Planification de perception et de mission en environnement incertain : Application à la détection et à la reconnaissance de cibles par un hélicoptère autonome

Caroline Ponzoni Carvalho Chanel

Université de Toulouse

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut Supérieur de l'Aéronautique et de l'Espace (ISAE)

Présentée et soutenue par :
Caroline PONZONI CARVALHO CHANEL
le vendredi 12 avril 2013

Titre :

Planification de perception et de mission en environnement incertain : Application à la détection et à la reconnaissance de cibles par un hélicoptère autonome

---

École doctorale et discipline ou spécialité :
EDSYS : Systèmes embarqués et automatique

Unité de recherche :
Équipe d'accueil ISAE-ONERA CSDV

Directeur(s) de Thèse :

M. Patrick FABIANI (directeur de thèse)
M. Florent TEICHTEIL-KONIGSBUCH (co-directeur de thèse)

Jury :

M. Malik GHALLAB
M. Olivier BUFFET
M. François CHARPILLET, rapporteur
M. Abdel-Illah MOUADDIB, rapporteur
M. Patrick FABIANI, directeur de thèse
M. Florent TEICHTEIL-KONIGSBUCH, co-directeur de thèse

Résumé

Les applications de l'Intelligence Artificielle à des agents en interaction physique avec le monde réel sont confrontées au besoin de planifier des actions avec information incomplète par rapport à l'état du monde. Les applications visées dans nos travaux sont des missions de robotique aérienne en environnement incertain. Dans ces problèmes, la perception est un moyen d'acquérir de l'information sur l'environnement afin de mieux agir pour atteindre le but de la mission, mais elle nécessite aussi des décisions et actions spécifiques de perception dans le but de recueillir l'information. Il s'agit donc de décider pour percevoir et de percevoir pour décider. Dans ce contexte, cette thèse s'attache à proposer un cadre de modélisation et de résolution de problèmes de planification de perception et de mission pour un drone hélicoptère qui évolue dans un environnement incertain et partiellement observé. Nous tenons compte de la représentation des actions et de leurs conséquences incertaines, des incertitudes d'observation, des contraintes impératives de sécurité sur les actions du drone et des contraintes d'exécution en temps réels, même pendant le calcul de la stratégie d'action. Dans un premier temps, nous proposons un cadre formel de modélisation de problèmes de décision séquentielle dans l'incertain unifiant la représentation des fonctions de perception de l'environnement et celle des actions pour l'accomplissement de la mission. Nous avons fondé notre travail sur le cadre des Processus Décisionnels Markoviens Partiellement Observables (POMDP), car il propose un schéma général applicable à cette problématique, et optimal pour les tâches de perception et de décision à long terme. Nous confrontons ce cadre aux données réelles obtenues en vol sur un scénario de détection et de reconnaissance de cibles : une attention spéciale est portée à la modélisation des sorties symboliques et incertaines de l'algorithme de traitement d'image en tant que fonction d'observation probabiliste du modèle POMDP obtenues par apprentissage (statistique hors ligne), et à la modélisation de la fonction de récompense qui représente les conséquences numériques et symboliques des actions du robot aérien. Nous proposons une analyse critique de la mise en œuvre en pratique du modèle POMDP et du critère d'optimisation associé, à la lumière d'une étude bibliographique sur les approches de décision pour la perception d'une part, et de perception pour la décision d'autre part. Une étude sur le compromis entre la prise d'information et la décision nous permet de proposer et analyser un critère d'optimisation mixte, qui peut favoriser la prise d'information afin de mieux valider les possibles décisions qui doivent être prises en ligne. Afin de respecter les contraintes de sécurité et les limitations physiques de nos robots aériens, nous proposons ensuite une approche pour tenir compte explicitement des préconditions sur les actions dans chaque état de croyance : le modèle AC-POMDP, qui sépare l'information concernant la vérification des propriétés du modèle (les préconditions), de celle qui renseigne sur le gain des récompenses ou sur l'observation des objets de l'environnement. Les équations d'optimisation proposées pour ce modèle permettent de ne sélectionner que des actions faisables avec certitude, lors de la mise à jour de la valeur de l'état de croyance. La structure du modèle AC-POMDP permet de plus d'optimiser la fonction de valeur paramétrée par des alpha-vecteurs définis sur des sous-espaces de croyance différents et réduits, contrairement au modèle POMDP classique qui optimise la fonction de valeur paramétrée par des alpha-vecteurs définis sur l'espace entier de croyance. Enfin, nous présentons une étude sur l'optimisation et l'exécution en parallèle de politiques POMDP en ligne et en temps contraint, permettant d'embarquer les algorithmes résultants sur nos robots aériens. Ceci comprend une implémentation des algorithmes de résolution en ligne de POMDP, ainsi qu'un cadre d'optimisation et d'exécution en parallèle de politiques POMDP basé sur une anticipation partielle probabilisée des états futurs du système. Nous avons embarqué ce cadre algorithmique sur les hélicoptères autonomes de l'Onera, et l'avons testé en vol et en environnement réel sur une mission de détection et reconnaissance de cibles.

Mots-clés : Planification pour la perception, POMDP, planification en ligne et en parallèle de l'exécution, robotique mobile, apprentissage statistique hors-ligne de modèle d'observation.

Table des matières

Table des figures

Liste des tableaux

Liste des Algorithmes

Introduction

La détection et la reconnaissance de cibles par des véhicules aériens inhabités (UAV) autonomes est un domaine de recherche actif [Wang *et al.*, 2012], en raison de la mise en place croissante de systèmes de drones dans des missions civiles et militaires. Dans ce genre de mission, la stratégie de décision de haut niveau est habituellement donnée dans les systèmes réels par une règle écrite à la main (survoler une zone donnée, atterrir, prendre une image, etc), qui dépend des événements stochastiques (détection d'une cible dans une zone donnée, cible reconnue) qui peuvent survenir lors de l'exécution de la règle de décision. A cause de la complexité de la construction automatique des règles de décision sous incertitude [Littman *et al.*, 1995,Sabbadin *et al.*, 2007], appelées politique, et de la méconnaissance des probabilités associées aux événements, peu de systèmes de drones construisent et optimisent des politiques automatiquement.

Quand les incertitudes dans l'environnement viennent de l'exécution imparfaite des actions, ou des observations de l'environnement, les politiques de haut niveau peuvent être automatiquement générées par l'optimisation de Processus Markovien Partiellement Observable (POMDP) [Smallwood et Sondik, 1973]. Ce modèle a été appliqué avec succès à la robotique mobile [Smith et Simmons, 2004, Candido et Hutchinson, 2011, Spaan, 2008], et aussi à la robotique aérienne [Miller *et al.*, 2009, Schesvold *et al.*, 2003, Bai *et al.*, 2011].

Des travaux précédents ont étudié des scénarios qui traitent à la fois du problème de perception et de décision dans une mission de robotique mobile. Le modèle formel choisi est souvent celui des POMDP. Il a été montré que ce modèle permet d'aborder à la fois les deux problématiques : par exemple, dans le scénario robotique de [Smith et Simmons, 2004] la perception est considérée comme un moyen pour arriver à l'accomplissement de la mission. Dans ce scénario, ramasser des pierres observées en tant que "bonnes" rapporte des gains à l'agent, mais le but final est d'arriver à un emplacement précis du terrain. Par contre, dans [Spaan et Lima, 2009], la perception est le but de la mission, puisque l'agent cherche à détecter et suivre des cibles. D'autre part, les scénarios étudiés sont connus à priori, en ce qui concerne par exemple leur taille, ce qui permet une résolution hors ligne. De plus, ces scénarios, même en étant considérés réalistes, ne prennent pas en compte des contraintes réelles, physiques ou de sûreté liées à la mission robotique, comme par exemple le fait que certaines zones du terrain ne sont pas exploitables même si une pierre dite "bonne" se trouve dans cette zone.

Dans ce contexte, cette thèse consiste à proposer un cadre de modélisation et de résolution de problèmes de planification de perception et de mission pour un drone hélicoptère qui évolue dans un environnement incertain et partiellement observable. Dans cette thèse, la perception est considérée non seulement comme une fin mais aussi comme un moyen pour aboutir au but de la mission : il s'agit ici de décider pour percevoir et de percevoir pour décider. De plus, des contraintes liées à la physique ou à la sûreté du système seront aussi considérées. Ces contraintes permettent de garantir que le plan d'actions résultant du calcul ne puisse pas engendrer des actions considérées comme dangereuses, ce qui est très important dans des

applications aéronautiques, par exemple pour obtenir les autorisations de vol. Un autre volet de cette thèse prend en compte le fait que le problème à résoudre ne peut être connu qu'en ligne. En effet, le nombre de cibles ou le nombre de zones qui composent l'environnement, sont des exemples de paramètres généralement inconnus avant le vol et qui doivent être automatiquement extraits en vol au début de la mission (par exemple, par des techniques de traitement d'image) afin de définir le problème de décision séquentielle à optimiser. Cette problématique suppose un parallélisme entre la résolution, soit la construction du plan, et l'exécution de ce même plan. Ainsi, à notre connaissance, aucune application dronique basée sur les POMDP n'a tenu compte de toutes ces caractéristiques à la fois.

Notre démarche nous a amenés dans une première partie à étudier les différentes approches de la littérature qui traitent de la planification pour la perception. Cette étude générale est présentée dans les deux premiers chapitres. L'état de l'art étudié cherche à poser les bases sur lesquelles nous nous sommes appuyés pour développer les différentes approches proposées dans cette thèse. Le chapitre 1 a pour objectif de présenter la problématique de la planification pour la perception, ainsi que les différents formalismes de décision séquentielle sous observabilité partielle, étant donné notre application à la détection et à la reconnaissance de cibles en environnement partiellement observable. Le but de ce chapitre est de justifier le choix du formalisme POMDP comme cadre mathématique de notre étude. Le chapitre 2 présente une revue détaillée des POMDP, ainsi que différentes approches de résolution des POMDP hors ligne et en ligne.

Nos contributions constituent la deuxième partie de cette thèse que nous avons intitulée "décider pour percevoir et percevoir pour décider". Dans cette seconde partie, nous présentons les études et approches proposées dans ce travail de thèse pour chacune des différentes problématiques : la modélisation du problème de détection et de reconnaissance de cibles, la planification pour la perception nécessaire à cette fin, la prise en compte de contraintes sur les actions lors de la planification, et la résolution en ligne du problème. Cette deuxième partie est constituée de 4 chapitres qui sont brièvement décrits ci-dessous.

Le chapitre 3 permet de modéliser le problème dual de perception-décision. Pour cela nous avons d'abord défini un scénario de détection et reconnaissance de cibles dans un environnement incertain. Son but est de détecter des cibles et de chercher à reconnaître leurs modèles parmi une base de données. Ce scénario comporte deux types de mission : une mission d'exploration, où l'on cherche à identifier l'état caché de l'environnement dans lequel l'agent autonome évolue ; et une mission d'atterrissage, où l'on cherche à identifier une cible en particulier parmi un certain nombre de cibles possibles, et ce afin d'atterrir à proximité de la cible recherchée. Ce type d'application nous a menés à l'étude du traitement d'information du capteur caméra choisi, afin de déterminer le type de réponse obtenu avec un tel capteur. Les informations disponibles nous amènent à un cadre probabiliste et d'observabilité partielle de l'environnement. Comme pour les études présentées dans l'état de l'art, nous avons choisi de modéliser ce scénario en tant que POMDP, car il propose un schéma général, qui permet de prendre en compte deux types d'incertitude : l'une liée aux effets des actions, et l'autre liée à la sortie du traitement d'image ; et optimal pour les tâches de perception et de décision à long terme. Étant donné que les actions du drone autonome dans cette application peuvent être considérées comme déterministes, le comportement du drone hélicoptère peut être considéré comme certain. Par contre, la sortie du traitement d'image, qui est basé sur l'algorithme publié dans [Saux et Sanfourche, 2011], ne peut pas être considérée comme telle, ce qui nous a amenés à apprendre un modèle statistique de la sortie du traitement d'images à partir de donnés réelles de l'environnement. Ces données ont été collectées lors de trois campagnes de prise d'image de voitures (nos cibles) dans la zone expérimentale où évolue le drone hélicoptère. L'apprentissage hors ligne du modèle d'observation seul a été préféré à la mise en oeuvre de techniques d'apprentissage par renforcement dans la mesure

où, d'une part l'apprentissage par renforcement des effets des actions et de la fonction de récompense n'étaient pas nécessaires, et d'autre part nous ne pouvions pas prendre le risque d'endommager un moyen expérimental coûteux en lors d'une phase d'apprentissage de la stratégie.

Dans le chapitre 4, nous proposons une analyse critique de la mise en œuvre pratique du modèle POMDP et du critère d'optimisation associé, en nous appuyant sur l'étude bibliographique des différentes approches de décision pour la perception et de perception pour la décision. Pour cela, nous avons étudié le compromis entre la prise d'information et la décision dans les deux cadres d'application mis en évidence dans le chapitre 3. Le premier cadre d'application se rapporte à la mission d'exploration qui, selon [Araya-López *et al.*, 2010, Candido et Hutchinson, 2011, Eidenberger et Scharinger, 2010], appartient à une classe de problèmes de nature différente, où l'agent doit interagir avec son environnement et identifier l'état caché du système comme dans les problèmes de perception active. Nous nous sommes intéressée à ce cadre d'application de la décision séquentielle car il suppose que la structure de la fonction de récompense doit être différente, c'est-à-dire qu'elle doit être aussi basée sur une mesure de l'incertitude de l'état de croyance. Ainsi, cette fonction de récompense diffère de celle utilisée dans le cadre classique des POMDP, qui est basée sur les paires état-action. Dans cette intention, un bilan des travaux en perception active, nous a permis d'identifier le critère le plus utilisé par cette communauté scientifique [Paletta et Pinz, 2000, Deinzer *et al.*, 2006]. D'autre part, des travaux précédents en planification, et en particulier dans le cadre des POMDP, ont défini des critères mixtes pour l'application à des problèmes de perception, sans toutefois implémenter des algorithmes pour les résoudre. A cet effet, nous avons choisi de coupler le critère classique des POMDP avec le critère classique de la perception active, en nous basant sur les travaux de [Mihaylova *et al.*, 2002]. L'étape suivante a consisté à adapter des algorithmes déjà existants de résolution de POMDP à ce nouveau critère, tout en tenant compte des points soulevés par [Araya-López *et al.*, 2010], comme par exemple la propriété de convexité et les approximations linéaires nécessaires à ce nouveau critère. De plus, une évaluation a été réalisée par la suite, afin de démontrer l'apport d'un critère mixte qui aurait pour but de favoriser la prise d'information par le drone hélicoptère afin de mieux valider les possibles décisions qui doivent être prises en ligne. La proposition de ce critère mixte a donné lieu à trois communications [Carvalho Chanel *et al.*, 2010c, Carvalho Chanel *et al.*, 2010b, Carvalho Chanel *et al.*, 2010a], toutefois dans ces communications l'évaluation du critère mixte diffère de celle présentée dans ce chapitre. En revanche, nous proposons aussi une autre approche de modélisation qui ajoute au modèle des états but fictifs au moyen d'actions de classification. Avec cette modélisation, un tel critère mixte basé sur une mesure de l'incertitude sur l'état de croyance n'est plus forcément nécessaire puisque la fonction de récompense peut dépendre seulement de valeurs définies selon les paires état-action. Une étude comparative des ces approches a été mené afin de vérifier leur équivalence en termes de prise d'informations. Le deuxième cadre d'application se rapporte à la mission d'atterrissage, un problème de décision séquentielle qui est directement modélisé en tant que POMDP classique en ce qui concerne la structure de la fonction de récompense. Nous nous sommes aussi intéressée à ce cadre applicatif afin de vérifier que le POMDP est capable de gérer implicitement l'acquisition d'information pour atteindre le but de la mission, qui est d'atterrir à côté d'une cible en particulier parmi d'autres cibles présentes ou non dans l'environnement. Pour ce cadre d'application, nous avons aussi évalué des politiques obtenues à partir du critère mixte proposé et une étude comparative des stratégies a aussi été menée. Les résultats démontrent que pour ce type de mission l'optimisation explicite d'une mesure de l'incertitude de l'état de croyance n'est pas nécessaire. Nous pensons en effet et démontrons dans cette thèse que si l'on ajoute au modèle des états buts fictifs (au moyen d'actions de classification ou d'une action d'atterrissage), un tel critère mixte basé sur une mesure de

l'incertitude de l'état de croyance ne serait plus nécessaire dans de nombreux cas pratiques (y compris en perception active pure). L'étude du critère mixte permet en fait d'ajuster les récompenses d'un modèle POMDP classique équivalent. Ces deux approches sont donc, en pratique, complémentaires. En ce sens, ces études nous apportent des réponses en ce qui concerne les avantages et inconvénients de l'application du POMDP dans ces différents problèmes à caractère applicatif.

Dans le chapitre 5, nous nous sommes intéressée à prendre en compte certaines contraintes physiques et de sûreté liées à la mission du drone hélicoptère, comme par exemple, atterrir seulement si une zone est "atterrissable", ou garantir que le changement de zone se fera à une altitude de vol donnée. On a vérifié que le modèle standard des POMDP permet de modéliser ces contraintes sur les actions, mais au prix d'une complexité de résolution plus élevée, puisque il est nécessaire d'ajouter des états et des observations supplémentaires au modèle. Étant donné que les contraintes de sûreté sont primordiales dans des applications en aéronautique, notre objectif a été de proposer une approche générale alternative au modèle classique capable de prendre en compte les contraintes liées à la mission. Tout d'abord nous avons mis en évidence la façon dont on modélise ces contraintes classiquement, parce que le modèle classique des POMDP nous permet de modéliser des contraintes sur les actions à partir d'un réglage de coûts de type potentiomètre. Dans ce type de modélisation, en particulier pour la robotique, le modèle d'observation est augmenté et structuré selon $\Omega = \mathcal{O} \times \Theta$, où $\mathcal{O}$ représente les observations "standards" et Θ les observations sur la "faisabilité" des actions, résultant ainsi en un modèle où des informations des sémantiques différentes sont en quelque sorte confondues. Nous cherchons donc à proposer une façon alternative pour traiter ce problème, qui nous conduit à un nouvel algorithme qui exploite explicitement la séparation entre l'information concernant la vérification des propriétés du modèle (les préconditions), et celle qui renseigne sur la présence du but de mission ou sur la nature des objets de l'environnement. Nous avons appelé ce nouveau modèle AC-POMDP (*Action Constrained POMDP*). Pour la comparaison de ce nouveau modèle AC-POMDP, nous avons proposé une transformation entre le modèle AC-POMDP et le modèle POMDP, qui nous a permis de démontrer que les AC-POMDP sont inclus dans les POMDP. Comme nous considérons que l'information sur la faisabilité des actions a une sémantique différente, ceci nous permet des générer des sous-ensembles d'actions basés sur l'état de croyance. Ainsi, les équations d'optimisation proposées pour le modèle AC-POMDP permettent de ne sélectionner que des actions faisables avec certitude, lors de la mise à jour de la valeur de l'état de croyance. Pour l'évaluation des AC-POMDP nous proposons un algorithme appelé PCVI - (*PreCondition Value Iteration*). PCVI tire profit de la structure des AC-POMDP afin d'optimiser la fonction de valeur paramétrée par des α-vecteurs eux-mêmes définis sur des sous-espaces de croyance différents et réduits, contrairement au modèle POMDP qui optimise la fonction de valeur paramétrée par des α-vecteurs qui sont définis sous l'espace entier de croyance. Ainsi, les évaluations ont pour but de vérifier la complexité algorithmique du nouveau modèle ainsi que la garantie du respect des contraintes. Les travaux présentés dans ce chapitre ont donné lieu à deux communications [Carvalho Chanel *et al.*, 2011b, Carvalho Chanel *et al.*, 2011a].

Le chapitre 6 s'intéresse enfin à la résolution en ligne du problème de planification, étant donné que certains paramètres du problème de planification ne peuvent pas être connus avant le vol, c'est-à-dire avant le début de la mission. Par exemple, le nombre de zones à explorer ne peut être défini au préalable. Ceci est un facteur déterminant dans le problème qu'on veut résoudre. Il n'est donc pas possible de résoudre le problème hors ligne et d'embarquer seulement le plan d'action calculé. Il est important de noter que la taille et la complexité du problème ne permettent pas d'optimiser la politique complète avant d'exécuter la première action. En fait, contrairement aux approches POMDP classiques d'optimisation en ligne, il est impératif de contrôler le temps consacré à chaque mise à jour de Bellman.

Les contraintes temporelles de calcul sont dures et imposées par les architectures d'exécution dans laquelle les algorithmes POMDP sont embarqués. Dans cette intention nous débutons le chapitre par l'application robotique de la mission d'atterrissage. Ensuite, nous présentons la génération automatique du problème de planification qui a été mise en place, et nous présentons l'architecture embarquée sur l'hélicoptère autonome. Pour la résolution en ligne, nous avons adapté des algorithmes de résolution en ligne déjà existants pour les POMDP à la plate-forme de planification embarquée sur les drones hélicoptère de l'Onera. Cette plate-forme propose un cadre de résolution et d'exécution original, qui nous permet d'anticiper l'évolution probabiliste du système afin d'optimiser des morceaux de politique en parallèle de l'exécution. Ainsi nous avons appelé ce cadre d'optimisation anticipée et d'exécution en parallèle "AMPLE" (*Anytime Meta PlannEr*). Pour valider l'approche de planification en ligne proposée nous avons réalisé des vols expérimentaux et nous avons évalué statistiquement le cadre de résolution. Les évaluations statistiques ont été faites grâce à des simulations très réalistes de type *hardware-in-the-loop* afin de vérifier le bon fonctionnement de toutes les parties logicielles concernées. Ces simulations ont consisté à mettre en œuvre l'architecture de supervision et de planification et le traitement d'image qui sont actuellement embarqués sur le drone hélicoptère. Ces travaux ont donné lieu à quatre communications [Carvalho Chanel *et al.*, 2012a, Carvalho Chanel *et al.*, 2012b, Carvalho Chanel *et al.*, 2012c, Carvalho Chanel *et al.*, 2013].

Première partie

Etat de l'art

Chapitre 1

Planification, perception et formalismes de décision

Ce premier chapitre a pour objectif de présenter la problématique de la planification pour la perception, ainsi que les différents formalismes de décision séquentielle sous observabilité partielle étant donné notre application à la détection et à la reconnaissance de cibles en environnement partiellement observable. Nous débuterons par la présentation de la planification et de la perception en robotique. Ensuite, une attention spéciale sera donnée aux différents critères de performance liés à la décision pour la perception dans le cadre de la perception active. Après, nous aborderons des formalismes de décision séquentielle existants dans un cadre probabiliste, où le but est de planifier des actions qui sont conditionnées aux observations de l'environnement. Le cadre probabiliste se base sur la prise en compte des sorties produites par les systèmes de traitement d'information, qui sont indispensables pour la perception. Cette étude bibliographique justifie le choix des Processus Décisionnel de Markov Partiellement Observables (POMDP) comme modèle formel de notre application. Ensuite, une discussion approfondie sur différents travaux qui utilisent les POMDP comme modèle formel pour la planification en robotique est présentée. Nous conclurons avec un bilan de cette étude bibliographique et par les intuitions en ce qui concerne les pistes de recherche explorées lors de ce travail de thèse.

1.1 Planification, perception et action en robotique

"*La planification est le processus de choix et d'organisation des actions tout en anticipant leurs effets*" [Ghallab *et al.*, 2004]. Le choix et l'organisation des actions a pour objectif d'atteindre de la meilleure façon possible des buts prédéfinis. La planification cherche à définir le comportement, soit le plan d'actions, conditionnel ou pas, qui sera exécuté au fur et à mesure que le système robotique évolue dans l'environnement.

Les actions d'un robot peuvent l'amener à changer son état, comme par exemple les déplacements sur une grille, mais aussi, changer l'état de l'environnement où le robot se trouve, par exemple, en ramassant un objet. Il existe plusieurs approches de la planification, en fonction des actions possibles, des effets des actions, de l'observabilité parfaite ou imparfaite de l'état, du nombre d'acteurs impliqués.

La planification de mouvements, un cas particulier du problème général de la planification d'actions pour le contrôle de systèmes dynamiques, consiste à déterminer la séquence d'actions, dont les effets peuvent ou non être déterministes, dans l'espace de configuration

d'un robot pour l'amener d'une position initiale à une position finale. La planification de la perception consiste quant à elle, à planifier des actions pour la prise d'information afin d'identifier l'état de l'environnement (diagnostic, identification, reconnaissance). La planification pour la navigation et l'exploration est une combinaison des deux problèmes : mouvement et perception, avec l'objectif d'explorer une zone ou d'atteindre un but de mission. La planification pour la coordination multi-robots cherchera quant à elle, à construire un plan pour que les robots atteignent le but commun.

1.1.1 Séquence d'action, plan conditionnel et politique

Une séquence d'actions $a_1, ..., a_n$, peut être conçue de différentes manières avec plus ou moins d'impact sur le problème de planification devant être résolu. Une séquence d'actions peut être décrite, par exemple, comme un ensemble de points qui forment une trajectoire de référence à laquelle s'ajoute une déviation admise. De cette façon, le problème de planification est réduit à un problème d'optimisation en dimension finie. Dans des problèmes de planification déterministe, c'est-à-dire où l'évolution du système robot-environnement a une dynamique déterministe, et où l'on cherche à atteindre un état but, le résultat de la planification est un plan, soit une séquence d'actions, qui amène de manière certaine de façon optimale ou pas, le robot à l'état but.

Un plan conditionnel, par contre, est défini comme une séquence d'actions qui peut à un moment donné t dépendre de l'état du système ou d'un événement extérieur. A cet instant t, différents plans d'actions peuvent être utilisés suivant l'état du système.

Une politique est généralement définie comme des actions qui suivent un plan conditionnel. Une politique est une fonction $\pi : s \rightarrow a$, c'est-à-dire une fonction qui définit une action a pour tout état s. Une politique est le résultat de la planification pour les problèmes dit non-déterministes et/ou probabilistes. Dans ces problèmes, les actions du robot peuvent amener le système à différents états, et le calcul, en général hors ligne, du plan dépend de l'évolution du système à chaque instant. De cette façon une action est planifiée pour chaque état du système. La question qui se pose alors est : quelle est la meilleure politique que l'on peut appliquer étant donné un problème de planification non-déterministe et/ou probabiliste ? La réponse sera donnée dans la section 1.3 où nous détaillerons les différents cadres de décision séquentielle pour des problèmes de planification probabiliste.

1.1.2 Perception

Une des exigences appliquées à des robots mobiles est qu'ils doivent agir en dépit des incertitudes sur leur environnement et sur les effets de leurs actions. La perception ou prise d'information est ce qui permet au robot d'acquérir des données sur l'environnement. Ceci est fondamental, afin que l'acteur estime et corrige son état après l'exécution d'une action (voir figure 1.1). La perception est la brique qui permet au robot de mettre à jour son estimation d'état, et ce afin de réaliser l'action la plus adaptée par rapport à l'estimation d'état courante. Les robots perçoivent leur environnement grâce à des capteurs, soit de vision, soit de position. Ces capteurs peuvent être hétérogènes. Par exemple un robot peut utiliser un radar pour estimer la distance à un mur après un déplacement vers ce mur. Ou encore, un robot peut avoir à sa disposition une caméra qui lui permet d'identifier des objets. Ainsi, la perception est une brique indispensable pour que le robot puisse interagir avec son environnement.

1.2 Perception active

La perception active vise à maximiser la connaissance sur l'environnement. Elle peut être aussi définie par la prise de décision de l'agent en tenant compte des effets de ses

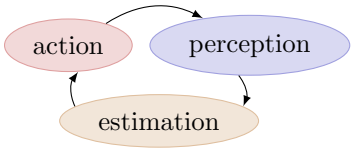


FIGURE 1.1 – cycle action-perception-estimation.

actions sur ses capacités de perception de l'environnement [Spaan, 2008, Mihaylova *et al.*, 2002, Dutta Roy *et al.*, 2004]. La figure 1.2 montre un exemple de boucle de contrôle pour la perception active. Dans le cas présent, le robot ou le système intelligent doit contrôler son capteur (par exemple : une caméra) pour acquérir les caractéristiques informatives qui permettent de mieux distinguer des objets grâce à un système d'interprétation s'appuyant sur une base d'apprentissage. À partir des données obtenues par le système d'interprétation, le robot ou le système intelligent met à jour une croyance courante, puis décide ensuite de manière autonome du contrôle à appliquer sur son capteur (changer la focale, l'angle de vue, la hauteur, ou le champ de vision).

La perception active est vue comme un défi pour plusieurs raisons. La fonction *d'interprétation*, qui est responsable de la traduction de l'information acquise, est une thématique de recherche actuelle [Wang *et al.*, 2012, Saux et Sanfourche, 2011, Eidenberger et Scharinger, 2010, Defretin *et al.*, 2010]. Plusieurs algorithmes de traitement d'information, qui dépendent de manière ad hoc du capteur utilisé, sont proposés dans la littérature. La fonction de *pilotage*, qui est responsable de la prise de décision, quant à elle, se rattache à la planification pour la perception, où la solution du problème dépend d'un critère d'optimisation qui est, dans la majorité des cas, une fonction multi-objectif [Mihaylova *et al.*, 2002, Spaan, 2008, Araya-López *et al.*, 2010]. Ce critère d'optimisation gère le compromis entre le gain d'information acquis par le système et le coût des actions. Ce dernier est souvent lié au coût du traitement numérique, tels que le temps ou le nombre d'opérations qui sont particulièrement importants lorsqu'on travaille en temps réel.

Une autre difficulté est liée aux incertitudes dans le modèle du robot, de l'environnement et des capteurs. Ces incertitudes amènent à des modèles dynamiques et de mesure non-linéaires. En plus, dans la plupart des cas, les informations sur l'environnement acquises par les capteurs ne se rapportent pas à toutes les variables d'état du système, ce qui conduit à un système partiellement observable. Pour cela, une croyance courante sur l'état du système, qui est une estimation de l'état courant, souvent modélisée par une distribution de probabilité sur les états possibles, doit être maintenue à tout instant. Comme cette croyance est maintenue

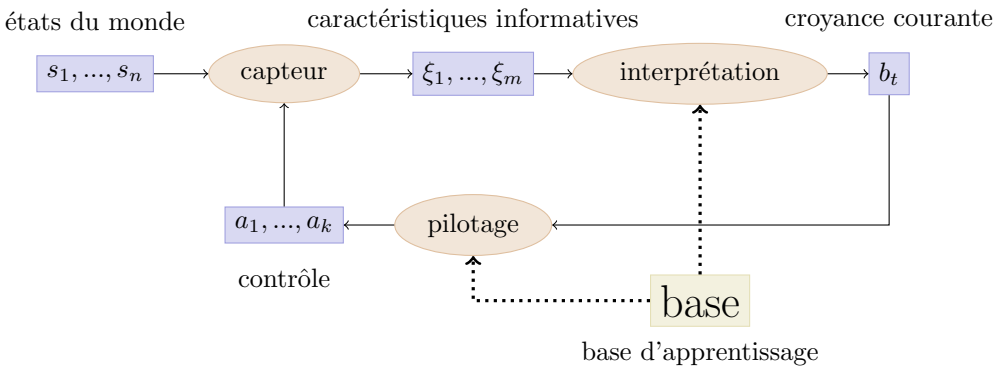


FIGURE 1.2 – Boucle fermée à événement discrets et évolution non-linéaire de la perception active.

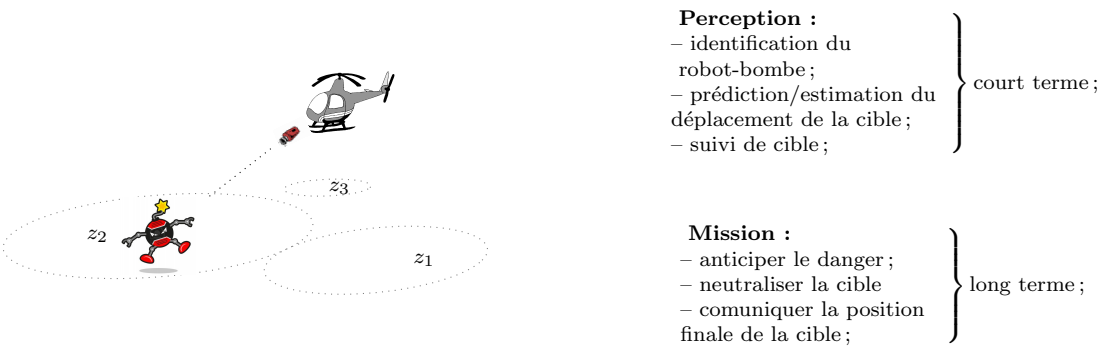


FIGURE 1.3 – Scénario de perception/mission pour la reconnaissance de cible.

à tout instant, lors de la résolution du problème de planification nous sommes confrontés à une explosion combinatoire du nombre (d'états) de croyances courantes possibles. Enfin, le modèle du robot et les modèles des capteurs sont non linéaires ; même s'il existe des méthodes pour linéarisation de ces modèles, malheureusement beaucoup de problèmes ne peuvent pas être traités par la linéarisation, comme par exemple le filtre de Kallman étendu. Ils nécessitent un développement de techniques spécifiques pour le choix des actions.

Dans le cas général, choisir les actions de perception requiert un compromis entre actions de court et de long terme. Le robot doit prendre les décisions qui vont l'amener au but de sa mission, ainsi que celles qui vont lui permettre de prendre plus d'informations en relation avec son environnement. Les deux types d'actions sont nécessaires pour atteindre au mieux l'objectif de la mission. Par exemple, prenons un cas d'application où un drone hélicoptère doit identifier une cible considérée comme dangereuse, afin d'anticiper ses actions, ou de la neutraliser (scénario de la figure 1.3). Des actions de court terme sont par exemple, les actions de perception : identifier, prédire le déplacement de la cible, suivre la cible ; et les actions de long terme sont celles définissant la fin de la mission : anticiper le danger, neutraliser la cible, communiquer la position finale de la cible.

Comme autres exemples de perception active, appliqués à des scénarios bien définis qui dépendent du compromis court et long terme, on peut citer :

- des applications pour la saisie des objets, où l'incertitude concerne la position, l'orientation ou l'identité des pièces manipulées par le robot [Eidenberger *et al.*, 2009, Eidenberger et Scharinger, 2010] ;
- la navigation de robots mobiles : en environnement connu, en partant d'une position et d'une orientation avec le but d'atteindre une position et orientation spécifiées dans un temps prédéterminé [Simmons et Koenig, 1995, Burgard *et al.*, 1997, Thrun *et al.*, 2005, Candido et Hutchinson, 2011] ;
- des applications en vision active via sélection des paramètres comme la distance focale, l'angle de vue, et ce afin de réaliser l'identification et la reconnaissance des objets [Deinzer *et al.*, 2003, Eidenberger *et al.*, 2008, Spaan, 2008, Defretin *et al.*, 2010] ;

1.2.1 Formalisation du problème de la perception active

Suivant [Mihaylova *et al.*, 2002], la perception active peut être exprimée comme un problème de génération de trajectoires dans un environnement stochastique dynamique décrit par le modèle :

$$s_{t+1} = f(s_t, u_t, \eta_t) \tag{1.1}$$
$$b_{t+1} = h(s_{t+1}, w_{t+1}, v_{t+1}) \tag{1.2}$$

où, s est le vecteur d'état, f et h les fonctions non nécessairement linéaires du modèle du système et du modèle de mesure, b le vecteur de mesure, η et v sont respectivement le bruit du système dynamique et le bruit de mesure ; u est le vecteur de commande de la dynamique du système, et w est le vecteur de commande de la fonction de mesure avec les paramètres des capteurs (comme la distance focale, etc). Les indices t et $t+1$ indiquent l'axe temporel.

L'état du système est influencé par les entrées u et w. Ici, nous ne ferons aucune différence entre u et w, et nous les noterons comme une action a. Les systèmes conventionnels, pour lesquels la perception n'est pas contrôlée, sont formés seulement par des composantes d'estimation et de contrôle, dont l'on suppose les entrées données et connues. Les systèmes autonomes sont contraints d'adapter ces entrées de manière à obtenir la meilleure estimation possible pour remplir au mieux leurs objectifs.

Un critère de performance multi-objectif, appelé fonction de valeur, est nécessaire pour quantifier chaque séquence d'actions $(a_1, a_2, ..., a_n)$: le gain attribué à l'acquisition d'informations et celui associé aux coûts des actions [Mihaylova *et al.*, 2002]. Ainsi :

$$W = \min_{a_1,...,a_n} \left\{ \sum_{j=1}^{n} \rho_j U_j + \sum_{l=1}^{n} \beta_l C_l \right\} \tag{1.3}$$

Ce critère est composé par la somme pondérée de deux composantes :

- Les termes en j représentent l'espérance des mesures des incertitudes par rapport à l'état réel du système ; ou encore, par rapport à la précision nécessaire pour atteindre le but. Dans le cadre bayésien, l'estimation de l'incertitude est basée sur une distribution de probabilité sur les états atteignables du système ;
- Les termes en l représentent l'espérance des coûts, par exemple, ceux associés aux déplacements du robot : énergie, temps, distances aux obstacles, distance au but.

U_j et C_l, sont fonctions de la séquence $a_1, ..., a_n$. Les pondérations ρ_j et β_l attribuent un poids différent aux deux termes, et sont des paramètres réglés arbitrairement par le concepteur.

Dans la suite nous présentons quelques critères de performance liés à l'incertitude généralement utilisés dans la littérature de la perception active. Ces critères de performance peuvent être séparés selon la prise en compte dans les décisions du court ou long terme. Nous rappelons que, en ce qui concerne notre application de détection et reconnaissance de cibles, nous souhaitons mettre en place des décisions séquentielles à long terme, parce que la perception n'est pas seulement une fin en soi, mais aussi un moyen, indispensable au bon accomplissement de la mission tout en tenant compte des coûts liés aux actions de déplacement ou de changement d'angle de vue.

1.2.2 Critères de performance liés à l'incertitude

La décision autonome associée à la perception active se base très fortement sur une mesure de l'incertitude de la croyance courante, souvent définie par une distribution de probabilité sur les états du système dans le cadre bayésien. Dans la perception active pour la reconnaissance d'une scène ou d'un objet, l'utilité d'une observation est déterminée par le gain d'information. Plusieurs travaux utilisent l'entropie de Shannon pour quantifier l'information associée à l'état de croyance [Burgard *et al.*, 1997, Deinzer *et al.*, 2003, Eidenberger *et al.*, 2009, Eidenberger et Scharinger, 2010]. D'autres utilisent la mesure d'information mutuelle entre deux points de vue différents [Deguchi et Ohtsu, 2006, Yu *et al.*, 2009], ou encore, la divergence de Kullback–Leibler [Candido et Hutchinson, 2011]. Ces fonctions sont présentées dans la suite, en nous basant sur l'ouvrage [Cover et Thomas, 2006].

Définition 1.2.1 *L'entropie de Shannon d'une variable aléatoire discrète X sur un ensemble $\mathcal{X}$ est définie par :*

$$H(X) = -\sum_{x\in\mathcal{X}} p(x)\log(p(x)) \tag{1.4}$$

où $p(x)$ est la fonction de masse de probabilité, c'est-à-dire, la fonction qui donne la probabilité d'un résultat élémentaire d'une expérience telle que $p(x) = Pr(X = x)$, $x \in \mathcal{X}$. Notons que, $0 \leq H(X) \leq \log(|\mathcal{X}|)$, où $|\mathcal{X}|$ représente le nombre total d'éléments de l'ensemble $\mathcal{X}$, et que $H(X)$ est maximale quand X est une distribution équiprobable sur tous les éléments $x \in \mathcal{X}$, ce qui représente, dans le cadre bayésien, l'absence totale d'information.

L'entropie jointe de deux variables aléatoires X et Y avec une fonction de masse conjointe $p(x, y)$ peut être établie également : $H(X, Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x, y)\log(p(x, y))$.

Définition 1.2.2 *L'entropie d'une variable aléatoire Y conditionnée à la variable aléatoire X, $H(Y|X)$ est définie par :*

$$\begin{aligned} H(Y|X) &= \sum_{x\in\mathcal{X}} p(x)H(Y|X = x) &(1.5)\\ &= -\sum_{x\in\mathcal{X}} p(x)\sum_{y\in\mathcal{Y}} p(y|x)\log(p(y|x)) &(1.6)\\ &= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x, y)\log(p(y|x)) &(1.7)\end{aligned}$$

Définition 1.2.3 *On considère deux variables aléatoires X et Y avec une fonction de masse jointe $p(x, y)$ et avec des fonctions de masse marginales $p(x)$ et $p(y)$. La mesure d'information mutuelle $I(X, Y)$ est l'entropie relative entre la distribution jointe et le produit des distributions marginales :*

$$I(X, Y) = \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x, y)\log\frac{p(x, y)}{p(x)p(y)} \tag{1.8}$$

L'information mutuelle permet de mesurer le gain d'information d'une variable aléatoire par rapport à une autre, et peut être écrite en fonction de l'entropie conditionnelle :

$$\begin{aligned} I(X, Y) &= H(X) + H(Y) - H(X, Y) &(1.9)\\ &= H(X) - H(X|Y) &(1.10)\\ &= H(Y) - H(Y|X) &(1.11)\end{aligned}$$

Décision à court terme :

Dans [Burgard *et al.*, 1997], la localisation active est présentée avec l'objectif d'estimer la localisation du robot à partir de données extraites de capteurs. Ceci en supposant que durant le procédé de localisation, l'agent dispose d'un accès total ou partiel aux commandes de ses capteurs et de ses actionneurs. L'idée clé de cette approche est que l'efficacité de la localisation est améliorée par les commandes actives de la direction des déplacements du robot et de ses capteurs. Le principe est de contrôler les actionneurs du robot afin de minimiser l'espérance de l'incertitude de la distribution de probabilité b sur les positions possibles, soit l'état de croyance ou l'hypothèse courante. Ce critère est modélisé par l'espérance de l'entropie de Shannon sur l'état de croyance $E_a(H(b))$, et le coût du déplacement en jeu $c(a)$.

$$a^* = \arg\min_a(E_a(H(b)) + \sigma c(a)), \text{ avec } \sigma \geqslant 0 \tag{1.12}$$

La localisation active, présentée dans [Burgard *et al.*, 1997], calcule une distribution de probabilité b sur toutes les localisations possibles dans l'environnement. Donc, pour réduire l'incertitude sur l'estimation de l'état, le robot doit choisir l'action qui peut l'aider à mieux distinguer les différentes positions. Autrement dit, il doit diminuer l'entropie de sa distribution de probabilité sur ses positions possibles après chaque action. L'avantage de cette approche est que les coûts de déplacement sont pris en compte, mais, par contre, contrairement à ce que nous souhaitons, il n'y a pas de projection à long terme. Le critère de performance ici peut être vu comme un critère local. Le robot choisit l'action qui lui rapporte *immédiatement* une mesure plus certaine sur sa position avec un coût minimum.

Le même critère de performance est utilisé dans [Eidenberger *et al.*, 2008] pour la sélection des points de vue ; par contre ce travail ne tient pas compte des coûts associés aux actions. De plus, une fois qu'un point de vue est choisi par l'agent, il est ensuite pénalisé, afin d'éviter que l'agent réalise des prises de vue avec le même jeu de paramètres. Grâce au critère de minimisation local, l'estimation de l'état va être améliorée à chaque instant de décision, par contre l'optimalité vis-à-vis d'un critère global (projection à long terme) ne peut pas être garantie.

Dans [Deguchi et Ohtsu, 2006] le critère est défini en termes de réduction d'incertitude et d'ambiguïté de l'observation. L'objectif est de réduire le nombre d'étapes pour la décision finale de reconnaissance d'un objet parmi d'autres enregistrés dans la base de données. Les autheurs se sonnent comme objectif de réduire le nombre d'étapes pour la reconnaissance, mais il définissent toutefois un critère local de décision, c'est-à-dire à court terme. Le critère cherche à sélectionner à l'étape t le point de vue immédiat de l'étape $t+1$ qui maximise le gain d'information entre b_t et b_{t+1}, qui est quantifié par la mesure d'information mutuelle :

$$I_a(b_t, b_{t+1}) = H(b_t) - H(b_{t+1}) \tag{1.13}$$

$$= H(b_t) - H(b_t|a, o_t), \tag{1.14}$$

où l'estimation de l'état b_t est un vecteur dont chaque composante $b_t^i = Pr(Obj = obj_i)$ représente la probabilité à l'instant t que i soit l'objet observé, o_t représente l'image perçue après réalisation de l'action a. Ensuite, l'action optimale est évaluée non seulement en termes de gain d'information, mais aussi en termes de coût pour obtenir cette nouvelle image. Les auteurs considèrent le temps dépensé T_a comme coût pour amener la caméra jusqu'à la nouvelle position. L'action choisie à l'instant t sera donc celle qui maximise l'information mutuelle avec un temps minimal :

$$a^* = \arg\max_a \frac{I_a(b_t, b_{t+1})}{T_a}. \tag{1.15}$$

Une fois de plus nous tenons à remarquer que, contrairement à ce que nous souhaitons, il n'y a pas de projection à long terme. Le critère de performance peut être aussi vu comme un critère local. Grâce au critère de minimisation local, l'estimation de l'état pour la reconnaissance va être améliorée à chaque instant de décision, par contre l'optimalité vis-à-vis d'un critère global avec une projection à long terme ne peut pas être garantie.

