environnement, c'est-à-dire déterminer la valeur de chaque point d'intérêt, avec des agents qui ont différentes capacités pour les observer. Par exemple, considérons deux types de robots autonomes : des robots qui peuvent faire des observations locales précises, et des robots qui peuvent faire des observations globales mais imprécises, qui évoluent dans un environnement représenté par une grille, comme

illustré dans la figure 4.3. Chaque type de robot peut potentiellement être complètement hétérogène, c'est-à-dire ne pas avoir les mêmes actionneurs ni les mêmes capacités d'observation. En revanche, ils évoluent tous dans le même environnement, par conséquent ils partagent tous le même ensemble d'états joints et le même ensemble d'états de l'environnement.

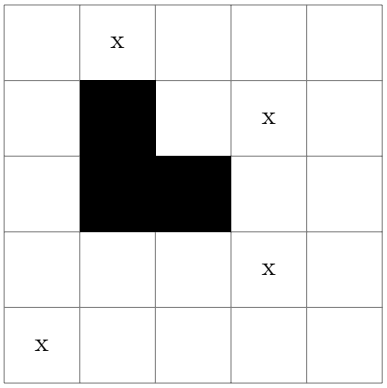


FIGURE 4.3 – Un environnement de taille 5x5 avec 4 points d'intérêt, marqués par une croix (x)

Nous considérons les hypothèses suivantes :

1. Chaque robot connaît le modèle des autres robots (ensemble d'actions, ensemble d'états, ensemble d'observations, fonction d'observation)
2. Chaque robot connaît l'état interne courant (par exemple la position, l'orientation, etc.) des autres robots
3. L'information récoltée est instantanément partagée à l'ensemble des robots sans aucun coût (c'est-à-dire qu'ils partagent le même état de croyance)
4. Les positions et les domaines des points d'intérêt sont connus à l'avance, ainsi que la disposition de la carte (obstacles, etc.)
5. Les points d'intérêt sont indépendants entre eux (l'état d'un point d'intérêt n'a aucune influence sur les états des autres points)

Nous sommes donc dans un problème à observabilité mixte (observabilité partielle sur l'environnement et observabilité totale sur l'état des agents) et centralisé (un planificateur central récolte les informations reçues de chaque agent dans le but de prendre une décision).

Nous allons désormais décrire le modèle de récolte d'information pour un agent et un point d'intérêt.

4.3.1 Les POMDP d'exploration

Les macro-actions de notre Meta-MDP sont des politiques de POMDP pour récolter de l'information par un agent sur un point d'intérêt. De ce fait, pour chaque agent i et chaque point d'intérêt j, nous définissons le modèle de POMDP augmenté suivant :

$$POMDP_{i,j}\langle S_i, \mathcal{E}, A_i \cup \{OBSERVE, STAY\}, T_{i,j}, R_{i,j}, \Omega_i, O_i\rangle \tag{4.3}$$

Les états

L'ensemble S_i d'états est l'ensemble des états internes pour l'agent i (par exemple sa position, son orientation, etc.). L'état d'un agent est complètement observable.

Les états de l'environnement

L'ensemble $\mathcal{E}$ est l'ensemble de tous les états possibles de l'environnement, tel que :

$$\mathcal{E} = DOM(x_1) \times \cdots \times DOM(x_n) \tag{4.4}$$

L'état de l'environnement est partiellement observable.

Les actions

L'ensemble A_i représente l'ensemble des actions que l'agent i peut effectuer, auxquelles nous ajoutons une action "STAY" (ne rien faire), et une action "OBSERVE", qui est une action pour observer le point d'intérêt proche de l'agent.

La fonction de transition

La fonction de transition $T_{i,j}$ est une fonction de transition classique de POMDP, où $T_i(s, a)$ est la distribution de probabilité sur l'espace d'états lorsque l'agent i exécute l'action a dans l'état s.

L'état de croyance

Pour représenter les connaissances d'un agent sur l'environnement, nous définissons également un état de croyance b comme dans les POMDP classiques. Cependant, dans notre cas, nous souhaitons mesurer les connaissances sur chaque point d'intérêt individuellement, donc, au lieu de définir un état de croyance sur l'état complet de l'environnement, nous définissons un ensemble d'états de croyance :

$$b = \{b(x_1), \ldots, b(x_n)\} \tag{4.5}$$

où $b(x_i)$ est l'état de croyance pour la variable d'environnement i, qui représente les connaissances actuelles sur le point d'intérêt i (une distribution de probabilité sur $DOM(x_i)$). De ce fait, lorsque l'agent évolue dans l'environnement et reçoit une observation, seul l'état de croyance du point d'intérêt correspondant est mis à jour.

La fonction de récompense

Notre objectif est de maximiser la quantité d'information que l'on possède sur l'environnement. Pour ce faire, nous allons utiliser l'entropie de Shannon, telle que nous l'avons présentée en section 3.2.2. L'entropie de Shannon pour une variable aléatoire, notée $H(x)$, est définie de la manière suivante :

$$H(x) = -\sum_{v \in DOM(x)} P(x = v) \log_2 P(x = v) \tag{4.6}$$

Cependant, pour les problèmes de décision orientés information, il est généralement plus approprié d'utiliser l'entropie de Shannon négative, qui mesure la quantité d'information plutôt que la quantité d'incertitude, et qui peut donc être utilisée directement comme fonction de récompense à maximiser :

$$H^-(x) = \log_2(|DOM(x)|) + \sum_{v \in DOM(x)} b(x = v) \log_2 b(x = v) \tag{4.7}$$

L'entropie de Shannon négative est maximale dans les coins du simplexe et minimale au centre [Araya et al., 2010], ce qui signifie que la récompense est maximale lorsqu'il n'y a aucune ambiguïté sur la variable.

Enfin, nous définissons la quantité d'information $I(b)$ contenue dans un état de croyance b comme la somme des entropies négatives de chaque point d'intérêt :

$$I(b) = \sum_{x \in X} H^-(x) \tag{4.8}$$

En introduisant la quantité d'information dans la fonction de récompense de l'agent, notre objectif sera d'optimiser dans le but de récolter le plus d'information possible. La fonction de récompense $R_{i,j}((s,b),a,(s,b)')$ définit la récompense que l'agent i recevra lorsqu'il appliquera l'action a dans l'état (s,b) (où s est l'état interne de l'agent, et b l'état de croyance) et transite dans l'état $(s,b)'$. Nous définissons $R_{i,j}((s,b),a,(s,b)')$ de la manière suivante :

$$R_{i,j}((s,b),a,(s,b)') = I(b') - H^-(x_j) + \begin{cases} H_{max}(x_j) & \text{si } s \models j \wedge a = OBSERVE \\ 0 & \text{sinon} \end{cases} \tag{4.9}$$

Ici, $H^-(x_j)$ est l'entropie négative du point d'intérêt assigné à l'agent, et $I(b')$ est la quantité d'information actuelle sur l'environnement dans l'état de croyance b. $H_{max}(x_j) = \log_2 |DOM(x_j)|$ est l'entropie maximale pour le point d'intérêt j.

Lorsque l'agent explore son environnement, il reçoit une récompense correspondant à la quantité d'information de son nouvel état de croyance, moins l'information du point d'intérêt qui lui a été alloué (comme s'il n'en tenait pas compte). Cependant, lorsque l'agent observe son point d'intérêt alloué (en exécutant l'action OBSERVE), il reçoit de manière optimiste la récompense maximale possible pour ce point d'intérêt, comme si le fait d'observer ce point allait complètement désambiguïser la situation sur ce point en particulier. Cette fonction de récompense permet donc à la fois de récolter de l'information sur l'ensemble des points d'intérêt de l'environnement, tout en accordant une attention plus particulière au point d'intérêt alloué à l'agent.

L'ensemble d'observations

L'ensemble Ω_i est l'ensemble des observations que l'agent i peut recevoir. Un agent peut recevoir une observation sur la valeur d'un point d'intérêt, l'ensemble d'observation représente donc l'ensemble des domaines de chaque point d'intérêt.

La fonction d'observation

La fonction d'observation $O_i(s',a)$ est une fonction d'observation classique pour un POMDP, qui donne une distribution de probabilité sur l'ensemble d'observations de l'agent, lorsque l'agent exécute une action a et transite dans l'état s'.

Ces modèles nous permettent de calculer des politiques d'exploration pour chaque agent et chaque point d'intérêt, de manière à ce que chaque agent possède une politique optimale pour aller explorer chaque point d'intérêt. Nous notons $\pi^*_{i,j}$ la politique optimale de l'agent i pour le point d'intérêt j, et $V^*_{i,j}$ la fonction de valeur de cette politique, telle que définie par l'équation de Bellman (voir l'équation 2.6). Malheureusement, une telle politique ne peut être calculée avec les algorithmes de résolution classiques de POMDP, dû à l'utilisation de l'entropie dans la fonction de récompense qui n'a pas la propriété d'être PWLC (*piecewise linear and convex* - linéaire par partie et convexe). Cependant, nous avons présenté dans la section 3.3.1 des techniques existantes pour résoudre de tels modèles, notamment les algorithmes de résolution de ρPOMDP [Araya et al., 2010].

4.3.2 Le Meta-MDP

Dans cette section, nous allons décrire notre modèle principal pour calculer une allocation optimale de points d'intérêt aux agents de manière à maximiser la quantité d'information récoltée sur le long terme en tenant compte des récompenses futures. Nous définissons un MDP $\langle S, \mathcal{E}, A, T, R\rangle$ où $S = \{S_1 \times \cdots \times S_k\}$ est l'ensemble des états joints des agents et $\mathcal{E}$ est l'ensemble des états possibles de l'environnement, comme défini en section 4.3.1. L'originalité de notre approche est que l'ensemble d'actions jointes est l'ensemble des allocations possibles de points d'intérêt aux agents, qui sont représentés par des politiques d'exploration que nous avons introduites dans la section précédente. Par exemple, si nous avons deux agents et trois points d'intérêts, alors l'ensemble d'actions est $A = \{(\pi^*_{1,1}, \pi^*_{2,2}), (\pi^*_{1,1}, \pi^*_{2,3}), (\pi^*_{1,2}, \pi^*_{2,1}), (\pi^*_{1,2}, \pi^*_{2,3}), (\pi^*_{1,3}, \pi^*_{2,1}),$ $(\pi^*_{1,3}, \pi^*_{2,2})\}$. L'objectif de ce modèle est de calculer une politique jointe optimale pour allouer des politiques d'exploration individuelles aux agents.

Nous définissons la fonction de récompense de la manière suivante :

$$R(\epsilon, a, \epsilon') = I(\epsilon') \tag{4.10}$$

Encore une fois, nous utilisons l'entropie de Shannon négative, telle que définie dans l'équation 4.7 pour mesurer la quantité d'information présente dans un état de croyance. L'objectif de ce modèle étant uniquement de récolter le plus d'information possible sur l'environnement, nous définissons la fonction de récompense comme étant uniquement l'entropie négative de l'état suivant.

La difficulté de ce modèle réside dans la fonction de transition. En effet, étant donné que les actions sont des politiques complètes de MDP à exécuter, nous devons calculer la distribution d'états possibles après avoir exécuté la politique de chaque agent. De plus, étant donné qu'il s'agit de politiques et non de plans, il est possible d'avoir des trajectoires infinies (par exemple un agent qui n'arrive jamais à atteindre son point d'intérêt). De ce fait, nous limitons la longueur des trajectoires à une certaine profondeur d donnée. Par exemple, pour deux agents 1 et 2 qui appliquent l'action a_1 et a_2 dans un état s sur d pas de temps, la fonction de transition est définie comme :

$$T^d(s, (a_1, a_2)) = \underbrace{(T_{1,a_1} \circ T_{2,a_2}) \circ \cdots \circ (T_{1,a_1} \circ T_{2,a_2})}_{d \text{ fois}} \tag{4.11}$$

où $\circ$ est l'opérateur de composition des fonctions, c'est-à-dire l'utilisation du résultat d'une fonction comme argument de la seconde, et T_{1,a_1} et T_{2,a_2} sont les fonctions de transition des MDP d'exploration des agents 1 et 2 appliquées aux actions d'exploration données par les politiques d'exploration individuelles π_{1,a_1} et π_{2,a_2}, avec a_1 et a_2 les méta-actions qui représentent les points d'intérêt assignés aux agents.

Étant donné que nous n'avons pas besoin des trajectoires complètes mais seulement des états finaux de ces trajectoires, et que nous connaissons l'état initial (car c'est l'état courant), nous pouvons calculer la distribution au temps d par **analyse du premier pas** (FSA [5]). En effet, le FSA est une technique couramment employée pour résoudre divers problèmes que l'on retrouve dans les chaînes de Markov, notamment le problème qui consiste à calculer la distribution d'états à un temps donné. Cette technique consiste à partir du premier pas (l'état initial), puis calculer successivement la distribution au pas de temps suivant en partant de la distribution au pas de temps actuel. Étant donné qu'il est possible de construire une chaîne de Markov à partir d'une politique de MDP, nous pouvons appliquer cette technique pour calculer la distribution d'états

5. first step analysis

après l'exécution des politiques des agents de la manière suivante :

$$T^0 = \{s\} \tag{4.12}$$

$$T^t = \{(T_{1,a_1} \circ T_{2,a_2}) \forall s \in T^{t-1}\} \tag{4.13}$$

où s est l'état initial, $t \leq d$ le pas de temps considéré, ainsi que T_{1,a_1} et T_{2,a_2} les fonctions de transition individuelles définies précédemment.

Étant donné que nous appliquons les fonctions de transition individuelles des agents successivement, l'ordre d'application est important. En effet, l'état de croyance des agents sur l'environnement ne sera pas forcément le même selon l'ordre dans lequel les agents l'observent. De ce fait, il peut être important de considérer l'ordre d'application des actions pour le calcul de la fonction de transition. Nous proposons en algorithme 3 un algorithme de programmation dynamique pour calculer cette fonction. Par souci de simplicité, l'algorithme est décrit pour le cas où il n'y a que deux agents. Nous considérons que, à un pas de temps donné, chaque agent a une probabilité équiprobable d'appliquer son action en premier, et chaque ordre est considéré.

Algorithme 3 : Meta-Fonction de Transition

Entrées : un état s, une action jointe a, une profondeur d
Output : T^d la distribution d'états possibles au temps d

```
1  T^0 : S → ℝ la distribution initiale
2  T^0(s) = 1
3  pour chaque t ∈ {1, ..., d} faire
4  |  T^t : S → ℝ
5  |  pour chaque s^{t-1} ∈ T^{t-1} faire
6  |  |  tr_1 = T_{2,a_2}(T_{1,a_1}(s^{t-1}, π_{1,a_1}), π_{2,a_2})
7  |  |  tr_2 = T_{1,a_1}(T_{2,a_2}(s^{t-1}, π_{2,a_2}), π_{1,a_1})
8  |  |  pour chaque s^t ∈ tr_1 faire
9  |  |  |  T^t(s^t) ± T^{t-1}(s^{t-1}) * 1/2 (tr_1(s^t) + tr_2(s^t))
10 |  |  fin
11 |  |  pour chaque s^t ∈ tr_2 faire
12 |  |  |  T^t(s^t) ± T^{t-1}(s^{t-1}) * 1/2 (tr_1(s^t) + tr_2(s^t))
13 |  |  fin
14 |  fin
15 fin
16 retourner T^d
```

Cet algorithme commence par calculer la distribution d'états au temps t puis utilise la distribution au temps t pour calculer la distribution au temps $t+1$. Comme nous l'avons évoqué, nous considérons l'ordre dans lequel les agents exécutent leur action, ce qui explique que les lignes 9 et 11-13 sont dupliquées. Cet algorithme peut facilement être étendu pour le cas où il y a n agents, en tenant compte du fait que pour n agents il y a $n!$ ordres d'exécution des actions possibles. Il est également possible de simplifier le problème en considérant que les agents effectueront toujours leurs actions dans le même ordre (par exemple toujours l'agent 1 en premier, puis l'agent 2), et dans ce cas il n'y a qu'un seul ordre d'exécution à calculer.

Cet algorithme permet de calculer une solution de meilleure qualité en fonction de la profondeur de calcul (plus la profondeur est grande, plus la solution sera précise), mais au coût du temps de calcul. Cependant, cet algorithme a un comportement **anytime**, c'est-à-dire qu'il est

capable de fournir une solution même s'il est arrêté avant la fin de son exécution, et la qualité de la solution augmentera alors en fonction du temps de calcul alloué.

La complexité de cet algorithme, pour calculer la fonction de transition d'un seul état, est de $O(d*|S|)$ (on itère d fois sur un sous-ensemble de l'espace d'états). Soit, pour calculer la fonction de transition du Meta-MDP, une complexité dans le pire des cas de $O(|S|*d*|S|)$, ce qui fait que calculer cette fonction de transition est très coûteux en temps de calcul, en particulier lorsque le nombre d'états est grand. De ce fait, nous allons présenter dans la section suivante des techniques permettant d'approximer une solution optimale dans le but de réduire le temps de calcul.

4.3.3 Approximation de la Meta-Fonction de transition

L'algorithme pour calculer la fonction de transition du Meta-MDP est très coûteux, comme nous l'avons montré dans la section précédente. Dans cette section, nous allons présenter deux techniques d'approximation probabilistes couramment utilisées dans le but de réduire les temps de calcul.

Échantillonnage Monte-Carlo

L'échantillonnage par méthode de Monte-Carlo [Metropolis and Ulam, 1949] est une méthode d'approximation qui utilise la génération aléatoire pour estimer une distribution de probabilité. Par exemple, pour un échantillon de taille $k = 10$, nous tirons aléatoirement 10 résultats possibles de la fonction de transition pour l'état courant et l'action choisie. Ensuite, l'échantillon résultant est considéré comme étant la distribution de probabilité de la fonction de transition. Admettons que après 10 tirages aléatoires nous ayons l'échantillon suivant : 3x s_1, 5x s_2, 2x s_3, nous allons considérer que la distribution de probabilité des états est $\{s_1 : 0.3, s_2 : 0.5, s_3 : 0.2\}$. Cette méthode permet généralement de fournir des solutions satisfaisantes à un certain nombre de problèmes. Malheureusement, pour notre problème en particulier, cette méthode n'est pas très efficace, car plus la profondeur augmente, et plus le nombre d'états résultants possibles augmente, avec des probabilités pouvant être relativement faibles. De ce fait, nous devrions choisir une taille d'échantillon suffisamment grande pour être représentatif, ce qui serait plus coûteux que notre algorithme original en temps de calcul. Nous avons donc choisi de ne pas retenir cette solution.

Élagage des états les moins probables

Cette méthode consiste à ne garder, à chaque pas de temps, que les états les plus probables, et élaguer les autres. De ce fait, nous réduisons le facteur de branchement de notre arbre de recherche à chaque niveau. Pour ce faire, nous définissons un paramètre γ qui représente le seuil à partir duquel les états sont élagués, et qui influera sur la précision de l'approximation. Par exemple, pour $\gamma = 0.75$, nous allons conserver les états jusqu'à une probabilité de 75%, et donc les 25% restants seront élagués. Ce processus est appliqué à chaque pas de temps durant le calcul de la fonction de transition (entre les lignes 17 et 18 de l'algorithme 3). Cette méthode d'approximation est décrite dans l'algorithme 4.

Pour illustrer notre propos, considérons un exemple où l'on souhaite calculer la fonction de transition du Meta-MDP à profondeur $d = 2$. Dans la figure 4.4, nous commençons par l'état initial à $d = 0$. Ensuite, lorsque l'agent se trouve dans l'état s^0, il peut exécuter trois actions qui mènent à la profondeur 1, comme illustré par la figure 4.5. Finalement, pour chaque état accessible depuis la fonction de transition à profondeur 1, nous calculons la distribution à profondeur 2, comme illustré par la figure 4.6. Pour cet exemple, à partir d'un état s et à profondeur $d = 2$, nous avons obtenu 8 états résultants.

Algorithme 4 : Approximation par élagage des états les moins probables

Entrées : La Meta-Fonction de transition au temps t T^t, le facteur d'élagage γ
Output : L'approximation de la fonction de transition au temps t

$S : \{\}$
$sum = 0$
Trier T^t par ordre décroissant de probabilité
pour chaque $(s, p) \in T^t$ **faire**
  $S = S \cup \{s\}$
  $sum = sum + p$
  si $sum > \gamma$ **alors**
    break
  fin
fin
T'^t :la nouvelle distribution de probabilité
`// Normalisation de la nouvelle distribution`
pour chaque $s \in S$ **faire**
  $T'^t[s] = T^t[s]/sum$
fin
retourner T'^t

FIGURE 4.4 – Exemple de fonction de transition à d=0 sans approximation

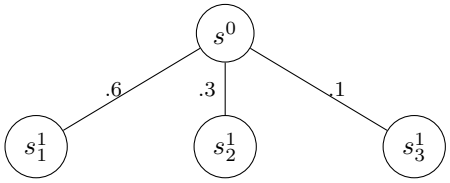


FIGURE 4.5 – Exemple de fonction de transition à d=1 sans approximation

Il est important de noter que la fonction de transition n'a pas toujours une structure d'arbre comme on peut le voir sur cet exemple, mais que plusieurs actions peuvent parfois mener au même état. Cela est pris en compte dans les algorithmes que nous avons présentés. Nous allons désormais reprendre le même exemple mais en appliquant un élagage des états les moins probables, comme décrit précédemment, avec un seuil d'élagage $\gamma = 0.75$. Nous commençons de la même manière avec l'état initial, illustré en figure 4.7. Ensuite, nous calculons la distribution des états accessibles à partir de l'état s^0. Cette fois-ci, nous n'allons garder que les états les plus probables. Concrètement, nous trions les états du plus au moins probable, puis nous conservons les états jusqu'à une probabilité de 0.75. L'état s_3^1 est donc élagué, comme illustré par la figure 4.8. Nous réitérons ce processus pour calculer la distribution d'états à profondeur $d = 2$. Cette fois-ci, l'élagage élimine les états s_2^2 et s_5^2 qui ont une probabilité totale de 0.23, comme illustré par la figure 4.9. Enfin, la figure 4.10 illustre la distribution finale obtenue par approximation.

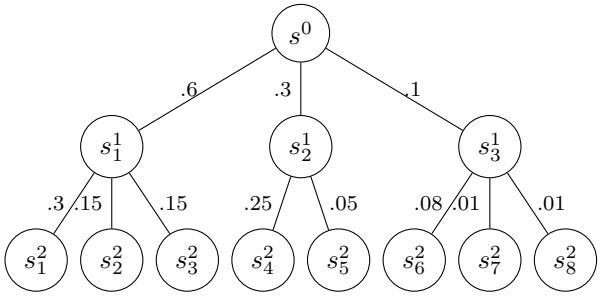


FIGURE 4.6 – Exemple de fonction de transition à d=2 sans approximation

FIGURE 4.7 – Exemple de fonction de transition à d=0 avec approximation

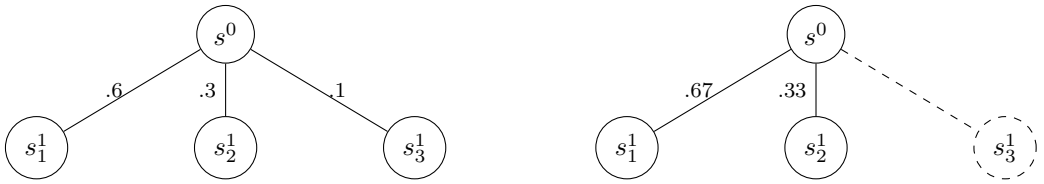


FIGURE 4.8 – Exemple de fonction de transition à d=1 avec approximation

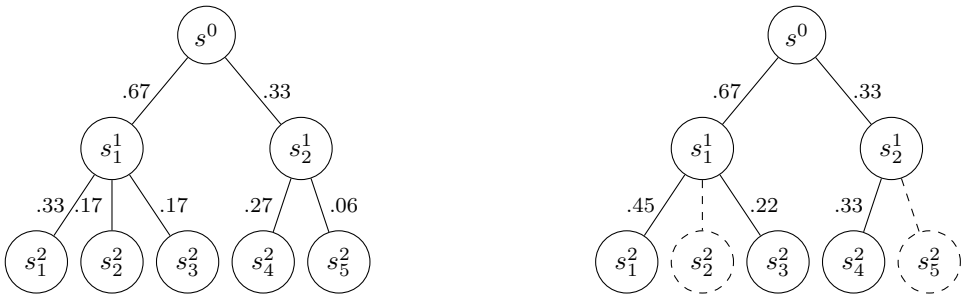


FIGURE 4.9 – Exemple de fonction de transition à d=2 avec approximation

Nous constatons que, pour cet exemple, avec $\gamma = 0.75$, le nombre d'états résultants est passé de 8 à 3, pour une profondeur $d = 2$. De ce fait, l'approximation permet de grandement réduire la taille de l'arbre de recherche. Cependant, cette approximation a un coût, car la précision de la solution est moindre, par exemple pour l'état s_1^2, la probabilité est de 0.3 pour la fonction de transition exacte, mais l'approximation donne une probabilité de 0.45.

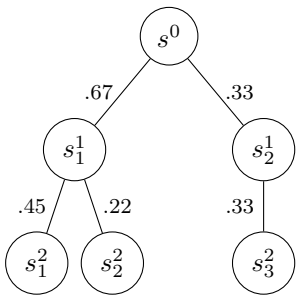


FIGURE 4.10 – Exemple de fonction de transition à d=2 avec approximation après élagage

4.4 Résolution

La résolution du Meta-MDP se déroule en deux parties : tout d'abord, pour chaque POMDP d'exploration de chaque point d'intérêt pour chaque agent, nous calculons une politique optimale, comme décrit en section 4.3.1. Ensuite, nous pouvons résoudre le Meta-MDP à proprement parler. Résoudre le Meta-MDP consiste à calculer une politique jointe $\pi(\epsilon) = (\pi_1, \ldots, \pi_k)$ où, pour chaque état ϵ, nous avons l'action jointe optimale dictée par la politique optimale, tel que décrit par la reformulation suivante de l'équation de Bellman :

$$V(\epsilon) = \max_{(\pi_1,\ldots,\pi_k)\in A} [R(\epsilon, (\pi_1, \ldots, \pi_k), \epsilon') + \gamma \sum_{\epsilon' \in \mathcal{E}} T^d(\epsilon, (\pi_1, \ldots, \pi_k), \epsilon') V(\epsilon')] \tag{4.14}$$

Comme pour les POMDP d'exploration, les techniques classiques de résolution de POMDP ne peuvent pas être utilisées pour résoudre le Meta-MDP, car la fonction de récompense utilise une nouvelle fois l'entropie de Shannon, qui n'est pas PWLC. Il est cependant possible d'utiliser d'autres techniques de résolution telle que la discrétisation de l'espace d'état de croyance afin de résoudre le POMDP comme un MDP classique. Par ailleurs, une des difficultés de notre modèle étant la fonction de transition, il pourrait être intéressant d'utiliser des méthodes à base d'apprentissage par renforcement pour estimer une politique optimale sans avoir à connaître le modèle de transition.

4.5 Complexité

Les modèles d'exploration ainsi que le Meta-MDP étant basés sur des POMDP, la complexité pour résoudre ce modèle est identique à celle pour résoudre un POMDP, c'est-à-dire PSPACE-complet à horizon fini [Papadimitriou and Tsitsiklis, 1987], et indécidable à horizon infini [Madani et al., 1999]. Cependant, le fait de hiérarchiser le problème permet de réduire la taille de l'espace de recherche au niveau du Meta-MDP, ce qui peut potentiellement être avantageux en pratique. De plus, comme dans le cadre des MDP hiérarchiques, le fait de calculer plusieurs politiques au lieu d'une seule engendre un coût supplémentaire, mais ce coût peut être compensé par la réutilisation des politiques dans d'autres problèmes similaires. Enfin, la complexité du calcul de la fonction de transition du Meta-MDP est un point négatif de notre modèle, bien que nous avons proposé une méthode d'approximation qui permet de réduire cela. Il est donc assez difficile d'évaluer concrètement la complexité théorique de notre modèle par rapport aux autres modèles de la littérature, et c'est pour cela que nous évaluerons dans la partie expérimentations les capacités de notre modèle en pratique.

4.6 Conclusion

Dans ce chapitre, nous avons présenté un modèle théorique basé sur les POMDP pour faire de la récolte d'information multi-agents hétérogènes dans un environnement partiellement connu et partiellement observable. Ce modèle, que nous avons nommé Meta-MDP, est un modèle hiérarchique qui consiste à calculer des politiques d'exploration individuelles pour chaque agent et chaque point d'intérêt afin d'allouer les points d'intérêt aux agents de manière optimale sur le long terme en tenant compte des capacités de chacun, qui peuvent être différentes. Nous avons formalisé ce modèle et proposé un algorithme pour calculer la Meta-Fonction de transition, qui est un des points clés de ce modèle. Le coût de calcul de la Meta-Fonction de transition étant important, nous avons également proposé un algorithme d'approximation basé sur un élagage statistique des états les moins probables, à chaque niveau de l'arbre de recherche, ce qui permet de réduire le facteur de branchement de manière conséquente. Enfin, nous avons proposé une analyse de la complexité théorique de notre modèle, bien que des travaux plus approfondis soient nécessaires. Nous proposerons dans la partie III une évaluation expérimentale de ce modèle. Dans le chapitre suivant, nous allons proposer un cas d'application concret du Meta-MDP à l'exploration et la reconnaissance active dans un environnement qui sera cette fois totalement inconnu (hormis les dimensions). En effet, dans ce chapitre nous avons émis plusieurs hypothèses qui ne sont pas toujours réalistes, comme la connaissance a priori de la disposition de l'environnement, ainsi que le nombre et la position des points d'intérêt.