[Eidenberger *et al.*, 2009] et [Eidenberger et Scharinger, 2010] modélisent le problème de perception active en tant que Processus Décisionnel Markovien Partiellement Observable (POMDP). La règle de décision optimale est définie par une fonction de valeur (programmation dynamique) telle que :

$$V_t(b_t) = \max_{a_t} \left[r(b_t, a_t) + \gamma \int p(o_{t+1}|a_t, b_t) V_{t-1}(b_{a_t}^{o_{t+1}}) do_{t+1} \right], \text{ avec} \tag{1.16}$$

$$r(b_t, a_t) = -\rho E_o[H(b_{a_t}^{o_{t+1}})] + \int c(s, a_t) b_t(s) ds \tag{1.17}$$

où, b_t représente l'état de croyance (distribution de probabilité sur les états s) à l'instant t, $b_{a_t}^{o_{t+1}}$ représente l'état de croyance immédiat après exécution de a_t et observations de o_{t+1}, $E_o[H(b_{a_t}^{o_{t+1}})]$ représente l'espérance de l'entropie sur les observations possibles après la réalisation de l'action a_t, $c(s, a_t)$ le coût associé à l'action a_t et aux états s, ρ une constante qui permet d'équilibrer les deux critères et γ le facteur d'actualisation. Toutefois, le POMDP en question n'est pas résolu pour l'obtention d'une politique. La décision, pour l'obtention de l'action optimale $\pi_t(b_t) = a_t^*$, est calculée en ligne par un choix glouton d'action, tel que :

$$\pi_t(b_t) = \arg\max_{a_t} \left[r(b_t, a_t) \right], \tag{1.18}$$

Dans [Filliat et Meyer, 2000], la contribution consiste en des décisions de moyen terme par rapport à des déplacements et à la construction incrémentale des états à partir d'un modèle POMDP dégradé. Le robot cherche à construire la carte de son environnement à partir du choix d'actions qui lui permettront de découvrir et d'acquérir plus d'information sur son environnement, ou du choix d'actions de déplacement vers des directions pour lesquelles il n'a encore aucune donnée. Malheureusement, dans ce travail aucun critère de décision n'est explicitement formalisé.

Décision à long terme :

Dans [Deinzer *et al.*, 2003], l'agent doit choisir des points de prise de vue pour faciliter la classification, en évitant des prises de vue ambiguës, ou en excluant certaines hypothèses d'identification. L'agent dispose d'une caméra contrôlable. La modélisation du problème attribue une récompense plus importante au choix du point de vue qui augmente la quantité d'information acquise en diminuant l'incertitude de l'état de croyance b sur les états du système. La mesure de l'incertitude de l'état de croyance est exprimée par l'entropie. A l'instant t, le processus de décision, c'est-à-dire le choix du point de vue (par une politique π), aura pour but la maximisation de l'espérance accumulée et pondérée des récompenses futures. La récompense, ici, ne dépend pas des coûts liés aux mouvements de la caméra, mais seulement de la quantité d'information acquise :

$$\pi_t^*(b) = \arg\max_{\pi} E\left[R_t \mid b_t = b, \pi \right], \text{ avec} \tag{1.19}$$

$$R_t = -\sum_{n=0}^{\infty} \gamma^n H^{\pi}(b_{t+n+1}). \tag{1.20}$$

L'avantage de ce travail est la projection sur le long terme représentée par l'espérance de la somme pondérée des récompenses futures. Comme exprimé dans les équations 1.19 et 1.20, la récompense à l'instant t se rapporte seulement sur la somme des mesures de l'incertitude futures de l'état de croyance. Ceci est contraire à ce que nous souhaitons faire, où les coût associés aux déplacement de la caméra devront être pris en compte dans le critère, étant donné que notre agent hélicoptère dépense du carburant pour ses déplacements.

Dans [Deutsch *et al.*, 2004], le même genre de critère de performance est utilisé pour déterminer une séquence d'actions optimale affectant des niveaux de zoom de la caméra dans une tâche de suivi d'objet pour un horizon de décision k donné. L'action est sélectionnée afin de minimiser l'espérance de l'entropie de l'estimation d'état conditionnelle aux actions et observations passées. Le modèle utilisé pour l'estimation d'état est un filtre de Kalman étendu. Pour déterminer les actions optimales avant d'obtenir des observations, les auteurs utilisent l'entropie conditionnelle de l'estimation d'état courante par rapport à une séquence d'actions $\langle a \rangle^k$ et observations $\langle o \rangle^k$ données. En moyennant sur toutes les séquences d'observations $\langle o \rangle^k$ possibles, il est possible de retrouver la séquence optimale d'actions (cf. [Deutsch

et al., 2004]), telle que :

$$\langle a\rangle^{k*} = \arg\min_{\langle a\rangle^k} H(b_{t+k}|\langle o\rangle^k, \langle a\rangle^k), \text{ où} \quad (1.21)$$

$$H(b_{t+k}|\langle o\rangle^k, \langle a\rangle^k) = \int p(\langle o\rangle^k|\langle a\rangle^k) H(b^+_{t+k}) d\langle o\rangle^k, \text{ avec} \quad (1.22)$$

$$H(b^+_{t+k}) = -\int p(b_{t+k} \mid \langle o\rangle^k, \langle a\rangle^k) \log(p(b_{t+k} \mid \langle o\rangle^k, \langle a\rangle^k)) db_{t+k} \quad (1.23)$$

Le critère présenté ici s'appuie sur la projection à long terme, en considérant les actions et les observations passées, par contre comme dans [Deinzer *et al.*, 2003], les coûts associés aux changements de niveaux de zoom ne sont pas pris en compte. On pourrait admettre que ces coûts sont négligeables en pratique. En revanche, pour notre application, un changement de zoom se traduit par un changement d'altitude de vol, ce qui entraîne une dépense de carburant. Nous ne pouvons pas raisonnablement négliger ces coûts si l'on veut obtenir un modèle réaliste.

D'autre travaux modélisent leur critère de performance pour des décisions à long terme : on peut citer [Spaan, 2008, Sridharan *et al.*, 2008] où les problématiques sont représentées par des modèles POMDP factorisés ou hiérarchiques. Le critère utilisé est le critère classique des POMDP. Le critère ainsi que le modèle POMDP seront présentés dans le chapitre 2. Le critère classique des POMDP permet d'élaborer des stratégies plus robustes pour le traitement d'images et le suivi des objets d'intérêt que d'autres procédés plus réactifs de traitement d'images ou de pistage [Sridharan *et al.*, 2008]. Mais avant de discuter ces travaux plus en détail (section 1.4), nous présentons la décision séquentielle en environnement partiellement observable, car, pour notre application de détection et reconnaissance de cibles il est préférable d'optimiser des décisions à long terme. Après chaque action, nous pouvons/devons observer l'état du système, pour mettre à jour une croyance courante, afin de choisir la meilleure action pour la suite. De plus, l'observation obtenue n'est pas précise, puisque elle se rapporte à la sortie du traitement d'information. Ceci nous place dans un cadre de décision séquentielle en environnement partiellement observable.

Dans la suite, une revue sur la décision séquentielle en environnement partiellement observable, ainsi que sur les différents formalismes de décision séquentielle, sera présentée. Nous cherchons ici à justifier notre choix du modèle POMDP comme formalisme de décision séquentielle en environnement partiellement observable.

1.3 Décision séquentielle en environnement partiellement observable

La décision séquentielle correspond au processus d'évaluation des actions futures et de détermination de la meilleure séquence d'actions pour satisfaire les buts spécifiques d'un problème de planification [Eidenberger et Scharinger, 2010]. Les problèmes de décision séquentielle nécessitent que la valeur de l'état soit observée entre deux étapes de décision [Pralet *et al.*, 2010a].

Dans sa thèse, M. Littman [Littman, 1996] suggère que le problème de décision séquentielle constitue la tâche fondamentale d'un système intelligent quand celui-ci interagit avec son environnement. La décision séquentielle consiste à répondre à la question : *Que dois-je faire maintenant ?* La décision séquentielle est la réponse à cette question quand : *maintenant* est un état (de croyance) courant parmi un ensemble fini ou infini d'états, *faire* est une action parmi un ensemble fini ou infini d'actions, et *dois* représente l'évaluation à long terme des coûts ou des récompenses associés aux actions et/ou l'état du système. Il s'agit donc ici de décision rationnelles (optimisation d'un critère).

Quand, entre deux étapes de décision, l'état du système ne peut être observé que partiellement, ou de manière imprécise, on se réfère à des problèmes de décision séquentielle en environnement partiellement observable.

Selon le problème de planification différents formalismes pourront être utilisés pour la modélisation et la résolution du problème, c'est-à-dire les actions du système intelligent ont des effets déterministes ou non-déterministes, si les états sont complètement ou partiellement observables, ou mesurés de manière imprécise, ou encore, si l'agent cherche à maximiser ou à minimiser des récompenses ou des coûts à long terme,

Nous nous sommes concentrée sur des problèmes de décision séquentielle où les états sont partiellement observables dans un cadre probabiliste, parce que les observations acquises dans le cadre de la planification pour la perception sont souvent partielles et imprécises. La sortie du processus de traitement d'information est en effet souvent modélisée par la probabilité d'observer telle ou telle caractéristique d'un objet (couleur, identité) sachant l'état actuel du système (orientation du capteur, position du robot), et/ou les états, les actions et les observations passés, avec éventuellement un intervalle de confiance [Deinzer *et al.*, 2003, Deutsch *et al.*, 2004, Deguchi et Ohtsu, 2006, Spaan, 2008, Eidenberger et Scharinger, 2010].

Dans la suite, nous présenterons quelques formalismes pour la décision séquentielle en environnement partiellement observable en nous appuyant sur l'ouvrage [Pralet *et al.*, 2010a].

1.3.1 Différents formalismes pour la décision séquentielle en environnement partiellement observable

Formalismes basés sur SAT Le formalisme *SAT - Satisfiabilité* s'appuie sur la logique propositionnelle. La syntaxe de la logique propositionnelle est basée sur des variables booléennes. Un littéral est une variable booléenne ou sa négation ; une clause est une disjonction des littéraux ; et une formule booléenne est une conjonction de clauses. Le problème de satisfaction SAT consiste à déterminer si, dans une formule booléenne, des valeurs peuvent être affectées aux variables afin de rendre la formule vraie [Davis et Putnam, 1960]. Rendre la formule booléenne vraie équivaut à décider quelles valeurs l'on doit donner aux variables de décision.

Une extension du formalisme SAT a été proposée par [Littman *et al.*, 2001], dite *SAT stochastique*, afin de prendre en compte des variables aléatoires. Ces variables aléatoires ont une sémantique différente de celle des variables de décision puisque nous ne pouvons pas leur affecter des valeurs. Donc, le problème lié à la planification sous incertitude est de déterminer s'il existe une séquence d'actions pour laquelle la probabilité de la trajectoire dans l'espace d'état jusqu'au but soit supérieure à un certain seuil [Littman *et al.*, 2001].

Le non-déterminisme est représenté par une somme sur les valeurs possibles de la variable aléatoire multipliées par leur probabilité. La résolution de ces problèmes est basée sur une séquence d'agrégations du type max et $\sum$ sur la formule booléenne qui décrit le problème. Prenons l'exemple donné par [Pralet *et al.*, 2010a]. Considérons la formule : $(x \vee y) \wedge (y \vee \neg z) \wedge (\neg y \vee z)$. Soit y une variable aléatoire (variable non contrôlable), avec une distribution de probabilité uniforme sur ses deux valeurs possibles, c'est-à-dire, que la variable y peut prendre la valeur vrai ou faux avec la probabilité de 0.5. Alors : il existe une valeur de x pour toute valeur de y, et une valeur de z de façon à ce que la formule booléenne ait une probabilité d'être vraie :

$$\max_x \sum_y \max_z (0.5 \times (x \vee y) \wedge (y \vee \neg z) \wedge (\neg y \vee z)). \tag{1.24}$$

L'opérateur max cherche à rendre la formule vraie, et la somme traduit une espérance sur les valeurs prises par y. La valeur finale de la formule booléenne devra être supérieure à un

certain seuil pour être considérée vraie. Si la valeur de la formule peut être vérifiée vraie, une politique optimale qui permet que la formule soit toujours satisfaite peut être décrite par : *"assigner x en tant que vrai, ensuite, si y prend la valeur vraie, assigner z en tant que vrai ; sinon, si y prend la valeur faux, assigner à z la valeur faux."*

La prise en compte de la notion d'observabilité partielle pour les problèmes de perception et mission, qui dérive de l'imprécision sur les observations dépendant des états du système, peut être intégrée à ce formalisme en définissant comme variable aléatoire les observations possibles après la réalisation d'une action. Par contre, ce formalisme doit supposer un horizon de décision fini, connu a priori. Ceci n'est pas évident à déterminer dans le cas général de la planification pour la perception et les buts de mission en raison des incertitudes et des cycles éventuels dans le graphe des états visités pendant l'exécution de la politique. De plus, l'utilisation des variables booléennes risque d'être un facteur limitant pour l'utilisation de ce formalisme dans des applications réalistes où les variables sont généralement n-aires.

Formalisme basé sur CSP Le *CSP - Problème de satisfaction de contraintes* est un modèle graphique basé sur des fonctions de contraintes (préférences ou impossibilités). L'objectif est de trouver s'il existe des valeurs de variables vérifiant les contraintes. Ceci revient à résoudre le problème d'optimisation $\max_V(\wedge_{c\in C} c)$, où V représente l'ensemble des variables, et C l'ensemble de contraintes. Ce problème peut être résolu par élimination d'opérateurs (max) sur la conjonction de contraintes. Le CSP est dit cohérent [Pralet *et al.*, 2010a] si les variables affectées vérifient les contraintes.

Comme dans le cas du formalisme SAT, le formalisme CSP a été étendu à *SCSP - Problème stochastique de satisfaction de contraintes* [Walsh, 2002] dans le but de modéliser des problèmes de décision qui prennent en compte des variables aléatoires non-contrôlées et l'observabilité partielle. La variable aléatoire, c'est-à-dire la variable non-contrôlée peut prendre n'importe quelle valeur avec une certaine probabilité, et l'observabilité partielle concerne la variable qui doit être observée entre deux étapes de décision. Ceci transforme le problème de décision en un problème de décision séquentielle.

Définition 1.3.1 *Le SCSP est formellement défini par un triplet (V, P, C) :*
- *V est une séquence de variables. L'ordre dans lequel les variables apparaissent dépend de leur séquence dans les étapes de décision. V est l'ensemble constitué par des variables de décision et des variables aléatoires ;*
- *P est un ensemble de fonctions. La multiplication de ces fonctions donne une distribution de probabilité sur les variables aléatoires. Ces fonctions ne dépendent pas des variables de décision (hypothèse de contingence) ;*
- *C est l'ensemble des contraintes.*

Les SCSP modélisent les incertitudes probabilistes des variables incontrôlables. Dans les SCSP, l'objectif est de maximiser la probabilité que toutes les contraintes soient satisfaites, ou de s'assurer que la probabilité qu'elles soient satisfaites soit supérieure à un certain seuil.

Dans les SCSP on alterne décision et observation, la politique est un arbre où les nœuds sont les variables. Le nœud racine est la première variable en V. Et le dernier nœud en profondeur est la dernière variable de V. Les branches sont les valeurs possibles des variables de décision, ou les valeurs possibles des variables aléatoires. Chaque feuille peut être associée à une valeur de V. Celle-ci est affectée à 1 si les contraintes sont respectées tout le long de la trajectoire, et 0 dans le cas opposé. A chaque feuille une probabilité de réalisation peut être associée, et la valeur d'une politique SCSP, qui est calculée seulement à partir des feuilles affectées à 1 pour garantir le respect des contraintes, est la somme sur toutes les feuilles pondérées par leur probabilité de réalisation. Le SCSP est satisfait si la valeur de la

politique est supérieure à un certain seuil, ou le SCSP est optimal si et seulement si la valeur de la politique est maximale.

De même que pour les SSAT, l'observabilité partielle (i.e. l'imprécision sur les observations dépendant des états du système) peut être aussi intégrée à ce formalisme en définissant comme variable aléatoire les observations possibles après la réalisation d'une action. Un autre avantage de ce formalisme est que les contraintes de sûreté liées à la non réalisation de certaines actions, comme par exemple l'atterrissage dans une zone dangereuse même si la cible recherchée se trouve dans cette zone, pourraient être prises en compte proprement dans le modèle. Par contre, ce formalisme considère un horizon fini et connu a priori [Tarim *et al.*, 2006], contrairement à notre application de planification pour la perception et les buts de mission. De plus, dans le SCSP, l'hypothèse de contingence est une restriction forte [Pralet *et al.*, 2010a]. Celle-ci est violée quand des variables de décision affectent l'état des variables aléatoires, par exemple, dans le cas où le changement de position du robot amène au changement de cible observée, l'observation dépendant ici de la composante de l'état *identité* de la cible.

Le travail de C. Pralet [Pralet et Verfaillie, 2008] propose dans le cadre déterministe des CSP une façon de tenir compte de l'horizon inconnu de décision. L'approche introduit la notion de *timelines*, afin de définir des variables de dimension représentant le nombre possible et inconnu d'étapes de décision permettant au CSP de considérer un horizon h qui n'est pas connu à l'avance. La contribution majeure est le modèle *CNT - Constraint Network on Timelines* et l'algorithme qui permet sa résolution afin d'obtenir une solution satisfaisante pour le problème. Ce travail n'a pas encore été généralisé aux SCSP.

Formalisme basé sur réseaux bayésiens Les réseaux bayésiens (RB) permettent de représenter des distributions de probabilité jointes sur un ensemble de variables à partir d'une connaissance a priori de la relation entre ces variables [Djian *et al.*, 1995, Marchand et Chaumette, 1997].

Définition 1.3.2 *Formellement, un réseau bayésien est un triplet (V, G, P), tel que :*
- *V est un ensemble fini de variables ;*
- *G est une graphe orienté acyclique sur V ;*
- *$P = \{P_{x|parents_G(x)} | x \in V\}$ est un ensemble de distributions de probabilités conditionnelles sur chaque variable $x \in V$ étant donné ses parents dans G ;*

Dans les réseaux bayésiens les nœuds représentent les variables aléatoires discrètes et les arcs représentent la causalité entre les variables. Ces réseaux peuvent être utilisés pour représenter la connaissance disponible sur un domaine particulier. La structure du graphe doit être définie en fonction de l'application. Elle permet de maintenir des tables de probabilité associées aux variables du réseau, qui quantifient la connaissance. Ces tables permettent notamment de modéliser l'incertitude associée aux observations (dans le cas d'un système partiellement observable). De plus, l'influence d'une observation peut être propagée sur les variables du réseau suivant la causalité entre ces variables. Ainsi, les réseaux bayésiens permettent de maintenir des croyances par rapport aux informations collectées et aux liens causaux entre les variables.

Une extension des réseaux bayésiens qui permet de représenter l'évolution temporelle des variables aléatoires en fonction d'une séquence discrète a été proposée : il s'agit des *DBN - Dynamic Bayesian Networks* [Dean et Kanazawa, 1989]. Les DBN permettent des décrire la dépendance bayésienne d'une variable s_t à l'instant t en fonction de ses variables parentes s_{t-1} à l'instant $t-1$. Ainsi, ils permettent une représentation graphique aisée des processus stochastiques discrets, contrôlés ou non. Des formalismes de décision séquentielle tels que les MDP et les POMDP factorisés (présentés plus tard) s'appuient sur les DBN pour représenter

de manière compacte et graphique les fonctions de transition et d'observation d'état [Boutilier *et al.*, 2000, Williams *et al.*, 2005].

Les *ID - Diagrammes d'Influence* [Howard et Matheson, 2005] sont une extension des réseaux bayésiens à la notion de décision. Les ID ajoutent des variables de décision $\{d_1, ...d_q\}$ aux réseaux bayésiens, et des utilités (coûts ou récompenses) sont associées aux variables de décision.

Définition 1.3.3 *Un diagramme d'influence (ID) est un modèle graphique défini sur trois ensembles de variables S , D et U organisées dans un graphe acyclique orienté, où,*

- *S est l'ensemble des variables aléatoires, représentées par des cercles. Pour chaque $x \in S$, une distribution de probabilité conditionnelle $P_{x|pa_G(x)}$ de x sachant ses parents dans le graphe est définie ;*
- *D est l'ensemble des variables de décision, représentées par des carrés. Pour chaque $x \in D$, $pa_G(x)$ est l'ensemble des variables parentes qui doivent être observées avant que la décision x soit prise. De cette façon, les arcs qui arrivent sur ces variables de décision sont les arcs informatifs, car ils définissent l'information disponible avant décision.*
- *U est l'ensemble de variables d'utilité, représentées par des diamants. Pour chaque $u \in U$, une fonction coût additive $U_{pa_G(u)}$ de domaine $pa_G(u)$ est définie. Les variables d'utilité sont des feuilles dans le graphe acyclique dirigé.*
  L'ID suppose une trajectoire $d_1 \rightarrow d_2 \rightarrow ... \rightarrow d_q$ qui contient toutes les variables de décision, où l'ordre est complètement déterminé.

L'objectif est de déterminer la valeur des variables de décision. La politique assigne une valeur aux variables de décision pour toute valeur assignée aux variables aléatoires. Afin de déterminer la politique, sa valeur est calculée par l'espérance de l'utilité. On cherchera à maximiser la valeur de la politique. Une limitation de cette approche est l'hypothèse d'une trajectoire finie $d_1 \rightarrow d_2 \rightarrow ... \rightarrow d_q$ où l'ordre est complètement déterminé, ce qui est difficile à déterminer a priori pour notre application de détection et reconnaissance de cibles. Toutefois, les diagrammes d'influence peuvent capturer les incertitudes sans l'hypothèse de contingence contrairement aux SCSP, et les décisions peuvent influencer les états du système. Malheureusement, pour l'utilisation des ID, les algorithmes d'optimisation sont moins développés que pour d'autres formalismes [Pralet *et al.*, 2010a].

Formalisme basé sur les Processus Décisionnels Markoviens Les *MDP - Processus Décisionnels Markoviens* sont un formalisme longtemps étudié pour les décisions dans des domaines stochastiques [Puterman, 1994, Sigaud et Buffet, 2008]. Dans ce modèle il est possible à tout instant d'observer l'état du système avec certitude et l'agent intelligent peut choisir une action parmi d'autres, mais les effets des actions peuvent amener à plusieurs états différents suivant un modèle de transition probabiliste en fonction de l'état de départ. Des récompenses et/ou des coûts sont associés aux paires état-action.

Définition 1.3.4 *Un MDP est un n-uplet (S, A, T, R) où :*

- *S est l'ensemble fini d'états de l'agent et de son environnement.*
- *A est l'ensemble fini d'actions.*
- *T est une fonction de probabilité de transition entre états $T : S \times S \times A \rightarrow [0, 1]$, telle que $p(s_{t+1}|s_t, a_t)$ est la probabilité conditionnelle d'atteindre l'état s_{t+1} sachant l'état s_t et l'action a_t.*
- *R est une fonction de récompense $R : S \times A \rightarrow \mathbb{R}$, telle que $r(s_t, a_t)$ est la récompense immédiate associée à la réalisation de l'action a_t dans l'état s_t.*

Les MDP sont séquentiels : ils ne s'agit pas d'un, mais de plusieurs problèmes de décision en séquence qu'un agent doit résoudre, où chaque décision courante influence la résolution du problème qui suit [Puterman, 1994, Sigaud et Buffet, 2008]. Ce caractère séquentiel des décisions se retrouve typiquement dans les problèmes de planification probabiliste, qui généralisent les approches de plus court chemin dans un environnement stochastique. L'objectif de résoudre un MDP est de trouver une politique optimale associant à tout état une action optimale. Résoudre ce problème décisionnel de Markov peut être considéré comme contrôler l'agent afin qu'il se comporte de manière optimale à long terme, c'est-à-dire, de façon à maximiser son revenu moyen accumulé. Les solutions sont des politiques, ou stratégies, ou encore règles de décision, qui spécifient l'action à entreprendre dans chaque état possible, sous l'hypothèse que l'agent possède *a priori* une connaissance parfaite du processus et de son état à tout instant. Si les observations de l'état du processus sont incomplètes, alors le modèle MDP sera insuffisant.

D'autre part, les Processus Décisionnels Markoviens Partiellement Observables (POMDP) modélisent les situations où l'agent n'a accès qu'à des informations partielles sur le processus à contrôler. Un processus décisionnel de Markov partiellement observable est un MDP dans lequel l'agent ne connaît pas l'état réel du processus : l'agent n'a accès qu'à une observation partielle et bruitée de cet état [Sigaud et Buffet, 2008]. Pour cela l'agent maintient une distribution de probabilité sur les états du système, i.e. un état de croyance, qui est mis à jour après chaque décision prise et observation reçue. Les POMDP définissent une fonction de probabilité d'observation $O : \Omega \times S \times A \rightarrow [0, 1]$, où Ω est l'ensemble fini d'observations. La fonction d'observation est telle que $p(o_t|s_t)$ est la probabilité d'observer le symbole o_t sachant l'état courant s_t.

Notons que la fonction d'observation correspond au modèle bayésien usuel de traitement du signal qui ramène les états de l'agent et de son environnement à des distributions de probabilité sur ses variables d'état. Les observations permettent à l'agent de mettre à jour son état de croyance après chaque réalisation d'une action. Ce caractère séquentiel se rapproche de celui exigé pour la perception active, où l'on a besoin d'observer afin de mettre à jour la croyance courante pour ensuite décider.

Les POMDP offrent un formalisme raisonnable, avec des variables d'états, d'actions et d'observations discrètes, pour traiter les problèmes de décision séquentielle où l'on dispose d'une observabilité partielle de l'état. L'objectif de résoudre un POMDP est de calculer une politique optimale, i.e. qui maximise le revenu moyen accumulé au cours du temps, avec ou sans horizon connu a priori. Ceci permet ainsi d'obtenir une politique projetée sur le long terme, qui détermine pour tout état de croyance une action optimale. Ce formalisme de décision correspond à ce que nous cherchons à faire, c'est-à-dire prendre des décisions afin d'acquérir des informations tout en tenant compte des buts de la mission grâce à une projection du raisonnement sur le long terme.

Des solutions exactes des POMDP ne peuvent être calculées que pour des problèmes avec un petit nombre d'états [Littman, 1997, Kaelbling *et al.*, 1998], en raison de la complexité inhérente à l'espace continu d'états de croyance. Pour des problèmes de plus grande taille, des méthodes de résolution approchées sont nécessaires [Pineau *et al.*, 2003, Smith et Simmons, 2004, Spaan et Vlassis, 2005, Ross et Chaib-Draa, 2007, Kurniawati *et al.*, 2008, Bonet et Geffner, 2009]. Le modèle POMDP et les algorithmes de résolution exacte et approximative seront détaillés au Chapitre 2.

Dans la suite nous approfondissons notre étude bibliographique sur quelques travaux où les POMDP ont été choisis comme formalisme de décision pour la robotique. Nous apporterons une attention spéciale au modèle et au critère de performance utilisés, à la prise en compte de contraintes sur la réalisation de certaines actions, ainsi qu'aux méthodes de résolution en ligne ou hors ligne éventuellement proposées pour la résolution. Cette focalisa-

tion sur ces trois points est justifiée par les caractéristiques de notre mission de détection et de reconnaissance de cibles en temps réel par un drone autonome.

1.4 Les POMDP comme formalisme de décision séquentielle dans des applications de robotique

Dans cette section nous allons aborder quelques travaux où le modèle POMDP est utilisé comme formalisme de décision. Nous cherchons dans cette section à présenter au lecteur une revue des différentes applications des POMDP robotique mobile ou aérienne : la perception active, la prise en compte de contraintes de sûreté et l'application à la détection et à la reconnaissance de cibles en ligne. Le formalisme des POMDP ainsi que les méthodes de résolution seront présentés dans le chapitre 2.

Dans [Smith et Simmons, 2004] un premier scénario réaliste mêlant à la fois des actions de prise d'information et des actions menant à l'accomplissement de la mission pour la robotique mobile a été traité. Le problème d'un robot qui cherche à ramasser des pierres dites "bonnes" ou "mauvaises" est modélisé en tant que POMDP. La contribution majeure de ce travail repose sur l'algorithme de résolution *HSVI - Heuristic Search Value Iteration*, un algorithme hors ligne de résolution approchée des POMDP. Ce type de problème de détection, dont l'objectif est d'agir de manière à acquérir des connaissances sur certaines variables d'états, peut être directement formalisé sous forme de POMDP. En effet, il s'agit de récompenser une action que mène l'agent autonome à un état désiré.

Dans [Spaan et Lima, 2009], les auteurs ont proposé un cadre de décision pour la sélection dynamique de capteurs, dans ce cas précis un réseau de caméras. Le but est de détecter et suivre des cibles d'intérêt dans un environnement de taille définie mais sans aucune connaissance a priori sur la position des cibles. Le cadre de décision choisi est celui des POMDP, qui s'avère dans cette application être une approche satisfaisante pour la modélisation et pour la décision, puisque ce modèle doit gérer de manière intrinsèque la prise d'information. En effet, la fonction de récompense a été adaptée afin d'intégrer dans celle-ci la quantité d'information apportée par différents capteurs dans chaque situation. Les mesures d'incertitude concernant les observations ont été associées aux états de capteurs, c'est-à-dire modélisées directement sur les paires état-action.

[Sridharan *et al.*, 2008] propose un nouveau modèle POMDP hiérarchisé pour la planification du traitement d'information et d'actions de détection. Cette approche permet de planifier une séquence d'actions de prise d'information afin de déterminer l'état du système. Le but est de déterminer si la scène en face du robot contient un objet d'une certaine forme et d'une certaine couleur. La scène est découpée en plusieurs régions d'intérêt, et pour chaque région un POMDP est généré suivant le type d'objet recherché (état but), et suivant le coût de traitement associé à la taille de la région d'intérêt de l'image. Une fois les sous-POMDP résolus, un autre POMDP de plus haut niveau cherche à déterminer l'ordre dans lequel les différentes régions d'intérêt seront traitées. Dans ce cas, l'objectif est de trouver l'objet recherché en appliquant la politique de plus haut niveau qui s'appuie sur les POMDP de plus bas niveau. Les auteurs démontrent que les politiques calculées avec le modèle POMDP hiérarchique présentent des résultats plus robustes que d'autres approches plus simples et réactives de planification continue du traitement visuel.

Ces travaux ont appliqué directement le modèle POMDP aux problématiques robotiques (réseau caméra ou robotique mobile) dans un cadre où la perception est un moyen pour atteindre le but de la mission. Ils démontrent que le modèle POMDP permet de gérer de manière intrinsèque l'acquisition d'information pour aboutir au but de la mission dans leurs applications, si le but de la mission peut être exprimé directement dans la fonction de récompense.

Dans la suite nous évoquerons quelques travaux où la perception active est une fin en soi, c'est-à-dire que le but est de reconnaitre l'état caché du système.

1.4.1 POMDP et perception active

Dans certains cas, l'objectif de la mission peut être directement exprimé en termes d'incertitude d'information sur l'état. Dans de tels cas, selon [Araya-López *et al.*, 2010], les POMDP peuvent ne pas être appropriés, parce que la fonction de récompense dépend de la paire état-action, et non de la connaissance de l'agent. Au lieu de cela, il semble nécessaire de s'appuyer sur un modèle de récompense qui dépend de l'état de croyance. L'état de croyance du POMDP fournit l'expressivité nécessaire pour ces problèmes, pourtant, il n'y a pas beaucoup de recherches sur des algorithmes pour résoudre les POMDP pour ce nouveau modèle de récompense basé sur les états de croyance, et non directement sur les états. Les modèles sont généralement obligés de s'adapter au cadre POMDP classique, ce qui implique de changer le modèle de récompense initial [Spaan et Lima, 2009]. On pourrait affirmer que l'acquisition d'information est toujours un moyen, pas une fin, et donc, un problème bien défini de décision avec observabilité partielle doit toujours être modélisé comme un POMDP normal. Toutefois, dans un certain nombre de cas, le concepteur décide de séparer la tâche de recherche d'informations de celle de l'exploitation des informations.

Pour ce genre de problème, [Araya-López *et al.*, 2010] propose une extension du modèle POMDP à des fonctions de récompense basées sur l'état de croyance. Le but est de formaliser un nouveau type de fonction de récompense ρ qui ne dépend plus seulement de la paire état-action. La fonction de récompense est aussi associée à une mesure d'information de l'état de croyance. La convexité de la fonction de valeur du POMDP pour un horizon h est démontrée pour le nouveau modèle de récompense, sous la condition que ρ est une fonction convexe. Aussi, les auteurs démontrent qu'il est possible d'approximer ρ correctement avec une fonction linéaire par morceaux et convexe (PWLC), et d'utiliser des algorithmes de résolution exacts ou approchés des POMDP après des modifications mineurs. Des résultats expérimentaux très récents [Araya López, 2013] montrent qu'une mesure d'information linéaire par morceaux permet d'obtenir des performances compétitives voire meilleures que des mesures non-linéaires telles que l'entropie de Shannon. Ces résultats indiquent implicitement qu'il existe une modélisation sous forme de POMDP classique (récompenses définies sur les états et non sur la croyance) permettant de résoudre des problèmes de perception active pure, ce qui corrobore nos études menées dans le chapitre 4.

Donnons un autre exemple de cas où le modèle POMDP est utilisé pour formaliser le problème de perception active [Eidenberger *et al.*, 2009] déjà discuté précédemment. Le critère d'optimisation est basé sur la théorie de l'information et les coûts des actions. Le problème est optimisé en ligne avec un horizon très court. Cette approche de résolution n'utilise pas des algorithmes de l'état de l'art des POMDP, puisque les distributions de probabilité sont représentées par des ensembles de gaussiennes, et parce que le critère d'optimisation n'est plus linéaire par morceaux.

[Candido et Hutchinson, 2011] abordent également le problème d'acquisition d'information avec des POMDP continus s'adressant à la planification de mouvements d'un robot avec des contraintes d'évitement d'obstacle. L'approche optimise une fonction objectif qui considère la probabilité de collision avec un obstacle et la distance entre l'estimation d'état actuelle et le but à rejoindre. Des heuristiques basées sur la théorie de l'information sont utilisées afin de : (1) calculer des politiques pour la minimisation d'erreur d'estimation ; (2) calculer des politiques pour approcher au mieux le but. Ces heuristiques fournissent des politiques locales, c'est-à-dire des politiques qui ne peuvent être appliquées que sur un horizon de temps donné. Le problème d'optimisation est de trouver une loi de passage entre les politiques locales. Pour cela, une méthode approximative basée sur le filtrage particulaire

est proposée. Cette méthode estime la valeur de la fonction objectif de la politique locale courante et les coûts pour aller au but de la prochaine politique locale à chaque instant de passage, puis choisit la prochaine politique qui minimise la probabilité de collision et la distance au but. Les résultats présentés mettent en évidence le gain d'information obtenu par l'utilisation des plans favorisant la prise d'information tout en atteignant des configurations très proches du but. Par contre, même si la méthode vise une l'application en temps réel, le temps nécessaire pour les calculs ne le permet pas encore.

Une autre problématique souvent rencontrée dans des applications robotiques auxquelles nous nous intéressons, comme la détection et la reconnaissance de cibles par un hélicoptère autonome, est relative aux contraintes de sûreté liées au vol, par exemple imposées par les agences de régularisation. Dans le but de réviser les méthodes utilisées pour la prise en compte de ces contraintes dans le cadre POMDP, nous présentons dans la section suivante quelques travaux qui traitent du sujet.

1.4.2 POMDP et contraintes de sûreté

Dans de nombreuses applications réalistes, le concepteur des systèmes autonomes est souvent confronté à des contraintes physiques liées aux limites dynamiques de l'agent autonome ainsi que des contraintes de sûreté. Le plan doit garantir le respect de ces contraintes pour la protection des opérateurs et de l'agent robot contre un danger physique réel. Très peu de travaux se sont intéressés à ce genre de problématique en planification probabiliste.

Dans les Processus Décisionnels Markoviens (MDPs), l'approche classiquement utilisée pour contraindre certaines actions dans certains états est d'associer une grande pénalité (quantitative) aux actions indésirables en espérant que l'optimisation du processus conduise à proscrire ces actions de la politique optimale. Toutefois, des préconditions booléennes ont été introduites récemment de manière formelle [Younes et Littman, 2003] au cadre MDP, afin de produire des politiques contenant seulement des actions faisables ou désirables pour chaque état atteignable. Ceci a permis la conception d'algorithmes efficaces qui évaluent directement des préconditions booléennes au moment de de l'optimisation [Buffet et Aberdeen, 2009, Teichteil-Königsbuch *et al.*, 2010, Kolobov *et al.*, 2011].

Dans [Miller *et al.*, 2009] un modèle POMDP continu, i.e. avec des variables d'état et d'action continues, est utilisé pour la navigation d'un *UAV – Unmanned Aerial Vehicle* dans le but de suivre des cibles terrestres mobiles. La contribution est centrée sur une nouvelle méthode de résolution approximative de POMDP continu, appellé *NBO – nominal belief-state optimization*. Les limitations dynamiques apparaissent sous la forme des contraintes sur les mouvements de l'UAV, qui reposent sur un intervalle borné de l'accélération latérale de l'UAV. Les bornes ne changent pas pendant la mission, et la politique optimisée garantit le respect de ces limitations. Il est important de noter que le respect de cette contrainte dynamique est direct puisque pendant l'optimisation de la politique les actions dites dangereuses ne font pas partie de l'ensemble des actions applicables quelque soit l'état de l'UAV. Ce qui n'est pas le cas du problème auquel nous sommes confrontée, où, le fait qu'une zone puisse être considéré comme "non atterrissable" parmi d'autres qui sont "atterrissables" ne doit pas empêcher la présence de l'action "atterrir" dans le modèle.

Dans [Bai *et al.*, 2011] l'évitement des collisions entre avions est modélisée comme un POMDP continu pour générer automatiquement la politique de résolution de menace du système d'anti-collision. Cette problématique traite la contrainte de collision dans le modèle POMDP par des pénalités très importantes associées aux états de collision en espérant que l'optimisation du processus conduise à les éviter. Les actions de manœuvre de l'avion pour l'évitement sont aussi pénalisées. La contribution de ce papier est l'application de l'algorithme *MCVI - Monte Carlo Value Iteration* qui est capable de générer un graphe de politique directement à partir du POMDP continu sans une discrétisation de l'espace d'état et de

l'espace d'action. Les auteurs démontrent que leur approche a permis de réduire le risque de collision. On note que l'approche ne peut pas garantir que la collision sera évitée avec certitude. Pour différents coûts associés aux manœuvres, l'approche démontre le résultat suivant : plus la manœuvre d'évitement est coûteuse, plus le risque de collision augmente. Ce qui nous permet de conclure que l'évitement d'actions dangereuses doit être modélisé autrement que par une manipulation des coûts de type potentiomètre.

Dans [Pralet *et al.*, 2010b] la synthèse de contrôleur sûr dans des domaines non-déterministes et partiellement observables est traitée. Le contrôleur sûr doit optimiser des politiques qui garantissent qu'aucune action non réalisable ne sera exécutée. Ce contrôleur devra aussi garantir qu'aucune trajectoire menant au but n'atteindra des états indésirables. Dans ce cadre la politique est une fonction $\pi : o \rightarrow a$, c'est-à-dire, qui définit une action pour chaque observation. Les préconditions des actions sont définies selon une relation de faisabilité état-action. Le caractère partiellement observable indique que certaines variables d'états ne sont pas observables, mais d'autres sont complètement observables. Les observations renseignent avec certitude sur les variables d'état observables. Comme l'observation vient avant la décision, il est possible de définir une action faisable en s'appuyant sur l'observation reçue, étant donné que cette observation renseigne avec certitude sur les variables d'état clés pour la vérification de la faisabilité.

Dans [Teichteil-Königsbuch, 2012], un nouveau modèle théorique appelé *Path-Constrained MDP* est proposé afin de permettre au concepteur d'optimiser des politiques dont les chemins d'exécution sont *sûrs* au sens d'une logique temporelle du premier ordre. L'algorithme proposé, basé sur la programmation linéaire itérative, évalue la sûreté de la politique sur les chemins stochastiques possibles. Les chemins stochastiques peuvent être vus comme une chaîne de Markov et la politiques optimisée peut ainsi être validée avec des logiques temporelles probabilistes de type *PCTL - Probabilistic Real Time Tree Logic* [Hansson et Jonsson, 1994].

En raison de la mise en place croissante de systèmes de drones dans des missions civiles et militaires, la détection et la reconnaissance de cibles par des véhicules aériens inhabités (UAV) autonomes est aujourd'hui un domaine de recherche actif. En raison de la complexité du problème à traiter, la résolution en ligne est rarement envisagée alors que le problème n'est connu qu'en cours de mission. Nous présenterons dans la suite quelques travaux qui se sont intéressés à l'application en temps réel de politiques pour la problématique de détection et reconnaissance de cibles en ligne.

1.4.3 POMDP pour l'application à la détection et reconnaissance de cibles en ligne

Généralement, dans des missions de détection et de reconnaissance de cibles, la stratégie de décision de haut niveau est donnée par une règle écrite à la main (survoler une zone donnée, atterrir, prendre une image, etc), qui dépend des événements stochastiques (détection d'une cible dans une zone donnée, cible reconnue), qui peuvent survenir lors de l'exécution de la règle de décision. Contrairement à ce que nous voulons faire, nous nous intéressons à optimiser en ligne la politique pour ce genre d'application.

[Wang *et al.*, 2012] présente un cadre composé de détection, reconnaissance et suivi de cibles mobiles basé sur des vidéos capturées à partir de différentes altitudes de vol de l'UAV. Les différentes problématiques (détection, reconnaissance et suivi) sont découplées, c'est-à-dire, qu'elles ne sont pas traitées dans un même formalisme. Chaque problématique est gérée par un module dans l'architecture de l'UAV. Les auteurs affirment que grâce à ce cadre découplé, les cibles mobiles peuvent être suivies en temps réel.

Peu de systèmes de drones construisent et optimisent des politiques en ligne, a cause de la complexité de la construction automatique des ces règles de décision sous incertitude [Sabbadin *et al.*, 2007]. Comme déjà discuté dans [Miller *et al.*, 2009,Schesvold *et al.*, 2003,Bai

et al., 2011], les approches présentées ont pour vocation d'être appliquées en temps réel mais elles ne sont pas encore viables, ni embarquées. De plus, dans ces applications, le problème POMDP est supposé connu avant que la mission ne commence, permettant aux concepteurs d'optimiser la politique du drone hors-ligne sans contrainte de temps. Ce n'est clairement pas notre cas, car nous ne connaissons ni la topologie des zones, ni le nombre de cibles à détecter.

Toutefois, dans une mission de détection et de reconnaissance [Wang *et al.*, 2012], certains paramètres du problème sont inconnus avant le vol. En effet, le nombre de cibles ou le nombre de zones qui composent l'environnement, sont des exemples de paramètres généralement inconnus avant le vol et qui doivent être automatiquement extraits au début de la mission afin de définir le problème de décision séquentielle à optimiser. Pour un traitement réaliste, la problématique en soi impose donc une planification en temps réel et en environnement incertain.

Dans la suite nous présentons la conclusion de cette étude bibliographique, la justification de notre choix du modèle POMDP comme formalisme de décision pour notre application, ainsi que les pistes de recherche explorées lors de cette thèse.

1.5 Conclusion et intuitions

Deux contraintes nous sont imposées pour le problème général de la perception active : l'une est relative au traitement du signal, qui ramène les états de l'agent et de son environnement à des distributions de probabilité autour de ses variables d'état, et l'autre est liée à des incertitudes sur les effets des actions de l'agent autonome. Pour traiter les incertitudes probabilistes liées aux capteurs, aux actions de l'agent robot et de l'environnement, plusieurs problèmes de décision en robotique peuvent être modélisés sous forme d'un Processus Décisionnel Markovien Partiellement Observable (POMDP).