Chapitre 5

Meta-MDP pour la récolte d'information mono-agent avec des capteurs hétérogènes

Sommaire

5.1 Introduction

Nous avons proposé dans le chapitre précédent un modèle de récolte d'information multi-agent hétérogènes basé sur les processus décisionnels de Markov. Cependant, ce modèle repose sur des hypothèses fortes sur les problèmes que l'on peut résoudre, en particulier sur les connaissances préalables des agents sur l'environnement. Dans ce chapitre, nous proposons une extension de ce modèle de manière à pouvoir résoudre des problèmes de récolte d'information sans aucune connaissance a priori sur l'environnement, hormis ses dimensions. De ce fait, nous devrons considérer le problème de l'exploration de carte et de la détection de points d'intérêt conjointement avec le problème de la récolte d'information sur ces points d'intérêt. Plus précisément, nous considérons des applications robotiques où un robot autonome, évoluant dans un environnement stochastique et partiellement observable, embarque des capteurs hétérogènes dans le but de construire une carte de l'environnement ainsi que de détecter et reconnaître de potentiels points d'intérêt. Nous considérons des capteurs globaux qui sont capables de couvrir une grande portion de la carte et de détecter les obstacles, mais qui ne sont généralement pas efficaces pour la

reconnaissance du type d'obstacle (par exemple des capteurs laser), et des capteurs locaux qui ne peuvent observer qu'une petite portion de l'environnement mais qui sont capables de reconnaître les types d'obstacles avec une grande précision (par exemple des caméras). Durant la suite de ce chapitre, nous considérerons les obstacles qui ne sont pas des murs comme étant des points d'intérêt. Les approches classiques utilisent la perception active pour construire la carte en envoyant le robot proche des endroits inconnus pour découvrir de nouvelles zones, comme nous l'avons présenté dans l'exploration à base de frontières, mais reconnaissent les points d'intérêt passivement, c'est-à-dire en utilisant leur caméra constamment durant le processus d'exploration, mais étant donné que la portée du capteur permettant de construire la carte est plus grande que celle du capteur pouvant observer les points d'intérêt, ces approches sont généralement mauvaises pour reconnaître des points d'intérêt qui sont proches des murs ou dans les coins des pièces. Dans le cadre de ces travaux, nous considérerons l'observabilité partielle uniquement dans le processus de reconnaissance des points d'intérêt, et non pas dans la position du robot. L'observabilité partielle sur la position du robot pourrait être prise en compte avec des techniques de *Simultaneous localization and mapping*(SLAM), comme nous l'avons présenté dans la section 3.4. Nous allons principalement nous appuyer sur les travaux d'exploration à base de frontière, et leur extension au cas stochastique que nous avons présenté en section 3.4.2 pour étendre le modèle de Meta-MDP que nous avons proposé au cas où nous avons un seul agent qui ne connaît pas la disposition de l'environnement à l'avance ni les éventuels points d'intérêt et leurs positions, et le but va être, à chaque instant, de déterminer quelle stratégie le robot doit employer : explorer l'environnement, en se déplaçant proche des zones inconnues et en utilisant le capteur global, ou récolter de l'information sur un point d'intérêt, en se rapprochant d'un point pour l'observer avec le capteur local. La difficulté supplémentaire est que, étant donné que l'agent n'a aucune connaissance a priori sur l'environnement, il est très difficile de calculer une politique hors-ligne, c'est-à-dire calculer la politique de l'agent en amont de la mission. En effet, lorsque l'environnement n'est pas connu, si l'on souhaite calculer une politique hors-ligne, il est nécessaire que l'état de la grille d'occupation représentant l'environnement fasse partie de l'état de notre modèle. Cependant, pour une grille d'occupation de taille $n \times m$, il existe $3^{n \times m}$ arrangements (avec répétition) de grilles d'occupation possibles (chaque cellule pouvant être soit libre, soit occupée, soit inconnue), et ce sans même tenir compte des éventuels points d'intérêt présents dans l'environnement. La figure 5.1 représente une grille d'occupation de taille 82x62 (image de taille 410x310 px avec une résolution de 5 px par cellule), il y a donc $3^{82 \times 62}$ états possibles pour cette grille.

FIGURE 5.1 – Grille d'occupation : en blanc, les cellules libres ; en noir, les cellules occupées ; et en gris, les cellules inconnues

De ce fait, il est en pratique impossible de calculer une telle politique hors ligne. Pour cela, nous proposons un modèle où les politiques sont calculées en ligne, durant la mission, ce qui nous permet de considérer l'état actuel de la grille d'occupation comme étant une connaissance de l'agent. Dans ce chapitre, nous nous intéresserons particulièrement au modèle de prise de décision qui permet à un robot de construire une politique qui dicte la stratégie à adopter pour un état donné (explorer ou reconnaître). Le processus de classification des points d'intérêt en eux-mêmes ne sera pas abordé.

Les travaux présentés dans ce chapitre ont été acceptés à la conférence internationale *40th Symposium on Applied Computing* (SAC 2025), track *Intelligent Robotics and Multi-Agent Systems* (IRMAS), et seront présentés publiquement en 2025 à Sicile en Italie [Gandois et al., 2025], sous la forme d'un poster.

5.2 Définition du modèle

La carte de l'environnement est représentée comme une grille d'occupation à deux dimensions, où chaque cellule contient une valeur de niveau de gris, telle qu'une cellule occupée est noire et une cellule libre est blanche. La grille d'occupation est mise à jour à chaque fois que le robot reçoit une information d'un capteur. Un état de l'environnement est défini comme la position (x, y) du robot dans la grille d'occupation et son angle de rotation $\theta \in \Theta$. L'angle de rotation est discrétisé de 0° à 360° avec un pas de 45°, ce qui représente les huit directions ordinales et cardinales. De ce fait, pour une grille de longueur W et de hauteur H, l'ensemble d'états S est défini de la manière suivante :

$$S = W \times H \times \Theta \tag{5.1}$$

À chaque pas de temps, le robot peut effectuer quatre actions :

$$A = \{droite, gauche, avancer, reculer\} \tag{5.2}$$

Une rotation vers la gauche correspond à une rotation de 45°, et une rotation vers la droite correspond à une rotation de -45°. Nous maintenons également un état de croyance b pour chaque point d'intérêt. Un état de croyance est une distribution de probabilité sur l'ensemble des valeurs possibles pour un point d'intérêt (cet ensemble est supposé comme étant une donnée du problème). Cela nous permet de garder en mémoire l'information que nous possédons actuellement à propos d'un point d'intérêt que l'on a détecté et d'optimiser en conséquence. Nous notons $\mathcal{B}$ l'ensemble des états de croyance pour chaque point d'intérêt. Formellement,

$$\mathcal{B} = \langle b_1, \ldots, b_n \rangle \tag{5.3}$$

où b_i est l'état de croyance pour le point d'intérêt i, et $b_i(x)$ est la probabilité que ce point d'intérêt soit à la valeur x. Étant donné que les calculs sont effectués en ligne, il n'est pas nécessaire d'avoir une connaissance préalable du nombre de points d'intérêt dans la pièce, ni de leur position.

Ce modèle fonctionne de la manière suivante : à chaque fois que le robot reçoit une nouvelle information (c'est-à-dire une information qu'il ne connaît pas encore) de la part d'un capteur, l'algorithme suivant est appliqué :

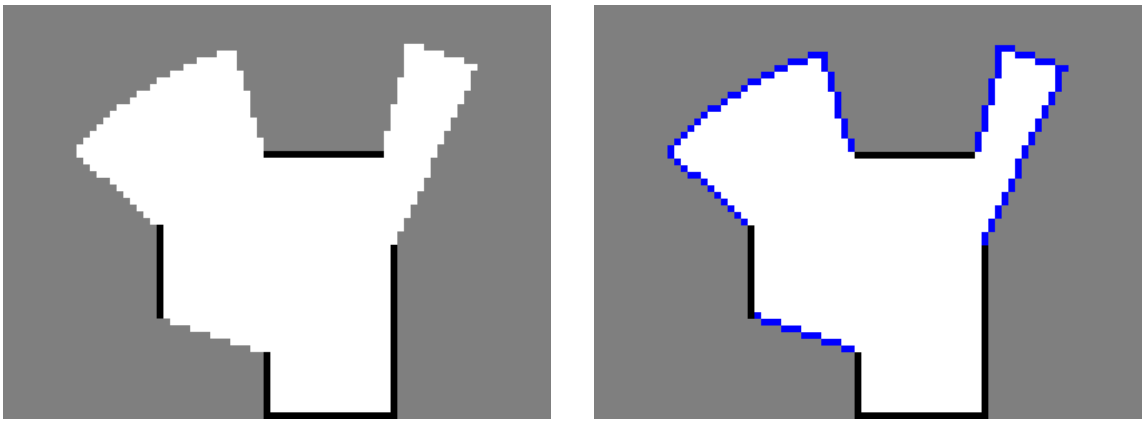
1. Mise à jour des frontières entre les cellules connues et inconnues
2. Détecter les points d'intérêt dans la grille d'occupation
3. Générer le MDP d'exploration de carte

4. Calculer une politique d'exploration de carte
5. Pour chaque point d'intérêt non reconnu, calculer une politique de reconnaissance pour ce point
6. Calculer la méta-politique pour choisir la meilleure politique à exécuter (exploration de carte ou reconnaissance d'un point d'intérêt)
7. Exécuter la méta-politique

Nous allons détailler dans les sections suivantes chaque étape de cet algorithme.

5.2.1 Mise à jour des frontières

Les frontières, comme dans l'exploration à base de frontière, sont l'ensemble des cellules connues qui sont adjacentes à une ou plusieurs cellules inconnues. Les cellules adjacentes à l'intérieur de la frontière globale peuvent être regroupées en sous-ensembles pour former des segments de frontière, mais dans le cadre de ces travaux nous considérerons la frontière dans sa globalité, que nous noterons $\mathcal{F}$. La figure 5.2.1 illustre une grille d'occupation (a) et la frontière entre les zones connues et les zones inconnues (b).



(a) Grille d'occupation (b) Frontières

FIGURE 5.2 – Grille d'occupation et frontières entre le connu et l'inconnu (en bleu)

Formellement, lorsque le robot détecte une nouvelle cellule c qui est libre dans la grille d'occupation, cette cellule est considérée dans la frontière $\mathcal{F}$ si :

$$\exists c' \in neighbors(c), IsUnknown(c') \tag{5.4}$$

5.2.2 Algorithme de détection de points d'intérêt

Afin de pouvoir calculer des politiques de reconnaissance pour les points d'intérêt, nous devons d'abord différencier les murs des points d'intérêt à l'aide de la grille d'occupation. Nous considérerons ici un cas simple où, lorsqu'une cellule occupée est isolée à moins d'une certaine distance d des autres cellules, nous considérons qu'il s'agit d'un point d'intérêt, et non un segment de mur. Formellement, une cellule c est un point d'intérêt si :

$$\nexists c' \in W \times H, c' \neq c, IsOccupied(c'), dist(c, c') < d \tag{5.5}$$

Pour des cas plus complexes, des algorithmes de détection plus robustes peuvent être utilisés [Zou et al., 2023]. Une fois qu'un point d'intérêt a été détecté, nous commençons à maintenir

un état de croyance sur la valeur de ce point. Nous considérerons ici un état de croyance initial uniforme sur l'ensemble des valeurs possibles, mais en pratique il est également possible de déterminer un état de croyance initial plus précis, par exemple lorsque l'on travaille avec des capteurs laser, des techniques de détection d'objet depuis un nuage de point de laser peuvent être utilisées.

5.2.3 Modèle d'exploration de carte

L'objectif de ce modèle est de calculer une politique pour que le robot explore son environnement. L'exploration est faite grâce à un capteur longue portée, typiquement un capteur laser, qui nous permet de construire une grille d'occupation correspondant à l'état réel de la carte. Ce modèle repose sur un MDP $\langle S, A, T, R_{exp} \rangle$ où l'ensemble S d'états et l'ensemble A d'actions sont définis dans les équations 5.1 et 5.2. La fonction de transition T est une fonction de transition classique pour un MDP, qui représente comment l'environnement évolue lorsque le robot exécute une action dans un état donné. La partie la plus importante de ce modèle repose sur la fonction de récompense. Le but de ce modèle est d'explorer la carte, mais également d'éviter les zones inconnues, et surtout, éviter les cellules connues comme étant occupées. De ce fait, nous utilisons la fonction de récompense pour appliquer des récompenses aux cellules appartenant à la frontière car ce sont les cellules qui sont connues comme étant libres et où l'on peut trouver de nouvelles informations. De plus, dû à la portée du capteur, le robot n'a pas nécessairement besoin de se déplacer directement sur la cellule en elle-même pour recevoir de l'information, nous pouvons donc appliquer une récompense sur les cellules proches de la frontière. Concrètement, une cellule de frontière peut contenir une récompense arbitraire (par exemple 1), et cette récompense est propagée dans un rayon autour de cette cellule. Formellement, la fonction de récompense d'une cellule $R(c)$ est définie de la manière suivante :

$$\begin{cases} 1 & \text{si } c \in \mathcal{F} \\ 0 & \text{sinon} \end{cases} \tag{5.6}$$

Et, pour chaque cellule $c \notin \mathcal{F}$ qui n'est pas occupée,

$$R(c) = \sum_{c' \in \mathcal{F}} \frac{R(c')}{dist(c, c')} \tag{5.7}$$

Ceci est la première étape pour calculer la fonction de récompense pour chaque cellule. Cependant, pour être exhaustif, nous devons définir une fonction de récompense pour chaque cellule et chaque angle de rotation du robot, car par exemple, si il y a un obstacle (comme un mur) entre le robot et la cellule inconnue, alors le robot ne peut pas forcément l'observer. De ce fait, pour chaque cellule $c \in W \times H$ et chaque $\theta \in \Theta$, $R_{exp}(c, \theta)$ est défini de la manière suivante :

$$\begin{cases} R(c) & \text{si } isObservable(c, \theta) \\ 0 & \text{sinon} \end{cases} \tag{5.8}$$

où *isObservable* est une macro qui, étant donné l'état actuel du robot, vérifie si le robot peut actuellement observer la cellule ou non, en utilisant par exemple un algorithme de raycasting.

5.2.4 Calculer une politique d'exploration de carte

Une fois que le modèle d'exploration de carte est généré, nous pouvons calculer une politique optimale pour explorer la carte. La politique est calculée en utilisant l'algorithme Value Iteration que nous avons présenté dans la section 2.2.5 à horizon fini avec un facteur d'atténuation.

Le choix de l'horizon pour calculer une politique est important car si l'on choisit un horizon trop faible alors le robot ne sera pas capable d'atteindre les récompenses, et si l'on choisit un horizon trop élevé alors le temps de calcul peut devenir trop long. La solution proposée par [Le Gloannec et al., 2010], que nous avons présentée en section 3.4.2, et que nous avons également utilisée, est d'incrémenter l'horizon durant le calcul afin d'être garanti d'avoir un horizon minimal qui permette au robot d'atteindre les récompenses. Nous notons π_{exp} la politique obtenue par ce processus :

$$\pi_{exp}(s) = \arg\max_{a \in A} \left(R_{exp}(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^*(s') \right) \tag{5.9}$$

5.2.5 Modèle de reconnaissance de point d'intérêt

Le but du modèle de reconnaissance de point d'intérêt est, étant donné un point d'intérêt i et son état de croyance courant $b_i \in \mathcal{B}$, calculer une politique optimale pour atteindre ce point d'intérêt et l'observer, tout en continuant de récolter de l'information en chemin (par exemple pour cartographier). Ce modèle repose sur un POMDP d'exploration comme nous l'avons décrit en section 4.3.1. Formellement, pour un point d'intérêt i, nous définissons un $POMDP_i\langle S, A \cup \{OBSERVE\}, T, R, \Omega, O\rangle$, où :

- S est l'ensemble des états de l'environnement tel que défini en équation 5.1
- A est l'ensemble d'actions tel que défini en équation 5.2, auquel on ajoute une action OBSERVE qui permet au robot d'utiliser son capteur de reconnaissance
- T est une fonction de transition classique
- Ω et O sont respectivement l'ensemble d'observations et la fonction d'observation. Ces paramètres peuvent être déterminés par exemple en utilisant un classifieur qui est capable de classifier un ensemble d'objets selon une distribution de probabilité donnée

La fonction de récompense de ce modèle dépend de l'état de croyance actuel du point d'intérêt mais également du gain possible en information pour l'exploration. Afin de récompenser le robot pour son état de croyance, nous allons une nouvelle fois utiliser l'entropie de Shannon, définie de la manière suivante :

$$H^-(i) = \log_2(|DOM(i)|) + \sum_{v \in DOM(i)} b_i(x) \log_2 b_i(x) \tag{5.10}$$

où $DOM(i)$ est l'ensemble des valeurs possibles pour ce point d'intérêt. Nous pouvons maintenant définir la fonction de récompense pour la reconnaissance de point d'intérêt de la manière suivante :

$$R(c,\theta) = R_{exp}(c,\theta) + H^-(i) \tag{5.11}$$

5.2.6 Modèle de sélection de politique

L'objectif de ce modèle est de sélectionner la meilleure stratégie à appliquer à chaque pas de temps, c'est-à-dire soit la cartographie, soit la reconnaissance d'un point d'intérêt (et lequel). Ce modèle repose sur un Meta-MDP $\langle S, A, T, R\rangle$, où :

- S est l'ensemble des états de l'environnement tel que défini en équation 5.1
- A est l'ensemble d'actions qui sont des politiques qui peuvent être exécutées

Dans cette configuration, l'ensemble d'actions est l'ensemble des politiques (exploration et reconnaissance de point d'intérêt) qui peuvent être exécutées. De manière à réduire les temps de calcul, nous avons intégré un paramètre k, qui permet de nous restreindre aux k objets les plus

proches (ainsi que la politique d'exploration de carte), ce qui signifie qu'à chaque pas de temps nous avons au plus $k+1$ méta-actions disponibles. Par exemple, pour $k = 3$, l'ensemble d'actions est : $A = \{\pi_{exp}, \pi_1, \pi_2, \pi_3\}$. Évidemment, si nous avons moins de k points d'intérêt à reconnaître, alors moins d'actions seront disponibles. Un cas particulier de cette approche est lorsque $k = 1$, alors dans ce cas, la stratégie résultante consiste à toujours choisir entre explorer la carte ou reconnaître le point d'intérêt le plus proche. La fonction de transition est la partie importante de ce modèle. Nous devons évaluer quels sont les états accessibles (et leur probabilité) après avoir exécuté une politique de MDP pour H pas de temps. Pour cela, nous utilisons l'algorithme que nous avons introduit en section 4.3.2 qui utilise la technique de l'analyse du premier pas, ainsi que la méthode d'approximation que nous avons proposée.

Étant donné que le but de ce modèle est de récolter le plus d'information sur l'environnement et de reconnaître les points d'intérêt, nous allons une nouvelle fois utiliser l'entropie de Shannon définie dans l'équation 5.10 dans notre fonction de valeur, ce qui nous amène à la reformulation suivante de l'équation de Bellman :

$$V(s) = R_{exp}(s) + \sum_{b \in \mathcal{B}} H^{-}(b) + \gamma \max_{a \in A} \sum_{s' \in S} T(s, a, s')V(s') \tag{5.12}$$

Résoudre cette équation nous fournit une politique optimale qui dicte, dans chaque état, quel capteur le robot doit utiliser, c'est-à-dire le capteur global pour cartographier l'environnement, ou le capteur local pour reconnaître un point d'intérêt (et lequel). Pour ce faire, des algorithmes comme Value Iteration peuvent être utilisés.

5.2.7 Exécution de la méta-politique

Une fois la méta-politique calculée, l'agent exécute la méta-action correspondant à son état courant, c'est-à-dire soit la politique d'exploration de la carte, soit une politique de reconnaissance de point d'intérêt. Les politiques de reconnaissance de points d'intérêt sont exécutées jusqu'à ce que l'agent observe réellement le point choisi, tandis que la politique d'exploration est exécutée jusqu'à ce que l'agent reçoive une nouvelle information.

5.3 Complexité

La complexité de ce modèle est identique à celle du Meta-MDP que nous avons présenté en section 4.3.2. Cependant, le fait de n'avoir qu'un seul agent réduit la taille de l'espace d'état de manière considérable, et l'introduction du paramètre k pour limiter le nombre de points d'intérêts considérés, permettent un meilleur passage à l'échelle que le Meta-MDP. Le paramètre k est important d'une part car il n'y a besoin de calculer que k politiques de reconnaissance au lieu de n, mais surtout, l'espace d'actions du Meta-MDP est réduit à seulement $k + 1$ actions (k reconnaissances et 1 exploration), ce qui fait que la résolution du Meta-MDP est grandement simplifiée.

5.4 Conclusion

Dans ce chapitre, nous avons présenté un modèle pour l'exploration de carte et la reconnaissance de points d'intérêt dans un environnement stochastique et partiellement observable avec des capteurs hétérogènes. Ce modèle se base sur le modèle Meta-MDP que nous avons décrit précédemment et permet de calculer une politique pour différents objectifs qui sont quel type

de capteur le robot doit utiliser (capteur global pour cartographier l'environnement, ou capteur local pour explorer un point d'intérêt). Nous avons également introduit un paramètre à notre modèle k qui permet de restreindre le nombre de points d'intérêts considérés et donc de réduire la complexité du modèle en pratique.

Troisième partie

Évaluation expérimentale

Chapitre 6

Récolte d'information multi-agents dans un environnement partiellement connu

Sommaire

6.1 Introduction

Dans ce chapitre, nous allons évaluer le modèle de Meta-MDP que nous avons introduit dans le chapitre 4 sur un problème de récolte d'information sur des points d'intérêt dans un environnement partiellement connu et partiellement observable avec des agents hétérogènes.

6.2 Protocole expérimental

Pour évaluer notre modèle, nous allons considérer un domaine de grille où nous avons plusieurs points d'intérêt, qui sont représentés par des variables booléennes, placés dans une grille à deux dimensions, qui peuvent représenter par exemple la présence d'un danger dans cet endroit. Le but va être d'évaluer tous les points d'intérêt dans le moins de temps possible. La figure 6.1 illustre un exemple d'environnement. Pour cette expérimentation, nous aurons deux agents : un agent autonome (par exemple un robot terrestre) qui possède une capacité d'observation locale mais très précise (qui reçoit presque toujours des informations correctes), que nous noterons par la suite AAL, et un agent autonome (par exemple un drone) qui possède une capacité d'observation globale mais peu précise (qui reçoit régulièrement des informations erronées), que nous noterons par la suite AAG.

Le AAL peut effectuer 4 actions différentes : HAUT, DROITE, BAS, GAUCHE. Comme nous l'avons présenté dans le modèle, nous ajoutons à cela également deux actions : STAY

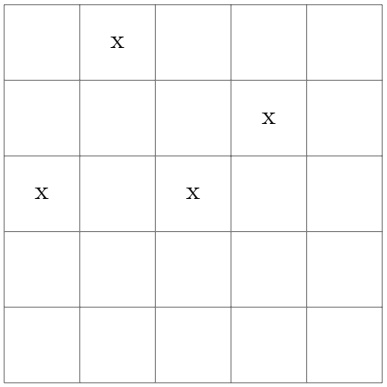


FIGURE 6.1 – Une grille de taille 5x5 avec 4 points d'intérêt (marqués par la lettre "x").

et OBSERVE. L'action STAY est une action qui ne fait rien et le robot reste dans la même cellule sans recevoir aucune information. Les 4 actions de déplacement déplacent le robot de 1 cellule dans la direction choisie et, si la nouvelle cellule contient un point d'intérêt, il reçoit une information correspondant à la table 6.1. Lorsque le robot utilise l'action OBSERVE dans une

Real value \ Observation	True	False
True	0.55	0.45
False	0.45	0.55

TABLE 6.1 – Fonction d'observation du AAL pour les actions de déplacement.

cellule contenant un point d'intérêt, il reçoit une information correspondant à la table 6.2.

Real value \ Observation	True	False
True	0.80	0.20
False	0.20	0.80

TABLE 6.2 – Fonction d'observation du AAL pour l'action OBSERVE.

Le AAG possède le même ensemble d'actions que le AAL, mais ses observations sont moins précises. Il a seulement 60% de chance d'observer le vrai état d'un point d'intérêt lorsqu'il applique l'action OBSERVE, et 55% lorsqu'il applique une action de déplacement.

6.3 Modèles de comparaison

Nous allons comparer notre modèle par rapport à deux autres approches que nous n'allons pas décrire dans les détails ici. La première approche est une approche basée sur les MPOMDP (Multi-Agent POMDP) que nous avons évoquée dans la section 2.4.1, qui est la solution optimale lorsque l'on considère les hypothèses de notre modèle (partage d'information, système centralisé). La seconde approche utilise un algorithme d'allocation de tâche optimal pour directement allouer les points d'intérêt aux robots en utilisant la fonction de valeur des politiques d'exploration

individuelles comme récompenses. La différence entre le Meta-MDP et le simple algorithme d'allocation de tâche est que le Meta-MDP considère les récompenses à long terme tandis que l'algorithme d'allocation de tâche ne considère que les récompenses directes de manière myope (relatif au niveau macro, car les politiques d'exploration sont toujours calculées de manière optimale pour une exploration à long terme). L'approche par allocation de tâche peut être vue comme un cas particulier du Meta-MDP où l'horizon de calcul est 1 et la profondeur de la Meta-fonction de transition est équivalente à l'horizon des politiques d'exploration. Les résultats attendus sont que le MPOMDP fournit une meilleure solution que le Meta-MDP, qui est supposé être meilleur que l'allocation de tâche. Cependant, le Meta-MDP offre un meilleur passage à l'échelle théorique que le MPOMDP.

Nous allons évaluer ces trois approches selon les critères suivants : le nombre moyen de pas de temps (actions exécutées par les robots) nécessaires pour complètement évaluer la situation (c'est-à-dire déterminer complètement l'état de tous les points d'intérêt). Par souci de simplicité, nous avons résolu ces modèles en utilisant un espace d'état de croyance discret. Chaque état de croyance pour chaque point d'intérêt possède 10 valeurs possibles, réparties de la manière suivante : $\{0.0, 0.11, 0.22, \ldots, 1.0\}$ (qui est largement suffisant pour résoudre ce type de problème). Nous avons également évalué le Meta-MDP pour différents paramètres de profondeur, avec et sans approximation, afin d'évaluer l'impact de l'approximation sur les temps de calcul et sur la qualité de la solution fournie.

6.4 Résultats

Dans cette section, nous allons présenter les résultats expérimentaux que nous avons obtenus sur l'exemple que nous avons décrit précédemment. Nous allons commencer par évaluer la qualité de la solution et la comparer aux deux autres approches tout en montrant l'évolution en fonction de différents paramètres de profondeur et en ajoutant l'approximation. Ensuite, nous allons mesurer comment le temps de calcul du Meta-MDP évolue lorsque l'on augmente la profondeur, avec et sans approximation. Enfin, nous allons discuter de la complexité en temps des différents modèles que nous avons évalués.

6.4.1 Qualité de la solution

La figure 6.2 montre les résultats pour les trois approches. Le paramètre de profondeur utilisé pour la Meta-Fonction de transition est 6 sans aucune approximation (ce qui correspond au maximum de profondeur que l'on a pu calculer en un temps raisonnable). L'axe des abscisses représente le nombre de pas de temps nécessaires pour complètement évaluer la situation, et l'axe des ordonnées est la probabilité que le modèle évalue complètement la situation dans le nombre de pas de temps donné. Pour commencer, nous pouvons remarquer que les trois modèles donnent des résultats similaires, qui sont dans le même ordre de grandeur, avec la même tendance, et l'écart type semble relativement équivalent. Comme nous l'avions imaginé, le modèle basé sur un MPOMDP est le plus efficace, avec un nombre de pas de temps moyen de 19.85. L'approche de référence (l'algorithme d'allocation de tâche) est très proche de la solution optimale avec un nombre de pas de temps moyen de 20.01. Notre Meta-MDP montre de bonnes performances (20.49 pas de temps en moyenne), ce qui représente 96.73% de la solution optimale. Le nombre de pas de temps moyen nécessaire à chaque modèle est résumé dans la table 6.3.

Nous allons désormais évaluer comment les performances du Meta-MDP évoluent lorsque l'on augmente la profondeur de calcul et la Meta-fonction de transition, et comment l'approximation impacte ses performances. Pour chaque expérimentation, nous avons utilisé la méthode

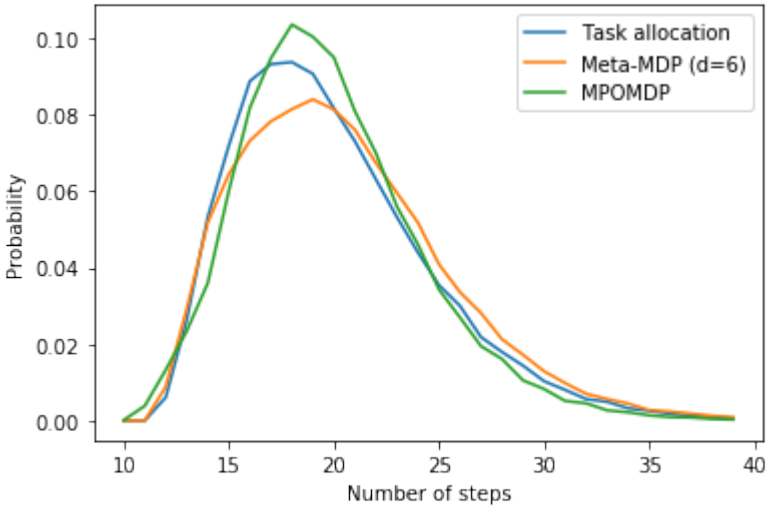


FIGURE 6.2 – Nombre de pas de temps nécessaires pour complètement évaluer la situation, 100 000 échantillons.

Model	Average steps required
MPOMDP	19.85
Meta-MDP (d=6)	20.49
Task allocation algorithm	20.01

TABLE 6.3 – Nombre moyen de pas de temps nécessaires pour complètement évaluer la situation avec chaque modèle.

d'approximation que nous avons présentée dans la section 4.3.3 (élagage des états les moins probables) avec un seuil d'élagage γ de 0.75.

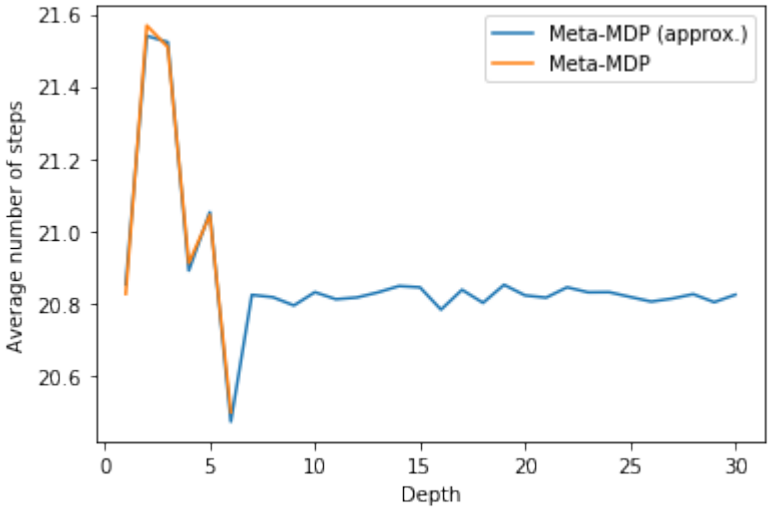


FIGURE 6.3 – Nombre moyen de pas de temps nécessaires pour complètement évaluer la situation en utilisant le Meta-MDP à différentes profondeurs, avec et sans approximation.

La figure 6.3 montre les résultats lorsque l'on augmente le paramètre de profondeur de la Meta-Fonction de transition avec et sans approximation. Nous pouvons constater que les performances du Meta-MDP ne sont pas dégradées par l'approximation, tandis que nous pouvons calculer à des profondeurs plus importantes. Nous pouvons également remarquer que la solution ne semble pas s'améliorer lorsque l'on augmente la profondeur (il se passe même l'effet inverse par moment), ce qui n'est pas le résultat que l'on attendait. Afin de comprendre ce qu'il se passe, nous avons également comparé dans la figure 6.4 le nombre d'allocations (meta-pas de temps, c'est-à-dire le nombre de fois qu'un nouveau point d'intérêt est alloué à chaque robot) nécessaires pour évaluer complètement la situation. Nous pouvons voir que plus la profondeur augmente, moins de réallocations sont effectuées, et encore une fois l'approximation ne semble pas avoir d'influence majeure. La raison de ce résultat est que le Meta-MDP, tel que nous l'avons défini, cherche à minimiser le nombre de méta-pas de temps, et non pas le nombre de pas de temps global. Plus précisément, étant donné que la fonction de récompense du Meta-MDP ne prend en compte que l'entropie de l'état suivant, le robot cherchera à optimiser de manière à récolter le plus d'information possible dans un seul pas de temps, ce qui mène à plus d'actions par méta-pas de temps, et réduit le nombre de méta-pas de temps. Nous discuterons de ce point par la suite.