De plus, le problème décisionnel traité ici exige un compromis entre une planification court terme associée à la perception de l'environnement, et une planification long terme associée à l'accomplissement de la mission. Il est nécessaire de pouvoir prendre en compte dans un même cadre formel la planification des actions de l'agent visant la réalisation des objectifs qui lui sont assignés (mission), et des actions visant à recueillir l'information nécessaire à la mise en œuvre de ces actions (perception). Si les actions suivent un plan conditionnel (une politique), le problème est souvent modélisé comme un POMDP. Ceci peut être défini pour un horizon de raisonnement fini ou infini. Il existe une récompense associée à chaque état et à chaque action prise dans cet état. L'objectif étant de trouver les actions (politique optimale) qui maximisent la récompense tenant compte de leurs effets à long terme.

Ces arguments justifient le choix des Processus Décisionnels Markoviens Partiellement Observables comme modèle de l'agent et de son environnement pour le problème d'optimisation traité dans ce sujet de thèse. Ce modèle sera formellement défini dans le chapitre 2, ainsi que les algorithmes de résolution disponibles dans la littérature.

Le modèle POMDP semble être une bonne approche pour traiter le problème de la perception active, puisque celui-ci peut fournir une politique optimale à long terme, ou quasi-optimale si la méthode de résolution est une méthode approchée. Par contre, pour les applications purement épistémiques, où l'on cherche à identifier l'état caché du système, [Araya-López *et al.*, 2010, Eidenberger *et al.*, 2009, Candido et Hutchinson, 2011] démontrent qu'il est possible d'adapter le critère de performance classique des POMDP afin de tenir compte d'une mesure d'incertitude liée à la méconnaissance de l'état du système par l'agent. Les travaux présentés précédemment traitent de nouveaux critères de performance dans un cadre POMDP, toutefois [Candido et Hutchinson, 2011, Eidenberger *et al.*, 2009] ne prennent pas en compte dans le calcul de la politique les effets à long terme des actions : horizon limité

dans le calcul de politiques locales, ou politique aveugle (h=1).

Une première piste de recherche dans notre thèse a donc été d'étudier le compromis entre la prise d'information et la décision dans deux cadres applicatifs, à partir d'une implémentation d'un critère mixte pour les POMDP. Notons qu'une étude similaire a été menée dans la thèse de [Araya López, 2013] en parallèle de notre thèse. Alors que la thèse de [Araya López, 2013] a étudié différentes mesures d'incertitude pour la perception pure, nous avons plutôt étudié l'intérêt de combiner une mesure d'incertitude entropique à des récompenses classiques définies sur les états. Ainsi, nous proposons un critère mixte qui fournit des politiques (quasi-)optimales à long terme contrairement aux politiques réactives de [Eidenberger *et al.*, 2009]. D'autre part, on pourrait affirmer que l'acquisition d'information est toujours un moyen, pas une fin, et donc qu'un problème bien défini de décision avec observabilité partielle doit être modélisé sous forme d'un POMDP classique [Spaan et Lima, 2009]. Nous pensons en effet que si l'on ajoute au modèle des états buts fictifs (au moyen d'actions de classification ou d'une action d'atterrissage), un tel critère mixte basé sur une mesure de l'incertitude de l'état de croyance ne serait plus nécessaire dans de nombreux cas pratiques (y compris en perception active pure). Un tel critère mixte permettrait en fait d'ajuster les récompenses d'un modèle POMDP classique (critère non mixte) équivalent. Ces deux approches seraient donc, en pratique, complémentaires. Ces études sont déclinées dans le chapitre 4 de cette thèse.

D'autre part, la prise en compte des contraintes de sûreté dans des applications utilisant le modèle POMDP est actuellement très limitée. Comme déjà discuté, dans les Processus Décisionnels Markoviens (MDPs), des préconditions booléennes ont été introduites de manière formelle [Younes et Littman, 2003], dans le but de produire des politiques contenant seulement des actions faisables ou désirables pour chaque état. À notre connaissance, l'utilisation formelle de préconditions n'a jamais été adaptée pour des Processus Décisionnels de Markov *Partiellement Observables*, en dépit des besoins pratiques encore plus marqués : l'observabilité partielle de l'environnement de l'agent robot impose de prendre de précautions accrues quant aux conséquences à long terme en termes de sûreté des actions exécutées. La vérification à l'optimisation et à l'exécution des préconditions n'est immédiate que lorsque l'état du système est complètement observable. Dans des systèmes partiellement observables, la vérification de l'applicabilité d'une action n'est pas une tâche facile [Pralet *et al.*, 2010b] : il s'agit, à l'optimisation comme à l'exécution, d'inférer efficacement à partir du modèle un ensemble d'actions applicable dans un état de croyance donné. La recherche sur les POMDP reste très focalisée sur les moyens d'améliorer l'efficacité des algorithmes visant à produire une politique pour le modèle POMDP standard (discutées dans le chapitre 2), plutôt que sur des améliorations permettant de confronter le modèle standard aux applications réelles.

En conséquence, une seconde piste de recherche, traitée dans le chapitre 5 de cette thèse, a été de formaliser la prise en compte des préconditions dans un cadre POMDP. Les préconditions permettent de tenir compte des contraintes de sûreté liées à la mission robotique, contrairement à [Bai *et al.*, 2011], où la modélisation d'une infaisabilité sur les actions est faite par un réglage de coûts de type potentiomètre, c'est-à-dire, en associant un coût très élevé associé aux paires état-action indésirables.

De plus, très peu de travaux se sont intéressés à la résolution *en ligne* du problème de détection et reconnaissance de cibles [Wang *et al.*, 2012] par un UAV autonome, ce qui est abordé dans une troisième piste de recherche. Dans [Carvalho Chanel *et al.*, 2012c, Carvalho Chanel *et al.*, 2013] nous avons étudié une mission de détection et reconnaissance de cible par un UAV autonome, modélisée par un POMDP défini en ligne une fois que le nombre de zones à explorer a été observé en début de mission. Nous croyons que ce travail présente l'une des premières implémentations d'une mission de détection et de reconnaissance de cibles basée sur un modèle de type POMDP, qui est optimisé en ligne par une approche original.

Celle-ci consiste à exécuter et optimiser en parallèle la politique sur des états de croyance futurs probables. L'expérimentation en vol de cette approche est discuté dans le chapitre 6 de cette thèse.

Chapitre 2

Processus Décisionnels de Markov Partiellement Observables - POMDP

Ce chapitre présente une revue des Processus Décisionnel Markoviens Partiellement Observables (POMDP). Les POMDP sont un cadre formel de décision séquentielle sous incertitude avec observabilité partielle des états du monde. Ce cadre est bien adapté à un grand nombre de problèmes réalistes où la décision est nécessaire malgré les incertitudes et l'accès limité aux états du système. Les POMDP supposent généralement des modèles complets et précis, avec une dynamique de transition stochastique pour les états, un modèle de suivi d'état imparfait, et une structure de récompenses. Le but est de trouver une stratégie optimale pour décider dans le "monde" modélisé. Nous présenterons le modèle POMDP, les différents algorithmes de résolution exacte et approximative de POMDP qui ont été proposés et les extensions pour des modèles factorisés.

2.1 Rappel du cadre formel des POMDP

Formellement, un POMDP est défini comme un n-uplet $\langle S, A, \Omega, T, O, R, b_0 \rangle$ où :

S : est un ensemble d'états ;

A : est un ensemble d'actions ;

Ω : est un ensemble d'observations ;

$T : S \times A \times S \to [0; 1]$: est une fonction de transition entre les états, où pour $\forall a_t \in A, \forall s_t \in S$, et $\forall s_{t+1} \in S$ on définit :

$$T(s, a, s') = Pr(s_{t+1} = s' \mid a_t = a, s_t = s), \forall t \tag{2.1}$$

comme la probabilité de transition de l'état s à l'état s' si l'action a est réalisée, quelque soit le triplet (s, a, s'). Sachant que T est une probabilité conditionnelle, nous avons : $\sum_{s' \in S} T(s, a, s') = 1, \forall (s, a)$;

$O : \Omega \times S \times A \to [0; 1]$: est une fonction d'observation, où pour $\forall o_{t+1} \in \Omega, \forall a_t \in A$, et $\forall s_{t+1} \in S$, on définit :

$$O(o, s', a) = Pr(o_{t+1} = o | s_{t+1} = s', a_t = a), \forall t \tag{2.2}$$

comme la probabilité d'observer o sachant que l'on a atteint l'état s' après l'exécution de a. Cette probabilité conditionnelle est définie pour tout triplet (o, s', a), et nous avons : $\sum_{o \in \Omega} O(o, s', a) = 1, \forall (s', a)$;

$R : S \times A \to \mathbb{R}$: est une fonction de récompense associée aux couples état-action $r(s, a)$. Cette fonction de récompense attribue une valeur qui quantifie l'utilité d'exécuter a quand l'état est s.

b_0 est une distribution de probabilité sur les états initiaux, appelée *état de croyance*. Nous avons :

$$b_0(s) = Pr(s_0 = s), \tag{2.3}$$

qui donne la probabilité que le système soit dans l'état s à l'instant $t = 0$. Cette distribution de probabilité est définie pour tout $s \in S$.

Comme exemple d'un POMDP, nous allons décrire le *Tiger Problem* [Cassandra, 2005] (voir figure 2.1). Ce problème consiste à choisir d'ouvrir la porte qui ne contient pas le tigre, sachant que l'agent ne connait pas l'état réel du système (il ne sait pas où le tigre se trouve). L'agent peut seulement écouter à chaque porte, ce qui fait office d'observation.

Il dispose de 3 actions, écouter, ouvrir la porte à gauche ou ouvrir la porte à droite. Chaque fois que l'agent choisit d'écouter, cette action a un coût ; si l'agent choisit d'ouvrir une des portes et que le tigre se trouve derrière cette porte, l'agent perd ; s'il ouvre la bonne porte il gagne. L'action d'ouvrir une des portes ramène à un état terminal. L'agent gagne le défi s'il choisit d'ouvrir la porte qui contient le cadeau.

Cet exemple représente un processus décisionnel de Markov partiellement observable, car l'agent ne connaît pas l'état réel du système $s_t \in S$, mais doit prendre des décisions en se basant sur son état de croyance $b_0, ..., b_t$. L'état de croyance est mis à jour à tout instant et tient compte de toutes les observations reçues $o_{t+1} \in \Omega$ et de toutes les actions réalisées $a_t \in A$. Dans cet exemple, les actions de l'agent ne modifient pas l'état réel du système.

2.1.1 Principe

A chaque instant t, l'agent n'a pas accès à l'état courant $s \in S$, mais peut seulement le percevoir partiellement sous forme d'une observation $o \in \Omega$, donnée par la fonction d'observation O. Quand il applique une action $a \in A$, il modifie l'état du processus avec une probabilité $p(s' \mid s, a)$ de l'amener dans l'état s'. Ensuite, l'agent ne perçoit que l'observation o qui dépend de la fonction $p(o \mid s', a)$. Pour chaque action appliquée, l'agent reçoit une récompense $r \in R$.

Il est à noter que même quand la fonction d'observation $O()$ est déterministe, c'est-à-dire qu'à chaque état n'est associée qu'une et une seule observation, l'agent peut ne pas connaître

FIGURE 2.1 – *Problème du tigre.*

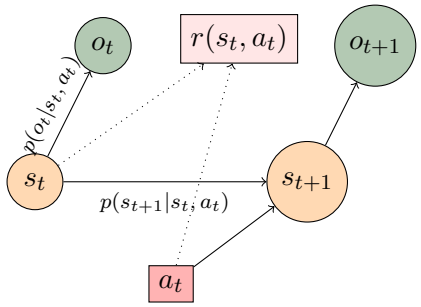


FIGURE 2.2 – Modèle de transition du POMDP.

l'état, car deux états peuvent être associés à une même observation. C'est d'ailleurs quand il y a ambiguïté sur les états qu'il est intéressant ou nécessaire d'utiliser les POMDP.

2.1.2 État de croyance

L'agent choisit ses actions en fonction des informations qui lui sont disponibles, mais une seule observation ne permet pas d'identifier directement l'état s du système. C'est pourquoi une certaine forme de mémoire est exigée. Ainsi, l'agent peut maintenir un historique de toutes les observations et actions exécutées. Nous définissons l'historique à l'instant t tel que :

$$h_t := \{a_0, o_1, ..., o_{t-1}, a_{t-1}, o_t\} \tag{2.4}$$

La taille de cet historique peut être très important en fonction du temps, car il capitalise l'ensemble des actions réalisées et des observations reçues. De plus, mémoriser tous les historiques possibles devient très compliqué car ce type de mémoire est exponentiel en l'horizon et le nombre d'observations. Il est connu que l'historique n'a pas besoin d'être représenté de manière explicite pour la sélection des actions, il suffit comme condition pour la sélection des actions de s'appuyer sur l'état de croyance courant [Smallwood et Sondik, 1973]. Ce résultat clé démontre qu'utiliser les états de croyance pour la définition des politiques optimales apporte autant d'information que l'utilisation d'un historique entier d'actions prises et des observations reçues [Smallwood et Sondik, 1973, Sigaud et Buffet, 2008].

Par conséquent, avec les modèles de transition d'état et d'observation, un POMDP peut être transformé en un MDP sur l'état de croyance [Sigaud et Buffet, 2008] : l'agent synthétise toute l'information disponible de son passé en utilisant un état de croyance $b(s)$, qui est une distribution de probabilité à posteriori :

$$b_t(s) := Pr(s_t = s|o_t, a_{t-1}, o_{t-1}, ..., a_0) \tag{2.5}$$

L'état de croyance peut être de calculé de manière récursive en utilisant l'état de croyance b_{t-1}, l'action a_{t-1} et l'observation o_t, c'est-à-dire qu'il doit être mis à jour après chaque action réalisée et chaque observation reçue. Si pour un état s donné, à t, et un état futur s' à $t+1$, on réalise l'action a et on reçoit l'observation o, on peut alors calculer un nouvel état de croyance à $t+1$ en appliquant la règle de Bayes, cela en utilisant les fonctions de transition et d'observation. Ainsi :

$$
\begin{aligned}
b_o^a(s') &= p(s' \mid b, a, o) \\
&= \frac{p(o, s' \mid b, a)}{p(o \mid b, a)} \\
&= \frac{p(o \mid s', a)p(s' \mid b, a)}{p(o \mid b, a)} \\
&= \frac{p(o \mid s', a)p(s' \mid b, a)}{\sum_{s,s'} p(o \mid b, a, s, s')p(s, s' \mid b, a)} \\
&= \frac{p(o \mid s', a) \sum_{s \in S} p(s' \mid s, a)b(s)}{\sum_{s' \in S} p(o \mid s', a) \sum_{s \in S} p(s' \mid s, a)b(s)}
\end{aligned} \tag{2.6}
$$

Les états de croyance forment un processus markovien : l'état de croyance à $t+1$ dépend seulement de l'état de croyance, de l'action, et de l'observation à t. C'est-à-dire, que l'état de croyance à $t+1$ synthétise toute l'information disponible jusqu'à l'instant t pour le contrôle [Smallwood et Sondik, 1973]. Le POMDP ainsi défini sur les états de croyance est difficile à résoudre car l'espace de b est défini sur l'espace continu de distributions de probabilités sur les états appartenant à l'ensemble S. De plus, ceci est un cas spécial d'un processus de Markov à état continu, où l'état est continu mais les probabilités de transition et d'observation d'état sont discrètes [Smallwood et Sondik, 1973].

Il est intéressant de considérer la nature de la distribution de l'état de croyance. L'état de croyance est défini sur un simplex Δ qui représente l'ensemble des distributions de probabilité sur les états, appelé aussi espace (continu) d'états de croyance sur l'espace d'état S. La représentation vectorielle de l'état de croyance est la répresentation la plus intuitive, étant donné que b peut être un vecteur sur un espace de taille $|S|$, et chaque composante de ce vecteur $b_t(s_i)$ représentera la probabilité d'être dans l'état s_i :

$$b_t(s_i) = Pr(s = s_i | a_0, o_1, ..., a_{t-1}, o_t).$$

Pour des espaces d'états très grands, calculer la mise à jour de l'état de croyance peut être très coûteux. Plusieurs travaux [Williams *et al.*, 2005, Sim *et al.*, 2008a, Ong *et al.*, 2009] se sont intéressés à explorer la structure de l'état de croyance pour aboutir à une représentation factorisée de l'état de croyance afin de développer des techniques plus efficaces pour la mise à jour.

Toutefois, calculer la politique optimale exacte ou approchée est une tâche encore plus difficile [Papadimitriou et Tsitsiklis, 1987], non seulement en raison de la taille de l'espace d'états, mais surtout de la taille de l'espace d'observations. L'optimisation de la politique du POMDP est réalisée par une recherche en largeur dans l'espace d'états de croyance. En commençant par l'état de croyance initial on énumère les différents aléas possibles, ce qui fait que le nombre d'états de croyance, qui dépendent de l'historique de paires action-observation, augmentera exponentiellement avec l'horizon de planification. Ces aspects seront discutés plus en détaille dans la section suivante.

2.1.3 Politique et fonction de valeur

L'objectif de l'agent est de choisir des actions qui permettront d'accomplir au mieux sa mission, d'où la nécessité de calculer une politique optimale. Une politique markovienne déterministe $\pi(b)$ est une fonction qui spécifie une action a pour chaque état de croyance, $\pi(b) : b \mapsto a$.

Pour un état de croyance b, et un horizon fini N (voir figure 2.3 pour un horizon $N = 2$), on construit un arbre dont la racine est un état de croyance b donné (plan conditionnel),

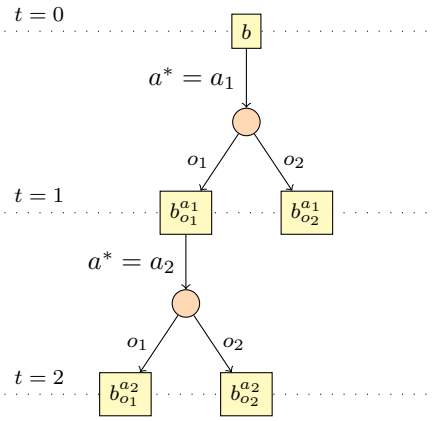


FIGURE 2.3 – Plan conditionnel pour un état de croyance b.

et à chaque étape du plan, un nœud action (cercles dans la figure 2.3) contient une action à effectuer. En fonction de l'observation reçue, on se déplace vers un nœud qui représente l'état de croyance obtenu (rectangles dans la figure 2.3).

Pour un horizon infini, soit un horizon de temps assez grand pour que la politique soit considérée stationnaire, l'objectif est de maximiser l'espérance de la somme décomptée de récompenses. Ou, si r définit un coût, l'objectif devient de minimiser le coût total espéré. A noter que, contrairement aux MPD, la politique $\pi(b)$ est une fonction définie sur l'ensemble continu de distribution de probabilité sur les états.

Une politique π peut être caractérisée par une fonction de valeur, $V^\pi(b)$, qui est définie par l'espérance des revenus totaux pondérés en fonction du temps, que l'agent recevra s'il suit la politique π en partant de b. La fonction de valeur réalise une projection d'un état de croyance sur une valeur dans $\mathbb{R}$. L'équation 2.7 met en évidence le critère γ-pondéré classiquement utilisé.

$$V^\pi(b) = E_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(b_t, \pi(b_t)) \middle| b_0 = b \right] \tag{2.7}$$

où γ est le facteur d'actualisation, $0 \leqslant \gamma < 1$, et :

$$r(b_t, \pi(b_t)) = \sum_{s \in S} r(s, \pi(b_t)) b_t(s) \tag{2.8}$$

Notons que dans la définition générale de la récompense, $r(b_t, \pi(b_t))$ est une espérance de gain par rapport à l'état de croyance. Le principal intérêt du facteur d'actualisation est d'assurer la convergence de la série en horizon infini [Sigaud et Buffet, 2008]. De plus, il représente la probabilité que le processus continue à chaque pas de temps.

Une politique π qui maximise V^π est appelée *politique optimale* π^*. Il est démontré qu'il existe une politique markovienne optimale, c'est-à-dire qui spécifie pour chaque b l'action optimale à effectuer à chaque étape, en supposant que l'agent agira de manière optimale dans les étapes suivantes.

La valeur d'une politique optimale π^* est définie par la fonction valeur optimale V^*, qui satisfait l'équation d'optimalité de Bellman, $V^* = \mathcal{L}V^*$:

$$V^*(b) = \max_{a \in A} \left[\sum_{s \in S} r(s, a) b(s) + \gamma \sum_{o \in \Omega} p(o \mid a, b) V^*(b^o_a) \right], \tag{2.9}$$

où, b^o_a étant donné par 2.6, représente l'état de croyance futur si l'on applique a et que l'on reçoit l'observation o. L'opérateur $\mathcal{L}$ est l'opérateur dynamique de Bellman. On garantit que la solution est optimale quand 2.9 converge pour tout b [Sigaud et Buffet, 2008].

Les développements théoriques qui ont été réalisés sur les différents critères pour les MDP [Puterman, 1994, Sigaud et Buffet, 2008] sont utilisables de la même manière et sont définis sur les états de croyance, dont l'opérateur d'itération de la fonction valeur, qui utilise l'équation de Bellman. Cette méthode très connue pour le calcul de la politique optimale s'appuie sur des itérations de type programmation dynamique [Smallwood et Sondik, 1973], qui consistent à faire converger la valeur progressivement vers le point fixe de l'équation 2.9 pour chaque état de croyance. Soit V une fonction de valeur, l'on peut initialiser la fonction de valeur par :

$$V_0(b) = \max_{a \in A} \sum_{s \in S} b(s) r(s,a), \tag{2.10}$$

et, l'on peut calculer la fonction de valeur pour chaque instant t en s'appuyant sur la fonction de valeur à $t-1$ par la propriété récursive de l'équation :

$$V_t(b) = \max_{a \in A} \left[\sum_{s \in S} r(s,a) b(s) + \gamma \sum_{o \in \Omega} p(o \mid a, b) V_{t-1}(b_a^o) \right]. \tag{2.11}$$

Cette mise à jour de la fonction de valeur maximise l'espérance de tous les gains futurs que l'agent recevra à la prochaine étape pour tout état de croyance b. La politique optimale à t peut être extraite directement de la fonction de valeur à l'étape précédente :

$$\pi^*(b) = \arg\max_{a \in A} \left[\sum_{s \in S} r(s,a) b(s) + \gamma \sum_{o \in \Omega} p(o \mid a, b) V_{t-1}(b_a^o) \right]. \tag{2.12}$$

En pratique, la solution exacte d'un POMDP à horizon infini est souvent difficile à calculer [Papadimitriou et Tsitsiklis, 1987]. La solution exacte d'un POMDP ne peut être calculée que pour des problèmes avec un petit nombre d'états [Cassandra, 1998]. Pour cela, les chercheurs se sont intéressés à développer des algorithmes qui approchent la solution optimale [Cassandra, 1998, Pineau *et al.*, 2003, Smith et Simmons, 2004, Spaan et Vlassis, 2005, Smith et Simmons, 2005, Kurniawati *et al.*, 2008].

Fonction de valeur linéaire par morceaux

En travaillant avec les états de croyance, il est possible d'exploiter des propriétés particulières de la fonction de valeur afin d'obtenir des algorithmes plus efficaces.

La fonction de valeur optimale pour un problème à horizon fini est linéaire par morceaux et convexe (PWLC) [Smallwood et Sondik, 1973], et pour un horizon infini V^*, elle peut être aussi approchée par une fonction de valeur PWLC. Ceci est dû au fait qu'on peut paramétrer la fonction de valeur V_n par un ensemble fini de vecteurs (hyperplans) $\left\{\alpha_n^i\right\}, i = 1, ..., |V_n|$ [Cassandra, 1998]. De plus, chaque vecteur définit une région de l'espace des états de croyance, où ce vecteur représente la valeur maximale de V_n (voir figure 2.4). Les α-vecteurs forment une partition de l'état de croyance. La convexité et la linéarité par morceaux de la fonction de valeur découle du fait que V_n réalise le maximum des valeurs des actions (équation 2.11), qui sont individuellement linéaires (représentées par des α-vecteurs) sur l'espace des états de croyance.

La fonction de valeur d'un b donné à l'instant n paramétrée par des α-vecteurs peut ainsi être définie par l'existence d'un ensemble d'α-vecteurs Γ_n tel que :

$$V_n(b) = \max_{\alpha_n^i \in \Gamma_n} \sum_{s \in S} b(s) \alpha_n^i(s), \text{ ou sous forme vectorielle} \tag{2.13}$$

$$V_n(b) = \max_{\alpha_n^i \in \Gamma_n} b \cdot \alpha_n^i \tag{2.14}$$

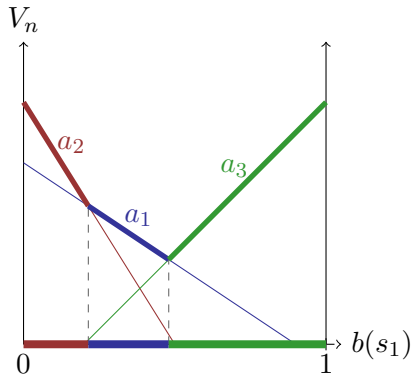


FIGURE 2.4 – α-vecteurs pour un problème à 2 états et 3 actions.

où $(\cdot)$ dénote le produit scalaire usuel. Pour un b donné, le gradient de la fonction de valeur est induit par le vecteur :

$$\alpha_n^b = \arg\max_{\alpha_n^i \in \Gamma_n} b \cdot \alpha_n^i \tag{2.15}$$

et la politique $\pi(b)$ par l'action associée à ce vecteur $a(\alpha_n^b)$, telle que : $\pi(b) = a(\alpha_n^b)$.

2.2 Itération exacte sur la fonction de valeur

Dans le cadre des POMDP, l'itération sur la fonction de valeur consiste à approcher V^* par l'application itérée de l'opérateur de programmation dynamique $\mathcal{L}$ à partir d'une fonction de valeur initiale V_0 supposée linéaire par morceaux et convexe. Pour l'opérateur $\mathcal{L}$, le produit intermédiaire estimé $(V_1, V_2, ...)$, sera aussi linéaire par morceaux et convexe [Smallwood et Sondik, 1973].

Des algorithmes qui manipulent directement des α-vecteurs afin de calculer les fonctions de valeur intermédiaires du problème ont été proposés dans la littérature. Par exemple, l'algorithme proposé par [Smallwood et Sondik, 1971] calcule toutes les projections $\mathcal{L}V_n$ possibles qui peuvent être construites, en appliquant la projection de la valeur autant de fois que nécessaire sur les paires action-observation.

Dans la suite, nous présentons les calculs nécessaires pour l'obtention de ces projections. Tout d'abord, il est nécessaire de créer les α-vecteurs associés à la valeur immédiate des actions, puis les projections pour toutes les paires action-observation sont calculées :

$$\Gamma^{a,*} \leftarrow \quad \alpha^{a,*}(s) \quad = r(s,a) \tag{2.16}$$

$$\Gamma^{a,o} \leftarrow \quad \alpha_i^{a,o}(s) \quad = \gamma \sum_{s' \in S} p(s'|s,a)p(o|s',a)\alpha_i(s'), \forall \alpha_i \in V_n. \tag{2.17}$$

Ensuite, l'opérateur de somme croisée doit être appliqué pour compléter la génération de l'ensemble des α-vecteurs associés à la projection à $n+1$:

$$\Gamma^a = \Gamma^{a,*} \bigoplus \Gamma^{a,o_1} \bigoplus \Gamma^{a,o_2} \bigoplus \ldots \bigoplus \Gamma^{a,o_{|\Omega|}} \tag{2.18}$$

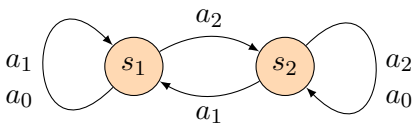
On définit la somme croisée d'ensembles par : soit deux ensembles $P = \{p_1, p_2, \ldots, p_m\}$ et $Q = \{q_1, q_2, \ldots, q_k\}$, la somme croisée produit un troisième ensemble tel que $P \bigoplus Q = \{p_1 + q_1, p_1 + q_2, \ldots, p_1 + q_k, \ldots, p_m + q_1, p_m + q_2, \ldots, p_m + q_k\}$.

Finalement nous prenons l'union sur tous les ensembles Γ^a :

$$V_{n+1} \leftarrow \mathcal{L}V_n = \bigcup_a \Gamma^a \tag{2.19}$$

Pour illustrer le fonctionnement de cette procédure, nous proposons de considérer le problème du tigre présenté au début de ce chapitre (voir figure 2.1). Nous illustrons une seule itération du calcul de la fonction de valeur.

Exemple 2.2.1 *Le problème du tigre consiste à choisir d'ouvrir la porte qui ne contient pas le tigre ; l'agent ne connait pas l'état réel du système (il ne sait pas où le tigre se trouve). L'agent peut seulement écouter à chaque porte, ce qui fait office d'observation. Il dispose de* 3 *actions, écouter, ouvrir la porte à gauche ou ouvrir la porte à droite. Chaque fois que l'agent choisit d'écouter, cette action lui coûte* 1*. Si l'agent choisit d'ouvrir une des portes et que le tigre se trouve derrière cette porte, l'agent perd* 100*. S'il ouvre la bonne porte il gagne* 10*. L'action d'ouvrir une des portes réinitialise le système (voir schéma du problème dans la figure 2.5). L'agent gange le défi à chaque fois qu'il choisit d'ouvrir la porte que contient le cadeau.*



récompenses

R	a_0	a_1	s_2
s_1	-1.0	10	-100
s_2	-1.0	-100	10

écouter (a_0)

T	s_1	s_2
s_1	1.0	0.0
s_2	0.0	1.0

O	s_1	s_2
o_1	0.85	0.15
o_2	0.15	0.85

ouvrir à gauche (a_1) ou à droite (a_2)

T	s_1	s_2
s_1	0.5	0.5
s_2	0.5	0.5

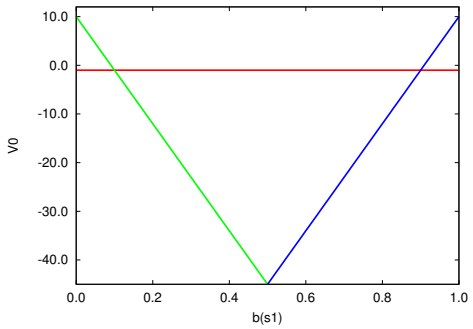
O	s_1	s_2
s_1	0.5	0.5
s_2	0.5	0.5

FIGURE 2.5 – Illustration des états, des actions et des observations, ainsi que des fonctions de transition, d'observation et de récompense pour le problème du tigre.

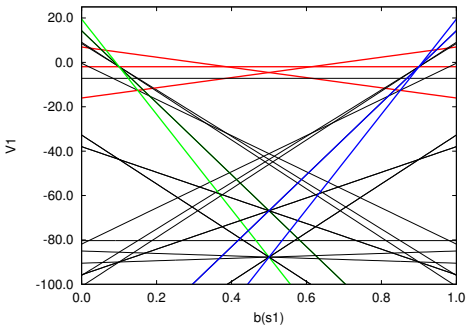
Pour commencer à résoudre ce problème, un ensemble initial de α-vecteurs V_0 est extrait directement de la fonction de récompense, à savoir un α-vecteur par action :

$$V_0 \leftarrow \alpha^a(s) = r(s,a), \forall a \in A. \tag{2.20}$$

La figure 2.6(a) montre la fonction de valeur initiale paramétrée par ces α-vecteurs. Cette figure montre seulement la dimension $b(s_1)$, puisque $b(s_2) = 1 - b(s_1)$. Les figures 2.7 et 2.8 décrivent les étapes qui amènent à la solution pour un horizon $t = 1$, avec un facteur d'actualisation $\gamma = 0.95$. La première étape consiste à projeter V_0 pour chaque paire action-observation (équation 2.17). La seconde étape concerne la somme croisée avec les revenus immédiats (équation 2.18). L'étape finale concerne l'union (équation 2.19). La fonction de valeur à l'étape V_1 est ainsi construite, et montrée sur la figure 2.6(b).



(a) α-vecteurs pour $t = 0$. Le vecteur en rouge est associé à l'action a_0, le vecteur en bleu à a_1, et le vecteur en vert à a_2.



(b) α-vecteurs pour $t = 1$. Les vecteurs en rouge sont associés à l'action a_0, les vecteurs en bleu à a_1, et les vecteurs en vert à a_2.

FIGURE 2.6 – Fonction de valeur pour les deux premières itérations du problème du tigre.

$$\Gamma^{a_0} = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \qquad \Gamma^{a_1} = \begin{bmatrix} 10 \\ -100 \end{bmatrix} \qquad \Gamma^{a_2} = \begin{bmatrix} -100 \\ 10 \end{bmatrix}$$

$$\bigcup_{a \in A}$$

$$V_0 = \left\{ \begin{bmatrix} -1 \\ -1 \end{bmatrix} \begin{bmatrix} 10 \\ -100 \end{bmatrix} \begin{bmatrix} -100 \\ 10 \end{bmatrix} \right\}$$

FIGURE 2.7 – Itération exacte sur la fonction de valeur à $t = 0$.

$$\Gamma_0^{a_0,o_0} = \left\{ \begin{bmatrix} -0.1425 \\ -0.8075 \end{bmatrix} \begin{bmatrix} 1.425 \\ -80.75 \end{bmatrix} \begin{bmatrix} -14.25 \\ 8.075 \end{bmatrix} \right\} \quad \Gamma_0^{a_0,o_1} = \left\{ \begin{bmatrix} -0.8075 \\ -0.1425 \end{bmatrix} \begin{bmatrix} 8.075 \\ -14.25 \end{bmatrix} \begin{bmatrix} -80.75 \\ 1.425 \end{bmatrix} \right\} \quad \Gamma^{a_0,*} = \left\{ \begin{bmatrix} -1 \\ -1 \end{bmatrix} \right\}$$

$$+ \qquad + \qquad +$$

$$\Gamma^{a_0} = \left\{ \begin{bmatrix} -1.95 \\ -1.95 \end{bmatrix} \begin{bmatrix} 6.932 \\ -16.05 \end{bmatrix} \begin{bmatrix} -81.89 \\ -0.382 \end{bmatrix} \begin{bmatrix} -0.382 \\ -81.89 \end{bmatrix} \begin{bmatrix} 8.5 \\ -96 \end{bmatrix} \begin{bmatrix} -80.32 \\ -80 - 32 \end{bmatrix} \begin{bmatrix} -16.05 \\ 6.932 \end{bmatrix} \begin{bmatrix} -7.175 \\ -7.175 \end{bmatrix} \begin{bmatrix} -96 \\ 8.5 \end{bmatrix} \right\}$$

$$\vdots$$

$$\Gamma_0^{a_2,o_0} = \left\{ \begin{bmatrix} -0.475 \\ -0.475 \end{bmatrix} \begin{bmatrix} 4.75 \\ -47.5 \end{bmatrix} \begin{bmatrix} -47.5 \\ 4.75 \end{bmatrix} \right\} \quad \Gamma_0^{a_2,o_1} = \left\{ \begin{bmatrix} -0.475 \\ -0.475 \end{bmatrix} \begin{bmatrix} 4.75 \\ -47.5 \end{bmatrix} \begin{bmatrix} -47.5 \\ 4.75 \end{bmatrix} \right\} \quad \Gamma^{a_2,*} = \left\{ \begin{bmatrix} -100 \\ 10 \end{bmatrix} \right\}$$

$$+ \qquad + \qquad +$$

$$\Gamma^{a_2} = \left\{ \begin{bmatrix} -100.95 \\ 9.05 \end{bmatrix} \begin{bmatrix} -95.72 \\ -37.97 \end{bmatrix} \begin{bmatrix} -147.9 \\ 14.27 \end{bmatrix} \begin{bmatrix} -95.72 \\ -37.97 \end{bmatrix} \begin{bmatrix} -90.5 \\ -85 \end{bmatrix} \begin{bmatrix} -142.7 \\ -32.75 \end{bmatrix} \begin{bmatrix} -147.9 \\ 14.27 \end{bmatrix} \begin{bmatrix} -142.7 \\ -32.75 \end{bmatrix} \begin{bmatrix} -195 \\ 19.5 \end{bmatrix} \right\}$$

$$V_1 \leftarrow \bigcup_{a \in A} \Gamma^a = \left\{ \begin{bmatrix} -1.95 \\ -1.95 \end{bmatrix} \begin{bmatrix} -6.932 \\ -16.05 \end{bmatrix} \begin{bmatrix} -81.89 \\ -0.382 \end{bmatrix} \cdots \begin{bmatrix} -147.9 \\ 14.275 \end{bmatrix} \begin{bmatrix} -142.7 \\ -32.75 \end{bmatrix} \begin{bmatrix} -195 \\ 19.5 \end{bmatrix} \right\}$$

FIGURE 2.8 – Itération exacte sur la fonction de valeur à $t = 1$

Malheureusement, à chaque étape, un nombre de vecteurs exponentiel en $|\Omega|$ est généré : $|V_{n+1}| = |A||V_n|^{|\Omega|}$. Les régions associées à de nombreux vecteurs seront vides, et seront donc inutiles (vecteurs en noir dans la figure 2.6(b) de l'exemple 2.2.1). Des algorithmes comme [Littman, 1997,Monahan, 1982] s'intéressent à les identifier pour, ensuite, les ignorer. Cette opération, qu'on appelle élagage, exige la résolution d'un programme linéaire pour chaque α-vecteur, et est plutôt coûteuse [Sigaud et Buffet, 2008] puisqu'elle ajoute des coûts de traitement, surtout quand l'espace d'état est grand. De plus, nous rappelons que dans le pire cas, la programmation linéaire a une complexité exponentielle en nombre d'observations.

Dans [Littman, 1994], une autre approche est envisagée : étant donné les ensembles de vecteurs résultants des projections $\Gamma^{a,o}$ et des sommes croisées Γ^a, et étant donné un état de croyance b, on considère le vecteur qui maximise la valeur de b en ajoutant ce vecteur dans V. Après, on cherche une évidence, soit un point témoin, pour lequel ce vecteur est sous-optimal. Quand on rencontre un point témoin, on ajoute le vecteur optimal de ce point à la représentation actuelle de V. Ensuite, on vérifie la dominance de ce nouveau vecteur afin de trouver un autre point témoin, et ainsi de suite. Cet algorithme, qui a pour nom *witness*, calcule une solution exacte. Ces vérifications exigent la résolution d'un programme linéaire, qui est construit pour chaque vérification de la dominance d'un α-vecteur afin de obtenir une représentation parcimonieuse de V.

[Littman, 1997] propose de réaliser des opérations d'élagage de manière incrémentale. La vérification de la dominance de chaque vecteur est faite à des moments différents : après la

projection pour le calcul des $\Gamma_i^{a,o}$, après la somme croisée avec les revenus immédiats et après l'union des vecteurs sur les actions. De cette façon, nous pouvons obtenir une représentation plus compacte de la fonction de valeur.

En général, calculer une solution ou une politique optimale pour les POMDPs devient un problème insoluble pour des problèmes réalistes. Ceci exige des techniques de résolution approchée. Dans la suite nous présenterons quelques travaux qui se sont intéressés à approcher la fonction de valeur des POMDP par des algorithmes approchés (équation de valeur modifiée).

2.3 Méthodes pour l'approximation de la fonction de valeur

Approximation par une politique myope. Une politique myope [Hauskrecht, 2000, Smith et Simmons, 2004] est une politique où la même action est toujours réalisée indépendamment de l'état de croyance et des observations. La valeur associée à une politique myope est clairement une borne inférieure de la valeur optimale du POMDP. La fonction de valeur résultante d'une politique myope est spécifiée par un *ensemble de α-vecteurs de taille $|A|$*, où chaque vecteur correspond à la valeur espérée si l'on suit la politique myope correspondante. Ces α-vecteurs peuvent être calculés par :

$$\alpha_{t+1}^a = r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a)\alpha_t^a(s'), \tag{2.21}$$

où, $\alpha_0^a = \min_{s \in S} \frac{r(s,a)}{1-\gamma}$. Ces α-vecteurs constituent l'ensemble Γ_t à partir du quel on obtient la borne inférieure de la valeur pour un état de croyance b, telle que :

$$V_t(b) = \max_{\alpha \in \Gamma_t} \sum_{s \in S} b(s)\alpha(s)$$

Approximation par MDP. Dans [Littman *et al.*, 1995] des approximations basées sur la résolution du MDP sous-jacent à un POMDP ont été proposées. La première est connue comme *MDP approximation*. Soit $\hat{V}_n$ une fonction linéaire décrite pas un vecteur α_n^{MDP} qui correpond à la fonction de valeur $V_n^{MDP}(s),\ \forall s \in S$, la fonction de valeur du POMDP $\hat{V}^{MDP}$ peut être approchée par :

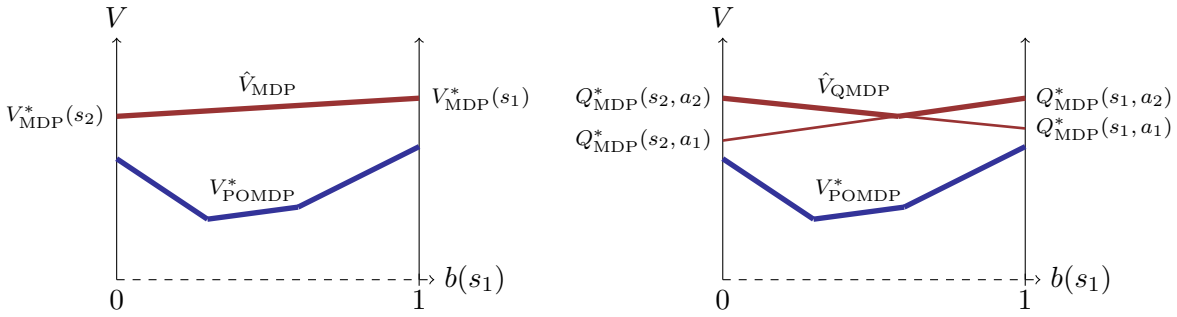
$$\begin{aligned} \hat{V}_{n+1}^{MDP}(b) &= \sum_{s \in S} b(s) \max_{a \in A} \left[r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a)\alpha_n^{MDP}(s') \right] && (2.22) \\ &= \mathcal{L}_{MDP} \hat{V}_n^{MDP}(b) && (2.23) \end{aligned}$$

ainsi, $\hat{V}^{MDP}$ est décrite par une fonction linéaire avec les composantes :

$$\alpha_{n+1}^{MDP}(s) = V_{n+1}^{MDP}(s) = \max_{a \in A} \{ r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a)\alpha_n^{MDP}(s') \} \tag{2.24}$$

Cette application de l'opérateur de mise à jour nous ramène toujours à une fonction linéaire (composée d'un seul α-vecteur) (voir figure 2.9(a)). De plus, celle-ci fournit une politique optimiste vis-à-vis du POMDP, et est connue comme une borne supérieure de la vraie fonction de valeur du POMDP [Hauskrecht, 2000].

Approximation par des Q-fonctions (QMDP). Une variante à l'approximation MDP utilise des Q-fonctions [Littman *et al.*, 1995] ; cette approximation diffère de la précédente



(a) Approximation par MDP

(b) Approximation par Q-fonction (QMDP)

FIGURE 2.9 – Approximations basées sur le MDP sous-jacent au POMDP pour un problème à 2 états et 3 actions.

essentiellement par une permutation des opérateurs de somme sur les états et de maximum sur les actions. Elle approche la fonction de valeur du POMDP par :

$$\hat{V}(b) = \max_{a \in A} \sum_{s \in S} b(s) Q_{MDP}(s,a), \text{ avec} \tag{2.25}$$

$$Q_{MDP}(s,a) = r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) V^*_{MDP}(s') \tag{2.26}$$

La règle de mise à jour de cet opérateur d'approximation peut être définie par :

$$\hat{V}^{QMDP}_{n+1}(b) = \max_{a \in A} \sum_{s \in S} b(s) \left[r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) \max_{\alpha_n \in V_n} \alpha^{MDP}_n(s') \right] \tag{2.27}$$

$$= \mathcal{L}_{QMDP} \hat{V}^{MDP}_n(b) \tag{2.28}$$

Q^*_{MDP} est la Q-fonction optimale pour le MDP sous-jacent d'un POMDP. Cette approximation $\hat{V}^{QMDP}$ est linéaire par morceaux et convexe *avec* $|A|$ *fonctions linéaires* (voir figure 2.9(b)), où chaque fonction linéaire correspond à une action. Cette approximation est aussi une borne supérieure de la fonction de valeur optimale du POMDP telle que, $\mathcal{L}\hat{V}_n \leq \mathcal{L}_{QMDP}\hat{V}_n \leq \mathcal{L}_{MDP}\hat{V}_n$ [Hauskrecht, 2000].

Méthode de la borne la plus informative (FIB). Les approches présentées précédemment ignorent l'observabilité partielle et s'appuient sur le modèle complètement observable sous-jacent. Afin d'améliorer ces approximations, [Hauskrecht, 2000] propose une méthode d'approximation basée sur le vrai opérateur de mise à jour de la fonction de valeur du POMDP. En manipulant l'opérateur max le plus à droite de l'équation :

$$V_{n+1}(b) = \max_{a \in A} \left\{ \sum_{s \in S} r(s,a) b(s) + \gamma \sum_{o \in \Omega} \max_{\alpha_n \in V_n} \sum_{s' \in S} p(o \mid s', a) \sum_{s \in S} p(s' \mid s, a) b(s) \alpha_n(s') \right\}, \tag{2.29}$$

et en extrayant la somme sur les états, on peut obtenir l'approximation $\hat{V}_{n+1}$ telle que :

$$\hat{V}_{n+1}(b) = \max_{a \in A} \left\{ \sum_{s \in S} r(s,a) b(s) + \gamma \sum_{o \in \Omega} \sum_{s \in S} b(s) \max_{\alpha_n \in V_n} \sum_{s' \in S} p(o|s',a) p(s'|s,a) \alpha_n(s') \right\} \tag{2.30}$$

$$= \max_{a \in A} \left\{ \sum_{s \in S} b(s) \left[r(s,a) + \gamma \sum_{o \in \Omega} \max_{\alpha_n \in V_n} \sum_{s' \in S} p(o|s',a) p(s'|s,a) \alpha_n(s') \right] \right\} \tag{2.31}$$

$$= \mathcal{L}_{FIB} \hat{V}_n(b) \tag{2.32}$$

Algorithme 1: Approximation par la méthode en "dents de scie".

entrée : POMDP, b, $\overline{Q}$, $\overline{V}$
sortie : valeur $\overline{V}(b)$.

$\overline{V}(b) \leftarrow \max_{a \in A} \sum_{s \in S} b(s) \overline{Q}_a(s)$;
foreach $b' \in \mathcal{B} \backslash \{e_s | s \in S\}$ **do**
  $c(b') \leftarrow \min_{s \in S} \frac{b(s)}{b'(s)}, \ \forall s | b'(s) > 0, b(s) > 0$;
  $f(b') \leftarrow \overline{V}(b') - \sum_{s \in S} b(s) \overline{V}(e_s)$;
$b'^* \leftarrow \arg\min_{b' \in \mathcal{B}} c(b') f(b')$;
$\overline{V}(b) \leftarrow \min(\overline{V}(b), c(b'^*) f(b'^*) + \sum_{s \in S} b(s) \overline{V}(e_s))$;

L'approximation $\hat{V}_{n+1} = \mathcal{L}_{FIB} \hat{V}_n$ conduit après mise à jour une fonction linéaire par morceaux et convexe composée par au plus $|A|$ vecteurs, chacun correspondant à une des actions. Il a été prouvé que cette approximation est aussi une borne supérieure de la fonction de valeur optimale du POMDP, telle que : $\mathcal{L}\hat{V}_n \leq \mathcal{L}_{FIB}\hat{V}_n \leq \mathcal{L}_{QMDP}\hat{V}_n \leq \mathcal{L}_{MDP}\hat{V}_n$ [Hauskrecht, 2000].

Approximations par interpolation de la valeur. La fonction de valeur sur l'espace continu d'états de croyance peut être approchée par une règle d'interpolation ou d'extrapolation basée sur un nombre fini d'états de croyance avec lesquels une valeur est associée. L'approximation par interpolation la plus utilisée dans la littérature est l'interpolation linéaire [Hauskrecht, 2000]. Celle-ci est prouvée être une borne supérieure de la fonction de valeur optimale du POMDP [Lovejoy, 1991].

L'objectif est trouver la meilleure interpolation, celle qui amène à la borne supérieure la plus proche de la fonction de valeur optimale. Soit b un état de croyance, G une grille de points b_j dans l'espace d'états de croyance et $\{b_j, V(b_j) | b_j \in G\}$ un ensemble de paires état de croyance-valeur, la meilleure interpolation peut être trouvée par le problème d'optimisation linéaire suivant :

$$\hat{V}(b) = \min_{\overline{\lambda}} \sum_{j=1}^{|G|} \lambda_j V(b_j) \tag{2.33}$$

$$s.c. \quad 0 \leq \lambda_j \leq 1, \ \forall j \tag{2.34}$$

$$\sum_{j=1}^{|G|} \lambda_j = 1 \tag{2.35}$$

$$b = \sum_j \lambda_j b_j \tag{2.36}$$

Pour éviter la résolution du programme linéaire, [Hauskrecht, 2000] a proposé la méthode dite en "dents de scie" (*sawtooth* en anglais) qui fournit une borne supérieure de la fonction de valeur optimale. Cette méthode décrite dans l'algorithme 1 (voir [Poupart *et al.*, 2011]) permet d'approcher la valeur d'un état de croyance très rapidement en se basant sur les valeurs $\overline{V}$ (approximation par MDP) et $\overline{Q}$ (approximation par QMDP) d'un ensemble fini d'états de croyance $\mathcal{B} \in \Delta$. Cette approximation trouve la meilleure interpolation en utilisant les $|S| - 1$ points extrêmes, notés e_s, et un point intérieur $b' \in \mathcal{B}$ du simplexe.

D'autres méthodes d'approximation basées sur des grilles, sur ajustement de courbes ou encore sur le MDP sous-jacent inobservable ont été proposées. Nous invitons tout lecteur intéressé à se rapporter à [Hauskrecht, 2000, Poupart *et al.*, 2011] pour plus de détails.

Les méthodes présentées dans cette sous-section, qui évitent de calculer les projections pour la mise à jour de la fonction de valeur paramétrée par des α-vecteurs en utilisant à la place des approximations, sont aujourd'hui dépassées au niveau de la performance (récompenses accumulées) par des méthodes algorithmiques plus récentes. Ces méthodes plus récentes se focalisent sur la recherche d'un ensemble fini, échantillonné, et borné d'états de croyance dits atteignables et réalisent des itérations sur la fonction de valeur pour cet ensemble borné d'états de croyance. De plus, une partie de ces méthodes utilisent la borne inférieure et/ou supérieure de la fonction de valeur, qui sont souvent calculées par les méthodes approximatives présentées précédemment, comme heuristique dans l'initialisation de la valeur et l'exploration de l'espace d'états de croyance afin de construire un ensemble d'états de croyance pertinent.

2.4 Méthodes approximatives d'itération sur la fonction de valeur

La principale cause de complexité du calcul d'une solution exacte pour un POMDP est qu'à chaque étape du calcul de la fonction de valeur, l'action optimale doit être trouvée pour tout point de l'espace de croyance. Nous avons mis en évidence par l'exemple 2.2.1 que la taille de la fonction de valeur augmente exponentiellement à chaque pas de l'horizon de planification.

Une approche naturelle pour combattre cette complexité est de calculer la fonction de valeur *seulement* pour un ensemble *fini* d'états de croyance. Dans la suite, nous présenterons l'opérateur de mise à jour de la valeur utilisé ponctuellement sur certains états de croyance par des approches de résolution approximatives.

2.4.1 Opérateur de mise à jour de la valeur – *backup*

Soit une fonction de valeur V_n paramétrée par un ensemble d'α-vecteurs Γ et un état de croyance particulier b, on peut calculer le vecteur α_{n+1}^b de V_{n+1}, qui maximise la valeur de b à l'instant $n+1$ à partir de $\mathcal{L}V_n$ comme suit :

$$\alpha_{n+1}^b = \underset{\alpha_{n+1}^i \in \mathcal{L}V_n}{\arg\max}\ b \cdot \alpha_{n+1}^i, \tag{2.37}$$

où $\mathcal{L}V_n$ est l'ensemble (inconnu) de vecteurs. Cette opération appelée *backup* fournit le α_{n+1}^i qui maximise la valeur de b et repose sur les projections calculées par les équations 2.17, 2.18 et 2.19. Si l'on définit $r_a(s) = r(s,a)$ et si l'on utilise les équations 2.6, 2.11 et 2.14, l'opérateur de *backup* peut être obtenu par :

$$\begin{aligned}
V_{n+1}(b) &= \max_a \left[b \cdot r_a + \gamma \sum_o p(o \mid a, b) V_n(b_o^a) \right] \\
&= \max_a \left[b \cdot r_a + \gamma \sum_o p(o \mid b, a) \max_{\alpha_n^i \in V_n} \sum_{s'} b_o^a(s') \alpha_n^i(s') \right] \\
&= \max_a \left[b \cdot r_a + \gamma \sum_o p(o \mid b, a) \max_{\alpha_n^i \in V_n} \frac{\sum_{s'} p(o \mid s', a) \sum_s p(s' \mid s, a) b(s) \alpha_n^i(s')}{p(o \mid b, a)} \right] \\
&= \max_a \left[b \cdot r_a + \gamma \sum_o \max_{\alpha_n^i \in V_n} \sum_{s'} p(o \mid s', a) \sum_s p(s' \mid s, a) b(s) \alpha_n^i(s') \right]
\end{aligned} \tag{2.38}$$

donc :

$$V_{n+1}(b) = \max_a \left[b \cdot r_a + \gamma \sum_o \max_{\alpha_i^{a,o} \in \Gamma^{a,o}} b \cdot \alpha_i^{a,o} \right] \tag{2.39}$$

avec $\Gamma^{a,o}$ donné par l'équation 2.17. Ensuite, en appliquant l'égalité suivante :

$$\max_j b \cdot \alpha_j = b \cdot \arg\max_j b \cdot \alpha_j \tag{2.40}$$

deux fois en 2.39, le vecteur résultat de l'opération $backup(b)$ est obtenu par :

$$backup(b) = \arg\max_{\alpha_b^a \in \Gamma_b^a} b \cdot \alpha_b^a, \text{ avec} \tag{2.41}$$

$$\Gamma_b^a \leftarrow r_a + \gamma \sum_o \arg\max_{\alpha_i^{a,o} \in \Gamma^{a,o}} b \cdot \alpha_i^{a,o} \tag{2.42}$$

Il est important de noter que calculer le vecteur résultat de l'opération $backup(b)$ pour tout l'espace d'état de croyance est très coûteux. En effet, pour trouver l'ensemble minimal de α-vecteurs qui couvre l'ensemble des états de croyance b, on doit calculer tous les vecteurs $\bigcup_{b \in \Delta} backup(b)$ de $\mathcal{L}V_n$.

Dans les algorithmes approximatifs d'itération sur la valeur, le calcul de 2.41 est réduit à un nombre fini d'étapes, c'est-à-dire que cette opération est calculée seulement pour les états de croyances qui appartiennent à l'ensemble fini, échantillonné et borné d'états de croyance. Ceci conduit à un nombre fini de vecteurs, borné par la taille de l'ensemble des états de croyance.

L'utilisation des méthodes approximatives est principalement motivée par leur capacité à trouver de bonnes politiques partielles, définies uniquement pour les états de croyance choisis, pour résoudre des problèmes de tailles considérablement plus élevées, au prix d'une perte (souvent contrôlée) d'optimalité. De plus, ces méthodes supposent que la politique sera généralisée aisément dans le voisinage des états de croyance pour lesquels le calcul a été réalisé, c'est-à-dire que la politique pourra être utilisée pour d'autres points "proches" de l'espace de croyance qui n'appartiennent pas à l'ensemble fini B.

Ainsi, différentes méthodes approximatives d'itération sur la valeur ont été développées pour la résolution hors ligne et en ligne des POMDP. Les approches dites hors ligne sont capables d'obtenir des politiques quasi-optimales en temps raisonnable. Les méthodes de résolution en ligne se sont appuyées sur les différents développements proposés par les approches hors ligne, et arrivent aujourd'hui à fournir des politiques partielles convenables très rapidement.

Dans la suite, nous présentons d'abord les méthodes approximatives d'itération sur la valeur développées pour la résolution hors ligne. Les méthodes de résolution en ligne seront présentées dans un second temps.

2.4.2 Méthodes de résolution hors ligne

Les méthodes de résolution hors ligne s'intéressent en approcher une fonction de valeur afin d'obtenir une politique ϵ-optimale. Par exemple, ils réalisent des itérations sur la valeur jusqu'à ce que la différence entre les valeurs successives (entre t et $t+1$) des états de croyance considérés est plus petite qu'un seuil donné ϵ.

Par suite, nous présentons quelques algorithmes de résolution hors ligne de l'état de l'art du domaine POMDP. Nous les classons selon la méthode employée pour la mise à jour de la valeur des états de croyance (synchrone ou asynchrone) et selon la façon qu'ils choisissent les états de croyance qui seront considérés pour l'approximation de la fonction de valeur (construction stochastique ou heuristique de l'ensemble d'états de croyance fini et borné).

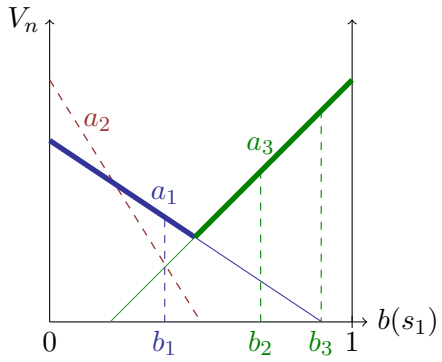


FIGURE 2.10 – Fonction de valeur à l'instant n pour un problème à 2 états et 3 actions. L'α-vecteur en rouge ne fait pas partie de la fonction de valeur, puisque aucun état de croyance de l'ensemble $\mathcal{B} = \{b_1, b_2, b_3\}$ n'a sa valeur maximisée par cet α-vecteur.

Ainsi, nous présentons dans la suite les approches dites *point-based*, les approches dites *trial-based* et les approches basées sur une recherche guidée de manière heuristique.

Méthodes *point-based*

Les méthodes appelées *point-based* construisent une fonction de valeur linéaire par morceaux, mais focalisent la construction des α-vecteurs et la mise à jour de la fonction de valeur sur un ensemble fini $\mathcal{B} \in \Delta$. Ce type de méthode met à jour à chaque itération non seulement les valeurs mais aussi les gradients (les α-vecteurs) pour tout point b (états de croyance) de $\mathcal{B}$ de manière synchrone. Il est indispensable de noter que l'ensemble de états de croyance $\mathcal{B}$ est composé par des états de croyance considérés atteignables. Ces états de croyances sont atteignables par le suivi d'une politique arbitraire en partant de b_0. Ainsi, l'espace de l'état de croyance peut être couvert par un ensemble d'états de croyance relativement petit avec une forte densité.

La figure 2.10 illustre une représentation parcimonieuse de la fonction de valeur à l'instant n d'un problème à 2 états et 3 actions. Il est à noter qu'aucun état de croyance de l'ensemble $\mathcal{B} = \{b_1, b_2, b_3\}$ n'a sa valeur maximisée par l'α-vecteur associé à l'action a_2. Ainsi, cet α-vecteur ne ferra pas partie de la fonction de valeur. Il est à noter que la généralité de la politique dépend fortement de la façon par laquelle l'ensemble $\mathcal{B}$ est constitué.

L'algorithme PBVI. L'algorithme d'itération sur la fonction de valeur *PBVI – Point-Based Value Iteration* [Pineau *et al.*, 2003], travaille sur un ensemble fini d'états de croyance. Cet algorithme est un des premiers algorithmes performants de l'état de l'art qui se focalise sur la construction des α-vecteurs et la mise à jour de la fonction de valeur pour un ensemble fini $\mathcal{B} \in \Delta$, tout en démontrant que l'erreur de l'approximation de la fonction de valeur est bornée et dépend de la densité de $\mathcal{B}$. Nous nous intéressons particulièrement à cet algorithme car il nous sera utile lors des chapitres 4 et 5 de cette thèse.

Les auteurs de PBVI utilisent l'opérateur de mise à jour de la valeur (*backup*) de l'équation 2.41 pour l'ensemble borné $\mathcal{B}$ d'états de croyance. Nous appellerons cet opérateur de mise à jour de la valeur $\tilde{\mathcal{L}}_{\text{PBVI}}$. Ceci permet de calculer à chaque étape la fonction de valeur de l'ensemble $\mathcal{B}$:

$$\tilde{\mathcal{L}}_{\text{PBVI}} V_n = \bigcup_{b \in \mathcal{B}} backup(b) \tag{2.43}$$

Algorithme 2: PBVI

entrée : POMDP, N
sortie : fonction de valeur V

1 Initialiser V_0, $n = 0$;
2 $\mathcal{B} \leftarrow b_0$;
3 **repeat**
4 $\quad n = n + 1$;
5 $\quad$ Étendre $\mathcal{B}$;
6 $\quad V_n \leftarrow \emptyset$;
7 $\quad$ Calculer toutes les projections $\Gamma^{a,o}$ de V_{n-1} (équation 2.17) ;
8 $\quad$ **for** $b \in \mathcal{B}$ **do**
9 $\quad\quad V_n \leftarrow \bigcup backup(b)$;
10 **until** $n < N$ ou $||\max_{\alpha_n \in V_n} \alpha_n \cdot b - \max_{\alpha_{n-1} \in V_{n-1}} \alpha_{n-1} \cdot b|| < \epsilon, \forall b \in \mathcal{B}$;

Plus précisément, l'algorithme 2 résume le fonctionnement de PBVI. PBVI initialise la fonction de valeur suivant l'équation 2.20 (ligne 1). Ensuite, l'ensemble d'états de croyance $\mathcal{B}$ est initialisé par l'état de croyance initial du POMDP (ligne 2). Avec un nombre fini d'itérations, l'algorithme effectue d'abord une expansion de l'ensemble d'états de croyance atteignables par des simulations stochastiques du POMDP (ligne 5).

L'opération de la ligne 5 s'effectue comme suit : pour chaque point $b \in \mathcal{B}$, un état s est tiré suivant $b(s)$, puis pour chaque action $a \in A$, une observation est tirée suivant $p(s'|s,a)$ et $p(o|s')$, ainsi un ensemble $\{b_{a0}, ..., b_{aj}\}$ est créé. La prochaine étape consiste à mesurer la distance euclidienne des points $\{b_{a0}, ..., b_{aj}\}$ par rapport à tout $b \in \mathcal{B}$, et le point le plus éloigné de tous les $b \in \mathcal{B}$ est choisi et intégré à $\mathcal{B}$.

Ensuite, PBVI calcule toutes les projections de V_{n-1} suivant l'équation 2.17 (ligne 7). Pour tout $b \in \mathcal{B}$, l'opération de mise à jour de la valeur est effectuée (équation 2.41). Celle-ci construit la nouvelle fonction de valeur V_n (ligne 9, équation 2.43). Si le nombre N d'itérations est atteint, ou si la plus grande différence de la valeur pour tout $b \in \mathcal{B}$ est inférieure à ϵ, l'algorithme s'arrête et la fonction de valeur approchée par les α-vecteurs générés est retournée.

L'algorithme PERSEUS. Un autre algorithme d'itération approximative sur la fonction de valeur a été proposé, l'algorithme PERSEUS [Spaan et Vlassis, 2004, Spaan et Vlassis, 2005]. PERSEUS explore l'environnement et recueille un ensemble d'états de croyance atteignables qui est fixé au début de la résolution. L'exploration concerne plusieurs trajectoires partant de b_0 à travers l'espace de croyance en échantillonnant aléatoirement des paires action-observation jusqu'à une certaine profondeur. Après, la valeur de V_0 est initialisée avec toutes les composantes égales à $\frac{1}{1-\gamma} \min_{s,a} r(s,a)$ [Zhang et Zhang, 2001, Spaan et Vlassis, 2005]. Cette approximation garantit que la fonction de valeur initiale sera une borne inférieure. L'idée principale est qu'à chaque itération de la mise à jour de fonction de valeur, on augmente la valeur de tous les point $b \in \mathcal{B}$ par la mise à jour de la valeur et du gradient de seulement un sous-groupe $\tilde{\mathcal{B}} \subseteq \mathcal{B}$. Pour cela, l'opérateur de mise à jour de la valeur est appliqué afin de calculer un α-vecteur qui maximise la valeur de b pour une itération n, et ensuite l'algorithme vérifie quels sont les autres $b' \in \mathcal{B}$ qui ont vu leur valeur augmenter. Les points b' dont la valeur a augmenté sont considérés comme ayant leur valeur mise à jour. Autrement dit, on construit une fonction de valeur $V_{n+1} = \mathcal{L}_{PERSEUS} V_n$ qui augmente la valeur V_n sur $\mathcal{B}$, telle que, $V_n(b) \leq V_{n+1}(b)$, $\forall\, b \in \mathcal{B}$.

Les deux algorithmes que nous venons de présenter réalisent les itérations sur l'ensemble $\mathcal{B}$ d'états de croyance de manière synchrone, c'est-à-dire, que la mise à jour de la fonction de valeur est réalisée pour tout état de croyance de $\mathcal{B}$ à chaque itération. Une autre forme de mise

à jour très répandue est l'itération asynchrone sur la valeur, connue sous la dénomination de *trial-based*.

Méthodes *trial-based*

Les méthodes *trial-based* sont basées sur des simulations de trajectoires dans l'espace d'états de croyance. Les états de croyance visités le long de la trajectoire ont leur valeur mise à jour dans l'ordre inverse. Plus précisément, une fois la trajectoire obtenue, la mise à jour s'effectue depuis l'état de croyance le plus profond de la trajectoire jusqu'à l'état de croyance initial. Dans la suite nous présenterons un algorithme que implémente cette approche.

L'algorithme RTDP-bel. L'algorithme RTDP-bel [Bonet et Gefner, 1998, Bonet et Geffner, 2009] est une adaptation de l'algorithme RTDP [Barto *et al.*, 1995], conçu initialement pour la résolution des MDP. RTDP-bel, en plus de discrétiser les états de croyance et maintenir une Q-fonction pour ces états de croyance discrétisés, suggère une transformation du modèle POMDP vers un POMDP orienté "but" avec un modèle de récompenses transformé en un modèle de coûts ($c(s,a)$ à la place de $r(s,a)$). Selon les auteurs, ceci est nécessaire, car l'algorithme de base RTDP suppose que la trajectoire atteindra après plusieurs étapes un état but avec probabilité 1 depuis chaque état initial possible, tout en essayant de minimiser les coûts. Les trajectoires sont exécutées en accord avec le modèle POMDP, jusqu'à l'état de croyance but ; à chaque étape une projection de l'état de croyance vers un état de croyance discrétisé proche est réalisée, afin de mettre à jour la valeur de cet état de croyance discrétisé. Dans cet algorithme, la fonction de valeur n'est pas représentée par des α-vecteurs, mais représentée par une table de hachage dont la clé d'entrée est cet état de croyance discrétisé. La fonction de discrétisation est très simple, celle-ci projette chaque $b(s)$ en $d(b(s)) = ceil(D * b(s))$, où D est un entier positif, et la fonction $ceil(x)$ fournit l'entier immédiatement supérieur ou égal à x. Cette discrétisation est faite seulement pour accéder à la valeur dans la table de hachage, et ne substitue pas un état de croyance du POMDP visité au cours d'une trajectoire. RTDP-bel n'offre aucune garantie de convergence en raison de la discrétisation pour l'accès à la table de hachage, car la valeur associée à un état de croyance discrétisé peut osciller. De plus, cette fonction de valeur approchée, qui est initialisée de façon heuristique, n'a pas de garantie de demeurer une borne inférieure.

Méthodes basées sur une recherche heuristique pour la résolution hors ligne

D'autres méthodes sont basées sur des simulations de trajectoires dans l'espace d'états de croyance. Contrairement aux méthodes précédentes les trajectoires simulées sont guidées : plus précisément, ces méthodes se basent sur les bornes supérieure et inférieure de la fonction de valeur pour indiquer successivement quel doit être le prochain état de croyance à visiter. Une fois que la trajectoire est considérée comme assez profonde (selon un critère défini), la mise à jour s'effectue comme pour les méthodes *trial-based* : depuis l'état de croyance le plus profond de la trajectoire jusqu'à l'état de croyance initial (de manière asynchrone).

L'algorithme HSVI. L'algorithme *HSVI – Heurisctic Search Value Iteration* a été proposé par [Smith et Simmons, 2004, Smith et Simmons, 2005], et depuis il démontre une des meilleures performances dans le domaine en ce qui concerne le temps de calcul et les revenus collectés. HSVI est basé sur le maintien d'une borne supérieure $\overline{V}$, et une borne inférieure $\underline{V}$ de la fonction de valeur optimale pour chaque état ce croyance visité.

La méthode de HSVI est illustrée dans l'algorithme 3. À partir des heuristiques basées sur les bornes, HSVI guide les trajectoires dans l'espace d'état de croyance en sélectionnant successivement : l'action qui est considérée comme la plus prometteuse pour l'état de croyance

Algorithme 3: HSVI [Smith et Simmons, 2005]

entrée : POMDP
sortie : fonction de valeur V

1 Initiliser $\overline{V}$ et $\underline{V}$;
2 **while** $\overline{V}(b_0) - \underline{V}(b_0) > \epsilon$ **do**
3 Explorer$(b_0, \overline{V}, \underline{V})$;

4 **fonction Explorer(**$b, \overline{V}, \underline{V}$**)**;
5 **if** $\overline{V}(b) - \underline{V}(b) > \epsilon\gamma^{-t}$ **then**
6 $a^* \leftarrow \arg\max_{a \in A} Q_{\overline{V}}(b, a)$;
7 $o^* \leftarrow \arg\max_{o \in \Omega} p(o|b, a^*)(\overline{V}(b_a^o) - \underline{V}(b_a^o))$;
8 Explorer$(b_{a^*}^{o^*}, \overline{V}, \underline{V})$;
9 $\underline{V} \leftarrow backup(b, \underline{V})$;
10 $\overline{V} \leftarrow \mathcal{L}\overline{V}$;

visité, c'est-à-dire, celle qui maximise la Q-valeur $Q_{\overline{V}}(b, a)$ (choix glouton) (ligne 6 de l'algorithme 3) ; puis, l'observation pour laquelle la différence $p(o|b, a^*)(\overline{V}(b) - \underline{V}(b))$ est maximale (ligne 7). Ces choix sont réalisés successivement, jusqu'à ce que la trajectoire puisse être considérée comme assez profonde. Autrement dit, le critère d'arrêt de la recherche est tel que : l'algorithme s'arrête quand l'erreur entre la valeur des bornes pour l'état de croyance visité est considéré comme "petite" par rapport aux paramètres ϵ, γ et t (ligne 5). Ensuite, il exécute la mise à jour de la valeur de la borne supérieure (équation 2.11 appliquée à la borne) et l'opération de mise à jour de la valeur (équation 2.41) pour l'état de croyance observé (lignes 9 et 10). La mise à jour de la valeur des états de croyance allant du plus profond jusqu'à l'origine des trajectoires (b_0) est assurée par la récursivité de la fonction *Explorer*.

Dans HSVI, la borne supérieure a une représentation du type point-valeur, c'est-à-dire qu'elle est représentée par des paires $[V(b), b]$. De plus, dans la version 1 de HSVI [Smith et Simmons, 2004], à chaque fois qu'un nouvel état de croyance b' est découvert, la valeur initiale de $\overline{V}(b')$ est approchée par la technique d'interpolation linéaire, présentée dans la section 2.3. Et dans la version 2 de HSVI [Smith et Simmons, 2005] par l'algorithme 1. L'opération de mise à jour de la borne supérieure s'appuie sur l'opérateur $\mathcal{L}$. La borne inférieure est initialisée par un ensemble de α-vecteurs résultants du calcul d'une fonction de valeur myope (voir section 2.3).

Le maintien d'une borne supérieure qui est mise à jour en parallèle des opérations de mise à jour de la valeur des états de croyance est une manœuvre considérée coûteuse. Afin de combattre cela, [Shani *et al.*, 2007] a proposé l'algorithme FSVI qui ne maintient pas de borne supérieure, ce qui se traduit par un gain considérable dans le temps de calcul.

L'algorithme FSVI. *FSVI – Forward Search Value Iteration* est un algorithme de résolution de POMDP orienté "but" basé sur les avantages des algorithmes RTDP-bel et HSVI. Contrairement à RTDP-bel, FSVI n'utilise pas de transformations, il fixe un nombre d'étapes de recherche si un "but" n'a pas été retrouvé. FSVI implémente la même structure récursive de recherche dans l'espace d'états de croyance que HSVI, mais contrairement à HSVI il ne maintient pas de borne supérieure. À la place, il propose un choix glouton d'actions d'exploration basé sur la Q-fonction de valeur du MDP sous-jacent.

Cet algorithme a démontré un gain considérable de temps dans la convergence de la fonction de valeur, ce qui met en évidence le fait qu'une politique de recherche gloutonne basée sur le MDP sous-jacent peut, dans certains cas, être suffisante. Par contre, dans les problèmes où la prise d'information sur le système est indispensable pour le bon déroulement de la

mission, FSVI a démontré un comportement très pauvre, dû au fait que la politique gloutonne de recherche basée sur le MDP sous-jacent, qui ne tient pas compte de l'observabilité partielle, conduit la recherche vers des états de croyance où le gain associé à $r(s,a)$ est important, et non vers des états de croyance où le gain d'information serait prioritaire.

L'algorithme SARSOP. Un autre algorithme basé sur HSVI est l'algorithme SARSOP proposé par [Kurniawati *et al.*, 2008]. SARSOP utilise la même stratégie de choix que HSVI, c'est-à-dire, que l'algorithme calcule des trajectoires dans l'espace d'état de croyance en sélectionnant l'action de manière gloutonne, et l'observation de façon à ce que la recherche aboutisse à un état de croyance pour lequel la contribution $\overline{V}(b) - \underline{V}(b)$ est maximale. SARSOP initialise les bornes supérieure et inférieure de la même façon que HSVI. La différence entre les algorithmes se trouve dans la manière d'arrêter la recherche en profondeur. HSVI arrête la recherche une fois que $\overline{V}(b) - \underline{V}(b) > \epsilon\gamma^{-t}$. SARSOP procède différemment. Il propage la valeur de la borne inférieure et une valeur approchée de la borne supérieure de b_0 tout au long de la trajectoire. Ceci afin de : (1) comparer à chaque étape de l'exploration la borne inférieure propagée et évaluée en b avec une prédiction de la valeur optimale de ce même b exploré ; et (2) de comparer à chaque étape de l'exploration la valeur approchée de la borne supérieure propagée et évaluée en b avec la borne supérieure courante de ce b. Dans SARSOP, le critère d'arrêt de la recherche est défini selon : $[(\hat{V} \leq L)$ et $(\overline{V}(b) \leq \max\{U, \underline{V}(b) + \epsilon\gamma^{-t}\})]$, où $\hat{V}$ est la valeur optimale prédite de b, L et U représentent les bornes propagées et évaluées en b, et $\overline{V}(b)$ et $\underline{V}(b)$ les bornes inférieure et supérieure courantes de ce même b exploré. Ainsi, la profondeur de recherche est déterminée de façon à privilégier une recherche de points moins profondes que HSVI. Ensuite, la mise à jour de la valeur des états de croyance allant du plus profond jusqu'à l'origine des trajectoires (b_0) est réalisée comme pour HSVI et FSVI. SARSOP est supposé (aucune preuve théorique n'est fournie) orienter la recherche vers des états de croyance qui sont les plus susceptibles d'appartenir à l'ensemble $\mathcal{B}^*$, c'est-à-dire à l'ensemble d'états de croyance optimal qui est atteint si une politique optimale est exécutée. Les résultats expérimentaux publiés montrent que SARSOP est aussi performant que HSVI en ce qui concerne les récompenses accumulées, tout en étant légèrement plus rapide. Nous invitions tout lecteur intéressé à consulter [Kurniawati *et al.*, 2008] pour plus de détails sur la prédiction de la valeur optimale des états de croyance explorés et la propagation des bornes de b_0.

Les algorithmes présentés ci-dessus, qui ont été développés dans le cadre d'une résolution hors ligne, sont aujourd'hui capables d'approcher des politiques quasi-optimales pour des POMDP de grand taille en temps raisonnable (de l'ordre des milliers d'états). Dans la suite nous nous focaliserons sur des algorithmes de résolution en ligne récemment développés, et qui utilisent les avantages des techniques présentées précédemment.

2.4.3 Méthodes de résolution en ligne

Grâce aux méthodes récemment développées pour la résolution hors ligne des POMDP, qui ont propulsé le gain de temps de résolution, la résolution en ligne des POMDP a commencé à être envisagée, particulièrement dans le domaine de la robotique. Une difficulté majeure de la planification en ligne est qu'elle doit répondre à des contraintes de temps dures liées aux architectures d'exécution dans laquelle les algorithmes POMDP sont embarqués.

Des approches récentes comme [Paquet *et al.*, 2006, Ross et Chaib-Draa, 2007] suggèrent de combiner des approximations calculées hors ligne avec des approches de résolution en ligne pour affronter de manière efficace la résolution des POMDP. L'idée est d'utiliser des politiques obtenues par des algorithmes classiques de calcul hors ligne pour guider la recherche

des algorithmes en ligne. Cette technique permet à des algorithmes en ligne de planifier pour un horizon court tout en respectant les contraintes de temps réel avec une bonne précision, car l'optimisation en ligne pour un horizon fixé permet de réduire l'erreur d'approximation de la fonction de valeur initiale calculée hors ligne [Paquet *et al.*, 2006].

Un algorithme de résolution en ligne (voir algorithme 4) est divisé en une phase de planification (lignes 5 à 7) et une phase d'exécution (lignes 8 à 11), qui sont réalisées alternativement [Ross *et al.*, 2008b]. Dans la phase de planification, l'algorithme calcule la meilleure action à exécuter pour l'état de croyance courant de l'agent. Ceci est normalement obtenu en deux étapes :

1. À partir d'un arbre constitué par les états de croyance dits atteignables depuis l'état de croyance courant par l'exécution des paires action-observation (voir figure 2.11), l'algorithme choisit (ligne 5) le prochain nœud à développer afin de continuer la construction de l'arbre (lignes 6 et 7). Dans cet arbre les nœuds subséquents sont calculés par la règle de Bayes (équation 2.6). Notez que toutes les actions et observations doivent être considérées ;

2. Ensuite, la valeur de l'état de croyance courant est estimée par la propagation des valeurs estimées des feuilles et de ses parents jusqu'à la racine, suivant l'équation de Bellman 2.9. La valeur des feuilles est souvent estimée par une fonction de valeur approchée hors ligne.

Cette phase de planification est répétée jusqu'à ce que la condition d'arrêt soit atteinte, ou jusqu'à ce que la durée maximum allouée à la planification soit atteinte. Une fois la planification terminée, l'exécution prend le relais en exécutant l'action qui maximise la valeur courante de l'état de croyance courant (ligne 8) ; après exécution, une observation est reçue de l'environnement (ligne 9) et la mise à jour de l'état de croyance est faite par la règle de Bayes (équation 2.6) (ligne 10). Finalement, la mise à jour de l'arbre d'états de croyance est réalisée (ligne 11).

Dans le but d'être plus efficaces, les algorithmes de résolution en ligne cherchent à limiter le nombre d'états de croyance qui constituent l'arbre de recherche en fixant la profondeur de recherche ou en choisissant de développer les nœuds les plus pertinents. De cette façon, les approches diffèrent dans la façon que les sous-routines des lignes 5 et 6 dans l'algorithme 4 sont implémentées. Suivant [Ross *et al.*, 2008b], nous pouvons classifier ces différentes approches en trois catégories : approches basées sur une recherche de type *branch-and-bound* ; approches basées sur une recherche par échantillonnage (Monte Carlo) ; et approches basées sur une recherche guidée de manière heuristique.

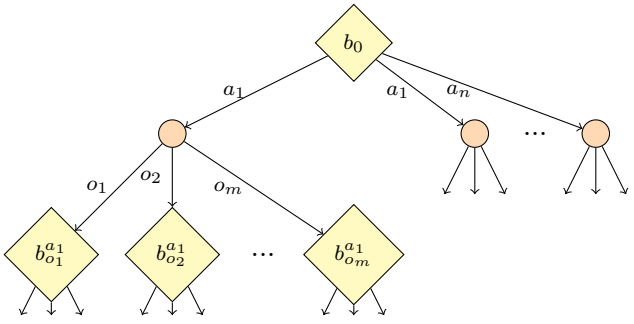


FIGURE 2.11 – Un arbre de recherche $\mathcal{T}$ en partant de b_0.

Algorithme 4: Schéma général des algorithmes de résolution en ligne pour les POMDP [Ross *et al.*, 2008b].

entrée : POMDP
statique : $\mathcal{T}$: arbre de recherche dans l'espace d'états de croyance ;
D : profondeur de recherche ;

```
Initiliser b_c ← b_0;
Initialiser T avec seulement b_c;
while mission non terminée do
    while délai pour la planification non écoulé do
        b* ← choisir le prochain nœud à développer ;
        Développer b* ;
        Mise à jour des parents de b* jusqu'à la racine ;
    Exécuter la meilleure action a* pour b_c ;
    Prendre une observation o ;
    Mettre à jour par la règle de Bayes (éq. 2.6) l'état de croyance b_c ← b_{c a*}^o ;
    Mettre à jour l'arbre pour que b_c soit le nouveau nœud racine ;
```

Méthodes basées sur une recherche de type *branch-and-bound*

Dans un algorithme basé sur une recherche de type *branch-and-bound* pour la résolution en ligne des POMDP, les bornes supérieure et inférieure de la fonction de valeur sont calculées au début du processus de résolution, et sont mises à jour lors du calcul en ligne. Ces bornes dirigent la recherche de type *branch-and-bound*, qui se compose d'un recensement systématique de tous les nœuds possibles, et au fur et à mesure que l'optimisation se déroule, la méthode élimine des sous-ensembles de nœuds sous-optimaux à l'aide des bornes supérieure et inférieure. Les bornes sont calculées au niveau des feuilles puis propagées vers les nœuds parents au cours de l'optimisation. Ainsi :

$$L_{\mathcal{T}}(b) = \begin{cases} \underline{V}(b), & \text{si } b \in \mathcal{F}(\mathcal{T}) \\ \max_{a \in A} QL_{\mathcal{T}}(b,a), & \text{sinon} \end{cases} \tag{2.44}$$

$$QL_{\mathcal{T}}(b,a) = r(b,a) + \gamma \sum_{o \in \Omega} p(o|b,a) L_{\mathcal{T}}(b_a^o), \tag{2.45}$$

$$U_{\mathcal{T}}(b) = \begin{cases} \overline{V}(b), & \text{si } b \in \mathcal{F}(\mathcal{T}) \\ \max_{a \in A} QU_{\mathcal{T}}(b,a), & \text{sinon} \end{cases} \tag{2.46}$$

$$QU_{\mathcal{T}}(b,a) = r(b,a) + \gamma \sum_{o \in \Omega} p(o|b,a) U_{\mathcal{T}}(b_a^o), \tag{2.47}$$

où, $\mathcal{F}(\mathcal{T})$ représente l'ensemble de nœuds feuilles de l'arbre de recherche $\mathcal{T}$, $U_{\mathcal{T}}(b)$ et $L_{\mathcal{T}}(b)$ les bornes supérieure et inférieure de $V^*(b)$ pour un état de croyance b, $QU_{\mathcal{T}}(b,a)$ et $QL_{\mathcal{T}}(b,a)$ représentent les bornes correspondant à $Q^*(b,a)$, et $\underline{V}(b)$ et $\overline{V}(b)$ les bornes utilisées pour les feuilles, souvent initialisées hors ligne par les méthodes approximatives discutées dans la section 2.3. Ces équations équivalent à l'équation de Bellman 2.9 appliquée aux bornes.

Étant donné ces bornes, le principe de la technique de *branch-and-bound* consiste à éliminer les nœuds considérés sous-optimaux : si une action a dans un état de croyance b a sa borne supérieure $QU_{\mathcal{T}}(b,a)$ plus petite que la borne inférieure d'une action $\tilde{a}$, $QL_{\mathcal{T}}(b,\tilde{a})$, nous savons que $Q^*(b,\tilde{a}) \geq Q^*(b,a)$; et donc a est sous-optimale. De cette façon, l'algorithme peut couper le sous-arbre de a, afin qu'aucun état de croyance atteint par a ne soit considéré dorénavant.

L'algorithme RTBSS. L'algorithme *RTBSS - Real-Time Belief Space Search* [Paquet *et al.*, 2006] est basé sur la méthode *branch-and-bound*. Dans RTBSS la sous-routine de la ligne 5 de l'algorithme 4 retournent simplement l'état de croyance courant. La sous-routine de la ligne 7 n'exécute aucune opération, car le nœud correspondant à l'état de croyance courant n'a pas de parents. La sous-routine de la ligne 4 met en place une recherche en profondeur d'abord jusqu'à un horizon D. Au fur et à mesure que les valeurs des bornes aux nœuds de la recherche en profondeur sont actualisées, l'algorithme coupe les actions considérées comme sous-optimales.

L'efficacité de RTBSS dépend grandement de la précision des bornes initiales qui sont calculées hors ligne. Il est connu que des algorithmes de résolution en ligne peuvent améliorer la précision d'une fonction de valeur approchée par un algorithme de résolution hors ligne sous-optimal [Puterman, 1994, Hauskrecht, 2000] [Paquet *et al.*, 2006] démontre que la technique de RTBSS améliore grandement, dans certain cas, la politique donnée par un algorithme de résolution hors ligne.

Méthodes basées sur une recherche par l'échantillonnage

Les méthodes basées sur une recherche par échantillonnage se sont intéressées à proposer une alternative de résolution pour des problèmes où le nombre d'observations est relativement grand. Quand le nombre d'observations est grand, le facteur de branchement est assez important lors du développement des nœuds de l'arbre de recherche. L'idée de ces approches est de concentrer la recherche sur les observations les "lus probables" en échantillonnant un sous-ensemble d'observations à chaque développement et en ne considérant seulement que les états de croyance atteignables par les observations qui ont été tirées. Cette stratégie est employée par les algorithmes proposés dans [McAllester et Singh, 1999, Bertsekas et Castanon, 1999].

Plus précisément, ces approches consistent à développer l'arbre de recherche avec un horizon D fixé, et, au lieu d'explorer toutes les observations pour chaque action, elles utilisent C observations qui sont échantillonnées suivant la probabilité $p(o|b,a)$. Ensuite, cette probabilité est approchée par les fréquences des observations. Ces approches s'appuient sur le fait qu'un petit nombre d'observations est suffisant pour avoir une bonne estimation de la valeur, en comptant sur les observations les plus probables dans le calcul de $Q^*(a,b)$. Pour ces algorithmes, la valeur de la borne inférieure des nœuds est donc approchée par :

$$QL_{\mathcal{T}}(b,a) \leftarrow r(b,a) + \gamma \sum_{o \in Z | N_o(Z) > 0} \frac{N_o(Z)}{C} L_{\mathcal{T}}(b_a^o) \tag{2.48}$$

où, $Z = \{o_1, o_2, .., o_C\}$ est un sous-ensemble de Ω de C observations, et $\frac{N_o(Z)}{C}$ est la fréquence observée pour chaque observation dans Z. L'inconvénient de ces approches est qu'aucun élagage au niveau des actions n'est plus possible, dû au fait que l'estimation par échantillonnage (Monte Carlo) ne garantit pas que les bornes seront correctement propagées [Hauskrecht, 2000]. De plus, dans [McAllester et Singh, 1999], la borne inférieure des feuilles est approchée par la récompense immédiate associée à $r(b,a)$. Ceci peut être amélioré en utilisant une fonction de valeur approchée par un algorithme de résolution hors ligne.

Dans [Bertsekas et Castanon, 1999], une technique d'échantillonnage est utilisée aussi pour approcher la valeur espérée pour chaque action, en supposant que la politique initiale (calculée hors ligne) sera exécutée dans la suite. Cette estimation est réalisée à partir de M trajectoires de profondeur D. Ainsi, le facteur de branchement $|A|$ influence seulement le calcul de $L_{\mathcal{T}}(b)$ au premier niveau. En conséquence, cet algorithme est plus efficace que [McAllester et Singh, 1999]. Par contre, la bonne performance de cette approche dépend fortement de la qualité de la politique initiale calculée hors ligne.

Méthodes basées sur une recherche heuristique pour la résolution en ligne

Contrairement aux méthodes de recherche de type *branch-and-bound* ou par échantillonnage, qui s'intéressent à réduire le facteur de branchement dans l'arbre, les méthodes basées sur une recherche heuristique s'intéressent à développer uniquement les nœuds les plus pertinents vis-à-vis de la diminution de l'erreur d'approximation de la fonction de valeur. Autrement dit, le nœud le plus pertinent est celui qui aidera l'algorithme à prendre des bonnes décisions le plus vite possible.

[Satia et Lave, 1973] ont proposé un des premiers algorithmes *pour la résolution en ligne* de POMDP (construction incrémentale de l'arbre de recherche). Cet algorithme de recherche heuristique suggère de résoudre le MDP sous-jacent d'un problème POMDP, en utilisant sont résultat d'optimisation comme borne supérieure de la fonction de valeur, et comme borne inférieure n'importe quelle politique dite raisonnable. L'heuristique que les auteurs proposent pour guider la recherche est basée sur la différence de valeur entre les deux bornes aux feuilles. La différence peut être *propagée* vers la racine et *pondérée* par la probabilité que cette feuille soit visitée au long du processus, afin de qualifier la contribution de cette feuille dans la diminution de l'erreur d'approximation de la fonction de valeur à la racine.