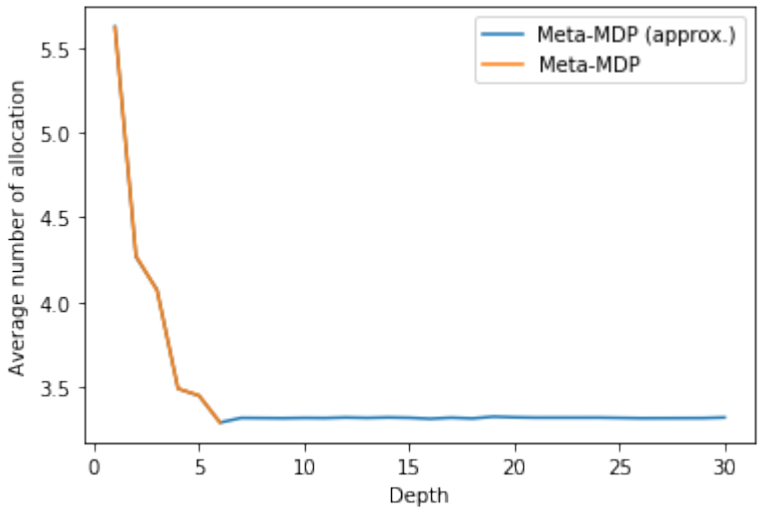


FIGURE 6.4 – Nombre moyen d'allocation de points nécessaires pour complètement évaluer la situation en utilisant le Meta-MDP à différentes profondeurs, avec et sans approximation.

6.4.2 Temps de calcul

Dans la figure 6.5, nous avons mesuré le temps de calcul de la politique du Meta-MDP lorsque nous augmentons la profondeur de calcul de la Meta-Fonction de transition et lorsque nous ajoutons l'approximation. Nous pouvons voir que le temps de calcul sans approximation croît de manière exponentielle tandis que, lorsque l'on utilise la méthode d'approximation, le temps de calcul croît de manière presque linéaire, ce qui conforte nos déclarations précédentes. Un point intéressant est qu'aux alentours de la profondeur 13, le temps de calcul avec approximation décroît puis, ensuite, augmente beaucoup plus lentement. Nous expliquons cela par le fait que l'algorithme de calcul de politique a convergé vers une politique optimale plus tôt que pour les profondeurs précédentes (une itération de moins), et nous supposons que le seuil d'élagage γ que nous avons utilisé est tel qu'augmenter la profondeur ne permet pas de découvrir beaucoup plus

d'états suffisamment probables.

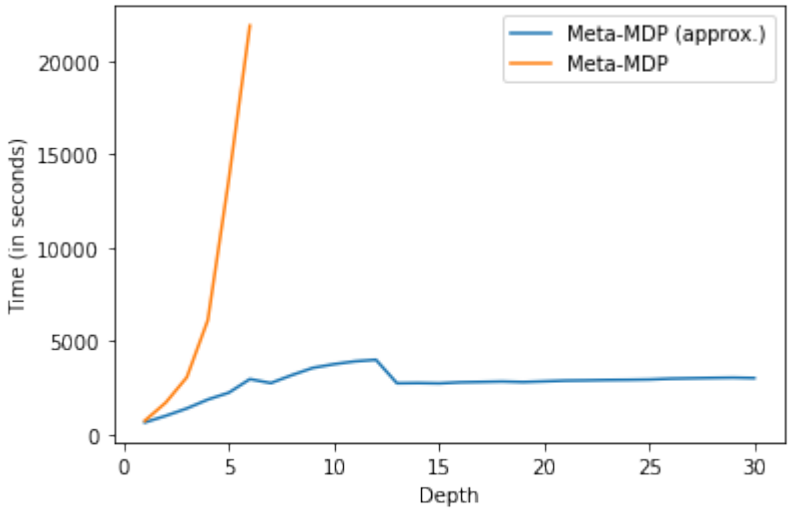


FIGURE 6.5 – Temps de calcul de la politique du Meta-MDP à différentes profondeurs, avec et sans approximation.

6.4.3 Complexité

Pour l'algorithme d'allocation de tâche, nous avons utilisé l'algorithme Hongrois [Kuhn, 1955], qui est un algorithme couramment utilisé pour l'allocation optimale de tâches. L'algorithme Hongrois a une complexité temporelle de $O(n^3)$ (avec n étant le nombre de tâches). Étant donné que nous n'avons pas beaucoup de tâches dans notre exemple, le temps de calcul pour cette approche est négligeable, et fournit un meilleur passage à l'échelle que les deux autres approches. Le modèle classique de POMDP est PSPACE-complet à horizon fini, et indécidable à horizon infini. Le modèle à base de MPOMDP étant l'extension multi-agent du POMDP, sa complexité est également PSPACE-complet. Concernant le Meta-MDP, l'approximation permet un meilleur passage à l'échelle que le Meta-MDP sans approximation, et dans le pire des cas, le Meta-MDP a une complexité au moins équivalente au MPOMDP, mais des travaux plus approfondis sont nécessaires pour établir une comparaison théorique entre le Meta-MDP et le MPOMDP.

6.5 Conclusion

Dans ce chapitre, nous avons évalué notre modèle de Meta-MDP pour de la récolte d'information dans un environnement stochastique partiellement observable avec des agents hétérogènes. Nous avons comparé ce modèle à une approche optimale définie par un MPOMDP et une approche de référence définie par un algorithme d'allocation de tâche dans un environnement représenté par une grille, et nous avons montré que le modèle donne des résultats encourageants mais qui pourraient néanmoins être améliorés en retravaillant la fonction de récompense du Meta-MDP. Nous avons également montré que la méthode d'approximation que nous avons proposée nous permet de réduire la complexité de notre algorithme pour calculer la Meta-fonction de transition avec des résultats similaires en termes de qualité de solution, par rapport à la solution optimale.

Chapitre 7

Exploration et reconnaissance active mono-agent dans un environnement totalement inconnu

Sommaire

7.1 Contexte

Au début des années 2010, la DGA [6] et l'ANR [7] se sont associés pour lancer le défi CAROTTE (CArtographie par un ROboT d'un TErritoire). L'objectif de ce défi, d'une durée de 3 ans, était de faire s'affronter différentes équipes de recherche dans un scénario d'exploration et de reconnaissance d'objets avec un système robotisé autonome évoluant dans une arène composée de différentes pièces. Les seules informations à disposition des équipes étaient la taille de l'arène à explorer, et la liste des différents types d'objets qui pourraient s'y trouver. Le but des robots était alors de cartographier leur environnement et de reconnaître les différents objets, puis sortir de l'arène avant la fin du temps imparti (30 minutes). La figure 7.1 illustre un exemple de carte générée lors de ce défi.

L'équipe qui était arrivée vice-championne de la compétition, dont faisait partie le laboratoire GREYC dans lequel cette thèse a été réalisée, avait développé une stratégie avec deux robots

6. Direction Générale de l'Armement
7. Agence Nationale de la Recherche

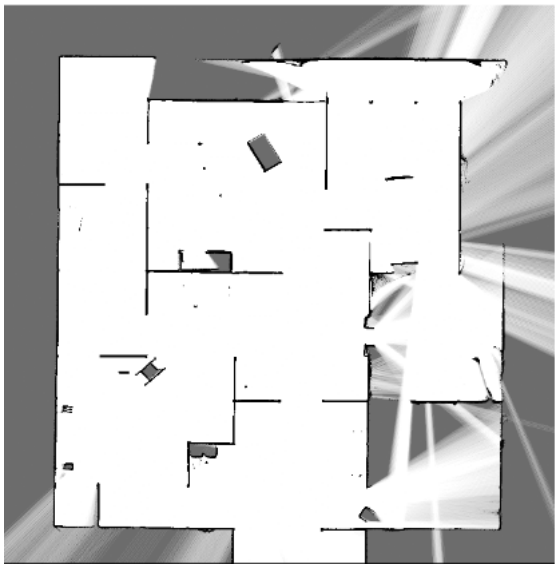


FIGURE 7.1 – Exemple de carte générée lors du défi CAROTTE [Matignon et al., 2015]

embarquant chacun un capteur laser pour cartographier l'arène et une caméra pour prendre des photos. Un algorithme de classification était utilisé pour reconnaître les éventuels objets pris en photos. La difficulté de cette approche était de trouver un équilibre entre exploration et reconnaissance, car les robots essayaient de maximiser la surface couverte par des photos tout en cartographiant l'entièreté de l'arène. Cependant, étant donné que le temps était limité et que le capteur laser permettait aux robots de cartographier la zone à longue distance, il n'était pas intuitif de se rendre dans des zones déjà couvertes par le capteur laser pour les prendre en photo. De ce fait, les robots pouvaient parfois manquer des objets qui se trouvaient proches des murs, car ils ne jugeaient ces zones pas suffisamment intéressantes pour s'y rendre.

La stratégie développée a été publiée [Matignon et al., 2015] et repose sur un DEC-MDP où la fonction de récompense est une somme pondérée entre l'exploration et la reconnaissance. Plus formellement, un robot calcule une fonction de récompense pour l'exploration, notée R_{exp}, et une fonction de récompense pour la reconnaissance, notée R_{cov}. Ces fonctions de récompenses sont définies à partir des frontières entre les zones couvertes par les différents capteurs (R_{exp} pour le capteur laser, et R_{cor} pour la caméra), comme nous l'avons présenté en section 3.4.2. Ensuite, la fonction de récompense du MDP est définie comme une somme pondérée entre la fonction de récompense pour l'exploration et la fonction de récompense pour la reconnaissance de la manière suivante :

$$R(s) = (1 - \alpha)R_{exp}(s) + \alpha R_{cov}(s) \tag{7.1}$$

où α est un paramètre permettant d'ajuster l'importance que le robot accorde à l'exploration par rapport à la reconnaissance. Lorsque $\alpha = 0$, le robot cherchera uniquement à explorer la carte, et lorsque $\alpha = 1$, le robot cherchera uniquement à faire de la reconnaissance. Nous avons présenté ici une version simplifiée de la fonction de récompense utilisée en pratique, car la stratégie développée utilisait une fonction de valeur distribuée [Matignon et al., 2012b] pour gérer la coordination entre les robots.

L'objectif de cette expérimentation n'est évidemment pas de reproduire le défi de l'époque à l'identique, mais simplement d'évaluer notre modèle sur un problème concret de récolte d'information qui consiste à explorer son environnement et reconnaître des objets.

7.2 Protocole expérimental

Nous avons évalué notre approche dans un environnement simulé ressemblant à une zone de bureaux comme illustré dans la figure 7.2 avec un seul robot embarquant deux capteurs hétérogènes. Le premier capteur est un capteur laser qui permet de détecter les obstacles et de

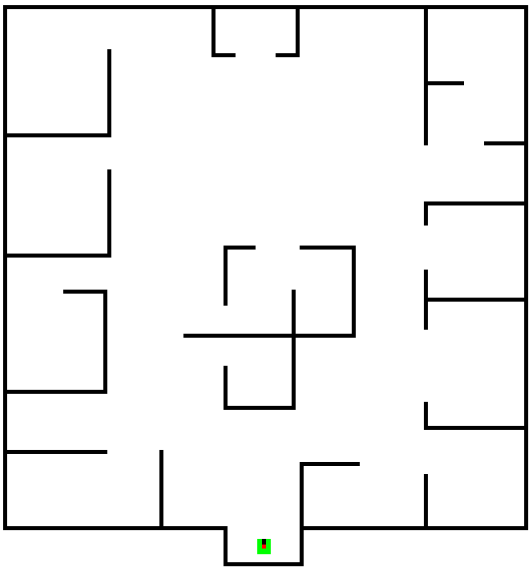


FIGURE 7.2 – L'environnement utilisé pour nos expérimentations

construire une carte de la zone, et une caméra qui permet de prendre des photos de l'environnement afin de reconnaître les objets. Les spécifications techniques de ces capteurs simulés sont décrites dans la table 7.1 et seront les mêmes pour toutes les expérimentations. Au début de la mission, le robot se situe au milieu en bas de la carte sans aucune connaissance au préalable sur l'environnement, hormis ses dimensions et les types d'objets qu'il peut y trouver. Un nombre aléatoire d'objets (entre 1 et 10) est placé aléatoirement dans l'environnement. Le but pour le robot est d'explorer entièrement la carte et de retourner à sa position initiale. Nous utilisons des grilles de taille 150x150 cellules, une cellule ayant une résolution de 5x5 px, ce qui est proche des valeurs permettant de représenter un environnement réel.

TABLE 7.1 – Caractéristiques des capteurs

Capteur	Portée	Angle
Laser	250px (50 cellules)	220°
Caméra	50px (10 cellules)	70°

7.3 Modèles de comparaison

Nous allons évaluer et comparer trois stratégies différentes pour récolter de l'information dans l'environnement (exploration de la carte et reconnaissance des objets) :

— Exploration active, reconnaissance passive
— Reconnaissance complète

— Exploration active, reconnaissance active (notre modèle)

Pour chaque stratégie et chaque nombre d'objets entre 0 et 10, nous avons mené 100 expérimentations. Les détails de chaque stratégie sont décrits dans les sections suivantes. Pour les illustrations des cartes générées, le code couleur est le suivant : les cellules grises sont des cellules qui n'ont pas été cartographiées par le robot, les cellules noires sont des obstacles, les cellules blanches sont des cellules libres, et les cellules vertes sont les cellules couvertes par la caméra. Les cellules de couleur rouge sont les cellules dans lesquelles un objet est présent (mais pas forcément reconnu).

7.3.1 Exploration active, reconnaissance passive

La stratégie de reconnaissance passive consiste à explorer la zone et cartographier la carte avec le capteur laser tout en prenant des photos avec la caméra, mais sans nécessairement essayer de se rapprocher des objets pour les reconnaître correctement. De ce fait, cette approche devrait reconnaître peu d'objets mais avec un temps d'exploration très court. La figure 7.3 illustre un exemple de carte construite avec cette stratégie. Nous pouvons constater que tous les objets n'ont pas été couverts par la caméra.

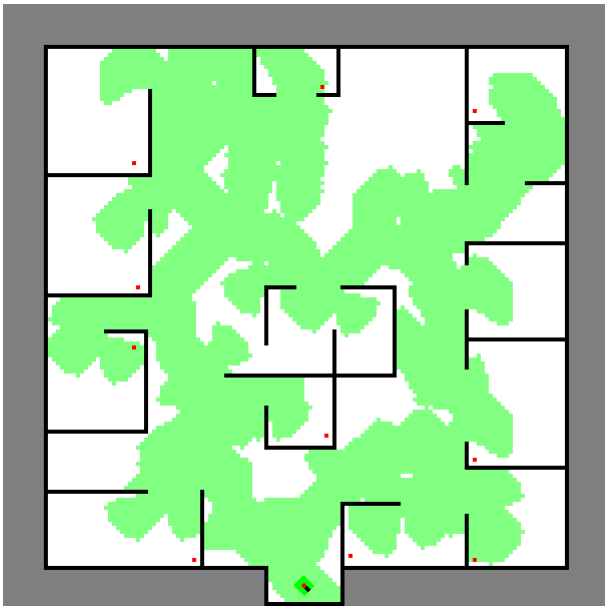


FIGURE 7.3 – Reconnaissance passive

7.3.2 Reconnaissance complète

La stratégie de reconnaissance complète consiste à prendre des photos de toute la zone de manière à reconnaître le plus d'objets possible durant l'exploration. Cette approche est garantie de reconnaître tous les objets, mais avec un temps d'exploration très élevé, car le robot ira systématiquement dans des zones inintéressantes qui ont déjà été balayées par le capteur laser. La figure 7.4 illustre un exemple de carte construite avec cette stratégie.