Plus précisément, étant donné un historique $h_d = \{a_0, o_1, ..., o_{d-1}, a_{d-1}, o_d\}$, on définit $Pr(h_d|b, \pi)$ comme étant la probabilité d'observer cet historique en partant de b et en suivant la politique π. Formellement, cette probabilité peut être calculée par :

$$Pr(h_d|b, \pi) = \prod_{i=1}^{d} Pr(o_i|b^{h_{i-1}}, \pi(b^{h_{i-1}})), \text{ où } b^{h_0} = b. \tag{2.49}$$

La contribution de cet historique à l'erreur d'approximation de $V^*(b)$ peut être définie par :

$$e_{\mathcal{T}}(b_c, b) = \gamma^{d(h_{\mathcal{T}}^{b_c,b})} Pr(h_{\mathcal{T}}^{b_c,b}|b_c, \pi^*)(V^*(b) - L_{\mathcal{T}}(b)) \tag{2.50}$$

où $h_{\mathcal{T}}^{b_c,b}$ représente l'historique d'actions et d'observations qui mène b_c à b dans l'arbre de recherche, et $d(h_{\mathcal{T}}^{b_c,b})$ la profondeur b dans l'arbre en partant de b_c. Comme en pratique on ne connait pas à l'avance V^* et π^*, il est proposé d'approcher la valeur de $e_{\mathcal{T}}(b_c, b)$ par :

$$\hat{e}_{\mathcal{T}}(b_c, b) = \gamma^{d(h_{\mathcal{T}}^{b_c,b})} Pr(h_{\mathcal{T}}^{b_c,b}|b_c, \hat{\pi}_{\mathcal{T}})(U_{\mathcal{T}}(b) - L_{\mathcal{T}}(b)) \tag{2.51}$$

où $\hat{\pi}_{\mathcal{T}}(b_c, b)$ représente la séquence d'actions qui mène de b_c à b.

À partir de cette approximation, nous pouvons formaliser l'heuristique de choix de nœud à développer que [Satia et Lave, 1973] utilise par :

$$b \leftarrow \underset{b \in \mathcal{F}(\mathcal{T})}{\arg\max}\, \gamma^{d(h_{\mathcal{T}}^{b_c,b})} Pr(h_{\mathcal{T}}^{b_c,b}|b_c, \hat{\pi}_{\mathcal{T}})(U_{\mathcal{T}}(b) - L_{\mathcal{T}}(b)) \tag{2.52}$$

où, $\mathcal{F}(\mathcal{T})$ représente l'ensemble de feuilles candidates.

L'algorithme BI-POMDP. [Washington, 1997] a proposé l'algorithme BI-POMDP, qui est basé sur l'algorithme AO* de [Nilsson, 1980, Hansen et Zilberstein, 2001] légèrement modifié pour ajouter les bornes supérieure et inférieure de la fonction de valeur. BI-POMDP pré-calcule comme borne inférieure le *min MDP*, c'est-à-dire, la valeur la plus basse pour chaque état pondéré par l'état de croyance. Il utilise comme borne supérieure l'approximation QMDP [Littman *et al.*, 1995]. Le choix du prochain nœud à développer s'appuie sur le choix du meilleur graphe, c'est-à-dire que la recherche va être orientée vers des feuilles qui sont atteignables en suivant la politique gloutonne.

L'algorithme AO* suggère de développer tous les nœuds qui appartiennent au meilleur graphe $\mathcal{G}$, par contre [Washington, 1997] propose de choisir celui qui maximise $U_{\mathcal{G}}(b) - L_{\mathcal{G}}(b)$. En conclusion, cette heuristique guide la recherche vers des nœuds qui sont atteignables par les actions les plus prometteuses, au sens où l'erreur entre la borne supérieure et inférieure est la plus importante.

Dans la suite de cette étude bibliographique, nous nous intéresserons à l'algorithme de résolution en ligne AEMS [Ross et Chaib-Draa, 2007], un des plus récents de la littérature qui se base sur les heuristiques de [Satia et Lave, 1973] et de [Washington, 1997]. Plus précisément, nous nous intéressons à cet algorithme car il nous sera utile lors du chapitre 6 de cette thèse.

L'algorithme AEMS. L'algorithme de recherche heuristique *AEMS - Anytime Error Minimization Search* a été proposé par [Ross et Chaib-Draa, 2007]. Cet algorithme se base sur les techniques heuristiques proposées par [Satia et Lave, 1973] et [Washington, 1997]. Son idée principale est de chercher à réduire le plus vite possible l'erreur d'approximation de la fonction de valeur au nœud racine [Satia et Lave, 1973], tout en donnant la préférence aux feuilles atteignables par une politique gloutonne [Washington, 1997]. Des études théoriques menés dans [Ross *et al.*, 2008a] présentent une l'analyse théorique des heuristiques utilisées dans AEMS, déterminant des conditions suffisantes pour que ces heuristiques garantissent d'obtenir des actions ϵ-optimale en temps limité.

Comme la plupart des algorithmes de résolution en ligne, AEMS dispose de deux phases : une pour la planification, et l'autre dédiée à l'exécution de la politique. Nous allons détailler la phase de planification.

Tout d'abord, AEMS initialise les bornes supérieure et inférieure qui seront utilisées par la suite pour guider l'optimisation. La borne supérieure est initialisée par l'approximation QMDP [Littman *et al.*, 1995], et la borne inférieure par une fonction de valeur calculée avec PBVI [Pineau *et al.*, 2003], en limitant le nombre d'itérations pour un nombre très faible d'états de croyance.

La première étape de la phase de planification qui consiste à choisir la prochaine feuille à développer (ligne 5 de l'algorithme 4), est réalisée suivant l'heuristique présentée dans l'équation 2.52. Ici, nous noterons :

$$\Upsilon^*_{\mathcal{T}}(b_c) \leftarrow \arg\max_{b \in \mathcal{F}(\mathcal{T})} \varkappa^*_{\mathcal{T}}(b), \text{ avec} \tag{2.53}$$

$$\varkappa^*_{\mathcal{T}}(b) = \gamma^{d(h^{b_c,b}_{\mathcal{T}})} Pr(h^{b_c,b}_{\mathcal{T}} | b_c, \hat{\pi}_{\mathcal{T}})(U_{\mathcal{T}}(b) - L_{\mathcal{T}}(b)) \tag{2.54}$$

Ensuite, AEMS développe et met à jour les bornes de la feuille choisie suivant l'algorithme 5. Pour la développer, AEMS commence par l'évaluation des bornes des prochaines feuilles associées aux paires action-observation (ligne 3 à 5). L'étape suivante est consacrée à la mise à jour de bornes $QL_{\mathcal{T}}(b,a)$ et $QU_{\mathcal{T}}(b,a)$ (lignes 6 à 7). Ensuite, le choix heuristique de l'observation se rapporte à celle dont la différence entre la valeur des bornes, pondérée par la probabilité que l'observation soit reçue, est maximale (ligne 8). La valeur heuristique de cette feuille est ensuite propagée au nœud parent b (nœud en développement) (lignes 9 et 10). Puis, le calcul de la valeur des bornes du nœud en développement est basée sur la valeur des bornes de ses feuilles, ainsi que sur la détermination de l'action optimale pour ce nœud (lignes 11 à 13). Finalement, le choix heuristique de l'action amenant à la feuille la plus pertinente est fait en considérant l'action qui maximise la borne supérieure [1] (ligne 14), ainsi que la propagation de la valeur des heuristiques de la feuille concernée (ligne 15 et 16).

1. [Ross et Chaib-Draa, 2007] présentent deux versions d'AEMS – AEMS1 et AEMS2 – pour lesquelles le choix de l'action heuristique $a^{\mathcal{T}}_b$ diffère, nous présentons ici le choix heuristique d'AEMS2, étant donné qu'AEMS2 présente des meilleurs résultats en termes de revenu moyen.

Algorithme 5: Sous-routine développer(b) d'AEMS [Ross *et al.*, 2008b].

entrée : POMDP ;
b : état de croyance à développer.
statique : b_c : l'état de croyance courant de l'agent ;
$\mathcal{T}$: arbre de recherche dans l'espace d'états de croyance ;
$\underline{V}$: borne inférieure ;
$\overline{V}$: borne supérieure.

1 **foreach** $a \in A$ **do**
2 **foreach** $o \in \Omega$ **do**
3 $b' \leftarrow$ mise à jour par la règle de Bayes avec b, a et o (équation 2.6) ;
4 $U_{\mathcal{T}}(b') \leftarrow \overline{V}(b')$;
5 $L_{\mathcal{T}}(b') \leftarrow \underline{V}(b')$;
6 $QL_{\mathcal{T}}(b,a) \leftarrow r(b,a) + \gamma \sum_{o\in\Omega} p(o|b,a) L_{\mathcal{T}}(b_a^o)$;
7 $QU_{\mathcal{T}}(b,a) \leftarrow r(b,a) + \gamma \sum_{o\in\Omega} p(o|b,a) U_{\mathcal{T}}(b_a^o)$;
8 $o_b^{\mathcal{T}} \leftarrow \arg\max_{o\in\Omega} \gamma p(o|b,a)(U_{\mathcal{T}}(b_a^o) - L_{\mathcal{T}}(b_a^o))$;
9 $\varkappa^*_{\mathcal{T}}(b,a) \leftarrow \gamma p(o_b^{\mathcal{T}}|b,a)(U_{\mathcal{T}}(b_a^{o_b^{\mathcal{T}}}) - L_{\mathcal{T}}(b_a^{o_b^{\mathcal{T}}}))$;
10 $\Upsilon^*_{\mathcal{T}}(b,a) \leftarrow b_a^{o_b^{\mathcal{T}}}$;
11 $L_{\mathcal{T}}(b) \leftarrow \max\{\max_{a\in A} QL_{\mathcal{T}}(b,a), L_{\mathcal{T}}(b)\}$;
12 $U_{\mathcal{T}}(b) \leftarrow \min\{\max_{a\in A} QU_{\mathcal{T}}(b,a), U_{\mathcal{T}}(b)\}$;
13 $a_b^* \leftarrow \arg\max_{a\in A} L_{\mathcal{T}}(b)$;
14 $a_b^{\mathcal{T}} \leftarrow \arg\max_{a\in A} QU_{\mathcal{T}}(b,a)$;
15 $\varkappa^*_{\mathcal{T}}(b) \leftarrow \varkappa^*_{\mathcal{T}}(b, a_b^{\mathcal{T}})$;
16 $\Upsilon^*_{\mathcal{T}}(b) \leftarrow b_{a_b^{\mathcal{T}}}^{o_{a_b^{\mathcal{T}}}^{\mathcal{T}}}$;

La mise à jour des valeurs des parents du nœud développé jusqu'au nœud racine doit être faite pour finaliser l'étape de planification. AEMS met à jour la valeur des nœuds parents, leurs bornes et les valeurs des heuristiques suivant l'algorithme 6.

Ainsi, AEMS combine les avantages des heuristiques de [Satia et Lave, 1973] et de [Washington, 1997]. Avant tout, AEMS encourage la recherche vers les feuilles où l'erreur $U_{\mathcal{T}}(b) - L_{\mathcal{T}}(b)$ pondérée par la probabilité que cette feuille soit rencontrée dans le futur est importante, comme dans [Satia et Lave, 1973]. Cette valeur heuristique et la feuille concernée sont propagées vers la racine grâce aux opérations des lignes 3 et 4 dans l'algorithme 6. De cette façon, la recherche est aussi focalisée vers les chemins les plus probables, ce qui permet d'aborder des problèmes avec un grand nombre d'observations. De plus, AEMS, comme BI-POMDP, concentre la recherche vers des feuilles atteignables par des actions qui semblent être optimales vis-à-vis de la borne supérieure.Ceci est utile dans des problèmes avec un grand nombre d'actions. D'autre part, il est possible d'élaguer les actions pour lesquelles $QU_{\mathcal{T}}(b,a) < L_{\mathcal{T}}(b)$, en évitant que des feuilles associées à des actions dominées soient développées, comme dans l'algorithme RTBSS (méthode *branch-and-bound*).

Il est à noter que l'heuristique utilisée par AEMS n'est pas très différente de celle de HSVI ou celle de SARSOP. Toutefois, nous rappelons que dans la résolution hors ligne, le choix du prochain état de croyance à visiter est réalisé à chaque appel de la fonction d'exploration (voir algorithme 3). Autrement dit, dans HSVI ou SARSOP, il n'y a pas de propagation d'une valeur heuristique vers l'état de croyance de départ. De plus, la profondeur de recherche n'est pas définie : on avance en profondeur jusqu'à ce que le critère d'arrêt soit atteint (comme par exemple $\overline{V}(b) - \underline{V}(b) > \epsilon\gamma^{-t}$ dans le cas de HSVI). Au contraire, AEMS construit l'arbre d'états de croyance petit-à-petit en donnant préférence au développement de la feuille la plus prometteuse selon la valeur heuristique qui a été propagée vers le nœud racine.

Dans la suite, nous terminerons notre étude bibliographique par une présentation des

Algorithme 6: Sous-routine de mise à jour des nœuds parents de b d'AEMS [Ross *et al.*, 2008b].

entrée : POMDP ;
b' : état de croyance b' à partir duquel tous les parents auront leur valeur mise à jour.
statique : b_c : l'état de croyance courant de l'agent ;
$\mathcal{T}$: arbre de recherche dans l'espace d'états de croyance ;

```
1 while b' ≠ b_c do
2     Prendre (b, a) tels que l'action a dans l'état de croyance b amène à l'état de croyance b' ;
3     Réaliser les opérations des lignes 6 à 10 de l'algorithme 5 pour (b, a) ;
4     Réaliser les opérations des lignes 11 à 16 de l'algorithme 5 pour b ;
5     b' ← b
```

modèles factorisé et avec observabilité mixte qui ont été récemment proposés dans la littérature.

2.5 Les modèles POMDP factorisés et avec observabilité mixte.

2.5.1 Le modèle POMDP factorisé

Dans les POMDP, les fonctions de transition d'état, d'observation et de récompense utilisent l'espace d'état pour la définition des domaines. Pour les problèmes avec un espace d'état important, des techniques ont été développées pour représenter ces fonctions de transition, d'observation et de récompense de manière compacte et structurée [Poupart, 2005].

Une des récentes avancées, en ce qui concerne la résolution des POMDP et la représentation des problèmes réalistes, exploite des propriétés structurelles du problème [Poupart, 2005], afin de résoudre des problèmes de grande taille sans énumérer les états individuels. Un lien avec ce qui a été développé pour les réseaux bayésiens dynamiques (DBN) peut être fait ici.

Les réseaux bayésiens permettent de représenter des distributions de probabilité jointes sur un ensemble de variables à partir d'une connaissance a priori entre la relation entre ces variables [Djian *et al.*, 1995, Marchand et Chaumette, 1997]. En particulier, les réseaux bayésiens dynamiques, *DBN - Dynamic Bayesian Network*, [Dean et Kanazawa, 1989, Murphy, 2002] sont une extension aux réseaux bayésiens et permettent de représenter l'évolution des variables aléatoires du réseau selon un axe temporel discret. Dans plusieurs domaines, il est possible de définir les fonctions de transition d'état, d'observation, et de récompense en fonction des variables d'états, variables d'observations et variables d'actions. Ceci permet une représentation compacte du problème sous forme de DBN, mais aussi graphique (voir figure 2.12).

L'état $s \in S$ d'un POMDP factorisé regroupe toutes les variables pertinentes pour décrire l'univers du problème. Une variable de cet univers est représentée par une variable d'état s_i avec un domaine $dom(s_i)$. Donc, chaque état s est une conjonction des N variables d'état instanciées $s_1, s_2, .., s_N$. La taille des états est exponentielle en N, avec $|S| = |dom(s_1)| \times |dom(s_2)| \times ... \times |dom(s_N)|$. De même, il est possible de décomposer l'espace d'actions A et l'espace d'observations Ω : chaque action a correspondrait à une instanciation jointe des variables d'action, et chaque observation o correspondrait à une instanciation jointe des variables d'observation.

Les probabilités conditionnelles $p(s'|s,a)$ et $p(o|s',a)$ peuvent être représentées par un graphe acyclique dirigé, comme celui de la figure 2.12. Dans ce graphe, les nœuds sont les variables d'état, d'action et d'observation, et les arêtes représentent les dépendances probabilistes. Les nœuds sont disposés en deux colonnes successives dans le temps. Chaque nœud s'_i a une table de probabilité conditionnelle associée $p(s'_i \mid parents(s'_i))$, qui spécifie

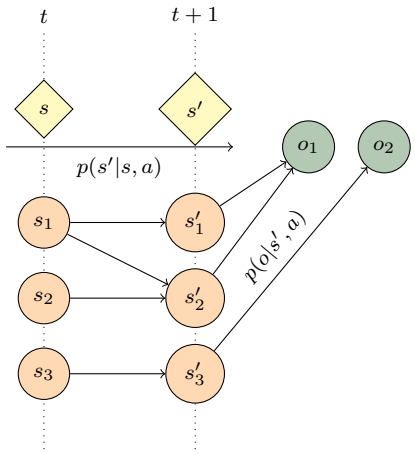


FIGURE 2.12 – Modèle factorisé de la fonction de transition d'état et d'observation d'un POMDP à 3 variables d'état et 2 variables d'observation pour une action a.

une distribution de probabilité conditionnelle par rapport à ses variables d'états parentes. L'utilisation des DBN permet la décomposition de la fonction de transition d'états sous forme d'un produit des distributions de probabilités conditionnelles de la fonction de transition de chaque variable d'état (voir figure 2.13(a)),

$$\begin{aligned} p(s'|s,a) &= p(s'_1, s'_2, s_3|s_1, s_2, s_3, a) \\ &= p(s'_1|s_1, a)p(s'_2|s_1, s_2, a)p(s'_3|s_3, a) \end{aligned}$$

et ceci à condition que s'_1, s'_2 et s'_3 soient indépendantes.

De même, la fonction d'observation peut être décomposée en produit des distributions de probabilités conditionnelles de chaque variable d'observation :

$$\begin{aligned} p(o|s',a) &= p(o_1, o_2|s'_1, s'_2, s'_3) \\ &= p(o_1|s'_1, s'_2)p(o'_2|s_3) \end{aligned}$$

à condition que o_1 et o_2 soient indépendantes.

De plus, pour la représentation de la fonction de récompense, d'autres propriétés des DBN peuvent être exploitées, comme par exemple l'additivité. Ainsi, la fonction de récompense peut être définie comme la somme de deux fonctions, où chacune est dépendante de certaines variables d'état et de certaines actions. Ou encore, nous pouvons citer une autre propriété : l'indépendance de contexte, basée sur le fait que certains variables d'état, d'action ou d'observation sont indépendantes les unes des autres dans certains contextes.

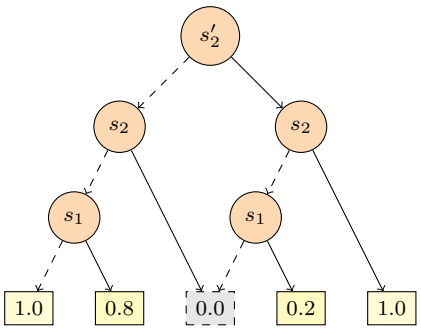
Par ailleurs, les fonctions de transition d'état et d'observation peuvent être représentées de manière très compactes à partir des tables de probabilités conditionnelles des DBN encodées en tant que diagrammes de décision. La taille des tables de probabilité est exponentielle en le nombre de parents d'une variable dans le pire cas, mais, en pratique, une variable aléatoire a peu de parents.

En ce sens, les *ADDs - Algebric Decision Diagrams* [Bahar *et al.*, 1997] proposent une représentation compacte de tables de probabilité des DBN. Les ADD sont des graphes de décision orientés acycliques dont les sous-graphes identiques sont fusionnés dans un seul sous-graphe, et les feuilles de valeur nulle ne sont pas stockées en mémoire. La figure 2.13(b) illustre l'ADD résultant de la table de probabilité conditionnelle de la variable d'état booléenne s'_1 (voir figure 2.13).

Dans sa thèse, [Poupart, 2005] propose de représenter les fonctions de transition, d'observation, de récompense ainsi que l'état de croyance par des ADD et propose, entre autres

s_1	s_2	s_2'	$p(s_2'\|s_1, s_2, a)$
F	F	F	1.0
F	F	V	0.0
F	V	F	0.0
F	V	V	1.0
V	F	F	0.8
V	F	V	0.2
V	V	F	0.0
V	V	V	1.0

(a) Table de probabilité conditionnelle pour l'état s_1 de la figure 2.12.



(b) Diagramme algébrique de décision pour la variable d'état s_2' de la figure 2.12

FIGURE 2.13 – Exemples : de table de probabilité conditionnelle et de diagramme algébrique de décision.

un des premiers algorithmes de résolution des POMDPs factorisés : l'algorithme Symbolic PERSEUS, qui est un algorithme d'itération approximative sur la fonction de valeur. L'idée est de représenter tous les vecteurs (dont les α-vecteurs) et matrices par des ADD, et de réaliser les opérations de mise à jour de la valeur et mise à jour de l'état de croyance de manière symbolique, en utilisant l'avantage de compacité des ADD.

D'autres chercheurs ont étendu les algorithmes HSVI [Sim *et al.*, 2008b] et FSVI [Shani *et al.*, 2008] de façon à travailler directement avec ce type de représentation factorisée du problème POMDP, sans avoir besoin d'énumérer tous sinon la plupart des états dans les tables de probabilités des transition d'état, d'observation ou de récompense. De plus, [Shani *et al.*, 2008] propose des opérations qui sont réalisées directement sur les ADD (ainsi très efficaces) pour les différents produits réalisés lors de la mise à jour de la valeur.

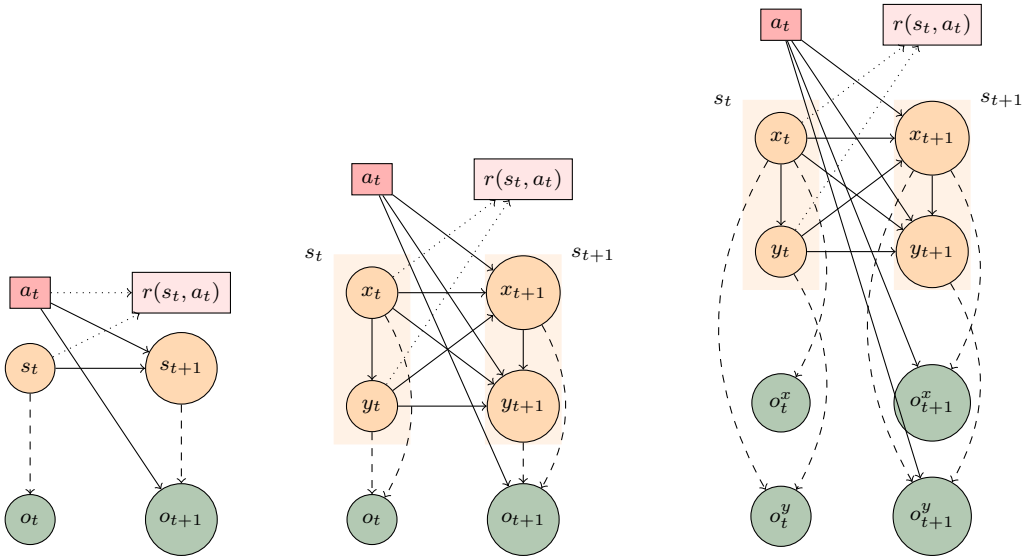
2.5.2 Les modèles POMDP avec observabilité mixte

Une autre extension pour le modèle POMDP récemment proposée est le modèle avec observabilité mixte [Ong *et al.*, 2009, López *et al.*, 2010], appelée MOMDP - *Mixed Observability Markov Decision Process*. Les MOMDP exploitent une structure particulière où certaines variables d'état sont complètement observables, ce qui permet de découpler le problème d'observabilité partielle de celui d'observabilité totale. Ce modèle factorisé met en évidence le fait que certaines variables d'état peuvent être observées complètement, ce qui conduit à un gain très important dans le temps de calcul des politiques. Nous rappelons qu'en effet la complexité de la mise à jours des algorithmes optimaux est exponentielle en le nombre d'observations du POMDP.

[Ong *et al.*, 2009] et [López *et al.*, 2010] proposent de factoriser l'espace d'état du POMDP en deux parties : une partie complètement observable représentée par une seule variable d'état x et une partie partiellement observable représentée par une variable d'état y. De cette façon, le couple (x, y) spécifie l'espace d'état complet, avec $|S| = |\mathcal{X}| \times |\mathcal{Y}|$, où $\mathcal{X}$ représente l'espace avec toutes les valeurs possibles de la variable x (resp. $\mathcal{Y}$ pour y).

Une des différences entre [Ong *et al.*, 2009] et [López *et al.*, 2010], est que l'approche proposée par [López *et al.*, 2010] factorise aussi l'espace des observations en accord avec la division de l'espace d'état. Dans les figures 2.14(b) et 2.14(c), nous représentons le modèle de transition pour les deux approches.

Nous allons maintenant définir formellement le MOMDP suivant [Ong *et al.*, 2009]. Un MOMDP est un n-uplet spécifié par $\{\mathcal{X}, \mathcal{Y}, A, \Omega, T_{\mathcal{X}}, T_{\mathcal{Y}}, O, R, (x_0, b_0)\}$, où la fonction de



(a) Modèle classique de transition du POMDP.

(b) Modèle de transition du MOMDP proposé par [Ong *et al.*, 2009].

(c) Modèle de transition du MOMDP proposé par [López *et al.*, 2010].

FIGURE 2.14 – Modèle factorisé de la transitions d'état et d'observation d'un POMDP et d'un MOMDP pour une action a_t.

probabilité conditionnelle $T_{\mathcal{X}}(x, y, a, x') = p(x'|x, y, a)$ donne la probabilité que la variable complètement observable ait la valeur x' sachant qu'on a réalisé l'action a dans l'état (x, y). La fonction de probabilité $T_{\mathcal{Y}}(x, y, a, x', y') = p(y'|x, y, a, x')$ donne la probabilité que la variable partiellement observable prenne la valeur y' sachant qu'on a réalisé l'action a dans l'état (x, y) et qu'on arrive à l'état x'.

L'avantage du modèle MOMDP devient claire quand on regarde de plus près les effets sur la représentation de l'état de croyance. Comme la variable x est complètement observable, la distribution de probabilité sur les états partiellement observables y est conditionnée par la valeur de x. Donc, l'agent a uniquement besoin de maintenir une distribution de probabilité sur les variables y, qui est noté $b_{\mathcal{Y}}$. De cette façon, l'état de croyance du modèle MOMDP est noté par le couple $(x, b_{\mathcal{Y}})$. On notera $\mathcal{B}_{\mathcal{Y}}$ l'espace d'états de croyance pour la variable y conditionnée par x : $\mathcal{B}_{\mathcal{Y}}(x) = \{(x, b_{\mathcal{Y}}), b_{\mathcal{Y}} \in \mathcal{B}_{\mathcal{Y}}\}$. $\mathcal{B}_{\mathcal{Y}}(x)$ est un sous-espace de $\mathcal{B}$, tel que $\mathcal{B} = \bigcup_{x \in \mathcal{X}} \mathcal{B}_{\mathcal{Y}}(x)$.

La mise à jour de l'état de croyance est maintenant définie par :

$$b^o_{\mathcal{Y},a}(y') = \eta \sum_{y' \in \mathcal{Y}} p(o|y', x', a) p(y'|x, y, a, x') p(x'|x, y, a) b_{\mathcal{Y}}(y) \tag{2.55}$$

où η est une constante de normalisation.

Dans le modèle MOMDP, la fonction de valeur peut être aussi paramétrée par des α-vecteurs. Celle-ci est alors notée par :

$$V(x, b_{\mathcal{Y}}) = \max_{\alpha \in \Gamma_{\mathcal{Y}}(x)} (\alpha \cdot b_{\mathcal{Y}}) \tag{2.56}$$

où α est maintenant un hyperplan sur l'espace $\mathcal{B}_y(x)$. De cette façon, la fonction de valeur sur l'espace complet d'états est paramétrée par un ensemble de $\Gamma_{\mathcal{Y}}(x)$, c'est-à-dire $\Gamma = \{\Gamma_{\mathcal{Y}}(x), x \in \mathcal{X}\}$.

Cette structure factorisée est utilisée de manière à réaliser toutes les opérations de mise à jour de l'état de croyance de la valeur sur un sous-espace plus petit. Dans [Ong *et al.*, 2009],

Algorithme 7: Opération de mise à jour de la valeur (*backup*) pour le modèle MOMDP de [Ong *et al.*, 2009]

entrée : MOMDP ;
$(x, b_{\mathcal{Y}})$: état de croyance dont la valeur est mise à jour.

foreach $a \in A$ **do**
  $\alpha_{a,x',o} \leftarrow \arg\max_{\alpha \in \Gamma_{\mathcal{Y}}(x')}(\alpha \cdot b^{o}_{\mathcal{Y},a})$;
foreach $a \in A$ **do**
  foreach $y \in \mathcal{Y}$ **do**
    $\alpha_a(y) \leftarrow$
    $r(x,y,a) + \gamma \sum_{o \in \Omega} \sum_{x' \in \mathcal{X}} \sum_{y' \in \mathcal{Y}} p(o|y',x',a) p(y'|x,y,a,x') p(x'|x,y,a) \alpha_{a,x',o}(y')$;
$\alpha' \leftarrow \arg\max_{a \in A}(\alpha_a \cdot b_{\mathcal{Y}})$;
$\Gamma_{\mathcal{Y}}(x) \leftarrow \alpha'$;

la compression du modèle POMDP en MOMDP permet un gain très important d'efficacité de l'algorithme d'itération approché sur la valeur, SARSOP, qui a été adapté à l'approche.

L'opération de mise à jour de la valeur (*backup*) pour un état de croyance $(x, b_{\mathcal{Y}})$ proposée dans [Ong *et al.*, 2009] est montrée dans l'algorithme 7. Les opérations de mise à jour des états de croyance et de la valeur présentées dans [López *et al.*, 2010] sont différentes, puisque la factorisation de l'espace d'observation du modèle MOMDP n'est pas la même. Nous invitons le lecteur intéressé à consulter directement [López *et al.*, 2010].

2.6 Conclusion et Bilan

Dans ce chapitre, nous avons présenté le modèle POMDP, les différents algorithmes de résolution exacte et approximative de POMDP qui ont été proposées et les extensions pour des modèles factorisés. Les algorithmes d'itération approchée sur la fonction de valeur pour une résolution hors ligne, tels que PBVI [Pineau *et al.*, 2003], PERSEUS [Spaan et Vlassis, 2005], HSVI [Smith et Simmons, 2005], FSVI [Shani *et al.*, 2007] et SARSOP [Kurniawati *et al.*, 2008] peuvent aujourd'hui calculer des politiques quasi-optimales pour des POMDP avec un espace d'état grand. Les algorithmes de résolution en ligne, tels que RTBSS [Paquet *et al.*, 2006] et AEMS [Ross et Chaib-Draa, 2007] se sont appuyés sur les différents développements proposés par les algorithmes de résolution hors ligne, en particulier pour approcher au mieux les bornes supérieure et inférieure de la fonction de valeur afin d'utiliser des heuristiques de recherche pertinentes. Ces approches peuvent aujourd'hui fournir des politiques partielles très rapidement. Les approches factorisées [Poupart, 2005, Sim *et al.*, 2008a, Shani *et al.*, 2008, Ong *et al.*, 2009, López *et al.*, 2010] ont aussi contribué à un gain d'efficacité algorithmique remarquable. Ces algorithmes peuvent aujourd'hui résoudre des problèmes réalistes de robotique.

Par contre, nous noterons que très peu des travaux se sont intéressés :

1. à l'implémentation algorithmique pour l'optimisation d'un critère mixte à long terme, contrairement à l'approche réactive de [Eidenberger et Scharinger, 2010], qui couple à la fois des récompenses définies sur les paires état-action avec une mesure de l'incertitude de l'état de croyance de l'agent. Ce type de critère dans un cadre POMDP peut s'avérer utile lorsque l'acquisition d'informations est importante [Candido et Hutchinson, 2011], par exemple pour les applications ayant une composante explicite de perception ;
2. à une prise en compte *formelle* de contraintes physiques [Miller *et al.*, 2009, Bai *et al.*, 2011], ou de contraintes permettant d'éviter la mise en danger du robot, ou encore qui sont parfois imposées par les agences de régulation ;

3. à la résolution en ligne des POMDP [Ross et Chaib-Draa, 2007, Sridharan *et al.*, 2008] où les contraintes temporelles sont dures et imposées par les architectures d'exécution dans laquelle les algorithmes POMDP sont embarqués. Plus précisément, les architectures d'exécution nécessitent une réponse immédiate (action à exécuter) des algorithmes de planification en ligne, ce qui est est très difficile à garantir pour deux raisons. Premièrement, les approches actuelles ne permettent pas de contrôler le temps passé dans chaque opération, comme par exemple $backup(b)$. Deuxièmement, dans certains cas, le problème à optimiser n'est pas connu à l'avance, ce qui empêche le calcul des bornes inférieure et supérieure hors ligne pour les heuristiques qui guident la recherche dans l'arbre d'états de croyance, très importantes pour les approches proposées dans [McAllester et Singh, 1999, Bertsekas et Castanon, 1999] par exemple.

Dans le chapitre suivant, nous présentons le modèle POMDP de la mission de détection et reconnaissance de cibles étudié dans le cadre de cette thèse. Puis, dans les chapitres subséquents, nous présenterons les travaux menés lors de cette thèse pour proposer des réponses alternatives et réalistes à chacun des points énumérés ci-dessus, de sorte à pouvoir résoudre en pratique cette mission de détection et de reconnaissance de cibles sur robot aérien en conditions réelles.

Deuxième partie

Percevoir pour décider et décider pour percevoir

Chapitre 3

Modélisation du problème de perception et de décision dans un cadre unifié

Ce chapitre a pour objectif de modéliser un scénario de détection et de reconnaissance de cibles par un hélicoptère autonome en environnement incertain. Ce scénario applicatif illustre le besoin dual de percevoir pour décider et décider pour percevoir. Nous avons vu que le modèle POMDP permet de prendre en compte deux types d'incertitude à la fois : une liée aux effets des actions, et l'autre liée à la sortie du traitement d'information. Tout d'abord, nous présenterons les variables d'états considérées dans ce modèle, ainsi que leur évolution dynamique comme effets des actions. Les coûts des actions est modélisée de façon réaliste, c'est-à-dire qu'ils reposent sur le temps de vol de l'agent hélicoptère lors de la réalisation des différentes actions. Ensuite, nous réalisons une étude du traitement d'information pour le capteur caméra choisi a été fait, afin de déterminer le type de réponse obtenu par le capteur pour la modélisation de la fonction d'observation, ce qui nous situe dans un cadre probabiliste et d'observabilité partielle de l'environnement. L'étude du traitement d'image est basée sur des données réelles qui ont été collectées lors de plusieurs campagnes de prise de vue de voitures en environnement réel. Finalement, la définition des buts de mission est présentée, celle-ci se décline en deux types de mission : une mission d'exploration pure, et une autre où vient se greffer un but d'atterrissage près d'une cible particulière. La première mission vise à identifier l'état caché du monde tout en tenant compte des coûts liés au déplacement de l'agent. L'autre mission consiste à identifier un modèle particulier de voiture parmi les différents modèles connus de la base de données afin d'atterrir proche de ce modèle recherché.

3.1 Définition d'un cadre unifié de perception et mission

Comme déjà mentionné, la détection et la reconnaissance de cibles par des véhicules aériens inhabités autonomes est un domaine de recherche actif [Wang *et al.*, 2012]. La stratégie de décision de haut niveau est généralement donnée par une règle écrite à la main dans ce genre de mission. De plus, la stratégie dépend des événements stochastiques comme par exemple, la détection d'une cible dans une zone donnée, ou le fait qu'une cible soit reconnue, qui peuvent survenir lors de l'exécution de la règle de décision. En raison de la complexité de la construction automatique des règles de décision séquentielle sous incertitude et observabilité partielle [Littman *et al.*, 1995, Sabbadin *et al.*, 2007], peu de systèmes

de drones construisent et optimisent des politiques automatiquement.

Des travaux précédents ont étudié des scénarios qui traitent à la fois du problème de perception et de décision en robotique mobile. Le modèle choisi est souvent celui des POMDP. Il a été montré que ce modèle permet de mener à la fois les deux problématiques de prise d'information et de satisfiabilité de but. Par exemple, dans le scénario robotique de [Smith et Simmons, 2004], la perception a été considérée comme un moyen pour satisfaire les buts symboliques de la mission. Par contre, dans [Spaan et Lima, 2009, Eidenberger *et al.*, 2009, Eidenberger et Scharinger, 2010], la perception de l'environnement était le but en soi de la mission. Cependant, le problème dual d'optimisation d'actions modifiant l'état physique et de celles affectant l'état des connaissances sous forme de POMDP dans le cadre des drones autonomes n'a pas été traité par d'autres personnes à notre connaissance.

Dans ce chapitre, nous avons étudié une mission de détection et reconnaissance de cible par un hélicoptère autonome, modélisée par un POMDP basé sur un nombre de zones à explorer et un nombre de cibles à détecter et à reconnaître. Nous pensons que ce travail est intéressant au moins pour deux raisons : (i) la mission de détection et reconnaissance de cibles est abordée comme un problème de planification séquentielle à long terme, avec des actions à la fois épistémiques (acquisition d'information par changement d'angle de vue, changement d'altitude, changement de zone) et affectant les buts de la mission (atterrissage, identification de la scène) ; (ii) le modèle d'observation du POMDP a été appris à partir de données réelles obtenues lors des campagnes de prise de vue de voitures, qui constituent les cibles à identifier, par un hélicoptère autonome.

3.1.1 Définition du problème de perception

La détection et la reconnaissance de cibles constituent le problème de perception à traiter. Il est connu que la détection et la classification d'objets peuvent souvent ne pas être atteintes par des mesures simples [Mihaylova *et al.*, 2002] : des stratégies de perception active sont nécessaires pour surmonter ces problèmes et permettre une approche autonome de reconnaissance de scène. La fusion d'une séquence d'observations améliore les résultats de détection [Eidenberger *et al.*, 2008] tout en tenant compte des coûts de déplacements ou de prise de vue supplémentaires. Dans ce cadre, une stratégie de planification est nécessaire, qui réalise un compromis évalué à long terme entre les bénéfices attendus d'une nouvelle mesure et les coûts implicites des actions.

Choix du capteur caméra pour la prise d'information

Dans notre application, au fur et à mesure que l'hélicoptère autonome évolue dans l'environnement, celui-ci doit acquérir de l'information concernant la présence et la nature des cibles en tenant compte de coûts associés au traitement d'information et au déplacement entre les différentes zones. L'information acquise est généralement traitée par des algorithmes qui dépendent de manière *ad hoc* de l'application et du capteur utilisé.

Dans des travaux connexes au sujet de cette thèse, comme par exemple [Eidenberger *et al.*, 2008, Spaan, 2008, Eidenberger et Scharinger, 2010], le capteur choisi est constitué d'une ou de plusieurs caméras. Pour ce type de capteur, différents algorithmes d'interprétation de scène sont aujourd'hui disponibles dans la littérature. Nous pouvons citer : *Unscented Transform* [Julier *et al.*, 2000] utilisé dans [Spaan et Lima, 2009] pour détecter la présence d'une personne dans la scène et pour estimer sa position ; [Sridharan et Stone, 2005] qui présente un algorithme de reconnaissance en ligne d'objets (ballon posé sur le sol) basé sur la segmentation couleur pour les robots mobiles ; [Denzler et Brown, 2002] qui utilise un classificateur basé sur la transformation d'un vecteur de pixels, qui caractérise l'image, en un vecteur de plus petite dimension qui regroupe les principaux composants d'intérêt de

l'image ; ou encore [Saux et Sanfourche, 2011] qui propose un algorithme de catégorisation des véhicules basé sur la mise en correspondance avec des modèles 3D à partir d'images aériennes.

L'information délivrée par ce type capteur peut aujourd'hui être traitée en ligne avec des résultats convenables. Il est important de noter que la plupart de ces algorithmes ont une sortie probabiliste, où l'information est souvent modélisée par une fonction gaussienne, ou encore par un ensemble de gaussiennes, qui associent une valeur moyenne et une matrice de covariance à une caractéristique. Le but est de trouver automatiquement la correspondance entre la réponse du classificateur et un objet du modèle. Cette classification dépend souvent de l'incertitude sur les classes d'objets et de la matrice de covariance.

D'autre part, dans une approche POMDP comme celle que nous proposons dans la suite, il est indispensable de bien connaître le fonctionnement et les limitations de l'algorithme de traitement visuel dans le but de pouvoir modéliser ses réponses en tant que fonction d'observation du POMDP. Ces aspects seront discutés dans la sous-section 3.2.3.

3.2 Modélisation du scénario en tant que POMDP

3.2.1 Variables d'état

Le nombre total d'états du POMDP dépend de plusieurs variables d'état qui sont discrétisées selon : le nombre de zones (N_z), le nombre d'altitudes de vol (N_h), le nombre de modèles de voiture (N_{models}) dans la base de données, ainsi qu'éventuellement un état terminal qui caractérise la fin de la mission. Nous supposons que les voitures, c'est-à-dire les cibles peuvent être dans n'importe quelle zone de l'environnement, et peuvent aussi représenter n'importe quel modèle connu de la base de données. Dans notre application, la base de données comporte 3 modèles possibles, soit $\{model_A,\ model_B,\ model_C\}$.

Les variables d'états sont telles que :

- z, avec N_z valeurs possibles, qui indique la position de l'hélicoptère autonome ;
- h, avec N_h valeurs possibles, qui indique l'altitude de vol de l'hélicoptère autonome ;
- $Id_{Ta_{z_1}}$ (respectivement $Id_{Ta_{z_2}}$, $Id_{Ta_{z_3}}$, etc), avec $N_{models} + 1$ valeurs possibles, qui indique l'identité ou l'absence d'un modèle de cible dans la zone 1 (respectivement dans la zone 2, dans la zone 3, etc.)

Ainsi, le nombre total d'états est donné par :

$$|S| = N_z \cdot N_h \cdot (N_{models} + 1)^{N_z} + T_s$$

où T_s représente l'état terminal.

Notez qu'aucune variable d'état ne formalise l'orientation des cibles dans les zones, et qu'aucune variable d'état ne formalise l'orientation du drone hélicoptère par rapport aux zones. Nous avons fait ce choix de modélisation car nous disposons d'un algorithme de traitement d'images qui peut tenir compte de cet absence d'information sur l'orientation tout en fournissant des réponses convenables. Cet aspect sera discuté dans la sous-section 3.2.3.

3.2.2 Dynamique du modèle

Les actions de haut niveau réalisées par l'hélicoptère autonome sont : changer de zone, changer d'altitude de vol, changer d'angle de vue. Le nombre d'actions de changement de zone dépend du nombre de zones considérées. Ces actions sont appelées $go\_to(\hat{z})$, où $\hat{z}$ représente la zone de destination. Changer l'altitude de vol dépend aussi du nombre d'altitudes considérées pour le vol de l'hélicoptère autonome. Ces actions sont appelées $go\_to(\hat{h})$, où $\hat{h}$ représente l'altitude désirée. L'action *change_view* change l'angle de vue de l'observation d'une zone,

la caméra pointant toujours au centre de cette zone. Donc le nombre total d'actions est : $|A| = N_z + N_h + 1$.

Avant de décrire la dynamique du modèle, nous présentons la notation utilisée : les variables primées représentent la variable de l'état successeur, et les variables non primées l'état courant. Ainsi nous définissons une fonction indicatrice et une fonction delta $\delta_x(x')$, comme suit :

Définition 3.2.1 $\mathbb{I}_{\{cond\}}$ *est la fonction indicatrice telle que :*

$$\mathbb{I}_{\{cond\}} = \begin{cases} 1, & \text{si } cond \text{ est vraie;} \\ 0, & \text{sinon.} \end{cases}$$

Définition 3.2.2 $\delta_x(x')$ *est la fonction Dirac telle que :*

$$\delta_x(x') = \begin{cases} 1, & \text{si } x = x'; \\ 0, & \text{sinon.} \end{cases} \tag{3.1}$$

Cette notation nous permet d'exprimer les différentes valeurs possibles prises par la variable de l'état successeur x'.

D'expérience, nous savons que le changement de zone, d'altitude et le changement d'angle de vue peuvent être considérées comme des actions déterministes. Cependant, le problème reste de type POMDP, car les observations des modèles des voitures sont probabilistes. De plus il est prouvé que la complexité pour résoudre un POMDP est essentiellement liée aux effets probabilistes des observations plutôt qu'aux effets probabilistes des actions [Sabbadin *et al.*, 2007].

En outre, chaque action du POMDP (changement de zone, d'altitude ou d'angle de vue, etc) donne lieu à une acquisition d'image et à l'exécution de l'algorithme de traitement d'image, qui fournit alors le symbole d'observation (voiture détectée, identifiée comme modèle A, etc) résultant de l'action du modèle POMDP. Comme la caméra est fixe, il est important de contrôler l'orientation de l'hélicoptère afin d'observer les différentes parties de l'environnement.

Nous allons maintenant décrire les modèles de fonctions de transition et de récompense du POMDP, formalisées par des équations mathématiques, qui reposent sur les variables d'état du problème.

Fonctions de transition et de récompense

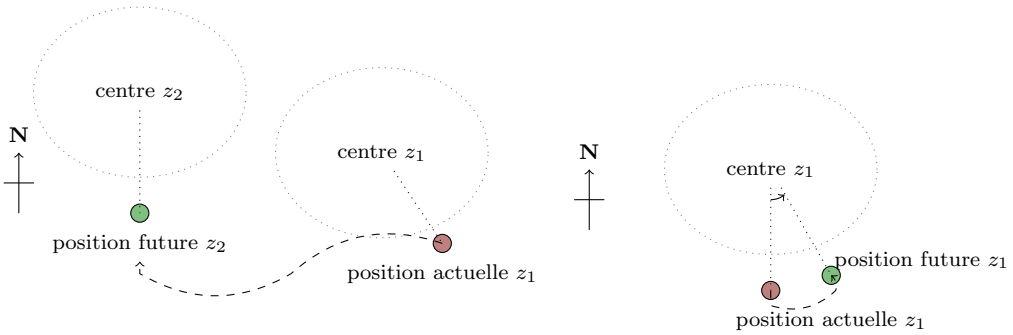
Pour définir la dynamique du modèle, nous caractérisons chaque action par une description textuelle expliquant comment les variables d'état évoluent une fois l'action appliquée, ainsi que la fonction de transition T, et la fonction de récompense R.

action go_to($\hat{z}$) : cette action amène l'hélicoptère autonome à la zone désirée. La dynamique est décrite ci-après, mais notez que si l'hélicoptère est dans un état terminal (T_s), cette action n'a ni effet ni coût associé (cas particulier non formalisé).

– fonction de transition :

$$T(s', \texttt{go\_to}(\hat{z}), s) = \delta_{\hat{z}}(z')\delta_h(h')\delta_{Id_{Ta_{z_1}}}(Id'_{Ta_{z_1}}) \ldots \delta_{Id_{Ta_{z_{N_z}}}}(Id'_{Ta_{z_{N_z}}})$$

Conformément à la définition de la fonction δ mentionnée précédemment, la fonction est différente de zéro seulement pour la transition vers l'état s' dans lequel les variables d'état "post-action" sont toutes égales aux variables d'état "pré-action", sauf la variable "zone" z' qui est égale à $\hat{z}$, la zone désirée.



(a) Changement de zone dans le plan horizontal.

(b) Changement d'angle de vue dans le plan horizontal.

FIGURE 3.1 – Schéma pour les actions de changement de zone et changement de vue pour une vue dans le plan horizontal.

– Fonction de récompense :

$$R(s, \texttt{go\_to}(\hat{\texttt{z}})) = -C_{z,\hat{z}} - C_{proc},$$

où $C_{z,\hat{z}}$ représente le coût du vol de z à $\hat{z}$. Ce coût modélise la consommation de carburant qui dépend de la distance entre les zones. La génération du POMDP prend en compte les coordonnées des zones. Ces coordonnées sont nécessaires pour générer un coût proportionnel à la distance entre zones. C_{proc} modélise le coût concernant le traitement d'information dans la zone d'arrivée (observation) qui suit l'exécution de l'action. Le temps de calcul de l'algorithme de traitement d'image est généralement assez court, si bien que $C_{z,\hat{z}} > C_{proc}$.

Il est important de noter qu'à chaque changement de zone, indépendamment de la position actuelle en cordonnées locales, l'hélicoptère autonome ira vers un point de rendez-vous dans la nouvelle zone avec une orientation déterminée (nord magnétique). La figure 3.1(a) illustre le schéma de l'action de changement de zone.

action go_to($\hat{\texttt{h}}$) : cette action amène l'hélicoptère autonome à l'altitude de vol désirée. Comme pour l'action go_to($\hat{\texttt{z}}$), si l'hélicoptère est dans un état terminal (T_s), cette action n'a ni effet ni coût.

– fonction de transition :

$$T(s', \texttt{go\_to}(\hat{\texttt{h}}), s) = \delta_z(z')\delta_{\hat{h}}(h')\delta_{Id_{Ta_{z_1}}}(Id'_{Ta_{z_1}}) \ldots \delta_{Id_{Ta_{z_{N_z}}}}(Id'_{Ta_{z_{N_z}}})$$

– Fonction de récompense :

$$R(s, \texttt{go\_to}(\hat{\texttt{h}})) = -C_{h,\hat{h}} - C_{proc},$$

où $C_{h,\hat{h}}$ représente le coût associé au changement d'altitude de h à $\hat{h}$ et modélise la consommation de carburant qui dépend de la différence d'altitude. C_{proc} modélise le coût du traitement de l'information qui a lieu à la fin de l'exécution de l'action (observation résultant du traitement d'image à partir de la nouvelle altitude de vol). Ces coûts sont typiquement moins élevés que le changement de zone. Nous avons $C_{z,\hat{z}} > C_{h,\hat{h}} > C_{proc}$.

action change_view : cette action est une action de haut niveau qui change l'angle de vue de l'hélicoptère autonome par rapport au centre de la zone explorée. Cette action de haut niveau ne change pas l'état du POMDP, parce que l'état du POMDP ne dépend que de la position z et de l'altitude h de l'hélicoptère. Mais, dans la réalité, cette action est traduite dans les composants d'exécution et de navigation du drone hélicoptère comme un déplacement circulaire autour du centre de la zone en question. Ce déplacement circulaire dépend de l'angle ϕ défini par le concepteur du système. Cette formalisation faisant abstraction du déplacement angulaire de l'hélicoptère nous permet d'éviter de rajouter une variable d'état dans le modèle, dont l'influence sur la valeur de la stratégie de haut niveau (POMDP) n'est pas sensible a priori. De cette façon, nous modélisons le message envoyé par le composant d'exécution de la politique du POMDP au composant de navigation comme un nouveau point de rendez-vous (*wait point* : W^p), qui est défini par :

$$\begin{array}{lcl} W^p_x & = & (x - zone_x)cos(\phi) + (y - zone_y)sin(\phi) + zone_x \\ W^p_y & = & -(x - zone_x)sin(\phi) + (y - zone_y)cos(\phi) + zone_y \\ W^p_{cap} & = & cap - \phi \end{array}$$

où, $\{x, y\}$ (respectivement $\{zone_x, zone_y\}$) définit la position de l'hélicoptère autonome (respectivement du centre de la zone) dans le plan horizontal par rapport aux coordonnées locales, et *cap* l'orientation de l'hélicoptère autonome par rapport au nord magnétique (voir figure 3.1(b)). Le coût de cette action est aussi proportionnel à la distance parcourue entre la position actuelle de l'UAV et le nouveau point de rendez-vous, et au coût du traitement d'information.

– fonction de récompense :

$$R(s, \texttt{change\_view}) = -C_{view} - C_{proc}.$$

Le coût C_{view} dépend de l'angle ϕ : plus l'angle qui définit le déplacement circulaire est important, plus ce coût est élevé. Nous avons $C_{view} > C_{proc}$.

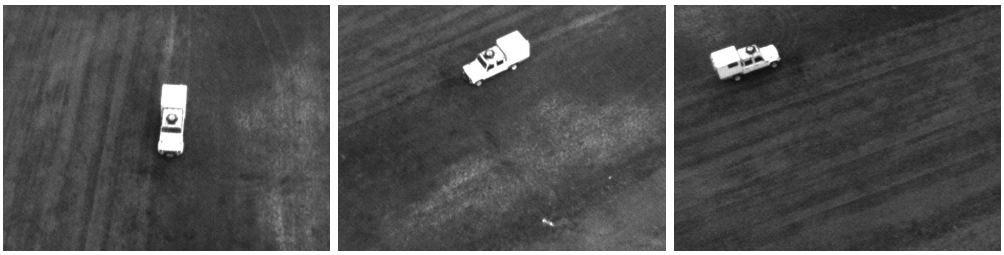
Dans la suite nous présentons le modèle d'observation de notre application.

3.2.3 Modèle d'observation

Le modèle de POMDP requiert une description fidèle, ou appropriée, des effets probabilistes des actions et des observations, ce qui est difficile à obtenir dans des applications réelles relativement complexes [Spaan et Lima, 2009]. Pour notre mission de détection et reconnaissance de cibles, nous avons appris automatiquement le modèle d'observation à partir de données réelles. Ce modèle repose sur un algorithme spécifique de traitement d'image (décrit dans la section 3.2.3). Pour notre application, 5 observations sont possibles ($|\Omega| = 5$) au dessus de la zone courante : {*voiture non détectée, voiture détectée mais non identifiée, voiture identifiée comme modèle A, voiture identifiée comme modèle B, voiture identifiée comme modèle C*}. L'apprentissage du modèle d'observation consiste à attribuer une probabilité a priori aux différentes sorties sémantiques possibles du traitement d'image pour une scène en particulier.

L'apprentissage hors ligne du modèle d'observation seul a été préféré à la mise en œuvre de techniques d'apprentissage par renforcement dans la mesure où, d'une part l'apprentissage des effets des actions et de la fonction de récompense ne sont pas utiles car connus, et d'autre part nous ne pouvions pas prendre le risque d'endommager un moyen expérimental coûteux au cours d'une phase d'apprentissage de la stratégie. Dans le cadre POMDP, des travaux se sont intéressés à l'apprentissage en ligne du modèle du POMDP sous-jacent [Ross *et al.*, 2011]. Une alternative possible est d'apprendre et de renforcer en ligne un modèle MDP dont les états sont les observations ou les croyances du modèle POMDP, ce qui revient à ne

Images pour le modèle A à 30 mètres d'altitude.

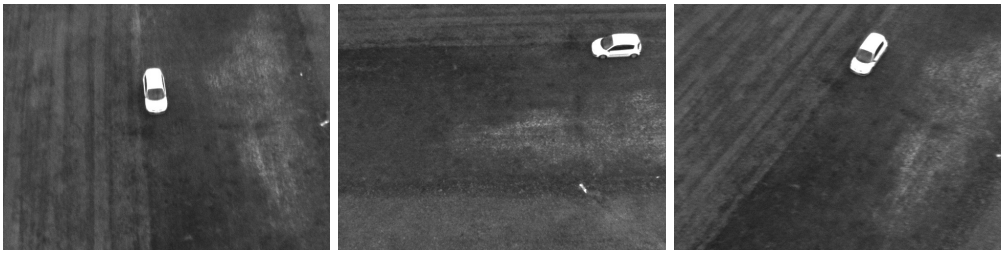


Images pour le modèle A à 40 mètres d'altitude.

Images pour le modèle B à 30 mètres d'altitude.

Images pour le modèle B à 40 mètres d'altitude.

Images pour le modèle C à 30 mètres d'altitude.



Images pour le modèle C à 40 mètres d'altitude.

FIGURE 3.2 – Exemple d'images recueillies lors des campagnes de prise de vue.

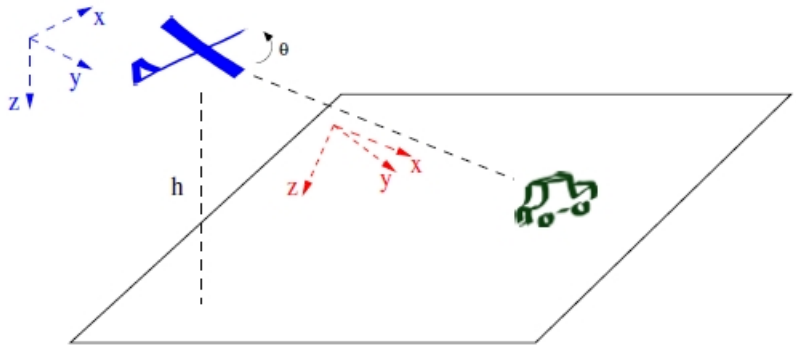


FIGURE 3.3 – Schéma de la prise de vue aérienne avec les paramètres dont l'algorithme de [Saux et Sanfourche, 2011] dépend (image issue de [Saux et Sanfourche, 2011]).

pas modéliser la dynamique sur l'état caché du système. Dans notre cas, nous cherchons à caractériser en laboratoire la réponse de l'algorithme de traitement d'image et la dynamique précise sur l'état caché du système, ce qui enduit un modèle de POMDP connu hors ligne mais résolu en ligne pour une scène particulière.

Campagnes de prise d'images

Pour apprendre le modèle d'observation du POMDP à partir de données réelles, nous avons effectué des campagnes de prise de vue en conditions réelles avec le drone et un ensemble de voitures connues. Dans la figure 3.2, nous montrons quelques images obtenues lors de ces campagnes. L'objectif a été d'acquérir un nombre suffisant d'images différentes à partir d'angles de vue différents et d'altitudes de vol différentes, de sorte à constituer une statistique suffisante pour l'apprentissage de la fonction d'observation du POMDP.

Dans la suite nous détaillons l'algorithme de traitement d'image que nous avons utilisé.

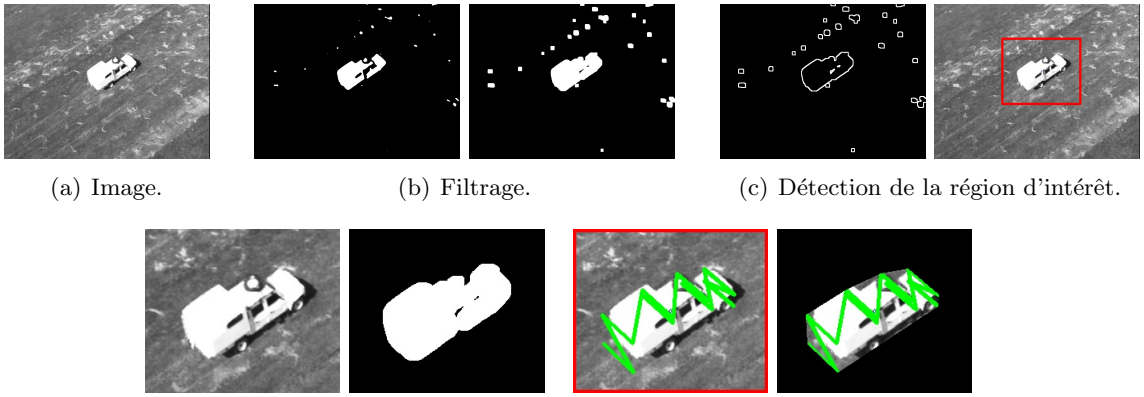
Algorithme de traitement d'image

L'observation des voitures est basée sur l'algorithme d'identification et de reconnaissance de véhicules par traitement d'image aérienne [Saux et Sanfourche, 2011]. Cet algorithme réalise une classification des véhicules basée sur des modèle 3D qui décrivent le profil des voitures. La description est assez simple puisque les catégories de voitures sont caractérisées seulement par la largeur, la hauteur et la longueur des véhicules.

Nous précisons que cet algorithme a été choisi pour plusieurs raisons :

- l'algorithme ne suppose pas une orientation a priori des véhicules dans la scène : cette flexibilité est d'une grande utilité lors de la modélisation puisque il n'est pas nécessaire de rajouter des variables d'état qui décrivent l'orientation des objets ;
- l'algorithme peut être enrichi sans difficulté par des modèles 3D de voitures plus fins ;
- l'algorithme dépend uniquement de paramètres tels que : l'altitude de vol, la focale de la camera et l'azimut de la caméra. L'azimut est l'angle entre l'hélicoptère et l'axe de vue de la caméra (voir figure 3.3) ;
- l'algorithme a été développé spécifiquement pour le traitement d'images aériennes ;
- l'algorithme est rapide et peut fournir une réponse en temps réel (moins d'une seconde) ;

L'algorithme de base de [Saux et Sanfourche, 2011] utilise comme données d'entrées : une ou plusieurs images de la scène, la région d'intérêt (ROI) de l'image exprimée en coordonnées image, la hauteur de vol de l'hélicoptère, l'angle d'azimut et la focale de la caméra. Toutefois l'algorithme est seulement capable de reconnaître des voitures de couleur claire, voire quasiment blanche (il traite les pixels de l'image sur la base des niveaux de gris).



(a) Image. (b) Filtrage. (c) Détection de la région d'intérêt.

(d) Région d'intérêt. (e) Mise en correspondance pour la région d'intérêt.

FIGURE 3.4 – Traitement d'image pour la détection et reconnaissance de cible basé sur l'algorithme de [Saux et Sanfourche, 2011].

Par ailleurs, comme le modèle POMDP suppose *une seule* observation après l'exécution de chaque action, l'algorithme raisonne sur une seule image à la fois. L'altitude de vol est un paramètre connu de l'architecture, ce qui rend possible l'utilisation de cette donnée d'entrée directement. La focale et l'azimut de la caméra utilisée sont aussi des paramètres connus de l'architecture, et nous supposons qu'ils ne changent pas au cours de la mission.

Notre version de l'algorithme de [Saux et Sanfourche, 2011] est basée sur la librairie OpenCV [1] [Bradski et Kaehler, 2008]. Il est important de noter que nous avons rajouté à cet algorithme une phase de détection automatique de la région d'intérêt (ROI), afin de ne plus dépendre des coordonnées image de la ROI comme paramètre d'entrée. Nous avons également enrichi la description des modèles 3D de voitures, avant définis selon la longueur, la hauteur et la largeur, par une description plus fine du profil des modèles. Le fonctionnement de l'algorithme de traitement d'image est décrit par la suite. Nous invitons le lecteur à regarder la figure 3.4 pour une illustration de son fonctionnement.

1. À partir de l'image qui provient de la caméra embarquée (figure 3.4(a)), une première phase de filtrage est réalisée sur l'image afin de mettre en évidence le contour probable du véhicule s'il est présent dans l'image (figure 3.4(b)) ;
2. La détection automatique de la zone d'intérêt est faite grâce au résultat du filtrage du contour (figure 3.4(c)). Elle cherchera à définir la région qui comprend le contour le plus grand (figure 3.4(d)). Si aucune région d'intérêt n'est obtenue, l'algorithme renvoie le symbole d'observation *voiture non détectée* ;
3. Si une région d'intérêt est détectée (figure 3.4(d)), l'algorithme génère une projection locale des gabarits 3D de la base de données de modèles de voiture sur la zone d'intérêt de l'image. Cette projection est réalisée pour plusieurs orientations du gabarit 3D, afin de la comparer avec l'image (figure 3.4(e)). La projection locale dépend uniquement de l'altitude, de la focale et de l'azimut de la caméra comme paramètres de prise de vue ;
4. le résultat de la projection locale de l'image de la zone d'intérêt est le choix du gabarit 3D qui maximise la similarité. La similarité est calculée par la distance de chanfrein [Akmal Butt et Maragos, 1998,Saux et Sanfourche, 2011]. Ainsi, l'algorithme renvoie le symbole associé au gabarit choisi : *voiture identifiée comme modèle A*, *voiture identifiée comme modèle B* ou *voiture identifiée comme modèle C*. Si le niveau de similarité est plus bas qu'un seuil défini a priori par l'expert, l'algorithme de traitement d'image renvoie le symbole *voiture détectée mais non identifiée*.

1. The OpenCV library, disponible en `http://opencv.willowgarage.com/wiki/`

Les différents seuils utilisés dans les étapes de filtrage et classification de l'algorithme sont des paramètres empiriques que nous avons fixés par expérience afin de traiter des images prises *pour des conditions environnementales très différentes*. Par exemple, étant donné un état et une image qui contient la voiture A, la fonction de traitement d'image ne renvoie pas nécessairement la réponse "modèle A", car cela dépend de l'algorithme de classification sous-jacent. Selon les conditions de prise de vue (image), en particulier l'intensité de la couleur blanche dans l'image, certains points de l'image seront éventuellement mis en correspondance avec les points d'un mauvais gabarit 3D et la réponse pourra ainsi être erronée.

Dans la suite, nous détaillons l'apprentissage hors-ligne réalisée afin de construire un modèle statistique de la fonction d'observation de notre modèle POMDP.

Apprentissage hors-ligne du modèle d'observation à partir de donnés réelles de l'environnement

L'apprentissage conduit à un modèle d'observation appris par l'intermédiaire d'une analyse statistique, réalisée hors ligne, des réponses de l'algorithme de traitement d'image en se basant sur les images prises lors de vols tests. Cette étude statistique repose sur la fréquence d'occurrence d'une observation connaissant l'état du système, pour un nombre d'expériences fixé. Plus précisément, pour approcher la fonction d'observation $O(o, s', a)$, nous avons compté le nombre de fois qu'un des cinq symboles a été renvoyé par le traitement d'image comme réponse pour un état donné s'. Ceci a pu être réalisé puisque chaque image a été étiquetée par l'état courant du drone. Nous pouvons ainsi affirmer que l'apprentissage mise en place est un apprentissage supervisé.

Il est à noter que la réponse de l'algorithme est déterministe si on connaît l'image qu'on lui donne en entrée. Toutefois, comme *plusieurs images* différentes sont possibles dans un même état, on ne peut connaître avec certitude l'image qui va être prise, et donc la réponse $p(o|s', a)$ de l'algorithme de traitement d'image. Le nombre d'images possibles dans un état donné étant potentiellement infini, nous avons statistiquement estimé la fonction $O(o_i, s', a) = \hat{p}(o_i|s', a)$, où o_i est une des 5 observations possibles, sur la base d'environ 500 images annotées par état :

$$\hat{p}(o_i|s', a) = \frac{1}{N_{exp}} \sum_{n=1}^{N_{exp}} \mathbb{I}_{\{o_n = o_i|s'\}}, \tag{3.2}$$

où N_{exp} représente le nombre d'expériences, soit le nombre de fois où l'algorithme de traitement d'image a été lancé pour les différentes images, et o_n le symbole obtenu pour l'expérience n. On utilise ainsi l'estimateur de la moyenne d'une distribution de Bernoulli, qui est égal à la probabilité de o_i contre les autres observations.Cet estimateur converge en probabilité vers $p(o_i|s', a)$ quand $N_{exp} \to \infty$ [Bass, 1967]. Notez que nous disposons de plus de 500 images par état, donc $N_{exp} \gg 1$ et ainsi nous admettons que les approximations statistiques sont suffisamment bonnes.

Nous pouvons aussi calculer des intervalles de confiance auxquels le vrai modèle est supposé appartenir. L'erreur d'approximation de la proportion ($\hat{p}(o_i = A|s', a) = \hat{p}$), soit l'erreur standard, peut être calculée à partir de l'écart-type suivant :

$$\sigma_{\hat{p}} = \sqrt{\frac{\hat{p}(1-\hat{p})}{N_{exp}}} \tag{3.3}$$

Avec ceci, il est possible d'associer des intervalles de confiance $I = [\hat{p} - l.\sigma_{\hat{p}}, \hat{p} + l.\sigma_{\hat{p}}]$ aux probabilités du modèle d'observation, où l représente le niveau de confiance associé à l'estimation de $\hat{p}$. Par exemple avec $l = 3$, nous garantissons avec $99,7\%$ que $[\hat{p} - 3.\sigma_{\hat{p}}, \hat{p} + 3.\sigma_{\hat{p}}] \ni \hat{p}$.

La table 3.1 montre un exemple de tableau de probabilité obtenu après apprentissage pour un état donné s' où l'hélicoptère autonome se retrouve par exemple dans la zone z_1 à une

observation (o_i)	$\hat{p}(o_i\|s',a)$	$\sigma_{\hat{p}}$	$I = [\hat{p} \pm 3\sigma_{\hat{p}}]$
voiture non détectée	0.0061	0.0023	[0 ; 0.0129]
voiture détectée mais non identifiée	0.1177	0,0095	[0,0891 ; 0,1462]
voiture identifiée comme modèle A	0,6809	0,0137	[0,6396 ; 0,7221]
voiture identifiée comme modèle B	0,0601	0,0070	[0,0391 ; 0,0812]
voiture identifiée comme modèle C	0,1351	0,0101	[0,1048 ; 0,1654]

TABLE 3.1 – Exemple d'une table de probabilité obtenue pour un état donné s' et action a. σ_p représente l'écart type de l'estimation.

altitude h_1 et observe la voiture $model_A$. A noter que cette statistique repose seulement sur la donné zone où l'agent hélicoptère se trouve, la cible qui est dans cette zone et de l'altitude de vol. Ainsi, la table de probabilité obtenue ne dépend pas du *nombre* de zones. La table dépend seulement des variables d'état z, h et Id_{Ta_z}. De cette façon, le modèle d'observation appris peut être utilisé tel quel lors de la génération en ligne du POMDP quelque soit le nombre de zones du problème.

Il est important de souligner que ce type d'apprentissage hors ligne est indépendant de l'algorithme de traitement d'image sous-jacent. Cette approche ne suppose pas une forme particulière (type de fonction) pour la sortie du traitement d'image contrairement à l'approche de [Denzler et Brown, 2002, Deinzer *et al.*, 2003] dont le modèle POMDP raisonne explicitement sur des gaussiennes avec matrices de covariances associées. Dans notre cas, l'information finale utilisée est uniquement l'étiquette renvoyée par l'algorithme de traitement d'image, et donc indépendante du modèle de raisonnement du traitement d'image. Toutefois, certaines applications pourraient nécessiter une prise en compte plus fine du modèle de perception (capteurs et traitements) au niveau des observations du POMDP.

Discussion

La fonction d'observation du POMDP, $O(o, s', a)$, ne tient pas compte des intervalles de confiance qui peuvent être déterminés lors de l'apprentissage hors ligne. Des travaux ont été réalisés dans le cadre MDP pour tenir compte des intervalles de confiance de la fonction de transition $T(s', a, s)$; on peut citer [Givan *et al.*, 2000] qui étudie une planification basée sur les intervalles de confiance des fonctions de transition, ou encore [Buffet et Aberdeen, 2005] qui proposent une version robuste de (L)RTDP [Bonet et Geffner, 2003]. Cette version robuste optimise une politique sur la base du pire modèle local, c'est-à-dire pour le modèle qui attribue plus de probabilité aux plus grandes pertes. Les résultats démontrent que le comportement de la politique est constant quelque soient les variations du modèle.

Peu de travaux se sont intéressés à ce problème dans un cadre POMDP. Ceci est dû principalement au fait que si les fonctions de transition et d'observation ne sont plus exactes, la mise à jour de l'état de croyance ne peut plus être réalisée correctement, ainsi que les projections des α-vecteurs lors de la mise à jour de la fonction de valeur. Ce sujet est actuellement un des points d'intérêt des chercheurs qui s'intéressent en embarquer des politiques ou à résoudre des problèmes réalistes pour lesquels la connaissance du modèle n'est pas parfaite. Deux approches ont été proposées dans la littérature à cet effet. Une approche propose une résolution hors ligne du POMDP avec des fonctions de transition et d'observation définies sur des intervalles de confiance, ce qui résulte en un arbre de politique à horizon limité au quel n'est associé ni fonction de valeur ni mise à jour d'un état de croyance [Ni et Liu, 2008]. Une deuxième approche repose sur l'apprentissage en ligne du modèle du POMDP sous-jacent [Ross *et al.*, 2011].

Dans la suite nous présenterons le différents buts de mission qui peuvent être considérés dans notre application robotique.

3.3 Définition du but de la mission

Notre modèle POMDP de détection et de reconnaissance de cibles peut engendrer deux types de mission : une mission d'exploration, et une mission d'atterrissage. Dans la première mission, seules des actions de prise d'information sont planifiées. Dans la deuxième, une action supplémentaire qui permet d'atterrir près d'une cible de choix est rajoutée au modèle POMDP.

3.3.1 Mission d'exploration

La mission consiste à détecter et identifier les modèles de voitures qui sont présents dans la scène, sans préférence pour un modèle particulier. Cela revient à identifier l'état caché du système robot-environnement. Les fonctions de récompense des actions de déplacement de l'agent ont été décrites dans la section 3.2.2, où nous avons présenté la dynamique du système. Notez, que ce type de mission, qui se ramène à un problème de perception active modélisé en tant que POMDP, a été traité de différentes façons dans la littérature [Eidenberger *et al.*, 2008, Eidenberger *et al.*, 2009, Eidenberger et Scharinger, 2010, Araya-López *et al.*, 2010]. Nous discuterons de ces travaux dans le chapitre 4, ainsi que les approches choisies pour traiter cette problématique dans ce travail de thèse. Nous montrerons en particulier que cette mission peut être résolue avec un modèle POMDP classique ou avec l'utilisation d'un critère mixte (de type ρPOMDP) basé d'une part sur une mesure de l'incertitude de l'état de croyance de l'agent, et d'autre part sur l'espérance de récompenses définies par les paires état-action.

3.3.2 Mission d'atterrissage

La mission consiste à détecter et identifier un modèle particulier de voiture dans la scène parmi les différents modèles de la base de données, et atterrir à côté de cette cible recherchée.

Pour la mission d'atterrissage, une action doit être rajoutée au modèle : nous la nommerons *land*, qui fait atterrir l'hélicoptère autonome sur une zone donnée. Cette action rapporte une récompense si la zone où l'hélicoptère autonome atterrit contient le modèle de voiture recherché.

action land : cette action finalise la mission de l'hélicoptère autonome, et l'amène à l'état terminal. Atterrir dans la zone où le modèle recherché se trouve, rapporte une récompense R_l. La variable mathématique $Se_{Ta} \in \{model_A, model_B, model_C\}$ encode un des modèles existants dans la base de donné, sachant que le modèle recherché peut changer d'une mission à une autre. Si l'hélicoptère est dans l'état terminal (T_s), cette action n'a ni effets ni coût.

- fonction de transition : $T(s', \texttt{land}, s) = \delta_{T_s}(s')$
- fonction de récompense :

$$R(s, \texttt{land}) = \begin{aligned} &\mathbb{I}_{\{(z=z_1)\&(Id_{Ta_{z_1}}=Se_{Ta})\}} R_l + ... + \mathbb{I}_{\{(z=z_{N_z})\&(Id_{Ta_{z_{N_z}}}=Se_{Ta})\}} R_l - \\ &\mathbb{I}_{\{(z=z_1)\&(Id_{Ta_{z_1}}\neq Se_{Ta})\}} C_l - ... - \mathbb{I}_{\{(z=z_{N_z})\&(Id_{Ta_{z_{N_z}}}\neq Se_{Ta})\}} C_l \end{aligned}$$

où $R_l \gg 0$ représente la récompense associée à une mission d'atterrissage réussie, autrement dit, l'hélicoptère atterrit dans la zone où la cible recherchée se localise, et C_l représente le coût d'une mission d'atterrissage manquée. Notez que : $C_{proc} < C_{h,\hat{h}} < C_{z,\hat{z}} \ll C_l$.

3.4 Bilan

Dans ce chapitre, nous avons formalisé le problème applicatif que nous cherchons à traiter en tant que POMDP. Une attention spéciale à été donnée à l'apprentissage du modèle d'observation à partir de données réelles de l'environnement. Le but de la mission reste très généraliste, il peut être épistémique ou symbolique. Dans le chapitre qui suit, nous verrons que si le but de mission est l'exploration (perception active), ou est l'atterrissage (perception active et satisfaction de but symbolique), les critères utilisés lors de l'optimisation peuvent varier. Nous expliciterons chacun de ces critères.

Chapitre 4

Étude du compromis entre la prise d'information et la décision

Dans ce chapitre nous étudions une possibilité de gérer le compromis entre la prise d'information et la décision dans deux cadres applicatifs. Le premier d'entre eux se rapporte à la mission d'exploration, où l'agent interagit avec son environnement pour identifier l'état caché du système. Nous nous sommes intéressée à ce problème de décision séquentielle pour la perception, car, selon les travaux de [Araya-López *et al.*, 2010, Candido et Hutchinson, 2011, Eidenberger et Scharinger, 2010], la fonction de récompense doit reposer sur une mesure de l'incertitude sur l'état de croyance. Sa forme est donc différente de celle utilisée dans le cadre classique des POMDP qui est, pour sa part, basée sur la paire état-action. Nous comparons donc deux approches d'optimisation des politiques. D'une part nous proposons un critère mixte pour les POMDP qui couple une mesure de l'incertitude sur l'état de croyance avec les récompenses définies par les paires état-action et nous développons un schéma algorithmique de résolution pour ce critère. D'autre part, nous proposons d'ajouter au modèle des états but fictifs au moyen des actions de classification afin de revenir à une modélisation sous-forme de POMDP classique (critère non mixte). Une étude comparative de ces approches a été menée afin de vérifier leur équivalence en termes de prise d'informations. Le second cadre d'application correspond à la mission d'atterrissage, qui s'apparente à un problème de décision séquentielle, dont la fonction de récompenses est modélisée à l'aide du formalisme classique des POMDP. La résolution de ce problème permet de vérifier que le POMDP est capable de gérer implicitement l'acquisition d'information pour atteindre le but de la mission. Pour ce cadre d'application, nous avons également évalué des politiques obtenues à partir du critère mixte proposé, et mené une étude comparative des stratégies. Nous montrons que si l'on ajoute au modèle des états buts fictifs (au moyen d'actions de classification ou d'une action d'atterrissage), l'utilisation du critère mixte basé sur une mesure de l'incertitude de l'état de croyance n'est plus nécessaire dans de nombreux cas pratiques. Ainsi, l'étude du critère mixte permet en fait d'ajuster les récompenses d'un modèle POMDP classique équivalent. Ces deux approches sont donc, en pratique, complémentaires.

Dans la décision séquentielle sous incertitude et observabilité partielle, l'agent a besoin d'acquérir de l'information sur l'environnement afin de mener à bien sa mission. Dans ce cas, la politique résultant de l'optimisation du problème doit intégrer une gestion du compromis

entre la prise d'information d'une part et la décision *finale* qui l'amènera à achever sa mission d'autre part.

Toutefois, suivant le cadre d'application, la modélisation du problème de décision séquentielle sous incertitude et observabilité partielle sous forme de POMDP est plus au moins directe : [Spaan et Lima, 2009], [Araya-López *et al.*, 2010] et [Smith et Simmons, 2004] présentent différents problèmes où le but de la mission peut ou non se modéliser dans la fonction de récompense. Nous nous sommes intéressée à l'étude de deux cas d'application particuliers : les mission d'exploration et d'atterrissage, qui ont été présentées au cours du chapitre précédent (chapitre 3).

Nous étudions tout d'abord la mission d'exploration. Pour celle-ci la modélisation du but à atteindre dans la fonction de récompense peut dépendre de l'état de croyance de l'agent, en s'appuyant sur une classe particulière de POMDP dénotée ρPOMDP et proposé dans [Araya-López *et al.*, 2010]. La résolution du nouveau critère d'optimisation associé à ce modèle de type ρPOMDP est obtenue par programmation dynamique. Nous proposons de mettre en œuvre un tel schéma algorithmique pour la résolution de ce ρPOMDP, ce qui constitue un axe de recherche novateur. Par la suite, nous nous sommes intéressée à l'étude du cadre relatif à la mission d'atterrissage, pour laquelle l'utilisation directe d'un modèle POMDP semble possible pour la modélisation et la résolution du problème [Smith et Simmons, 2004]. En conclusion, une comparaison des politiques obtenues est menée.

4.1 POMDP et mission d'exploration

Cette application vise à permettre la détection et la reconnaissance de cibles dans un environnement incertain et partiellement observable dans le cadre d'une mission d'exploration modélisée par un POMDP. Plus précisément, la mission consiste à détecter et à identifier des modèles de voitures dans la scène observée, sans préférence pour un modèle particulier (cf. chapitre 3). Selon [Spaan et Lima, 2009, Eidenberger *et al.*, 2009, Araya-López *et al.*, 2010, Eidenberger et Scharinger, 2010] cette mission se ramène à un problème de perception active.

4.1.1 Synthèse des travaux antérieurs sur les POMDP pour la perception active

Dans ce type d'application le but est d'acquérir suffisamment d'information pour identifier l'état caché du système. L'objectif de mission est généralement de minimiser l'incertitude sur l'état de croyance de l'agent. Comme déjà discuté dans le chapitre 1, dans ce genre de mission, le modèle classique des POMDP peut ne pas être approprié parce que la fonction de récompense est basée sur les paires état-action, et non pas sur la connaissance de l'agent.

On peut affirmer que l'acquisition d'information constitue toujours un moyen, et non une fin en soi. Dès lors, tout problème de décision avec observabilité partielle doit toujours être modélisé comme un POMDP. Toutefois, dans un certain nombre de cas, il semble difficile d'adapter le cadre général des POMDP, car il implique de modifier le modèle de récompense $r(s, a)$ (comme dans [Spaan et Lima, 2009]) de façon à y incorporer les gains d'information apportés par les capteurs.

[Araya-López *et al.*, 2010] a proposé une extension du modèle POMDP. Il définit une fonction de récompense ρ basée sur l'état de croyance de l'agent autonome, c'est-à-dire telle que ρ ne dépende plus uniquement de la paire état-action. La fonction de récompense est également associée par ailleurs à une mesure d'information de l'état de croyance. Suivant ce même objectif, [Eidenberger *et al.*, 2009, Eidenberger et Scharinger, 2010] présentent un autre exemple où le modèle POMDP est utilisé pour formaliser le problème de perception active. Le critère d'optimisation est alors fondé sur la théorie de l'information et les coûts des

différentes actions. La méthode de résolution proposée n'utilise pas d'algorithmes issus de l'état de l'art du POMDP, puisque les distributions de probabilité sont représentées par des ensembles de gaussiennes, et que le critère d'optimisation n'est plus linéaire par morceaux. Par contre, la méthode évalue la valeur des actions prises entre deux instants successifs grâce à une fonction de même forme que l'équation de Bellman.

Ce que nous retenons de ces travaux pour ce cadre d'application, est qu'ils permettent de définir des critères d'optimisation mixtes, basés sur une récompense attribuée aux paires état-action $r(s, a)$ d'une part, et sur la théorie de l'information, reposant généralement sur l'entropie de l'état de croyance b, ou l'information mutuelle d'autre part. Dans l'article de [Araya-López *et al.*, 2010] aucune implémentation algorithmique pour l'évaluation des politiques n'a pas été présenté ; et, [Eidenberger *et al.*, 2009,Eidenberger et Scharinger, 2010] proposent seulement une résolution partielle du modèle, pour l'obtention d'une stratégie myope (à 1 coup).

Pour cela, dans la suite de cette étude, nous nous sommes intéressée à définir une fonction de récompense qui dépend d'une mesure d'information contenue de l'état de croyance. Ainsi, nous nous approchons du critère utilisé dans [Eidenberger *et al.*, 2009, Eidenberger et Scharinger, 2010], en tenant compte toutefois d'un horizon de décision plus long, c'est-à-dire en s'intéressant à un critère de décision à long terme.

4.1.2 Étude d'un critère mixte

[Araya-López *et al.*, 2010] démontre que la convexité de la fonction de valeur d'un POMDP est préservée pour un horizon de longueur N, lorsque la fonction de récompense est convexe, même si celle-ci dépend de l'état de croyance. Il démontre également que si la fonction de récompense n'est pas linéaire, il est possible de l'approximer avec une fonction linéaire par morceaux et convexe – *Piecewise Linear and Convex* (PWLC). Il propose d'utiliser ensuite des algorithmes de résolution exacts ou approchés issus de l'état de l'art du domaine des POMDP moyennant quelques modifications.

Sur l'hypothèse que la fonction de récompense est effectivement convexe, nous proposons un critère mixte qui couple à la fois les récompenses associées aux paires état-action avec les récompenses associées à une mesure de l'incertitude de l'état de croyance. Le travail de [Araya-López *et al.*, 2010] a été publié au même moment où nous avons proposé ce critère mixte pour les POMDP [Carvalho Chanel *et al.*, 2010c, Carvalho Chanel *et al.*, 2010a, Carvalho Chanel *et al.*, 2010b]. De toute évidence l'approche ici présentée est un cas particulier du modèle général ρPOMDP.

Dans la suite nous définissons ce critère mixte, et nous démontrons que la politique associée à ce critère peut se calculer par programmation dynamique. Nous présentons une implémentation d'un algorithme basé sur un schéma d'optimisation d'itération approchée sur la valeur, que nous avons adapté au cas de ce critère mixte.

Définition 4.1.1 (Critère mixte) *Soit π une politique définie sur l'état de croyance. Nous définissons alors un critère d'optimisation qui se décompose en : l'espérance mathématique de la somme pondérée des récompenses attribuées aux actions choisies à laquelle s'ajoute l'espérance mathématique de la somme pondérée des entropies (négatives) des états de croyance*

successifs. Ces deux termes sont ensuite pondérés par une constante $\lambda \in [0,1]$ *tel que :*

$$\begin{aligned} J^\pi(b) &= (1-\lambda)V^\pi(b) + \lambda H^\pi(b), \text{ avec} \\ V^\pi(b) &= E_\pi\left[\sum_{t=0}^{\infty} \gamma^t r(b_t, \pi(b_t)) \middle| b_0 = b\right] \\ H^\pi(b) &= E_\pi\left[\sum_{t=0}^{\infty} \gamma^t H(b_t) \middle| b_0 = b\right] \end{aligned} \tag{4.1}$$

Notons que $V^\pi(b)$ *est la fonction de valeur du modèle POMDP classique, puisque :*

$$r(b_t, \pi(b_t)) = \sum_{s \in S} b_t(s) r(s, \pi(b_t)),$$

Notons aussi, que $H^\pi(b)$ *correspond à la fonction de valeur déjà proposée dans [Deinzer* et al.*, 2003]. En revanche, nous définissons ici l'entropie de l'état de croyance de la manière suivante :*

$$H(b) = \sum_{s \in S} b(s) \log(b(s))$$

de sorte que celle-ci est négative (pas de signe de moins). Cela permet à cette fonction d'entropie d'être convexe. De plus, l'entropie définie en tant que telle, joue le rôle d'une pénalisation dans le critère puisque $H(b) \leq 0,\ \forall b \in \Delta$*. La somme des deux termes forme une fonction convexe qui correspond à la définition du critère de performance donné par [Mihaylova* et al.*, 2002] (introduit dans l'équation 1.3).*

Il est alors possible de définir une équation de Bellman relative à ce critère, comme cela est démontré dans le théorème suivant :

Théorème 4.1.1 (Equation de Bellman pour le critère mixte.) *Le principe de Bellman appliqué au critère mixte pour une politique stationnaire* π *est donné par :*

$$J^\pi(b) = (1-\lambda) r(b, \pi) + \lambda H(b) + \gamma \sum_{o \in \Omega} p(o|b, \pi) J^\pi(b^o_\pi) \tag{4.2}$$

Preuve. À partir de l'équation 4.1 on peut remplacer $V^\pi(b)$ et $H^\pi(b)$ par leurs expressions sous forme de somme pondérée. La convergence des différentes sommes est assurée par le

facteur $\gamma \in [0,1[$ sachant que $r(b,\pi)$ et $H(b)$ sont bornées. On a donc :

$$
\begin{aligned}
J^\pi(b) &= (1-\lambda)E_\pi\left[\sum_{t=0}^{\infty}\gamma^t r(b_t,\pi)\middle| b_0=b\right] + \lambda E_\pi\left[\sum_{t=0}^{\infty}\gamma^t H(b_t)\middle| b_0=b\right] && (4.3)\\
J^\pi(b) &= (1-\lambda)E_\pi\left[r(b_0,\pi)|b_0=b\right] + (1-\lambda)E_\pi\left[\sum_{t=1}^{\infty}\gamma^t r(b_t,\pi)\middle| b_0=b\right] + \\
&\quad +\lambda E_\pi\left[H(b_0)|b=b_0\right] + \lambda E_\pi\left[\sum_{t=1}^{\infty}\gamma^t H(b_t)\middle| b_0=b\right] && (4.4)\\
J^\pi(b) &= (1-\lambda)\sum_{s\in S} r(s,\pi)b(s) + (1-\lambda)E_\pi\left[\sum_{t=1}^{\infty}\gamma^t r(b_t,\pi)\middle| b_0=b\right] + \\
&\quad +\lambda\sum_{s\in S} b(s)\log(b(s)) + \lambda E_\pi\left[\sum_{t=1}^{\infty}\gamma^t H(b_t)\middle| b_0=b\right] && (4.5)\\
J^\pi(b) &= (1-\lambda)\sum_{s\in S} r(s,\pi)b(s) + \lambda\sum_{s\in S} b(s)\log(b(s)) + \\
&\quad +\gamma(1-\lambda)\sum_{o\in\Omega} p(o|b,\pi)E_\pi\left[\sum_{t=0}^{\infty}\gamma^t r(b_t,\pi)\middle| b_0=b_\pi^o\right] + \\
&\quad \gamma\lambda\sum_{o\in\Omega} p(o|b,\pi)E_\pi\left[\sum_{t=0}^{\infty}\gamma^t H(b_t)\middle| b_0=b_\pi^o\right] && (4.6)\\
J^\pi(b) &= (1-\lambda)\sum_{s\in S} r(s,\pi)b(s) + \lambda\sum_{s\in S} b(s)\log(b(s)) + \\
&\quad +\gamma\sum_{o\in\Omega} p(o|b,\pi)\left((1-\lambda)V^\pi(b_\pi^o) + \gamma\lambda H^\pi(b_\pi^o)\right) && (4.7)\\
J^\pi(b) &= (1-\lambda)\sum_{s\in S} r(s,\pi)b(s) + \lambda\sum_{s\in S} b(s)\log(b(s)) + \gamma\sum_{o\in\Omega} p(o|b,\pi)J^\pi(b_\pi^o) && (4.8)
\end{aligned}
$$

Pour une politique stationnaire π donnée, la fonction de valeur J^π satisfait donc l'équation de Bellman 4.2. L'équation 4.2 montre que J^π correspond au critère γ-pondéré classique dans lequel la récompense est ajoutée à l'entropie de l'état de croyance courant. On est ramené à un problème de maximisation du critère pondéré précédent qui peut s'interpréter comme le calcul de récompenses *artificielles*. Celles-ci sont égales aux récompenses réelles $\sum_{s\in S} r(s,a)b(s)$ auxquelles s'ajoutent les entropies $H(b)$. ■

Proposition 4.1.1 *La fonction de valeur optimale $J^*(b)$ est la solution de l'équation de Bellman :*

$$J^*(b) = \max_{a\in A}\left\{(1-\lambda)r(b,a) + \lambda H(b) + \gamma\sum_{o\in\Omega} p(o|b,a)J^*(b_a^o)\right\} \qquad (4.9)$$

Preuve. On a :

$$J^\pi(b) = (1-\lambda)\sum_{s\in S} r(s,\pi)b(s) + \lambda\sum_{s\in S} b(s)\log(b(s)) + \gamma\sum_{o\in\Omega} p(o|b,\pi)J^\pi(b_\pi^o), \qquad (4.10)$$

et aussi, comme la convergence des différentes sommes est assurée par le facteur $\gamma \in [0,1[$,

on a pour toute politique $\pi = (a, \pi')$:

$$J^*(b) = \max_{\pi} \left\{ (1-\lambda) E_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(b_t, \pi(b_t)) \middle| b_0 = b \right] + \lambda E_\pi \left[\sum_{t=0}^{\infty} \gamma^t H(b_t) \middle| b_0 = b \right] \right\} \quad (4.11)$$

$$J^*(b) = \max_{(a,\pi')} \left[(1-\lambda) \sum_{s \in S} r(s,a) b(s) + \lambda \sum_{s \in S} b(s) \log(b(s)) + \gamma \sum_{o \in \Omega} p(o|b,a) J^{\pi'}(b_a^o) \right] \quad (4.12)$$

$$J^*(b) = \max_{a \in A} \left[(1-\lambda) \sum_{s \in S} r(s,a) b(s) + \lambda \sum_{s \in S} b(s) \log(b(s)) + \gamma \sum_{o \in \Omega} p(o|b,a) \max_{\pi'} J^{\pi'}(b_a^o) \right] \quad (4.13)$$

$$J^*(b) = \max_{a \in A} \left[(1-\lambda) \sum_{s \in S} r(s,a) b(s) + \lambda \sum_{s \in S} b(s) \log(b(s)) + \gamma \sum_{o \in \Omega} p(o|b,a) J^*(b_a^o) \right] \quad (4.14)$$

■

L'application de la programmation dynamique à l'équation de Bellman 4.2 permet de calculer une politique où la récompense immédiate se trouve pénalisée par l'entropie de l'état de croyance. Cette pénalisation sera d'autant plus importante que la distribution sur les états est peu précise. Ce calcul permet d'introduire les opérateurs de Bellman et de programmation dynamique relatifs à ce critère mixte.