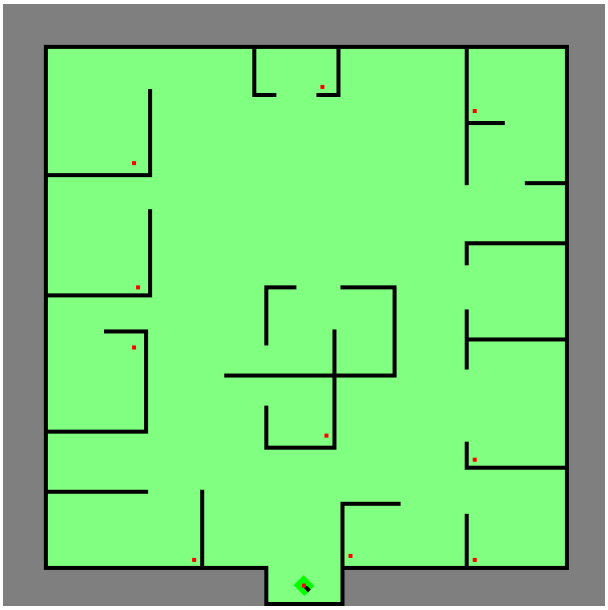


FIGURE 7.4 – Reconnaissance complète

7.3.3 Notre approche : exploration et reconnaissance active

La stratégie de reconnaissance active est celle que nous avons présentée dans le chapitre 5 (Meta-MDP pour la récolte d'information mono-agent avec des capteurs hétérogènes). Cette stratégie est supposée reconnaître tous les objets et cartographier toute la carte avec un temps d'exploration qui augmente selon le nombre d'objets reconnus. La figure 7.5 illustre un exemple de carte construite avec cette stratégie. Nous remarquons que tous les objets sont bien couverts par la caméra, mais également que la caméra n'a été utilisée que dans les zones présentant des objets, ce qui est également un avantage dans un contexte d'économie de ressources.

7.4 Critères de comparaison

Nous allons évaluer les trois stratégies énoncées précédemment par rapport aux critères suivants :

— Nombre d'objets reconnus
— Temps d'exploration
— Temps de calcul
— Temps total de la mission (temps d'exploration + temps de calcul)

Le temps d'exploration considère le temps réel passé par le robot à explorer la carte ou observer un objet, c'est-à-dire sans considérer le temps de calcul. Comme il s'agit d'un environnement simulé, il est difficile de déterminer de manière exacte le temps d'exploration réel. En revanche, nous pouvons mesurer le nombre de pas de temps nécessaires à l'exécution de la mission, puis choisir une valeur arbitraire comme échelle de temps. Dans les expérimentations d'exploration dans un environnement stochastique menées par [Le Gloannec et al., 2010], un pas de temps a été mesuré comme durant $1,8$ secondes, nous avons donc décidé de choisir cette valeur comme durée d'un pas de temps. Cependant, il est important de noter que cette valeur dépend grandement du robot employé et des capteurs utilisés. De plus, dans [Le Gloannec et al., 2010], cette durée comprend également le temps de calcul, ce qui ne sera pas le cas pour nos expérimentations.

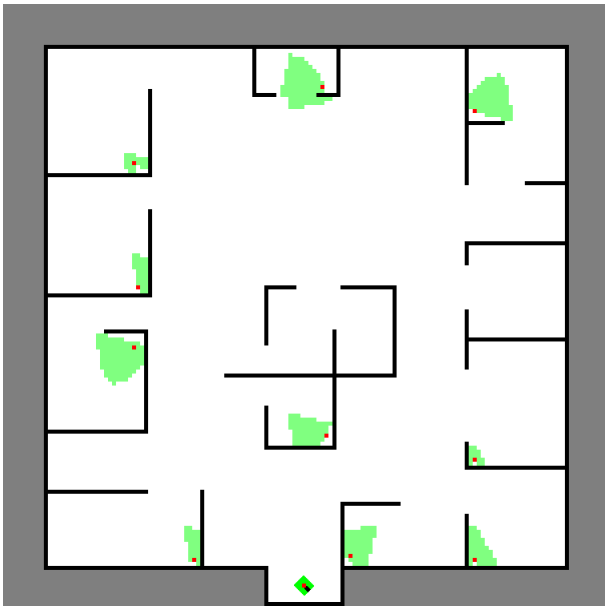


FIGURE 7.5 – Reconnaissance active

De ce fait, nous pouvons raisonnablement supposer que les temps d'exploration que nous allons mesurer seront légèrement surestimés par rapport à un environnement dans le monde réel.

7.5 Résultats

Nous avons déjà présenté dans les figures 7.5, 7.3, et 7.4 des exemples de cartes construites par chacune des stratégies après que le robot a terminé sa mission et est retourné à sa position initiale. Nous allons présenter dans les sections suivantes les résultats expérimentaux que nous avons obtenus pour chacune de ces stratégies pour les critères de comparaison que nous avons cités.

7.5.1 Temps d'exploration

En termes de temps d'exploration, la figure 7.6 montre le nombre moyen de pas de temps nécessaires pour explorer l'environnement avec les différentes stratégies. Nous pouvons constater que notre stratégie de reconnaissance active semble nécessiter un nombre de pas de temps qui croît linéairement en fonction du nombre d'objets. Les stratégies de reconnaissance passive et de reconnaissance complète semblent nécessiter un nombre de pas de temps presque constant, seulement le fait d'augmenter le nombre d'objets dans la pièce implique de devoir contourner ces objets, ce qui fait que le temps d'exploration augmente légèrement avec le nombre d'objets. La stratégie de reconnaissance complète nécessite bien plus de temps pour explorer la carte, car le robot cherche également à couvrir l'ensemble de l'espace avec sa caméra, tandis que la stratégie de reconnaissance passive nécessite le moins de temps, mais le robot ne cherche pas spécialement à reconnaître les objets.

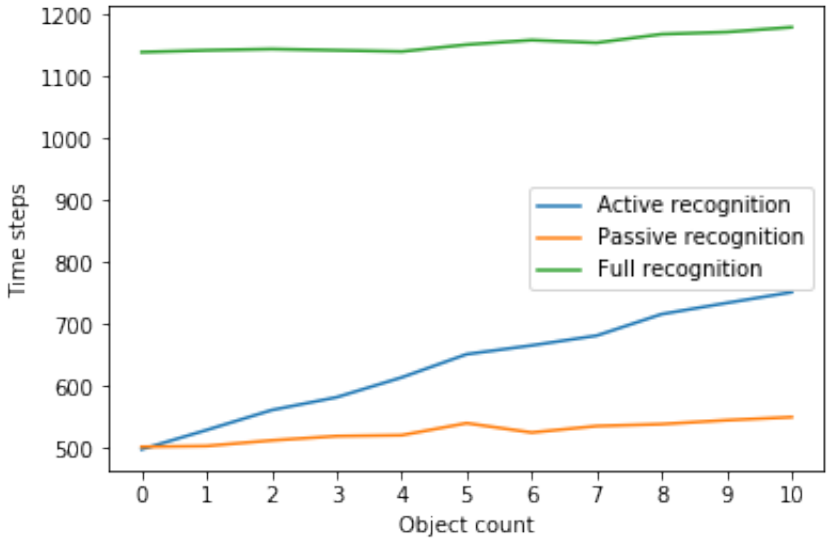


FIGURE 7.6 – Nombre de pas de temps requis pour explorer l'environnement

7.5.2 Nombre d'objets reconnus

Ces résultats se traduisent par ailleurs dans la figure 7.7 qui montre le nombre d'objets reconnus durant le processus d'exploration pour les différentes stratégies. Nous constatons clairement que la stratégie de reconnaissance passive réussit à reconnaître très peu d'objets, tandis que notre stratégie de reconnaissance active réussit à reconnaître tous les objets. La stratégie de reconnaissance complète permet également de reconnaître tous les objets ; par conséquent, la courbe pour cette stratégie est identique à la courbe de la reconnaissance active. De ce fait, elle n'est pas affichée sur le graphique.

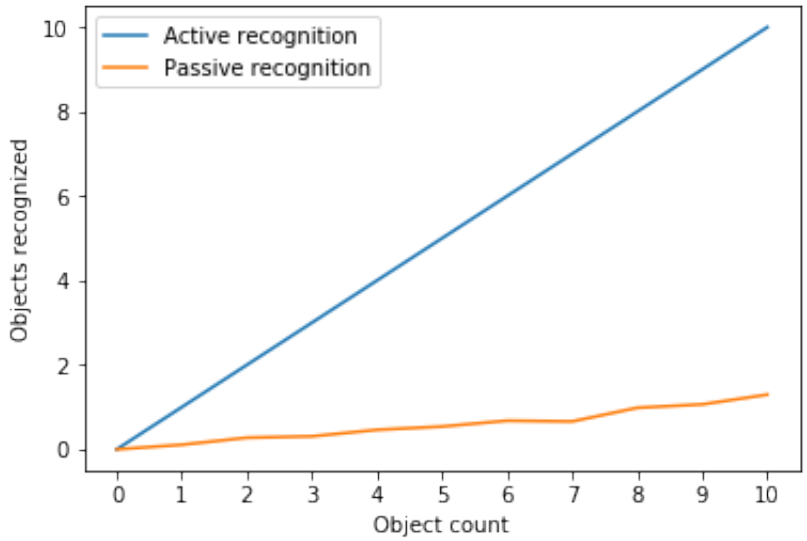


FIGURE 7.7 – Nombre d'objets reconnus durant l'exploration

7.5.3 Durée de la mission avec la reconnaissance active

Pour terminer, la figure 7.8 montre le temps (en secondes) nécessaires pour explorer l'environnement et reconnaître les objets, ainsi que les temps de calcul, pour notre stratégie de reconnaissance active. Étant donné que les politiques sont calculées sur place durant la mission, il est important de réduire les temps de calcul de manière à perdre le moins de temps d'exploration possible. Nous pouvons voir que le temps de calcul est presque constant voire semble diminuer lorsque l'on augmente le nombre d'objets. En effet, étant donné que nous restreignons le nombre d'objets considérés aux k plus proches objets, le temps de calcul ne devrait pas augmenter lorsqu'il y a plus de k objets dans l'environnement. Pour ces expérimentations, nous avons utilisé $k = 3$, ce qui signifie que le robot était capable de calculer une politique de reconnaissance pour les 3 objets les plus proches. De plus, lorsque le robot se dirige vers un objet pour le reconnaître, il exécute sa politique de reconnaissance jusqu'à l'avoir observé, ce qui explique la légère réduction du temps de calcul lorsque le nombre d'objets augmente. Comme nous l'avons évoqué précédemment, ces temps restent à prendre avec précaution dû à l'incertitude sur la durée d'un pas de temps par rapport à un environnement réel, mais il semblerait que les temps de mission aillent de 1200 secondes (20 minutes) à 1600 secondes (environ 27 minutes) avec des nombres d'objets allant de 0 à 10.

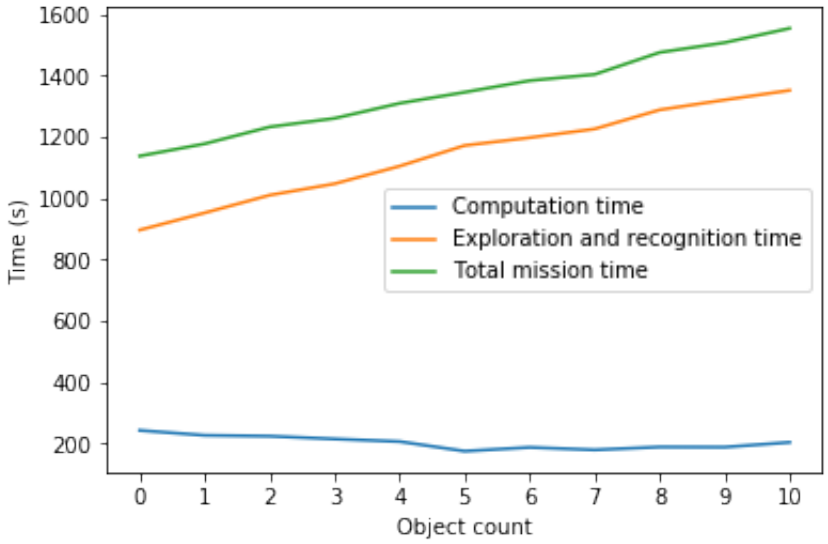


FIGURE 7.8 – Temps d'exploration pour la stratégie de reconnaissance active

7.6 Conclusion

Dans ce chapitre, nous avons évalué notre modèle de Meta-MDP pour la récolte d'information mono-agent avec des capteurs hétérogènes pour résoudre un problème concret d'exploration et de reconnaissance d'objets dans un environnement inconnu. Nous avons considéré un problème où un robot embarque deux capteurs différents (capteur laser et caméra) dans le but de cartographier une zone tout en reconnaissant de potentiels objets dans cette zone. Nous avons montré que cette stratégie permet de cartographier efficacement la zone tout en reconnaissant tous les objets qui sont présents de manière plus efficace que les approches de reconnaissance passive et de reconnaissance complète auxquelles nous nous sommes comparés. Par ailleurs, l'utilisation du

Meta-MDP en ne considérant qu'un sous-ensemble d'objets les plus proches permet de conserver un coût en temps de calcul relativement faible, ce qui est primordial lorsqu'il s'agit de planifier des stratégies sur place durant une mission. Cette stratégie permet d'avoir un temps d'exploration qui croît de manière linéaire en fonction du nombre d'objets présents dans l'environnement. Notre solution est plus générique que la solution développée lors du défi CAROTTE. En effet, la stratégie développée lors du défi repose sur un paramètre α qui ajuste l'importance entre l'exploration et la reconnaissance. Cependant, en fonction de l'environnement dans lequel l'agent évolue, et particulièrement la disposition de la carte, la valeur de α optimale peut grandement varier, ce qui est difficile à anticiper lorsque l'on ne connaît pas la structure de l'environnement à l'avance. Tandis que notre approche fonctionnera de la même manière indépendamment de l'environnement. Cependant, notre approche implique un coût supplémentaire quant à la détection des points d'intérêt et le calcul de plusieurs politiques de reconnaissance. Nous pensons que notre approche serait donc plus performante que l'approche développée lors du défi, mais des expérimentations complémentaires permettraient de s'en assurer, en particulier afin d'évaluer les différences de coûts en temps de calcul.

Conclusion

Chapitre 8

Conclusion et perspectives

8.1 Conclusion

L'objectif de cette thèse était de développer des solutions permettant de coordonner une flotte hétérogène de robots pour récolter de l'information dans un environnement inconnu. Le problème de la récolte d'information est un problème difficile à résoudre, en particulier lorsque l'observabilité est partielle, et encore plus lorsque nous avons plusieurs agents hétérogènes.

Les problèmes de prise de décision dans un environnement stochastique sont couramment résolus par le biais des processus décisionnels de Markov, ainsi que leurs extensions à l'observabilité partielle et aux systèmes multi-agents. Nous avons donc proposé un nouveau modèle basé sur les processus décisionnels de Markov afin de résoudre de tels problèmes de récolte d'information. Ce modèle, que nous avons nommé Meta-MDP, permet à une flotte hétérogène de robots de se coordonner dans le but de récolter de l'information efficacement sur leur environnement. Il s'agit d'un modèle hiérarchique fonctionnant en deux parties : nous commençons par calculer, pour chaque agent et chaque point d'intérêt, une politique permettant à cet agent de récolter de l'information sur ce point d'intérêt. Ensuite, nous calculons une méta-politique permettant d'allouer les points d'intérêts aux agents de manière optimale sur le long terme. Nous avons évalué expérimentalement ce modèle sur un problème simple et l'avons comparé à une approche optimale ainsi qu'à une approche myope et avons montré des résultats proches de l'optimal avec un meilleur passage à l'échelle théorique, bien que davantage de travail soit nécessaire afin de déterminer la complexité théorique de notre modèle.

Nous avons ensuite étendu ce modèle au cas où la carte de l'environnement n'est pas connue à l'avance, ce qui ajoute le problème de la cartographie de l'environnement à celui de la récolte d'information sur les points d'intérêt. Nous avons pour cela proposé une extension du Meta-MDP au cas où nous avons un agent embarquant plusieurs capteurs (typiquement un capteur laser pour cartographier et une caméra pour observer un point d'intérêt), et où le choix de la politique se porte cette fois sur la stratégie à adopter (cartographier ou récolter de l'information sur un point d'intérêt). Nous avons expérimenté ce modèle sur un problème concret de reconnaissance d'objet dans un environnement inconnu et avons montré de meilleurs résultats que des approches classiques de reconnaissance passive et de reconnaissance complète, avec un très bon passage à l'échelle selon le nombre de points d'intérêts. Néanmoins, ce dernier modèle ne permet pas la coordination de plusieurs agents, bien que des pistes de recherche permettant de travailler avec plusieurs agents existent.