Définition 4.1.2 *L'opérateur de Bellman, noté* $\mathcal{W}^\pi : \mathbb{R}^\Delta \to \mathbb{R}^\Delta$ *sachant que* $\mathbb{R}^\Delta \Leftrightarrow \{f : \Delta \to \mathbb{R}\}$. *Donc, pour tout* $J \in \mathbb{R}^\Delta$ *est défini par :*

$$\mathcal{W}^\pi J(b) = (1-\lambda) \sum_{s \in S} r(s,\pi) b(s) + \lambda \sum_{s \in S} b(s) \log(b(s)) + \gamma \sum_{o \in \Omega} p(o|b,\pi) J(b_\pi^o). \quad (4.15)$$

L'opérateur de programmation dynamique $\mathcal{W} : \mathbb{R}^\Delta \to \mathbb{R}^\Delta$ *est défini tel que :*

$$\mathcal{W} J(b) = \max_{a \in A} \left[(1-\lambda) \sum_{s \in S} r(s,a) b(s) + \lambda \sum_{s \in S} b(s) \log(b(s)) + \gamma \sum_{o \in \Omega} p(o|b,a) J(b_a^o) \right]. \quad (4.16)$$

Proposition 4.1.2 J^π *est l'unique point fixe de* $\mathcal{W}^\pi$, *et* J^* *est l'unique point fixe de* $\mathcal{W}$. *De plus, la politique stationnaire optimale est donnée par :*

$$\pi^*(b) \in \arg\max_{a \in A} \left[(1-\lambda) \sum_{s \in S} r(s,a) b(s) + \lambda \sum_{s \in S} b(s) \log(b(s)) + \gamma \sum_{o \in \Omega} p(o|b,a) J(b_a^o) \right] \quad (4.17)$$

Sachant que, les opérateurs $\mathcal{W}^\pi$ et $\mathcal{W}$ sont γ-Lipschitziens pour la norme sup [Sigaud et Buffet, 2008], on a :

$$\forall (J, J') \in (\mathbb{R}^\Delta)^2, \begin{cases} \|\mathcal{W}^\pi J - \mathcal{W}^\pi J'\| \leq \gamma \|J - J'\|_\infty \\ \|\mathcal{W} J - \mathcal{W} J'\| \leq \gamma \|J - J'\|_\infty \end{cases} \quad (4.18)$$

Preuve. En effet, pour tout $b \in \Delta$ et a^* tel que :

$$a^* \in \arg\max_{a \in A} \left[(1-\lambda) \sum_{s \in S} r(s,a) b(s) + \lambda \sum_{s \in S} b(s) \log(b(s)) + \gamma \sum_{o \in \Omega} p(o|b,a) J(b_a^o) \right] \quad (4.19)$$

on a :

$$0 \leq |\mathcal{W} J(b) - \mathcal{W} J'| = \gamma \sum_{o \in \Omega} p(o|a^*, b) |J(b_a^o - J'(b_a^o)| \quad (4.20)$$

$$\leq \gamma \sum_{o \in \Omega} p(o|a^*, b) \max_{o \in \Omega} |J(b_a^o - J'(b_a^o)| \quad (4.21)$$

$$\leq \gamma \max_{o \in \Omega} |J(b_a^o - J'(b_a^o)| \quad (4.22)$$

$$\leq \gamma \max_{b \in \Delta} |J(b) - J'(b)| = \gamma \|J - J'\|_\infty \quad (4.23)$$

Cette contraction de $\mathcal{W}$ assure donc l'existence d'un point fixe. De plus, grâce à la définition de toute politique optimale $\pi^*(b)$, on a :

$$\mathcal{W}^{\pi^*} J^* = \mathcal{W} J^* = J^* \tag{4.24}$$

J^* apparait donc comme le point fixe de $\mathcal{W}^{\pi^*}$. Or, par définition J^{π^*} est également le point fixe de $\mathcal{W}^{\pi^*}$. Du fait de l'unicité de ce point fixe, on a : $J^{\pi^*} = J^*$. Ceci montre que la politique $\pi^*(b)$ est optimale. ■

Bien que toujours convexe, le critère mixte n'est plus une fonction linéaire. Comme cela a été déjà discuté auparavant, [Araya-López *et al.*, 2010] remarque que si la fonction convexe de récompense n'est pas linéaire, il est possible de l'approximer à partir d'une fonction linéaire par morceaux et convexe (PWLC). Grâce à cela, nous pouvons utiliser des algorithmes approchés issus de l'état de l'art du domaine POMDP en les adaptant.

Adaptation d'algorithmes approchés de type *point-based* pour l'optimisation du critère mixte

Dans la suite, nous montrons qu'à partir d'une approximation linéaire du premier ordre il est possible d'adapter certains algorithmes approchés issus de l'état de l'art du domaine des POMDP afin d'obtenir une politique ϵ-optimale pour le critère mixte 4.1 proposé. Cette approximation du premier ordre repose sur la donnée de certains points de l'espace de l'état de croyance. Comme la plupart des algorithmes de type *point-based* procèdent à une mise à jour de la fonction de valeur du POMDP à partir d'un sous-ensemble d'états de croyance $\mathcal{B} \in \Delta$, nous considérerons que les ensembles de points utilisés par l'approximation linéaire d'une part et l'algorithme d'autre part sont les mêmes [Carvalho Chanel *et al.*, 2010c, Carvalho Chanel *et al.*, 2010a, Araya-López *et al.*, 2010].

Approximation linéaire du premier ordre de la fonction de valeur du critère mixte

Il est possible de démontrer que la fonction de récompense peut être représentée par un α-vecteur au point b^*, à partir d'une approximation linéaire du premier ordre de celle-ci. De la sorte que l'intégration de l'entropie dans la fonction de récompense n'empêchera pas de paramétrer la fonction de valeur par des α-vecteurs aux points $b^* \in \Delta$, en supposant la fonction de valeur initiale $J_0(b) = 0$.

Tout d'abord, nous définissons α_c, l'α-vecteur classique qui maximise la valeur $V(b)$, pour $b \in \Delta$. Cet α-vecteur est constitué des récompenses associées aux paires état-action et permet de modéliser la fonction de valeur $V(b)$ sous la forme vectorielle telle que : $V(b) = \alpha_c \cdot b$. Il est bien connu que les α-vecteurs classiques représentent le gradient de V par rapport à b.

$$V(b) = \alpha_c \cdot b, \text{ où } \alpha_c = \frac{\partial V(b)}{\partial b} \tag{4.25}$$

Ensuite, nous rappelons que l'approche critère mixte ici présentée est un cas particulier du modèle général ρPOMDP. Donc, la prise en compte de l'entropie modifie l'expression de la récompense immédiate $\rho(b, a)$ qui devient :

$$\rho(b, a) = (1 - \lambda) \sum_{s \in S} r(s, a) b(s) + \lambda \sum_{s \in S} b(s) \log b(s)$$

et, sous forme vectorielle nous définissons cette récompense immédiate par : $\rho(b, a) = \alpha_m \cdot b$.

Si l'on cherche à dériver la nouvelle récompense immédiate $\rho(b) = \alpha_m^a \cdot b$, on obtient :

$$\rho(b) = \alpha_m^a \cdot b = ((1 - \lambda)\alpha_c^a + \lambda \log(b)) \cdot b \tag{4.26}$$

$$\frac{\partial \rho(b)}{\partial b} = (1 - \lambda)\alpha_c^a + \lambda(\log b + \underline{\log}\, e) \tag{4.27}$$

où $\log b \equiv [\log b(s_1) \; ... \; \log b(s_n)]$ est le vecteur dont la i-ième composante vaut $\log b(s_i)$, et $\underline{\log}\, e \equiv [\log e \; ... \; \log e]$ un vecteur pour lequel toutes les composantes valent $\log e$. Ainsi, nous vérifions au passage que la dérivée de ρ par rapport à b n'est pas linéaire. Cependant, un développement en série au premier ordre permet d'obtenir une approximation linéaire au voisinage d'un point $b^* \in \Delta$. Dans ce cas, la récompense ρ et son gradient par rapport à b sont évalués au point $b^* \in \Delta$.

$$\begin{aligned}
\rho(b) &= \rho(b^*) + \frac{\partial \rho(b)}{\partial b}|_{b=b^*} \cdot (b - b^*) && (4.28)\\
&= ((1-\lambda)\alpha_c^a + \lambda \log(b^*)) \cdot b^* + ((1-\lambda)\alpha_c^a + \lambda(\log b^* + \underline{\log}\, e)) \cdot (b - b^*) && (4.29)\\
&= ((1-\lambda)\alpha_c^a + \lambda \log(b^*)) \cdot b^* + ((1-\lambda)\alpha_c^a + \lambda \log b^*) \cdot (b - b^*) + \lambda \underline{\log}\, e \cdot (b - b^*) && (4.30)\\
&= ((1-\lambda)\alpha_c^a + \lambda(\log b^*)) \cdot b + \lambda \underline{\log}\, e \cdot b - \lambda \underline{\log}\, e \cdot b^* && (4.31)
\end{aligned}$$

En utilisant la condition que pour tout b, la somme des composantes vaux 1 ($\sum_i b_i = 1$) étant donné que b est une distribution de probabilité, on voit que $\underline{\log} e \cdot b = \log e$ et donc $(\lambda \underline{\log}\, e \cdot b - \lambda \underline{\log}\, e \cdot b^*) = 0$. Ainsi il vient :

$$\begin{aligned}
\rho(b) &= ((1-\lambda)\alpha_c^a + \lambda(\log b^*)) \cdot b + \lambda \underline{\log}\, e \cdot b - \lambda \underline{\log}\, e \cdot b^* && (4.32)\\
\rho(b) &= ((1-\lambda)\alpha_c^a + \lambda(\log b^*)) \cdot b + \lambda \log e - \lambda \log e && (4.33)\\
\rho(b) &= ((1-\lambda)\alpha_c^a + \lambda(\log b^*)) \cdot b && (4.34)\\
\rho(b) &= \alpha_m^{b^*,a} \cdot b && (4.35)
\end{aligned}$$

Ceci démontre que nous pouvons utiliser $\alpha_m^{b^*,a}$ comme α-vecteur de tout point b au voisinage de b^*, en supposant la fonction de valeur initiale $J_0(b) = 0$. Ce résultat est important puisqu'il démontre à la fois que les algorithmes de type *point-based* peuvent être appliqués mais au prix d'une approximation linéaire des variations de J. Il apparaît également que pour chaque $b^* \in \Delta$ exploré l'algorithme devra conserver l'α-vecteur associé.

Nous rappelons qu'à la différence des travaux antérieurs, nous voudrons résoudre et implémenter le ρPOMDP reposant sur le critère proposé. Pour faire cela, nous avons dû modifier l'algorithme PBVI [Pineau *et al.*, 2003] afin de s'accommoder à cette nouvelle formulation. Le choix de l'algorithme PBVI se justifie par le fait qu'il offre une méthode d'exploration stochastique de l'espace des états de croyance, ce qui évite de modifier les heuristiques de recherche dans cet espace qui sont utilisées habituellement par d'autres algorithmes, par exemple, tels que HSVI [Smith et Simmons, 2005] et SARSOP [Kurniawati *et al.*, 2008].

Adaptation de l'algorithme PBVI

L'algorithme PBVI repose sur une méthode d'optimisation de type *point-based*. Celle-ci suppose qu'en mettant à jour non seulement la valeur mais aussi le gradient (le α-vecteur) pour chaque $b \in \mathcal{B}$, la politique calculée pourra être utilisée pour d'autres points de l'espace de croyance qui n'appartiennent pas à l'ensemble B. Nous rappelons ici que l'ensemble des états de croyance est composé de points considérés comme atteignables. Ces états de croyance sont atteignables en suivant une politique d'action arbitraire depuis une état de croyance initial b_0.

L'algorithme 8 décrit la procédure adaptée pour calculer les nouveaux α-vecteurs représentés par l'approximation linéaire du premier ordre. Avant de présenter le nouvel opérateur de mise à jour de la valeur – *backup* – nous souhaitons aborder un point technique qui nous semble essentiel dans le calcul du logarithme de b.

L'entropie définie négative, n'est pas une fonction lipschitzienne, c'est-à-dire une fonction continue à variation et à dérivée bornées. On voit clairement que $\log x \to -\infty$ quand $x \to 0$. Ceci pose un problème numérique lors de la résolution. Afin de s'en affranchir, nous proposons de borner inférieurement les composantes $\log b(s_i) = -\infty$ du vecteur $\log b$ et de définir un

Algorithme 8: PBVI pour le critère mixte

entrée : POMDP, N, $N_{\mathcal{B}}$
sortie : fonction de valeur J
1 Initialiser $J_0 \leftarrow \emptyset$, $n = 0$;
2 $\mathcal{B} \leftarrow b_0$;
3 **repeat**
4 | Étendre $\mathcal{B}$;
5 **until** $|\mathcal{B}| < N_{\mathcal{B}}$;
6 **repeat**
7 | $n = n + 1$;
8 | $J_n \leftarrow \emptyset$;
9 | Calculer toutes les projections $\Gamma_m^{a,o}$ de J_{n-1} (équation 2.17) ;
10 | **for** $b \in \mathcal{B}$ **do**
11 | | $J_n \leftarrow \bigcup backup_m(b)$;
12 **until** $n < N$ ou $||\max_{\alpha_n \in J_n} \alpha_n \cdot b - \max_{\alpha_{n-1} \in J_{n-1}} \alpha_{n-1} \cdot b|| > \epsilon, \forall b \in \mathcal{B}$;

seuil, de façon à approcher la composante $(-\infty)$ du gradient de la valeur d'un état de croyance par une constante suffisamment négative (assurer la pente). Il est à noter que cela ne change pas la valeur de l'état de croyance, étant donné que $x \log x \to 0$ quand $x \to 0$. Ainsi :

$$\log b(s_i) = \begin{cases} \log b(s_i), & \text{si } b(s_i) > 10^{-300}, \\ \log 10^{-300}, & \text{sinon.} \end{cases}$$

L'adaptation de l'algorithme PBVI (cf. l'algorithme 8) implique donc une modification de la mise à jour de la valeur ($backup_m$) (ligne 11). Dans ce cas, nous rappelons que la valeur :

$$\sum_{s \in S} ((1-\lambda) b(s) r(s,a) + \lambda b(s) \log(b(s)))$$

au voisinage d'un point $b \in \mathcal{B}$ peut être approchée par :

$$r_m^a = ((1-\lambda) r_a + \lambda \log b) \cdot b = \alpha_m^{b,a} \cdot b \tag{4.36}$$

où $r_a(s) = r(s,a)$. De cette façon le nouvel opérateur de *backup* peut être défini par :

$$backup_m(b) = \arg\max_{\alpha_b^a \in \Gamma_b^a} b \cdot \alpha_b^a, \text{ avec :} \tag{4.37}$$

$$\Gamma_b^a \leftarrow r_m^a + \gamma \sum_o \arg\max_{\alpha_i^{a,o} \in \Gamma_m^{a,o}} b \cdot \alpha_i^{a,o} \tag{4.38}$$

Les projections $\Gamma_m^{a,o}$ sont calculées de la même façon que dans la méthode classique (équation 2.17). Nous précisons une fois de plus que le calcul de mise à jour de la valeur, réalisée pour ce critère, impose que $|J_n| = |\mathcal{B}|$, c'est-à-dire que le nombre de α-vecteurs qui constitue la fonction de valeur à l'instant n est égale au nombre d'états de croyance atteignables. Ceci est dû à l'approximation linéaire du premier ordre qui permet de déterminer le gradient $J(b)$ pour $b \in \mathcal{B}$, à partir d'un α-vecteur particulier qui dépend du vecteur $\log b$.

Un autre point technique important est que, contrairement à la version classique de l'algorithme PBVI, la génération de points $b \in \mathcal{B}$ doit ici être faite dès le début de résolution et non au fur et à mesure des itérations. Ceci est un point délicat à gérer, puisque nous souhaitons éviter des erreurs d'approximation trop importants de la valeur viennent fausser l'ensemble de la démarche. Pour cela, nous avons choisi de créer l'ensemble des points de $\mathcal{B}$ en préambule à toute résolution.

Nous tenons aussi à remarquer que dû fait que l'approximation du premier ordre est basée sur les points $b \in \mathcal{B}$, aucune garantie de convergence pour un ϵ *très petit* n'est peut être donnée. La valeur associée à un état de croyance peut en faite osciller. Ceci a été vérifié notamment par l'expérience lors de nos tests en simulation pour l'évaluation de ce critère mixte. Même si l'opérateur de Bellman est une contraction dans l'espace des fonctions de valeur qui conduit in fine à un point-fixe pour ce critère mixte, la résolution ici décrite n'est pas exacte. Ainsi, l'on peut seulement garantir que l'erreur d'approximation sera borné [Araya-López *et al.*, 2010].

Dans la suite, nous nous sommes focalisée sur une évaluation des politiques qui peuvent être calculées sur la base de ce critère mixte. Comme celui-ci dépend explicitement de la valeur de la pondération λ, nous avons souhaité dans un premier temps évaluer l'effet de ce paramètre dans un problème simple de diagnostic.

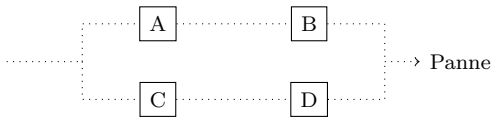
Évaluation du critère mixte par variation du paramètre λ

Pour jauger la pertinence du critère mixte, des politiques stationnaires ont été calculées pour différents valeurs du facteur de pondération λ. De la sorte, nous proposons d'étudier le critère défini précédemment selon une approche multi-objectif [Deb, 2001]. En effet, celui-ci peut être vu comme un compromis entre l'acquisition d'information que l'on cherche à maximiser et le coût des actions nécessaires à mettre en oeuvre pour cette prise d'information. L'objectif consiste donc à construire le front de Pareto associé à ces deux critères antagonistes et à déterminer l'ensemble de solutions non-dominées, c'est-à-dire toutes les solutions meilleures que les autres sur au moins un objectif. Lorsqu'on est en présence de plusieurs objectifs, il est rare qu'une solution permette de les optimiser simultanément. La plupart du temps, l'amélioration d'un objectif se fait au détriment d'un autre [Deb, 2001]. Ainsi, Le front de Pareto constitue un outil d'aide au choix d'une solution en fonction des critères que l'on cherche à minimiser ou maximiser.

La construction du Front de Pareto est ici réalisée en échantillonnant plus ou moins finement le paramètre de pondération λ et en résolvant pour chaque valeur de celui-ci le problème d'optimisation du critère mixte par programmation dynamique. Par conséquent, nous avons optimisé des politiques pour chaque valeur de λ en exploitant l'algorithme PBVI modifié. L'ensemble de résultats nous permet de tracer l'allure du Front de Pareto sachant que chaque politique calculée correspond à la maximisation des critères pour une valeur de λ.

Cette approche multi-objectif a tout d'abord été mise en ouvre sur un exemple simple de diagnostic afin de mettre en regard l'entropie de l'état de croyance de l'agent et le coût associé aux différentes actions réalisées. Le problème consiste ici à déterminer l'état caché d'un système en panne en réalisant des tests sur les différents composants de celui-ci (composants A, B, C et D). La figure 4.1 montre le schéma du système considéré. À chaque test est associé un coût différent (le test A coûte -2, les tests B et C coûtent -1 et le test D coûte -3). On suppose de plus que ces composants sont installés à des endroits plus au moins accessibles. Les tests renseignent l'agent sur l'état de fonctionnement du composant de manière imprécise : par exemple, le test du composant A renseigne avec une probabilité de 0.75 sur le véritable état (en marche ou en panne). Le but est d'identifier quel ou quels sont les composant du système qui sont effectivement en panne. L'agent, qui n'a aucune connaissance a priori sur l'état des composants, doit réaliser de tests à moindres coûts pour mener cette identification.

Dans cet exemple simple, les politiques pour le critère mixte proposé ont été calculées en faisant varier la constante λ pour un ensemble $\mathcal{B}$, où $|\mathcal{B}| = 1000$. Nous rappelons que la pondération λ permet également d'adapter la valeur de chaque critère, en donnant par exemple plus ou moins de poids à l'optimisation de la somme pondérée d'entropies : $H^{\pi}(b) = E_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t H(b_t) | b_0 = b\right]$.



Actions :
- test A (-2) ;
- test B (-1) ;
- test C (-1) ;
- test D (-3).

Observations :
- test A : vrai état (0.75) ;
- test B : vrai état (0.65) ;
- test C : vrai état (0.80) ;
- test D : vrai état (1.00).

FIGURE 4.1 – Schéma du système constitué de 4 composants : A, B, C et D.

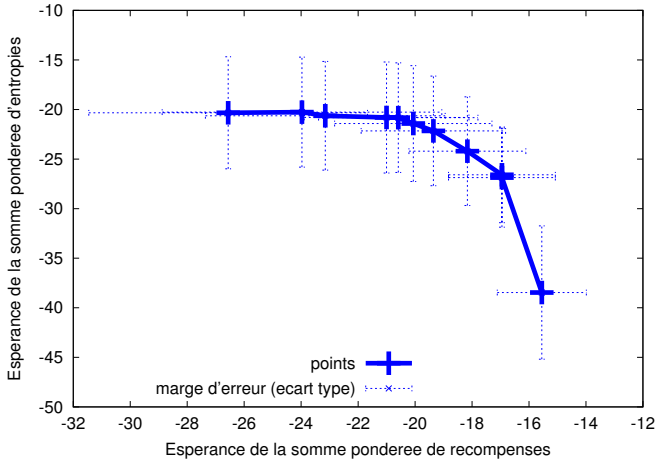


FIGURE 4.2 – Front de Pareto pour l'application de type diagnostic – $\lambda \in \{0, 0.1, 0.2, ..., 0.9, 1.0\}$.

.

La figure 4.2 montre le front de Pareto obtenu à la suite du traitement de cet exemple de diagnostic. Le balayage de différentes valeurs de λ permet d'approcher le front de Pareto associé à la maximisation des critères. Nous vérifions que l'amélioration de l'un des objectifs (récompenses sur les états ou entropie de l'état de croyance) se fait au détriment de l'autre. Le calcul des valeurs moyennes des critères a été effectué sur la base de 1000 simulations avec un horizon de 30 étapes de décision. Sur cette même figure, nous avons représenté l'écart type de chaque critère ce qui permet de mettre en évidence que cette méthode approchée conduit à des erreurs d'approximation qui ne sont pas négligeables.

Dans la figure 4.3(a), nous montrons la moyenne obtenue pour 1000 simulations à chaque étape de décision pour l'entropie négative de l'état de croyance, calculée comme :

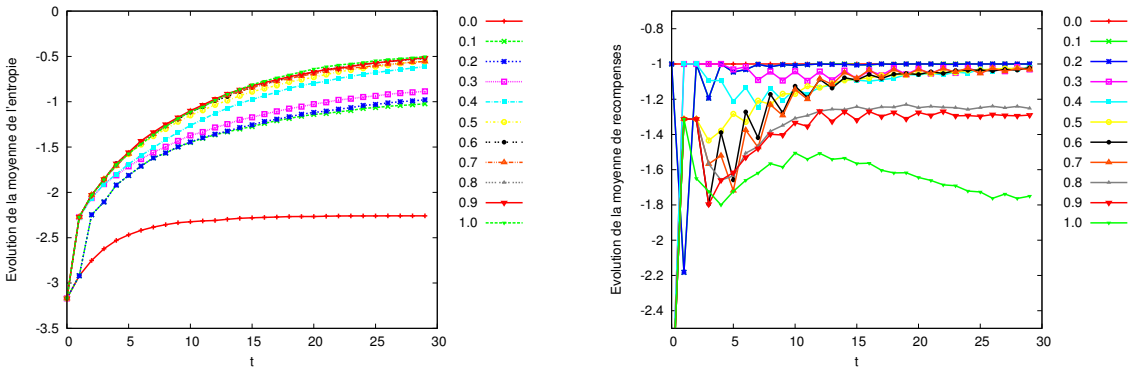
$$H_t = \frac{1}{k} \sum_{i=0}^{k} \sum_{s \in S} b_t(s) \log(b_t(s)) \tag{4.39}$$

où k représente le nombre de simulations. Et dans la figure 4.3(b), la moyenne de récompenses est calculée pour sa part comme suit :

$$V_t = \frac{1}{k} \sum_{i=0}^{k} r(s_t, \pi(b_t)) \tag{4.40}$$

Il est à noter que le simulateur connait l'état caché du système en attribuant les récompenses à chaque étape de décision.

Les courbes de la figure 4.3 montrent que plus le poids λ donné à l'entropie est élevé, plus l'entropie se trouve effectivement optimisée, ceci au prix de gains moyens moins élevés. Les différentes valeurs du facteur d'agrégation des critères λ sont indiqués à droite de chaque tracé.



(a) Moyenne de l'entropie de l'état de croyance à chaque instant t.

(b) Moyenne de la récompense à chaque instant t.

FIGURE 4.3 – Moyennes de l'entropie de l'état de croyance et de récompenses.

Nous montrons aussi l'espérance de la somme pondérée de récompenses et d'entropies sur la figure 4.4. L'espérance de la somme pondérée d'entropies est calculée comme :

$$H^{\pi}(b) = E_{\pi}\left[\sum_{i=0}^{t} \gamma^{i} H(b_i)|b_0 = b\right] \tag{4.41}$$

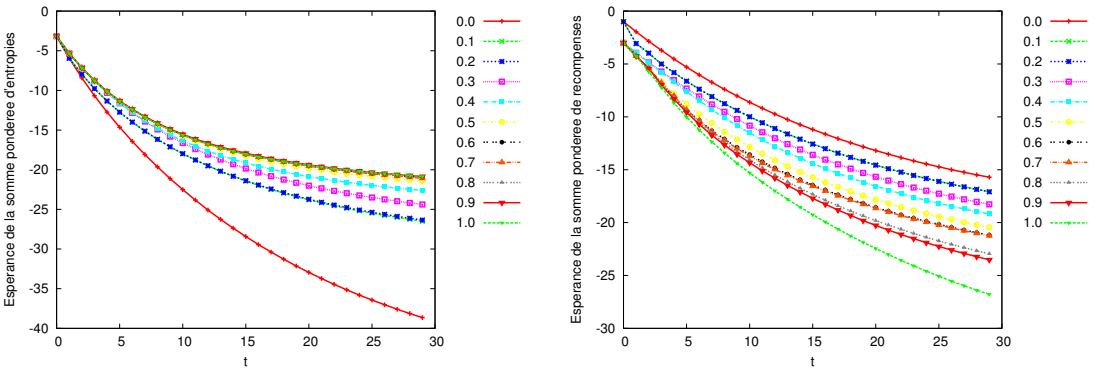
sur un passé de longueur t, et l'espérance de la somme pondérée de récompenses est telle que :

$$V^{\pi}(b) = E_{\pi}\left[\sum_{i=0}^{t} \gamma^{i} r(s_i, \pi(b_i))|b_0 = b, s_i\right] \tag{4.42}$$

pour le même horizon t.

Nous pouvons une fois de plus vérifier qu'en fonction de la valeur de λ le comportement de l'agent sera différent. Dans le cas où $\lambda = 0$, le critère optimisé correspond au critère classique des POMDP, qui n'intègre aucune optimisation de l'entropie de l'état de croyance pour ce problème ; ceci est vérifié par le fait que la politique contient seulement une action (test B) qui est la moins coûteuse. L'entropie de la croyance sur l'état du composant B est diminuée, mais aucune information par rapport aux autres composants n'est collectée (courbe rouge associée au symbole (–|–) dans la figure 4.3(a)). Au fur et à mesure que λ tends vers 1, la politique se rapproche d'un comportement de plus en plus investigateur malgré le coût des actions : plus l'agent se renseigne, plus l'entropie diminue au prix d'un coût d'actions plus important.

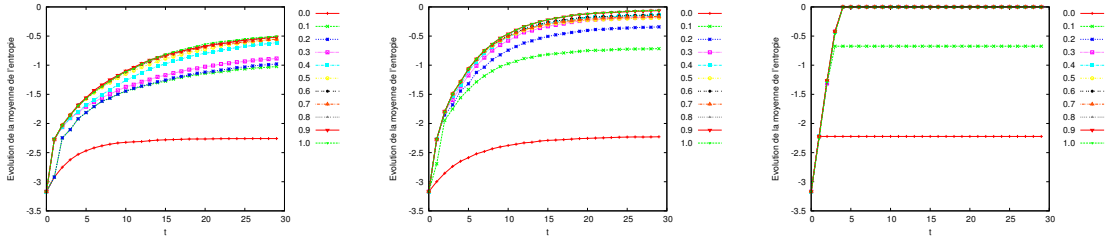
Nous tenons à faire remarquer que dans la figure 4.3(a) la moyenne de l'entropie associé à l'état de croyance n'atteint pas la valeur zéro sur ces simulations avec 30 étapes de décision. Ceci peut être expliqué par le fait que le modèle d'observation n'est pas exact. En effet, l'agent cherche à identifier l'état du système et à réduire l'entropie de son état de croyance, mais l'évolution de la moyenne de cette entropie dépend directement de la fonction d'observation. Pour démontrer empiriquement cela, nous avons recalculé des politiques à partir des modèles POMDP sur ce même exemple de diagnostic avec des modèles d'observation moins bruité (- test A : état véritable (0.85) ; - test B : état véritable (0.75) ; - test C : état véritable (0.90) ; test D : état véritable (1.00)) et non bruité (quelque soit l'action : état véritable (1.0)). Nous pouvons de cette façon illustrer sur la figure 4.5 que la vitesse de convergence de l'entropie dépend aussi de la précision du modèle d'observation : moins le modèle d'observation est bruité, moins grand est le temps nécessaire pour que la moyenne de l'entropie tende vers zéro.



(a) Espérance de la somme pondérée d'entropies de l'état de croyance pour chaque instant t.

(b) Espérance de la somme pondérée de récompenses pour chaque instant t.

FIGURE 4.4 – Espérance de la somme pondérée d'entropies de l'état de croyance et de récompenses.



(a) Moyenne de l'entropie de l'état de croyance du modèle de diagnostic.

(b) Moyenne de l'entropie de l'état de croyance du modèle de diagnostic moins bruité.

(c) Moyenne de l'entropie de l'état de croyance du modèle de diagnostic pas bruité.

FIGURE 4.5 – Moyennes de l'entropie de l'état de croyance en fonction du modèle d'observation.

Dans la suite nous évaluons le critère mixte pour la mission d'exploration traitée dans cette thèse, où but est d'acquérir suffisamment d'information pour identifier l'état caché du système. L'optimisation du critère devra tenir compte des coûts associés aux multiples déplacements de l'hélicoptère et de l'optimisation explicite de l'entropie de l'état de croyance.

Utilisation du critère mixte pour la mission d'exploration

Nous avons aussi réalisé le même type d'évaluation pour la mission d'exploration, décrite au chapitre 3. Dans celle-ci, l'agent autonome cherche à reconnaître l'identité des cibles dans les différentes zones, et ce, afin de reconnaître l'état caché du système. L'agent autonome doit se déplacer dans les différentes zones pour obtenir de l'information par rapport à la zone observée. La diminution de l'incertitude de son état de croyance devient le but de la mission. Nous rappelons que le critère mixte permettra de gérer le compromis entre la prise d'informations de l'environnement et les coûts associées aux différentes actions. Ces coûts sont ici proportionnels aux distances parcourues par l'hélicoptère.

Nous avons généré le modèle POMDP de cette mission d'exploration. Nous admettons que l'environnement est constitué de 3 zones distinctes séparées de 70 mètres les unes des autres. Initialement l'hélicoptère se trouve dans la zone 1 à une altitude de vol de 30 mètres, et aucune connaissance a priori sur l'identité ou la présence de cibles n'est disponible. Ce manque d'information se traduit par une distribution de probabilité uniforme sur les 64 états initiaux possibles.

Des politiques ont été calculées pour différentes valeurs de λ, et pour un ensemble d'états de croyance $\mathcal{B}$, tel que $|\mathcal{B}| = 5000$. Le temps moyen pour le calcul des politiques a été de 5 heures, pour $\epsilon \leq 2$. La valeur du seuil ϵ peut être considérée comme lâche, mais nous avons fait ce choix dû à l'approximation linéaire du premier ordre. Il apparaît que l'algorithme a tendance à osciller à partir d'un seuil lorsqu'on fait varier la valeur de la pondération λ. Ainsi, par l'expérience nous avons fixé ce seuil à 2.

Nous avons tracé le front de Pareto à partir de 12000 simulations de différentes politiques avec un horizon de 30 étapes de décision. Le front de Pareto obtenu est tracé sur la figure 4.6. Nous vérifions une fois de plus, que l'amélioration de l'un des deux objectifs (récompenses sur les états ou entropie de l'état de croyance) se fait au détriment de l'autre. Sur cette même figure, est également superposé l'écart type de chaque critère, qui permet de mettre en évidence encore une fois que cette méthode approchée de résolution conduit à des erreurs d'approximation significatifs.

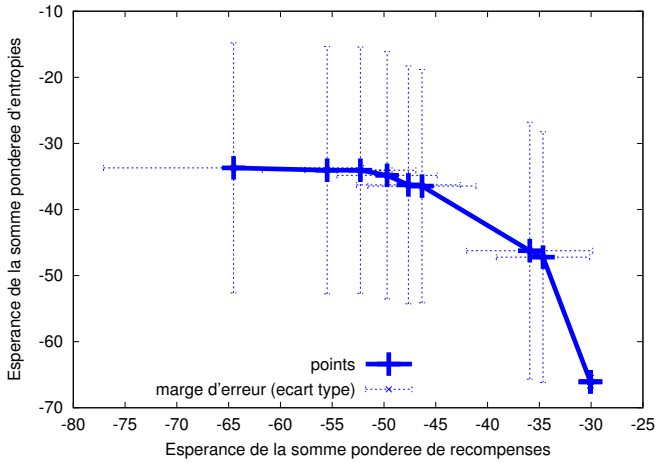
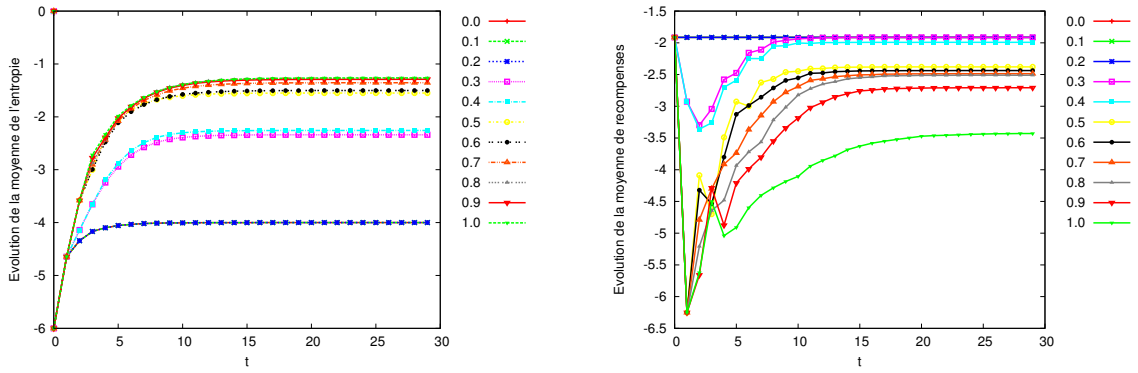


FIGURE 4.6 – Front de Pareto pour la mission d'exploration – $\lambda \in \{0, 0.1, 0.2, ..., 0.9, 1.0\}$.

.

L'évolution de la moyenne de l'entropie est tracée sur la figure 4.7(a). Nous pouvons ainsi voir que plus le poids de l'entropie est élevé, plus elle se trouve effectivement optimisée, ceci au prix de gains moyens moins élevés (figure 4.7(b)). Nous montrons aussi l'espérance de la somme pondérée de récompenses et d'entropies sur la figure 4.8 en accord avec les équations 4.41 et 4.42.

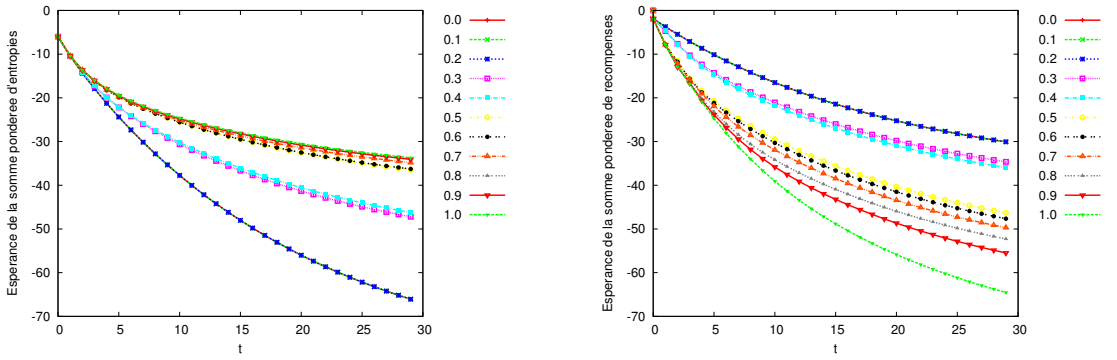
Comme précédemment, le comportement de l'agent sera différent en fonction de λ. Dans le cas où $\lambda = 0$, le critère optimisé se confond avec le critère classique du POMDP, qui ne prend pas en compte l'entropie associé à l'état de croyance, ce qui est à nouveau confirmé par le fait que la politique contient se réduit aux deux actions (goto h_1 et goto h_2) moins couteuses. L'entropie de l'état de croyance sur la zone 1, qui est la zone initialement observée, sera réduite ; par contre, comme aucune autre action de déplacement entre zones n'est présente dans la politique, l'entropie de l'état de croyance sur les autres zones ne sera pas réduite. Ceci explique le fait que la courbe rouge associée au symbole (–|–) sur la figure 4.7(a) reste bloquée à la valeur -4 à partir d'un certain temps (instant de décision t). Pour les cas où $\lambda = 0.1$ et 0.2, l'entropie n'a pas un poids suffisant vis-à-vis des récompenses associées au paires état-action moyennées sur l'état de croyance : ceci explique le fait que la politique confère au système le même comportement que pour $\lambda = 0$. Lorsque que nous faisons varier λ vers 1, la politique fournit à l'agent un comportement de plus en plus curieux malgré le coût des actions : plus l'agent acquiert de l'information, plus l'entropie de croyance est diminuée au prix d'un coût d'actions plus important.



(a) Moyenne de l'entropie de l'état de croyance à chaque instant t.

(b) Moyenne de la récompense à chaque instant t.

FIGURE 4.7 – Moyennes de l'entropie de l'état de croyance et de récompenses.



(a) Espérance de la somme pondérée d'entropies de l'état de croyance pour chaque instant t.

(b) Espérance de la somme pondérée de récompenses pour chaque instant t.

FIGURE 4.8 – Espérance de la somme pondérée d'entropies de l'état de croyance et de récompenses.

Toutes ces évaluations nous mènent à conclure que la détermination du bon λ à utiliser dépendra de l'importance que le concepteur du système autonome souhaitera donner aux différents critères. Ce résultat peut paraître décevant, puisque nous n'obtenons une manière unique de résoudre le compromis entre la prise d'information et la décision.

Nous discuterons dans la suite une approche alternative au critère mixte en se focalisant sur les avantages et inconvénients. Par la suite, nous comparerons les stratégies obtenues avec les différentes approches pour la mission d'exploration.

Comment faire autrement ?

Pour notre application, une approche alternative consiste à rajouter au modèle une action de terminasion de la mission, comme par exemple une action appelée *report*, qui permet à l'agent autonome de terminer sa mission en rapportant sa meilleure estimation de l'état du système lorsque sa croyance b tend à être certaine, telle que, $b(s) \rightarrow \delta_s$. Rappelons que le but est de reconnaître l'état caché du système observé. Une telle action peut dépendre d'un seuil de croyance η que l'on souhaite atteindre. Ainsi, il doit être pré-défini par le concepteur en fonction du coût "acceptable" pour obtenir un degré de certitude "suffisant". Si la croyance de l'agent sur un état donné du système dépasse ce seuil défini, cette action rapporterait une

récompense et conduirait à l'état terminal.

$$r(b, \texttt{report}) > 0, \text{ si } \max_{s \in S} b(s) > \eta \text{ et } r(b, \texttt{report}) \ll 0, \text{ sinon.}$$

Cette fonction de récompense est délicate. En effet, sa dépendance vis-à-vis des paires état-action n'est pas triviale à formaliser, puisque cette action ne dépend que de l'état de croyance courant de l'agent. Prenons le problème bien connu du tigre comme exemple : supposons que le but est de rapporter la position du tigre, l'agent disposant des actions écouter (coût de -1) et rapporter (récompense de 1 pour un $\eta = 0.9$ et de -2 sinon). La courbe de la fonction de valeur à $t = 0$ est tracée sur la figure 4.9. Cette figure illustre le fait que cette fonction de valeur est linéaire par morceau mais non convexe. Ceci empêche une application directe des algorithmes usuels de résolution des POMDP. Cependant cette approche montre que si on souhaite se ramener à une modélisation en POMDP classique on doit avoir une idée a priori de la solution pour fixer la valeur des récompenses.