Nous avons donc, d'une part, proposé un modèle de récolte d'information multi-agents hétérogènes dans un environnement partiellement inconnu et partiellement observable, et d'autre

part, étendu ce modèle pour faire de la récolte d'information dans un environnement totalement inconnu et partiellement observable avec un seul agent embarquant des capteurs hétérogènes, répondant ainsi à la problématique de cette thèse de deux manières complémentaires.

8.2 Perspectives

Les travaux que nous avons menés durant cette thèse peuvent faire l'objet de poursuites dans différentes directions. Nous proposons ici quelques perspectives pour de futurs travaux.

Analyse théorique Nous avons évoqué plusieurs fois la difficulté à déterminer concrètement la complexité du modèle Meta-MDP que nous avons proposé. La classe de complexité est certes la même que pour un MPOMDP classique, mais il pourrait être intéressant de creuser le sujet afin de déterminer dans quelles situations il est plus intéressant d'utiliser un modèle ou l'autre.

Amélioration du modèle Notre évaluation expérimentale du Meta-MDP a montré un défaut lié à sa fonction de récompense qui cherche à maximiser la quantité d'information récoltée par un meta-pas de temps, quitte à sacrifier le nombre de pas de temps global. Une idée de solution pourrait être de définir une fonction de récompense basée non pas sur le gain en information direct au niveau méta, mais plutôt sur le gain en information concret de la trajectoire avec un facteur d'atténuation selon la profondeur, en s'inspirant par exemple des fonctions de récompense des MDP hiérarchiques [Hauskrecht et al., 2013].

Exploration et récolte d'information multi-agents Le second modèle que nous avons proposé ne permet pas de travailler avec plusieurs agents à la fois. Pourtant, l'utilisation de plusieurs agents permettrait de gagner en efficacité, que ce soit pour l'exploration de la carte ou pour la récolte d'information sur les points d'intérêts, mais implique également d'autres complications quant à la cartographie de l'environnement, afin d'éviter les conflits entre les différents agents qui pourraient explorer les mêmes zones. Une piste de recherche qui a montré de bons résultats par le passé pour le problème de l'exploration multi-robots est l'utilisation des fonctions de valeur distribuées [Matignon et al., 2012b].

Nouveaux benchmarks Nous avons évalué notre premier modèle sur un problème relativement simple ce qui fait que les résultats que nous avons obtenus pourraient ne pas être représentatifs. Il serait donc intéressant de travailler sur d'autres problèmes plus complexes afin d'évaluer plus en détail les capacités de ce modèle. Enfin, pour le second modèle, ce sont les stratégies de comparaison que nous avons utilisées qui sont plutôt faibles, et il pourrait être intéressant de réfléchir à des stratégies plus avancées afin d'évaluer notre modèle contre d'autres stratégies, par exemple en utilisant des approches heuristiques.

Exploiter la structure du problème La grande difficulté du Meta-MDP est sa fonction de transition. Cependant, il peut être possible d'exploiter la structure des problèmes afin de simplifier le calcul de cette fonction, par exemple l'indépendance des transitions des agents, car si l'on suppose que les agents ont des modèles de transitions indépendants, alors nous pouvons travailler sur des états individuels au lieu d'états joints.

Algorithmes de résolution Nous n'avons pas proposé d'algorithme de résolution à proprement parler, et nous avons utilisé les algorithmes classiques mais qui ne sont pas toujours efficaces pour résoudre de tels problèmes. Une idée d'algorithme serait d'allouer les points d'intérêt aux agents à la manière d'un **best response** : on commence par allouer les points d'intérêt de manière aléatoire aux agents, puis on cherche la meilleure réponse pour chaque agent successivement, et ce jusqu'à convergence. L'algorithme JESP [Nair et al., 2003] fonctionne sur ce principe.

Contrôle décentralisé Le modèle de Meta-MDP que nous avons proposé repose sur des hypothèses fortes de communication et de partage d'information entre les agents, ce qui en fait un système centralisé. Dans des situations réelles, la communication entre les agents n'est pas toujours parfaite et il peut survenir des coupures de communication, ce qui rend de tels systèmes totalement inefficaces. Il pourrait donc être intéressant de travailler à décentraliser le modèle.

Actions duratives et Semi-MDP Les actions du Meta-MDP sont des politiques qui peuvent prendre plusieurs pas de temps à s'exécuter. Nous avons choisi de calculer la fonction de transition du Meta-MDP en bornant le nombre de pas de temps d'exécution, mais des travaux existent prenant en compte des actions pouvant prendre plusieurs pas de temps dans les MDP, comme les actions duratives [Mausam and Weld, 2008]. Par ailleurs, les MDP semi-markoviens (semi-MDP) permettent de tenir compte de durées variables et non déterministes des actions dans les MDP, ce qui est le cas lorsque l'on exécute une politique de récolte d'information. Les semi-MDP peuvent donc être une alternative intéressante à explorer.

Apprentissage par renforcement L'utilisation des techniques d'apprentissage par renforcement, et en particulier le Q-learning [Watkins and Dayan, 1992], peut être une solution intéressante pour estimer une politique optimale du Meta-MDP, car cela permettrait d'éviter le calcul de la Meta-Fonction de transition, qui est un processus coûteux. Ces techniques pourraient être utilisées lorsque la disposition de l'environnement est connue, ce qui est le cas dans le premier modèle que nous avons proposé. Lorsque la disposition de l'environnement n'est pas connue, il faudrait apprendre une politique sur l'ensemble des environnements possibles, ce qui prendrait trop de temps en pratique.

Bibliographie

[Amato et al., 2013] Amato, C., Chowdhary, G., Geramifard, A., Üre, N. K., and Kochenderfer, M. J. (2013). Decentralized control of partially observable markov decision processes. In *52nd IEEE Conference on Decision and Control*, pages 2398–2405. IEEE.

[Amigoni and Caglioti, 2010] Amigoni, F. and Caglioti, V. (2010). An information-based exploration strategy for environment mapping with mobile robots. *Robotics and Autonomous Systems*, 58(5) :684–699.

[Araya et al., 2010] Araya, M., Buffet, O., Thomas, V., and Charpillet, F. (2010). A pomdp extension with belief-dependent rewards. *Advances in neural information processing systems*, 23.

[Åström, 1965] Åström, K. J. (1965). Optimal control of markov processes with incomplete state information i. *Journal of mathematical analysis and applications*, 10 :174–205.

[Aulinas et al., 2008] Aulinas, J., Petillot, Y., Salvi, J., and Lladó, X. (2008). The slam problem : a survey. *Artificial Intelligence Research and Development*, pages 363–371.

[Bachrach et al., 2009] Bachrach, A., He, R., and Roy, N. (2009). Autonomous flight in unknown indoor environments. *International Journal of Micro Air Vehicles*, 1(4) :217–228.

[Bai et al., 2011] Bai, H., Hsu, D., Lee, W. S., and Ngo, V. A. (2011). Monte carlo value iteration for continuous-state pomdps. In *Algorithmic Foundations of Robotics IX : Selected Contributions of the Ninth International Workshop on the Algorithmic Foundations of Robotics*, pages 175–191. Springer.

[Bajcsy, 1988] Bajcsy, R. (1988). Active perception. *Proceedings of the IEEE*, 76(8) :966–1005.

[Bajcsy et al., 2018] Bajcsy, R., Aloimonos, Y., and Tsotsos, J. K. (2018). Revisiting active perception. *Autonomous Robots*, 42 :177–196.

[Bargiacchi et al., 2020] Bargiacchi, E., Roijers, D. M., and Nowé, A. (2020). Ai-toolbox : A c++ library for reinforcement learning and planning (with python bindings). *Journal of Machine Learning Research*, 21(102) :1–12.

[Bautin et al., 2011] Bautin, A., Simonin, O., and Charpillet, F. (2011). Towards a communication free coordination for multi-robot exploration. In *6th National conference on control architectures of robots*, pages 8–p.

[Bayes, 1958] Bayes, T. (1958). An essay towards solving a problem in the doctrine of chances. *Biometrika*, 45(3-4) :296–315.

[Bellenger, 2013] Bellenger, A. (2013). *Semantic decision support for information fusion applications*. PhD thesis, Rouen, INSA.

[Bellman, 1957] Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition.

[Bernstein et al., 2002] Bernstein, D. S., Givan, R., Immerman, N., and Zilberstein, S. (2002). The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4) :819–840.

[Bertsekas, 1995] Bertsekas, D. (1995). *Dynamic programming and optimal control*. Athena scientific.

[Beynier and Mouaddib, 2005] Beynier, A. and Mouaddib, A.-I. (2005). A polynomial algorithm for decentralized markov decision processes with temporal constraints. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 963–969.

[Boutilier et al., 1999] Boutilier, C., Dean, T., and Hanks, S. (1999). Decision-theoretic planning : Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11 :1–94.

[Boutilier et al., 2000] Boutilier, C., Dearden, R., and Goldszmidt, M. (2000). Stochastic dynamic programming with factored representations. *Artificial intelligence*, 121(1-2) :49–107.

[Boutilier et al., 1995] Boutilier, C., Dearden, R., Goldszmidt, M., et al. (1995). Exploiting structure in policy construction. In *IJCAI*, volume 14, pages 1104–1113.

[Boutilier and Poole, 1996] Boutilier, C. and Poole, D. (1996). Computing optimal policies for partially observable decision processes using compact representations. In *Proceedings of the national conference on artificial intelligence*, pages 1168–1175.

[Brooks, 1986] Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE journal on robotics and automation*, 2(1) :14–23.

[Brown et al., 2020] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33 :1877–1901.

[Burgard et al., 2003] Burgard, W., Moors, M., and Schneider, F. (2003). Collaborative exploration of unknown environments with teams of mobile robots. In *Advances in Plan-Based Control of Robotic Agents : International Seminar Dagstuhl Castle, Germany, October 21–26, 2001 Revised Papers*, pages 52–70. Springer.

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

[Capitan et al., 2013] Capitan, J., Spaan, M. T., Merino, L., and Ollero, A. (2013). Decentralized multi-robot cooperation with auctioned pomdps. *The International Journal of Robotics Research*, 32(6) :650–671.

[Cassandra et al., 1994] Cassandra, A. R., Kaelbling, L. P., and Littman, M. L. (1994). Acting optimally in partially observable stochastic domains. In *Aaai*, volume 94, pages 1023–1028.

[Cassandra et al., 1997] Cassandra, A. R., Littman, M. L., and Zhang, N. L. (1997). Incremental pruning : A simple, fast, exact method for partially observable markov decision processes. *Proceedings of the 13th conference on Uncertainty in artificial intelligence*.

[Charrow et al., 2015] Charrow, B., Kahn, G., Patil, S., Liu, S., Goldberg, K., Abbeel, P., Michael, N., and Kumar, V. (2015). Information-theoretic planning with trajectory optimization for dense 3d mapping. In *Robotics : Science and Systems*, volume 11, pages 3–12.

[Chung et al., 2004] Chung, T. H., Gupta, V., Burdick, J. W., and Murray, R. M. (2004). On a decentralized active sensing strategy using mobile sensor platforms in a network. In *2004 43rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601)*, volume 2, pages 1914–1919. IEEE.

[Dean and Kanazawa, 1989] Dean, T. and Kanazawa, K. (1989). A model for reasoning about persistence and causation. *Computational intelligence*, 5(2) :142–150.

[Dibangoye et al., 2016] Dibangoye, J. S., Amato, C., Buffet, O., and Charpillet, F. (2016). Optimally solving dec-pomdps as continuous-state mdps. *Journal of Artificial Intelligence Research*, 55 :443–497.

[Doshi et al., 2009] Doshi, P., Zeng, Y., and Chen, Q. (2009). Graphical models for interactive pomdps : representations and solutions. *Autonomous Agents and Multi-Agent Systems*, 18(3) :376–416.

[Dubois, 2018] Dubois, D. (2018). Uncertainty theories : a unified view. In *8th International Workshop on Reliable Engineering Computing (REC 2018)*.

[Durrant-Whyte and Bailey, 2006] Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping : part i. *IEEE robotics & automation magazine*, 13(2) :99–110.

[Eidenberger and Scharinger, 2010] Eidenberger, R. and Scharinger, J. (2010). Active perception and scene modeling by planning with probabilistic 6d object poses. In *2010 IEEE/RSJ international conference on intelligent robots and systems*, pages 1036–1043. IEEE.

[Fehr et al., 2018] Fehr, M., Buffet, O., Thomas, V., and Dibangoye, J. (2018). rho-pomdps have lipschitz-continuous epsilon-optimal value functions. *Advances in neural information processing systems*, 31.

[Fikes and Nilsson, 1971] Fikes, R. E. and Nilsson, N. J. (1971). Strips : A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4) :189–208.

[Floreano and Mondada, 1994] Floreano, D. and Mondada, F. (1994). Active perception, navigation, homing, and grasping : an autonomous perspective. In *Proceedings of PerAc'94. From Perception to Action*, pages 122–133. IEEE.

[Gandois et al., 2024] Gandois, A., Mouaddib, A.-I., Le Gloannec, S., and Alfalou, A. (2024). A meta-mdp approach for information gathering heterogeneous multi-agent systems. In *International Conference on Robotics, Computer Vision and Intelligent Systems*, pages 345–360. Springer.

[Gandois et al., 2025] Gandois, A., Mouaddib, A.-I., Le Gloannec, S., and Alfalou, A. (2025). Exploration and active recognition strategy using meta-mdp. In *Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing*, page To appear.

[Ghallab et al., 2004] Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated Planning : theory and practice*. Elsevier.

[Gmytrasiewicz and Doshi, 2005] Gmytrasiewicz, P. J. and Doshi, P. (2005). A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research*, 24 :49–79.

[Goldman and Zilberstein, 2003] Goldman, C. V. and Zilberstein, S. (2003). Optimizing information exchange in cooperative multi-agent systems. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 137–144.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

[Guo, 2003] Guo, A. (2003). Decision-theoretic active sensing for autonomous agents. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 1002–1003.

[Hansen et al., 2004] Hansen, E. A., Bernstein, D. S., and Zilberstein, S. (2004). Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, pages 709–715.

[Hausknecht and Stone, 2015] Hausknecht, M. and Stone, P. (2015). Deep recurrent q-learning for partially observable mdps. In *2015 aaai fall symposium series*.

[Hauskrecht, 2000] Hauskrecht, M. (2000). Value-function approximations for partially observable markov decision processes. *Journal of artificial intelligence research*, 13 :33–94.

[Hauskrecht et al., 2013] Hauskrecht, M., Meuleau, N., Kaelbling, L. P., Dean, T. L., and Boutilier, C. (2013). Hierarchical solution of markov decision processes using macro-actions. *arXiv preprint arXiv :1301.7381*.

[Howard, 1960] Howard, R. A. (1960). Dynamic programming and markov processes.

[Igl et al., 2018] Igl, M., Zintgraf, L., Le, T. A., Wood, F., and Whiteson, S. (2018). Deep variational reinforcement learning for pomdps. In *International conference on machine learning*, pages 2117–2126. PMLR.

[Ji and Carin, 2007] Ji, S. and Carin, L. (2007). Cost-sensitive feature acquisition and classification. *Pattern Recognition*, 40(5) :1474–1485.

[Keidar and Kaminka, 2014] Keidar, M. and Kaminka, G. A. (2014). Efficient frontier detection for robot exploration. *The International Journal of Robotics Research*, 33(2) :215–236.

[Koenig et al., 2001] Koenig, S., Tovey, C., and Halliburton, W. (2001). Greedy mapping of terrain. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 4, pages 3594–3599. IEEE.

[Kuhn, 1955] Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2) :83–97.

[Kurniawati et al., 2009] Kurniawati, H., Hsu, D., and Lee, W. S. (2009). Sarsop : Efficient point-based pomdp planning by approximating optimally reachable belief spaces.

[Lauri and Ritala, 2016] Lauri, M. and Ritala, R. (2016). Planning for robotic exploration based on forward simulation. *Robotics and Autonomous Systems*, 83 :15–31.

[Le Gloannec et al., 2010] Le Gloannec, S., Jeanpierre, L., and Mouaddib, A.-I. (2010). Unknown area exploration with an autonomous robot using markov decision processes. In *TAROS*.

[Littman, 1994] Littman, M. L. (1994). Memoryless policies : Theoretical limitations and practical results.

[Littman, 1996] Littman, M. L. (1996). *Algorithms for sequential decision-making*. Brown University.

[Liu et al., 2014] Liu, W., Gu, W., Sheng, W., Meng, X., Wu, Z., and Chen, W. (2014). Decentralized multi-agent system-based cooperative frequency control for autonomous microgrids with communication constraints. *IEEE Transactions on Sustainable Energy*, 5(2) :446–456.

[Lluvia et al., 2021] Lluvia, I., Lazkano, E., and Ansuategi, A. (2021). Active mapping and robot exploration : A survey. *Sensors*, 21(7) :2445.

[Lovejoy, 1991] Lovejoy, W. S. (1991). Computationally feasible bounds for partially observed markov decision processes. *Operations research*, 39(1) :162–175.

[Madani et al., 1999] Madani, O., Hanks, S., and Condon, A. (1999). On the undecidability of probabilistic planning and infinite-horizon partially observable markov decision problems. *Aaai/iaai*, 10(315149.315395).

[Markov, 1906] Markov, A. A. (1906). Rasprostranenie zakona bol'shikh chisel na velichiny, zavisyashchie drug ot druga. *Proceedings of the Imperial Academy of Sciences*, 15 :135–156.

[Matignon et al., 2012a] Matignon, L., Jeanpierre, L., and Mouaddib, A.-I. (2012a). Coordinated multi-robot exploration under communication constraints using decentralized markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pages 2017–2023.

[Matignon et al., 2012b] Matignon, L., Jeanpierre, L., and Mouaddib, A.-I. (2012b). Distributed value functions for multi-robot exploration. In *2012 IEEE International Conference on Robotics and Automation*, pages 1544–1550. IEEE.

[Matignon et al., 2015] Matignon, L., Jeanpierre, L., and Mouaddib, A.-I. (2015). Decentralized multi-robot planning to explore and perceive.

[Mausam and Weld, 2008] Mausam, M. and Weld, D. (2008). Planning with durative actions in stochastic domains. *J. Artif. Intell. Res. (JAIR)*, 31 :33–82.

[McCarthy et al., 2006] McCarthy, J., Minsky, M. L., Rochester, N., and Shannon, C. E. (2006). A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI magazine*, 27(4) :12–12.

[McDermott et al., 1998] McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., and Wilkins, D. (1998). Pddl - the planning domain definition language.

[Messias et al., 2011] Messias, J., Spaan, M., and Lima, P. (2011). Efficient offline communication policies for factored multiagent pomdps. *Advances in Neural Information Processing Systems*, 24.

[Metropolis and Ulam, 1949] Metropolis, N. and Ulam, S. (1949). The monte carlo method. *Journal of the American statistical association*, 44(247) :335–341.

[Mihaylova et al., 2003] Mihaylova, L., Lefebvre, T., Bruyninckx, H., Gadeyne, K., and De Schutter, J. (2003). A comparison of decision making criteria and optimization methods for active robotic sensing. In *Numerical Methods and Applications : 5th International Conference, NMA 2002 Borovets, Bulgaria, August 20–24, 2002 Revised Papers 5*, pages 316–324. Springer.

[Moravec and Elfes, 1985] Moravec, H. and Elfes, A. (1985). High resolution maps from wide angle sonar. In *Proceedings. 1985 IEEE international conference on robotics and automation*, volume 2, pages 116–121. IEEE.

[Moulin and Chaib-Draa, 1996] Moulin, B. and Chaib-Draa, B. (1996). An overview of distributed artificial intelligence. *Foundations of distributed artificial intelligence*, pages 3–55.

[Nair et al., 2003] Nair, R., Tambe, M., Yokoo, M., Pynadath, D., and Marsella, S. (2003). Taming decentralized pomdps : Towards efficient policy computation for multiagent settings. In *IJCAI*, volume 3, pages 705–711.

[Nair et al., 2005] Nair, R., Varakantham, P., Tambe, M., and Yokoo, M. (2005). Networked distributed pomdps : A synthesis of distributed constraint optimization and pomdps. In *AAAI*, volume 5, pages 133–139.

[Newell and Simon, 1956] Newell, A. and Simon, H. (1956). The logic theory machine–a complex information processing system. *IRE Transactions on information theory*, 2(3) :61–79.

[Newell and Simon, 1961] Newell, A. and Simon, H. A. (1961). Gps, a program that simulates human thought.

[Nicholson and Snyder, 1997] Nicholson, W. and Snyder, C. (1997). *Microeconomic theory*. Harcourt Brace College Publishers.

[Nilsson, 2009] Nilsson, N. J. (2009). *The quest for artificial intelligence*. Cambridge University Press.

[Oliehoek, 2012] Oliehoek, F. A. (2012). Decentralized pomdps. In *Reinforcement learning : state-of-the-art*, pages 471–503. Springer.

[Oliehoek et al., 2017] Oliehoek, F. A., Spaan, M. T., Terwijn, B., Robbel, P., et al. (2017). The madp toolbox : An open source library for planning and learning in (multi-) agent systems. *Journal of Machine Learning Research*, 18(89) :1–5.

[Oliehoek et al., 2008] Oliehoek, F. A., Spaan, M. T., Vlassis, N., and Whiteson, S. (2008). Exploiting locality of interaction in factored dec-pomdps. In *Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems*.

[Ong et al., 2010] Ong, S. C., Png, S. W., Hsu, D., and Lee, W. S. (2010). Planning under uncertainty for robotic tasks with mixed observability. *The International Journal of Robotics Research*, 29(8) :1053–1068.

[Papadimitriou and Tsitsiklis, 1987] Papadimitriou, C. H. and Tsitsiklis, J. N. (1987). The complexity of markov decision processes. *Mathematics of operations research*, 12(3) :441–450.

[Peot and Smith, 1992] Peot, M. A. and Smith, D. E. (1992). Conditional nonlinear planning. In *Artificial Intelligence Planning Systems*, pages 189–197. Elsevier.

[Pineau et al., 2003] Pineau, J., Gordon, G., Thrun, S., et al. (2003). Point-based value iteration : An anytime algorithm for pomdps. In *Ijcai*, volume 3, pages 1025–1032.

[Placed et al., 2023] Placed, J. A., Strader, J., Carrillo, H., Atanasov, N., Indelman, V., Carlone, L., and Castellanos, J. A. (2023). A survey on active simultaneous localization and mapping : State of the art and new frontiers. *IEEE Transactions on Robotics*, 39(3) :1686–1705.

[Puterman, 1994] Puterman, M. L. (1994). *Markov Decision Processes : Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley.

[Pynadath and Tambe, 2002] Pynadath, D. V. and Tambe, M. (2002). The communicative multiagent team decision problem : Analyzing teamwork theories and models. *Journal of artificial intelligence research*, 16 :389–423.

[Renoux, 2015] Renoux, J. (2015). *Contribution to multiagent p2lanning for active information gathering*. PhD thesis, Normandie Université.

[Renoux et al., 2015] Renoux, J., Mouaddib, A.-I., and Le Gloannec, S. (2015). A decision-theoretic planning approach for multi-robot exploration and event search. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5287–5293. IEEE.

[Russell and Norvig, 2020] Russell, S. J. and Norvig, P. (2020). *Artificial intelligence : a modern approach*. Pearson.

[Satsangi et al., 2015] Satsangi, Y., Whiteson, S., and Oliehoek, F. (2015). Exploiting submodular value functions for faster dynamic sensor selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.

[Schneider et al., 1999] Schneider, J., Wong, W.-K., Moore, A., and Riedmiller, M. (1999). Distributed value functions.

[Seuken and Zilberstein, 2008] Seuken, S. and Zilberstein, S. (2008). Formal models and algorithms for decentralized decision making under uncertainty. *Autonomous Agents and Multi-Agent Systems*, 17 :190–250.

[Shannon, 1948] Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, 27(3) :379–423.

[Silver and Veness, 2010] Silver, D. and Veness, J. (2010). Monte-carlo planning in large pomdps. *Advances in neural information processing systems*, 23.

[Simmons et al., 2000] Simmons, R. G., Apfelbaum, D., Burgard, W., Fox, D., Moors, M., Thrun, S., and Younes, H. L. S. (2000). Coordination for multi-robot exploration and mapping. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI)*, pages 852–858. AAAI Press / The MIT Press.

[Smallwood and Sondik, 1973] Smallwood, R. D. and Sondik, E. J. (1973). The optimal control of partially observable markov processes over a finite horizon. *Operations research*, 21(5) :1071–1088.

[Smith and Simmons, 2012] Smith, T. and Simmons, R. (2012). Heuristic search value iteration for pomdps. *arXiv preprint arXiv :1207.4166*.

[Sondik, 1978] Sondik, E. J. (1978). The optimal control of partially observable markov processes over the infinite horizon : Discounted costs. *Operations research*, 26(2) :282–304.

[Spaan, 2008] Spaan, M. T. (2008). Cooperative active perception using pomdps. In *AAAI 2008 workshop on advancements in POMDP solvers*.

[Spaan et al., 2010a] Spaan, M. T., Gonçalves, N., and Sequeira, J. (2010a). Multirobot coordination by auctioning pomdps. In *2010 IEEE International Conference on Robotics and Automation*, pages 1446–1451. IEEE.

[Spaan et al., 2010b] Spaan, M. T., Veiga, T. S., and Lima, P. U. (2010b). Active cooperative perception in network robot systems using pomdps. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4800–4805. IEEE.

[Spaan and Vlassis, 2005] Spaan, M. T. and Vlassis, N. (2005). Perseus : Randomized point-based value iteration for pomdps. *Journal of artificial intelligence research*, 24 :195–220.

[Sussman, 1973] Sussman, G. J. (1973). A computational model of skill acquisition.

[Sutton and Barto, 2018] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning : An introduction*. MIT press.

[Szer et al., 2012] Szer, D., Charpillet, F., and Zilberstein, S. (2012). Maa* : A heuristic search algorithm for solving decentralized pomdps. *arXiv preprint arXiv :1207.1359*.

[Thomas et al., 2021] Thomas, V., Hutin, G., and Buffet, O. (2021). Monte carlo information-oriented planning. *arXiv preprint arXiv :2103.11345*.

[Thrun et al., 2002] Thrun, S. et al. (2002). Robotic mapping : A survey.

[Turing, 1950] Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, LIX(236) :433–460.

[Varakantham et al., 2007] Varakantham, P., Marecki, J., Yabu, Y., Tambe, M., and Yokoo, M. (2007). Letting loose a spider on a network of pomdps : Generating quality guaranteed policies. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–8.

[Wang et al., 2007] Wang, Z., Huang, S., and Dissanayake, G. (2007). Multi-robot simultaneous localization and mapping using d-slam framework. In *2007 3rd International Conference on Intelligent Sensors, Sensor Networks and Information*, pages 317–322. IEEE.

[Watkins and Dayan, 1992] Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8 :279–292.

[Weizenbaum, 1966] Weizenbaum, J. (1966). Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1) :36–45.

[Weld, 1994] Weld, D. S. (1994). An introduction to least commitment planning. *AI magazine*, 15(4) :27–27.

[Weyns et al., 2004] Weyns, D., Steegmans, E., and Holvoet, T. (2004). Towards active perception in situated multi-agent systems. *Applied Artificial Intelligence*, 18(9-10) :867–883.

[Wooldridge, 2009] Wooldridge, M. (2009). *An introduction to multiagent systems*. John wiley & sons.

[Wurm et al., 2008] Wurm, K. M., Stachniss, C., and Burgard, W. (2008). Coordinated multi-robot exploration using a segmentation of the environment. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1160–1165. IEEE.

[Yamauchi, 1997] Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*, pages 146–151. IEEE.

[Yamauchi, 1998] Yamauchi, B. (1998). Frontier-based exploration using multiple robots. In *Proceedings of the second international conference on Autonomous agents*, pages 47–53.

[Zheng et al., 2011] Zheng, Y., Zhu, Y., and Wang, L. (2011). Consensus of heterogeneous multi-agent systems. *IET Control Theory & Applications*, 5(16) :1881–1888.

[Zhou and Roumeliotis, 2006] Zhou, X. S. and Roumeliotis, S. I. (2006). Multi-robot slam with unknown initial correspondence : The robot rendezvous case. In *2006 IEEE/RSJ international conference on intelligent robots and systems*, pages 1785–1792. IEEE.

[Zlot et al., 2002] Zlot, R., Stentz, A., Dias, M. B., and Thayer, S. (2002). Multi-robot exploration controlled by a market economy. In *Proceedings 2002 IEEE international conference on robotics and automation (Cat. No. 02CH37292)*, volume 3, pages 3016–3023. IEEE.

[Zou et al., 2023] Zou, Z., Chen, K., Shi, Z., Guo, Y., and Ye, J. (2023). Object detection in 20 years : A survey. *Proceedings of the IEEE*, 111(3) :257–276.

Coordination d'une flotte hétérogène de robots pour la récolte d'information dans un environnement inconnu

Résumé. Dans cette thèse, nous étudions le problème de la récolte d'information dans un environnement inconnu et partiellement observable avec des agents hétérogènes. Nous considérons un environnement composé de différents points d'intérêt, avec pour objectif de coordonner des agents hétérogènes dans le but de récolter de l'information sur ces points d'intérêt. L'hétérogénéité des agents peut apparaître sous différentes formes : plusieurs agents ayant des capacités d'observation différentes, des capacités d'embarquement différentes, des ressources différentes, ou bien un seul agent embarquant plusieurs capteurs hétérogènes. Dans un premier temps, nous avons proposé un modèle de récolte d'information avec plusieurs agents hétérogènes dans un environnement partiellement observable mais topologiquement connu. Ce modèle, que nous avons nommé Meta-MDP, est basé sur les processus décisionnels de Markov, et fonctionne en deux parties : premièrement, pour chaque agent et chaque point d'intérêt, nous calculons une politique pour récolter de l'information sur ce point. Ensuite, nous calculons une politique d'allocation des points d'intérêts aux agents de manière à optimiser la récolte d'information sur le long terme. Nous avons ensuite étendu ce modèle au cas où nous avons un agent embarquant plusieurs capteurs hétérogènes (typiquement un capteur laser et une caméra) dans un environnement inconnu dans le but de construire une carte de l'environnement tout en récoltant de l'information sur les éventuels points d'intérêt.

Mots-clés : Récolte d'information, Systèmes multi-agents, Robots hétérogènes, Processus Décisionnels de Markov

Coordination of a heterogeneous fleet of robots for information gathering in an unknown environment

Abstract. In this thesis, we study the problem of information gathering in an unknown and partially observable environment with heterogeneous agents. We consider an environment containing a set of interest points, with the objective of coordinating heterogeneous agents in order to gather information on these points. The heterogeneity of the agents can manifest in various ways : multiple agents with different observation capacities, different transport capabilities, varying resources, or a single agent equipped with multiple heterogeneous sensors. We started by proposing a model to gather information with multiple heterogeneous agents in a partially observable yet topologically known environment. This model, which we have named Meta-MDP, is based on Markov decision processes and operates in two stages : first, for each agent and each interest point, we calculate a policy to gather information on that particular point. Then, we compute a policy for allocating interest points to agents in a way that optimizes long-term information gathering. Then, we extended this model to the case where a single agent, equipped with multiple heterogeneous sensors (typically a laser sensor and a camera), operates in an unknown environment with the goal of building a map of the environment while simultaneously gathering information on potential interest points.

Keywords : Information Gathering, Multi-agent systems, Heterogeneous robots, Markov Decision Process