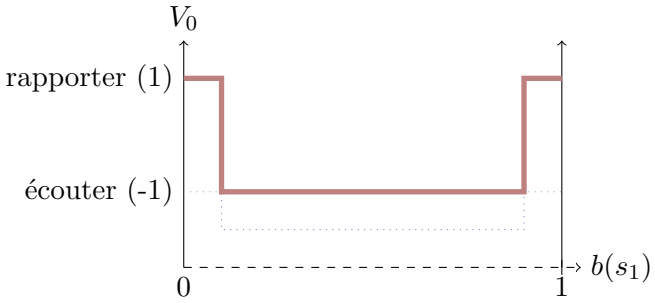


FIGURE 4.9 – Exemple d'un fonction de valeur à $t = 0$ non convexe pour le problème modifié du tigre.

Avec cette idée d'ajouter une action de type *report*, on peut vouloir se ramener à une modélisation sous forme de POMDP avec une fonction de récompense dépendant des paires état-action. Autrement dit, à la place de fixer η, on cherche à donner une récompense suffisamment grande où fait de rapporter correctement le véritable état. Ainsi, nous proposons en fait de résoudre la mission d'exploration par une autre approche. Plus précisément, nous rajoutons au modèle autant d'actions *report* qu'il y a d'états possibles, et ce, afin d'utiliser le *critère classique* du modèle POMDP dans un cadre de *perception active*, puisque chaque action *report* récompense l'agent si et seulement si l'état caché du système correspond à l'état rapporté.

4.1.3 Adaptation du modèle POMDP de la mission d'exploration par ajout d'actions

Cette nouvelle approche consiste à rajouter au modèle POMDP classique des actions de rapport de l'état du système, appelées *report* s_i, de sorte qu'une action supplémentaire est désormais offert à l'agent par état. La définition de telles actions est :

action report s_i : cette action permet à l'hélicoptère autonome d'affirmer que l'état réel du système qu'il observe est l'état s_i. Cette action conduit l'agent à l'état terminal du POMDP.

– fonction de récompense :

$$R(s, \texttt{report } s_i) = \mathbb{I}_{\{s=s_i\}} R_r - \mathbb{I}_{\{s \neq s_i\}} C_r$$

où $R_r > 0$ représente la récompense associée à l'action de rapporter l'état s_i lorsque celui-ci correspond effectivement à l'état caché du système que l'agent observe, et $C_r \gg 0$ le coût associé à cette action si l'état vrai n'est pas s_i.

Notons que dans cette approche, il est nécessaire d'ajouter une action par état, de façon à garantir que l'agent ne soit récompensé que pour l'état s_i concerné par l'action *report* s_i. L'avantage d'un tel modèle est que nous pouvons appliquer les algorithmes existants pour résoudre le POMDP et faire en suite des comparaisons avec les solutions obtenues par moyen d'un critère mixte.

D'autre part, nous rappelons que la complexité de la procédure mise à jour exacte de la valeur est liée au nombre d'α-vecteurs qui composent V_{n-1}, au nombre d'actions $|A|$ et qu'elle dépend de manière exponentielle du nombre $|\Omega|$ d'observations : $|V_{n+1}| = |A||V_n|^{|\Omega|}$. Ainsi, même si le facteur $|A|$ est relativement négligeable au regard de $|\Omega|$, nous pouvons être amenée à ajouter un nombre important d'actions au modèle en plus des actions standards de déplacement et de changement d'angle de vue. En revanche, si on se ramène au POMDP classique on peut appliquer des algorithmes efficaces basés sur la recherche heuristique, tels que HSVI [Smith et Simmons, 2005] et SARSOP [Kurniawati *et al.*, 2008], qui focaliseront l'optimisation de la valeur sur un petit nombre d'actions prometteuses selon l'état de croyance. Donc, le fait d'augmenter le nombre d'actions $|A|$ n'est pas si grave.

Pour évaluer cet approche, nous avons calculé des politiques à partir de différents algorithmes : PBVI (avec $|\mathcal{B}| = 5000$) que nous avons ré-implémenté ; HSVI [1] ; et SARSOP [2]. La durée de calcul de politiques a été limitée à 4 heures pour HSVI et SARSOP. Au bout de 4 heures de calcul ces deux algorithmes ont atteint une différence maximale entre les bornes supérieure et inférieure de la valeur de b_0 (état de croyance initial) de $\epsilon \leq 13$. Par suite, nous avons utilisé cette valeur pour ϵ comme critère d'arrêt de l'algorithme PBVI. Ainsi, pour PBVI, nous avons autorisé le processus itératif à se dérouler jusqu'à ce que l'erreur maximale sur la valeur des états de croyance appartenant à l'ensemble $\mathcal{B}$ soit égale ou inférieure à ϵ. Les résultats ont été obtenus au bout de 43 heures de calcul. Dans la table 4.1.3, nous quantifions le nombre d'α vecteurs $|V|$, le nombre d'états de croyance explorées $|\mathcal{B}|$ ainsi que le temps de calcul pour chacune des 3 résolutions.

algorithme	$\lvert\mathcal{B}\rvert$	$\lvert V\rvert$	temps (heures)
HSVI	n.a	47359	4
SARSOP	n.a	14783	4
PBVI	5000	2281	43

TABLE 4.1 – Résumé des performances associées aux différents algorithmes de résolution pour le modèle modifié (ajout d'actions de type *report*). n.a signifie que cette donnée n'est pas disponible pour l'utilisateur lors de la résolution.

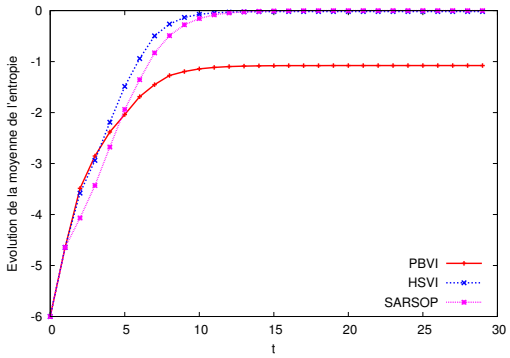
Nous pouvons constater que les algorithmes basés sur la recherche heuristique, c'est-à-dire HSVI et SARSOP, ont des performances supérieures à celle de l'algorithme PBVI qui repose sur une recherche stochastique. Il est indispensable de focaliser la recherche sur un nombre réduit d'actions. Nous avons augmenté le nombre d'états de croyance utilisés dans l'algorithme PBVI, c'est-à-dire $\mathcal{B}$ mais notre implémentation a atteint la limite de mémoire disponible. La machine que nous avons utilisé est un Intel Duo Core2 avec 2Gb de mémoire et 2.13GHz. Nous tenons à remarquer que le temps de calcul pour ces politiques est important, du à la quantité d'actions à évaluer.

Dans la figure 4.10, nous montrons la moyenne obtenue pour 12000 simulations à chaque étape de décision pour l'entropie négative (équation 4.39) et la moyenne de récompenses (équation 4.40). Il est à noter que le simulateur connait l'état caché du système en attribuant les récompenses à chaque étape de décision.

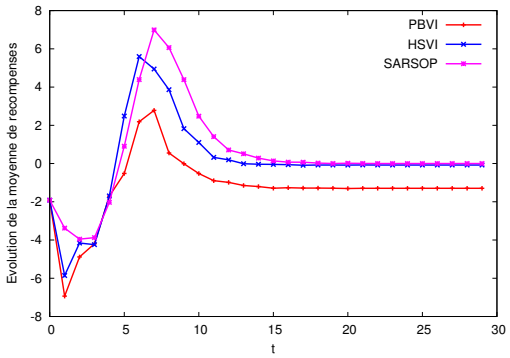
1. disponible en `http://www.cs.cmu.edu/~trey/zmdp/`
2. disponible en `http://bigbird.comp.nus.edu.sg/pmwiki/farm/appl/`

Nous montrons dans la figure 4.10(a) l'entropie qui permet d'évaluer l'incertitude de l'état de croyance de l'agent, illustrant ainsi l'évolution de la connaissance de celui-ci. Nous pouvons ainsi vérifier que l'incertitude de le l'état de croyance de l'agent tend vers zéro au fur et à mesure qu'il acquiert de l'information. Dans la figure 4.10(a), l'entropie moyenne de l'état de croyance tend vers zéro pour HSVI et SARSOP, et vers -1 pour PBVI. Ceci peut être expliqué par le fait que l'algorithme PBVI, en étant limité en nombre d'états de croyance, ne dispose pas d'actions optimisées pour certains états de croyance potentiellement rencontrés lors des simulations. Il peut se retrouver ainsi dans l'incapacité d'appliquer une action optimale à certains états de croyance dans certaines simulations. L'action utilisée est alors une action sous-optimale qui ne permet pas d'acquérir l'information manquante vis-à-vis de certains états de croyance dans lesquels se retrouve l'agent.

Dans la figure 4.10(b) l'évolution moyenne des récompenses nous montre que l'agent décide de rapporter l'état caché du système au bout de 10 étapes de décision en moyenne. Ceci est mis en évidence par le pic observé sur les fonctions de récompense observées entre les étapes de décision 5 et 10. La politique obtenue par l'algorithme PBVI a une courbe de valeur inférieure à celles des algorithmes HSVI et SARSOP. Cette observation s'explique par la même analyse que celle faite précédemment.



(a) Moyenne de l'entropie de l'état de croyance à chaque instant t.



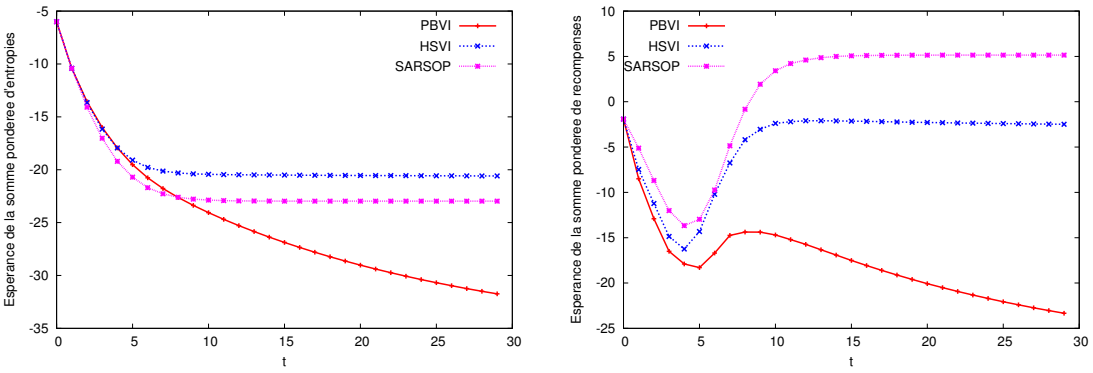
(b) Moyenne de la récompense à chaque instant t.

FIGURE 4.10 – Moyennes de l'entropie de l'état de croyance et des récompenses.

Il est à noter que le moment de prise de décision *finale* (action *report* s) est déterminé par le choix *a priori* de modélisation des C_r et R_r, ce qui correspond à une alternative au choix *a priori* d'une valeur de seuil η sur l'état de croyance.

Dans la figure 4.11, nous montrons les courbes de l'espérance de la somme pondérée d'entropies négatives H^π (équation 4.41) et de récompenses V^π (équation 4.42) pour chaque horizon t. L'espérance de la somme pondérée des récompenses est en effet le critère optimisé par la politique du POMDP lorsque celle-ci est simulée un nombre suffisant de foi. Ce critère est généralement utilisé comme mesure de la performance d'une politique. D'autre part, l'espérance de la somme pondérée des entropies met en évidence la vitesse de convergence de la croyance de l'agent. Une fois de plus les résultats caractérisant l'algorithme PBVI sont inférieurs à ceux de HSVI et SARSOP, toujours pour la même raison : PBVI est limité en nombre d'états de croyance et ne dispose donc pas d'actions optimales pour certains états de croyance rencontrées au cours des simulations.

Ces résultats nous permettent de conclure que cette approche, par ajouts de buts fictifs par moyen d'actions supplémentaires *rend possible* l'utilisation directe du critère classique ainsi que des algorithmes classiques de résolution des POMDP. Nous pouvons modéliser des récompenses sur ces actions *report* par des paires état-action, et résoudre ainsi le problème de perception active avec le formalisme classique de POMDP. Le point faible de la méthode



(a) Espérance de la somme pondérée d'entropies de l'état de croyance pour chaque instant t.

(b) Espérance de la somme pondérée de récompenses pour chaque instant t.

FIGURE 4.11 – Espérance de la somme pondérée des entropies de l'état de croyance et des récompenses.

concerne le nombre d'états qui conditionne de la même manière celui des actions. Ceci peut être un facteur limitant, au regard de l'algorithme de résolution employé. Toutefois, nous avons vérifié que l'utilisation des algorithmes efficaces basés sur la recherche heuristique nous permet surmonter ce problème. L'autre point faible de cette approche est qu'il faut choisir a priori la structure de récompense R_r et C_r pour obtenir ensuite un certain comportement. Ce choix est souvent empirique.

En évitant une modélisation explicite des actions de type *report*, nous avons vu qu'il était nécessaire d'encoder le besoin d'acquérir de l'information au sein du critère d'optimisation autrement que par les paires état-action. Dans ce cas, l'optimisation de l'incertitude associée à l'état de croyance de l'agent est explicite puisque celle-ci est retrouve directement modélisée dans la fonction de récompenses ρ. De la sorte, l'optimisation de la politique peut tenir compte de l'incertitude de l'état de croyance ainsi que des coûts/récompenses associés aux déplacement de l'agent. On met en évidence une limitation concrète du modèle POMDP classique qui ne permet pas de prendre en compte dans un même cadre l'optimisation de l'incertitude sur l'état de croyance et des coûts des actions.

Dans la suite, nous nous sommes intéressée à une comparaison des stratégies obtenues par les différentes approches pour la mission d'exploration. Cette comparaison particulière reste cependant délicate, puisque l'approche par ajouts d'actions présentée dans cette sous-section traduit directement au sein du modèle de récompense le moment de la prise de décision de rapporter l'état caché du système (R_r et C_r). De plus, la formulation du problème d'optimisation fondée sur un critère mixte, présentée dans la sous-section 4.1.2, nécessite de fixer un seuil sur l'état de croyance de l'agent pour la décision finale. La définition de ce seuil, qui implique d'arrêter toute observation pour identifier l'état caché, revient à l'utilisateur lors de l'application (simulation) de la politique. De la même façon, le choix des récompenses R_r et C_r est aussi fait par l'utilisateur, mais au moment de la modélisation.

4.1.4 Comparaison entre les approches par ajout d'action et de résolution avec un critère mixte pour la mission d'exploration

Étant donné que le critère mixte que nous avons proposé a été implémenté en modifiant l'algorithme PBVI, nous désirons comparer les stratégies obtenues sur la base de cet algorithme. Cette comparaison repose sur, d'une part, la stratégie obtenue à partir du critère mixte pour différentes valeurs de la pondération λ et la stratégie obtenue pour le modèle modifié qui intègre des actions de type *report*, d'autre part. Du fait des spécificités et des

différences importantes entre les deux approches, le recours à l'algorithme de résolution PBVI uniquement permet de mener une comparaison objective des politiques obtenues. Pour les deux approches nous avons fixé la taille l'ensemble d'états de croyance à 5000.

D'autre part, comme cela a déjà été démontré dans la section 4.1.3, la performance de l'algorithme PBVI est relativement pauvre comparée à celle des autres algorithmes. Afin de contourner cette limitation, nous avons moyenné nous résultats sur l'ensemble des trajectoires pour lesquelles la politique a rapporté un état dans le temps de mission imparti, fixé à 15 étapes de décision.

Pour le critère mixte, le moment de prise de décision finale, c'est-à-dire l'instant auquel le rapport de l'état caché du système est effectué, revient au concepteur du système robotique lorsque celui-ci met en œuvre la politique. Nous savons que ce choix est arbitraire, nous rappelons ici, que le choix de la valeur de R_r et C_r est également arbitraire. Ainsi, nous comparons par ailleurs différents seuils de prise de décision finale. Cette décision finale s'appuie sur l'état de croyance de l'agent. Nous supposons qu'une fois que l'état de croyance de l'agent à atteint certain niveau (c'est-à-dire, que la probabilité $b(s)$ a dépassé une valeur pré-définie) l'agent décidera de rapporter l'état s concerné et recevra en conséquence une récompense qui sera soit associée à une bonne, soit à une mauvaise décision, stoppant ainsi l'application la politique. La récompense attribuée est la même qui a été fixée pour le modèle de la sous-section 4.1.3 (ajout d'actions). On obtient ainsi une moyenne de la somme pondérée de récompenses qui peut être comparée à celle du modèle par ajout d'actions. Il est à noter que nous avons ainsi une comparaison qui nous semble juste entre deux manières relativement arbitraires de modéliser le comportement de l'agent, qui est obtenu par résolution d'un modèle classique (ajout d'action) ou modifié (critère mixte) de POMDP.

Éléments de comparaison

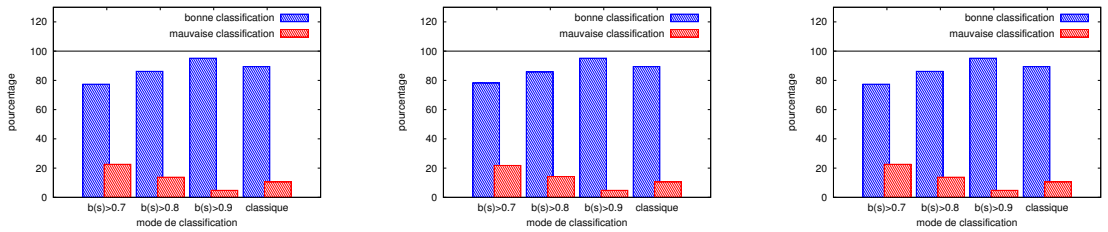
Pour comparer les approches, nous avons évalué quatre critères de performance :

1. le pourcentage de classifications bonnes, c'est-à-dire le nombre relatif de fois que l'agent a correctement classé l'état caché du système ;
2. le pourcentage de mauvaises classifications, c'est-à-dire le le nombre relatif de fois que l'agent a classé l'état caché du système de manière incorrecte ;
3. l'évolution de la moyenne de l'entropie de l'état de croyance et l'espérance de la somme pondérée des entropies, afin de vérifier la vitesse de convergence de la croyance de l'agent ;
4. l'évolution de la moyenne des récompenses et l'espérance de la somme pondérée ce celle-ci, afin de vérifier le coût engendré globalement par la politique ;

Le résultats moyens ont été calculés pour 1000 simulations de la politique à partir d'un état de croyance initial qui correspond à une distribution de probabilité uniforme sur les 64 états possibles. Cette croyance uniforme est indicative de la connaissance subjective de l'agent au début de mission. Autrement dit, elle permet de modéliser le fait que l'agent ne connait pas l'état initial, et qu'il n'a aucune préférence a priori. Nous avons simulé le système à partir d'une distribution initial uniforme pour rester dans un domaine connu de comparaison. Nous tenons à remarquer qu'en situation réelle il n'y a aucune raison pour que l'état de croyance initial répartit corresponde à l'état initiaux (ou à la fréquence d'occurrence des états initiaux) de l'agent et son environnement : au contraire l'état réel en expérimentation robotique est souvent un seul. De plus, pour cette comparaison nous avons choisi trois valeurs du facteur de pondération λ, $\lambda = \{0.5, 0.7, 0.9\}$, et trois seuils de prise de décision finale, $b(s) > 0.7$, $b(s) > 0.8$, $b(s) > 0.9$.

Résultats

Tout d'abord nous présentons dans la figure 4.12 les pourcentages de bonne et de mauvaise classifications. Les différentes colonnes sont associées aux seuils utilisés pour la classification pendant la simulation de la politique obtenue pour le critère mixte et au mot clef *classique* qui se réfère au modèle modifié pas ajouts d'actions qui optimise le critère classique des POMDP.



(a) Pourcentage de bonne et de mauvaise classification pour $\lambda = 0.5$.

(b) Pourcentage de bonne et de mauvaise classification pour $\lambda = 0.7$.

(c) Pourcentage de bonne et de mauvaise classification pour $\lambda = 0.9$.

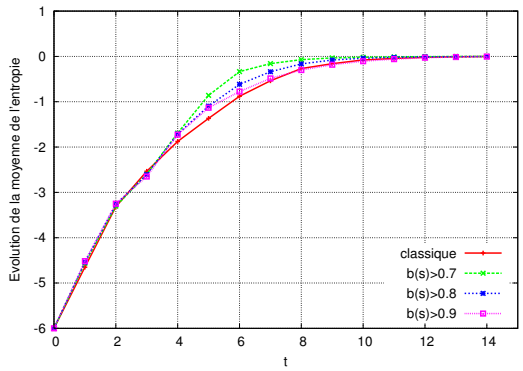
FIGURE 4.12 – Pourcentage de bonnes et de mauvaises classifications selon la valeur de λ.

Ces résultats montrent que selon le seuil spécifié par l'utilisateur les deux approches – critère mixte et ajout d'actions – peuvent être considérées comme équivalentes, en particulier pour un seuil $b(s) > 0.8$. De plus, pour le seuil $b(s) > 0.9$, le pourcentage de bonnes classifications est légèrement supérieur à celui obtenu à partir du critère classique relatif au modèle modifié. Il est à relever que ce résultat n'est pas nouveau. Il est connu de la communauté scientifique qui travaille dans le domaine de la perception active. Le recours à un critère mixte, comme celui défini par [Mihaylova *et al.*, 2002] (montré dans l'équation 1.3 du chapitre 1), permet en effet d'obtenir ce genre de performances. Toutefois, notre analyse comparative entre les deux approches est nouvelle à notre connaissance et n'a fait l'objet de publications.

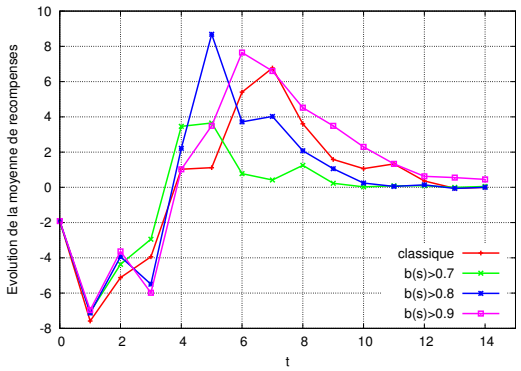
Il est aussi à noter que le pourcentage de bonnes classifications est toujours supérieur au seuil d'arrêt choisit. De plus ces résultats permettent de confirmer que la modélisation sous forme de POMDP classique (ajout d'actions) rend complètement implicite de "réglage" du seuil de décision au travers du choix du rapport entre R_r et C_r. Pour un utilisateur il est plus naturel de fixer un taux de bonnes/mauvaises classifications que l'on souhaite, que de définir le rapport de récompense (R_r et C_r).

Ainsi, nous pensons que cette comparaison est un outil très constructif : l'analyse du comportement d'une politique obtenue par moyen d'un critère mixte avec un seuil de décision nous aide à définir le *bon* rapport entre R_r et C_r afin de respecter un niveau de bonnes classifications. Nous avons fixé cette récompense à 50 et le coût associé à une mauvaise classification à 100. Nous pensons que plus la récompense sera petite plus l'agent autonome sera exigent sur son niveau de croyance pour la prise de décision *finale* de classification. De cette façon, nous pensons que ces approches en plus d'être considérées comme équivalentes peuvent être considérées comme complémentaires au regard d'un expert.

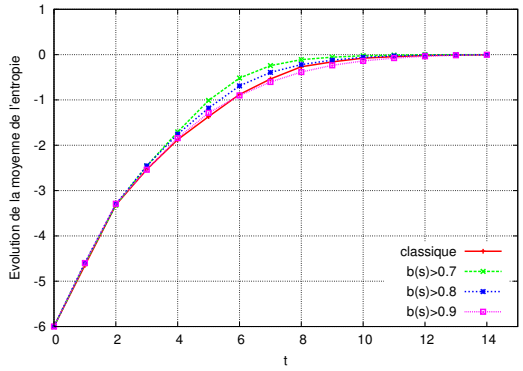
La figure 4.13 regroupe les résultats concernant l'évolution de la moyenne de l'entropie de l'état de croyance et des récompenses. Cette comparaison est encore plus délicate à réaliser que la précédente, étant donné que dans l'approche par ajout d'actions, les coûts et les récompenses donné à l'agent sont pris en compte directement dans le calcul de la politique, et dans le cas du critère mixte les récompenses associées à la classification ne sont pas prises en compte lors de l'optimisation. Les récompenses artificielles concernant les bonnes et les mauvaises classifications ont été ajoutées lors de la simulation de la politique.



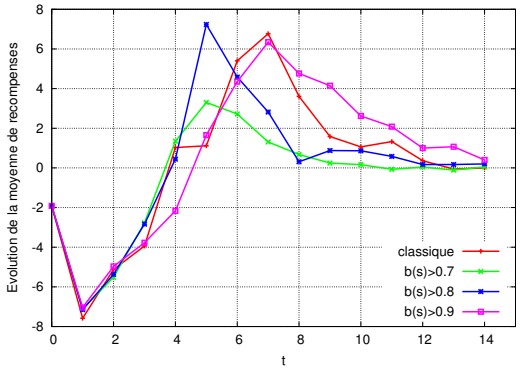
(a) Moyenne de l'entropie de l'état de croyance à chaque instant t pour $\lambda = 0.5$.



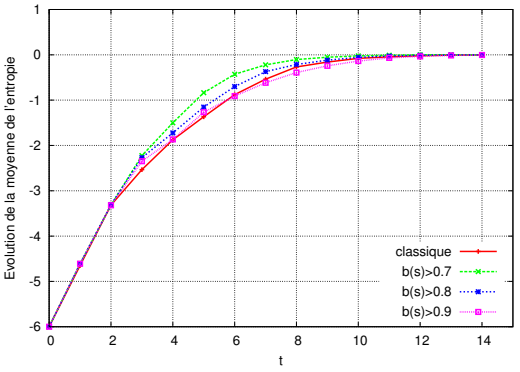
(b) Moyenne de récompenses à chaque instant t pour $\lambda = 0.5$.



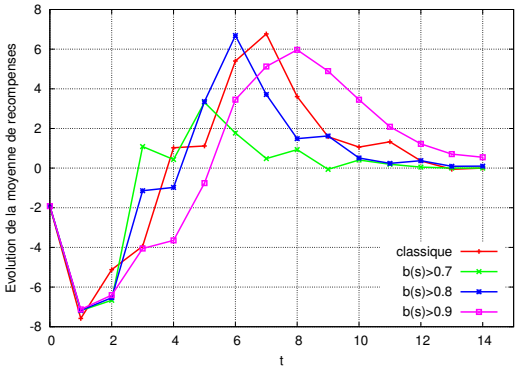
(c) Moyenne de l'entropie de l'état de croyance à chaque instant t pour $\lambda = 0.7$.



(d) Moyenne de récompenses à chaque instant t pour $\lambda = 0.7$.



(e) Moyenne de l'entropie de l'état de croyance à chaque instant t pour $\lambda = 0.9$.

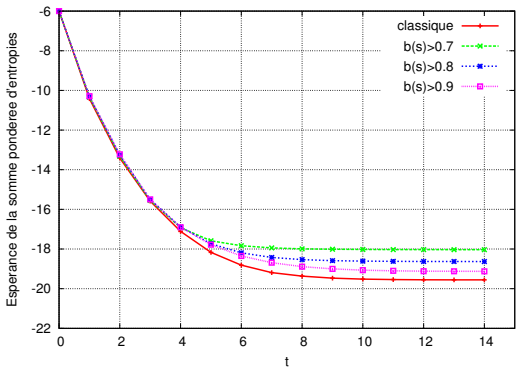


(f) Moyenne de récompenses à chaque instant t pour $\lambda = 0.9$.

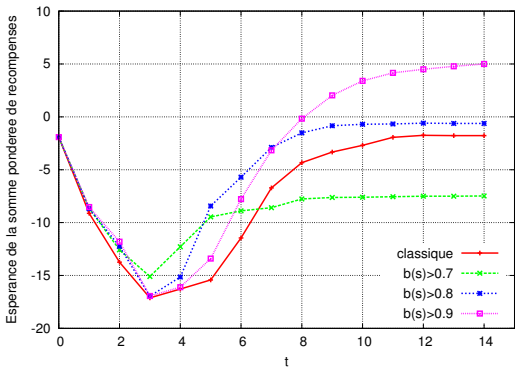
FIGURE 4.13 – Évolution moyenne de l'entropie de l'état de croyance et de récompenses selon la valeur de λ.

L'évolution de la moyenne de l'entropie de l'état de croyance suit quasiment les mêmes variations quelque soit la valeur de la pondération λ. Ceci n'est pas surprenant étant donné que pour des valeurs de λ supérieures à 0.5 (figure 4.7(a)), la vitesse de convergence de l'entropie reste la même. Ceci tend à démontrer une fois de plus que les deux approches – critère mixte et ajout d'actions – sont équivalentes.

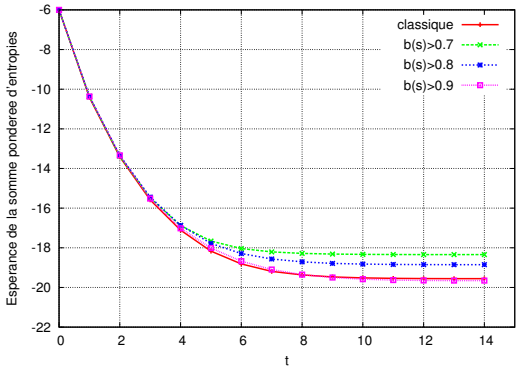
La différence entre ces deux approches est illustrée sur les courbes qui représentent la moyenne des récompenses à chaque instant t (figures 4.14(b), 4.14(d) et 4.14(f)). Sur ces figures nous pouvons observer que l'instant de prise de décision *finale* change selon le seuil fixé (pic des fonctions). Plus l'on est exigeant vis-à-vis de la probabilité $b(s)$ associée à l'état



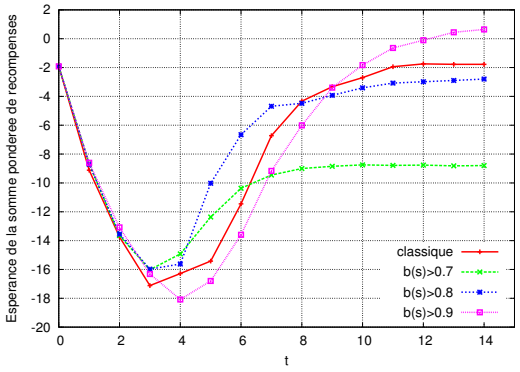
(a) Espérance de la somme pondérée d'entropies de l'état de croyance pour chaque instant t pour $\lambda = 0.5$.



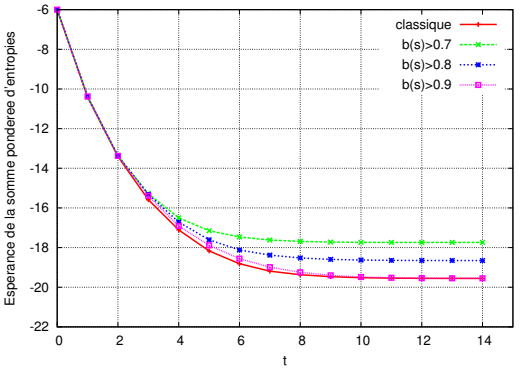
(b) Espérance de la somme pondérée de récompenses pour chaque instant t pour $\lambda = 0.5$.



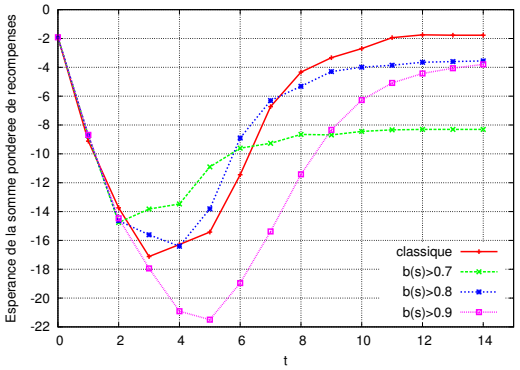
(c) Espérance de la somme pondérée d'entropies de l'état de croyance pour chaque instant t pour $\lambda = 0.7$.



(d) Espérance de la somme pondérée de récompenses pour chaque instant t pour $\lambda = 0.7$.



(e) Espérance de la somme pondérée d'entropies de l'état de croyance pour chaque instant t pour $\lambda = 0.9$.



(f) Espérance de la somme pondérée de récompenses pour chaque instant t pour $\lambda = 0.9$.

FIGURE 4.14 – Espérance de la somme pondérée d'entropies et de récompenses selon la valeur de λ.

caché, plus tard l'agent déclenchera sa décision *finale*.

Sur la figure 4.14, est tracée l'espérance de la somme pondérée des entropies et des récompenses pour les différentes valeurs de la pondération λ (selon équations 4.41 et 4.42). L'espérance de la somme pondérée des récompenses atteint une valeur supérieure pour le critère mixte pour les valeurs de $\lambda = 0.5$ et $\lambda = 0.7$ avec le seuil de décision supérieur à 0.7. Ceci peut être expliqué par le fait que les coûts associés aux déplacements de l'agent pèsent plus lors du calcul de la politique. Ainsi, l'agent cherchera de l'information de manière moins

coûteuse que dans le cas où $\lambda = 0.9$ par exemple.

Bilan partiel de la mission d'exploration

Nous pouvons aussi conclure que l'utilisation d'un critère mixte, non-linéaire, peut trouver une utilité dans la réalisation de missions exploratoires ou dans le traitement de problèmes de diagnostic. Dans le cas de ces missions, le but de l'agent est de réduire l'incertitude de son état de croyance. L'utilisation d'un critère non-linéaire que l'on optimise à partir des algorithmes issus de l'état-de-l'art de POMDP est possible, mais l'utilisateur de la méthode devra, soit utiliser des approximations linéaires du premier ordre basées sur un grand nombre d'état de croyance afin d'estimer au mieux le gradient de la fonction de valeur, soit mettre en place un algorithme de résolution dédié, pour lequel la fonction de valeur ne serait plus paramétrée par des α-vecteurs.

Nous pensons qu'un algorithme capable d'approcher une fonction de valeur non linéaire serait une bonne alternative, étant donné ce qui a été démontré tout au long de cette partie : l'opérateur de Bellman est une application contractante dans l'espace des fonctions de valeur. [Araya-López *et al.*, 2010] démontre qu'un algorithme capable d'approcher une telle fonction en utilisant un ensemble d'états de croyance, pourrait permettre de déterminer une politique ϵ-optimale, où l'erreur d'approximation ϵ peut être borné par $\frac{(R_{max}-R_{min}+C\delta_{\mathcal{B}}^{\alpha})\delta_{\mathcal{B}}}{1-\gamma}$, avec R_{max} et R_{min} les valeurs extrêmes des récompenses, $C\delta_{\mathcal{B}}^{\alpha}$ la borne supérieure de l'erreur de l'approximation linéaire des α-vecteurs et $\delta_{\mathcal{B}}$ la densité de couverture (points) de l'espace de croyance [Pineau *et al.*, 2003]. Nous avons vu que dû à la difficulté de l'utilisation d'une approximation linéaire du premier ordre, la valeur peut osciller pour certains états de croyance pour un ϵ éventuellement *petit*. De plus, le nombre d'états de croyance nécessaire pour approcher correctement la fonction de valeur peut s'avérer très grand : nous avons calculé des politiques pour des tailles d'ensemble égales à : 1000, 2000 et 5000. Nous avons observé que plus le nombre d'états de croyance est élevé, plus l'évolution moyenne de l'entropie tend vers une valeur proche de zéro (plus de points d'approximation). A moyen de calcul égaux par ailleurs notre algorithme n'a pas été capable de calculer une politique pour un ensemble d'états de croyance supérieur à 5000.

De plus, nous avons approfondi la réflexion sur la modélisation de la mission d'exploration de manière à ce que le critère optimisé soit le critère classique du POMDP. Notre réflexion nous a amené à étudier un modèle pour lequel on ajoute des buts fictifs au moyen d'actions dites *report s* pour chaque état s du système. Autrement dit l'agent décideur sera récompensé si et seulement si, il réalise l'action *report s* quand l'état s est l'état véritable du système. Il est à noter que l'optimisation de ces actions (buts) ne dépend plus de manière directe d'une mesure de l'incertitude de l'état de croyance puisque la récompense associée peut être modélisée pour les paires état-action. La politique appliquant ces actions réduira de manière implicite l'incertitude de l'état de croyance. L'ajout d'actions (buts) peut paraître un facteur limitant de l'approche, mais le retour vers une modélisation classique nous permet d'utiliser des algorithmes efficaces basés sur la recherche heuristique, qui focalisent l'optimisation vers les actions les plus prometteuses. Ceci permet d'éviter une évaluation exhaustive de toutes les actions du modèle. De plus, grâce à la comparaison des deux approches on peut proposer une structure des récompenses et des coûts du modèle POMDP classique tel que la politique obtenue corresponde à un taux de bonne classification comparable à celui obtenue à partir du critère mixte.

Ainsi, le critère mixte permet d'implémenter et d'optimiser une stratégie ρPOMDP qui respecte un taux de bonnes et mauvaises classifications défini comme le seuil de décision d'exécution d'une action *report s*, étant donné une pondération λ fixée dans le critère mixte. Il est à noter que la détermination de la pondération λ reste arbitraire. D'autre part le rapport

entre récompenses et coûts dans une modélisation classique définit de manière implicite ces taux de bonnes et de mauvaises classifications, qui ne peuvent être identifiés que par des simulations de la politique obtenue. D'une certaine manière l'approche par critère mixte peut être utilisée comme un outil constructif pour la détermination du *bon* rapport entre les récompenses et les coûts en regard des objectifs de taux de bonnes et mauvaises classifications de l'utilisateur (choix du paramètre η, puis réglage du paramètre λ juste suffisant, puis ajustement des récompenses sur les états terminaux dans le modèle classique) . Cela nous permet de conclure que ces approches sont non seulement comparables et équivalentes en termes de réduction d'incertitude mais aussi complémentaires.

Nous retenons donc que :

- l'utilisation d'un critère mixte non-linéaire est possible d'un point de vue algorithmique et parfois efficace ;
- il est possible d'évaluer et de comparer la politiques obtenue par un critère mixte (pour un seuil η et une pondération λ choisie) avec celle obtenue en choisissant des récompenses et des coûts adaptés pour le modèle classique (ajout d'actions) ;
- il ressort de la comparaison que le critère mixte permet de définir un taux acceptable des bonnes et des mauvaises classifications (choix d'un seuil η et ajustement du λ) ;
- il est possible de revenir ensuite vers une modélisation classique POMDP par ajout des buts fictifs en ayant des éléments objectifs de comparaison pour fixer le *bon* rapport entre R_r et C_r.

Dans la suite de ce chapitre, nous procédons à une évaluation de la politique pour la mission d'atterrissage. Il s'agit là d'un problème de décision séquentielle qui est modélisé directement à partir du formalisme POMDP classique en ce qui concerne la structure de la fonction de récompenses. Nous rappelons que nous nous sommes aussi intéressée à ce cadre applicatif afin de vérifier que le POMDP est capable de gérer implicitement l'acquisition d'information pour atteindre le but de la mission. Ce but est d'atterrir à côté d'une cible particulière reconnue parmi d'autres présentes ou non dans l'environnement.

4.2 POMDP et mission d'atterrissage

Notre deuxième cadre applicatif consiste à détecter et à identifier des cibles dans un environnement incertain et partiellement observable et ce afin d'atterrir à côté celle recherchée. Cette mission a été modélisée par un POMDP (voir chapitre 3). Plus précisément, les cibles s'apparentent à des modèles de voitures présentes dans la scène observé, avec une préférence pour un modèle en particulier. Cette mission se ramène de manière implicite à un problème de perception active. Dans cette mission d'atterrissage l'acquisition d'informations constitue un moyen, et non une fin, ce qui rend la mission assimilable à un problème de décision avec observabilité partielle qui peut être modélisé à partir d'un POMDP classique.

Dans ce type de mission nous n'avons pas besoin de rajouter d'actions supplémentaires de type *report* à la modélisation. Une seule action, appelée *land* (cf. chapitre 3) est nécessaire, puisque nous pouvons modéliser directement par des paires état-action les récompenses en cas d'atterrissage à côté de la bonne cible.

Nous rappelons ici la modélisation de cette action :

action land : cette action finalise la mission de l'hélicoptère autonome, et l'amène à l'état terminal. Atterrir dans la zone où le modèle recherché se trouve, rapporte une récompense R_l. La variable mathématique $Se_{Ta} \in \{model_A, model_B, model_C\}$ encode un des modèles existants dans la base de donné, sachant que le modèle recherché peut changer d'une mission à une autre. Si l'hélicoptère est dans l'état terminal (T_s), cette action n'a ni effets ni coût.

- fonction de transition : $T(s', \texttt{land}, s) = \delta_{T_s}(s')$

– fonction de récompense :

$$R(s, \texttt{land}) = \quad \mathbb{I}_{\{(z=z_1)\&(Id_{Ta_{z_1}}=Se_{Ta})\}} R_l + ... + \mathbb{I}_{\{(z=z_{N_z})\&(Id_{Ta_{z_{N_z}}}=Se_{Ta})\}} R_l - \mathbb{I}_{\{(z=z_1)\&(Id_{Ta_{z_1}}\neq Se_{Ta})\}} C_l - ... - \mathbb{I}_{\{(z=z_{N_z})\&(Id_{Ta_{z_{N_z}}}\neq Se_{Ta})\}} C_l$$

où $R_l > 0$ représente la récompense associée à une mission d'atterrissage réussie, autrement dit, l'hélicoptère atterrit dans la zone où la cible recherchée se localise, et $C_l \gg 0$ représente le coût d'une mission d'atterrissage manquée. Notez que : $C_{proc} < C_{h,\hat{h}} < C_{z,\hat{z}} \ll C_l$.

De cette façon, l'application du critère classique des POMDP devrait être suffisante pour obtenir une politique capable de gérer de manière implicite la prise d'information dans l'environnement.

Pour vérifier cette supposition, nous allons par la suite : premièrement, évaluer différentes politiques calculées à partir du critère classique ; deuxièmement, à partir du critère mixte proposé pour la mission d'atterrissage. Nous cherchons ici à vérifier : (1) si la politique calculée avec le critère classique des POMDP gère correctement la prise d'information que nous pourrons mesurer par la réduction d'incertitude de son état de croyance ; et (2), si une politique calculée avec le critère mixte apporte un gain à la vitesse de convergence de l'entropie de l'état de croyance dans le traitement de cette application. Nous allons aussi analyser les pourcentages de réussite de mission. Ce pourcentage se rapporte au nombre de fois où l'agent autonome a décidé de se poser à côté de la cible recherchée.

4.2.1 Évaluation de la politique calculée avec le critère classique des POMDP pour la mission d'atterrissage

Pour l'évaluation des politiques obtenues avec le critère classique des POMDP, nous avons utilisé l'algorithme PBVI [Pineau *et al.*, 2003], que nous avons ré-implémenté, pour traiter trois variantes de la mission : (1) où l'on cherche à atterrir à côté de la voiture correspondant au modèle A ; (2) au modèle B ; et (3) au modèle C. Dans chaque mission, nous avons fixé le nombre de zones à 3. L'hélicoptère autonome se trouve au début de la mission dans la zone 1, à une altitude de 30 mètres. L'agent n'a aucune connaissance à priori sur le nombre ni sur la nature des cibles qui sont effectivement présentes dans la scène. Cette croyance subjective est modélisée par une distribution de probabilité uniforme sur les 64 états possibles. Ainsi l'état de croyance initial est défini tel que :

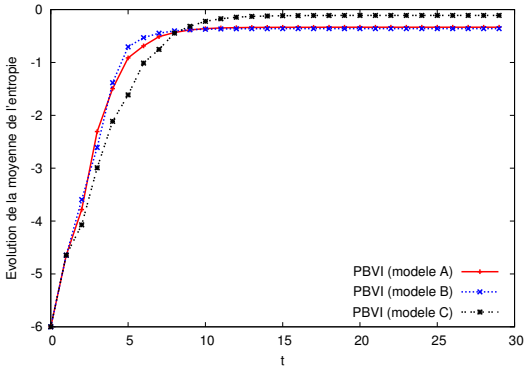
$$b_0 = \{z = z_1, h = h_1, Id_{Ta_{z_1}} = inconnu, Id_{Ta_{z_2}} = inconnu, Id_{Ta_{z_3}} = inconnu\}. \quad (4.43)$$

Le calcul de la politique avec l'algorithme PBVI a été fait pour un nombre d'états de croyance égal à 5000, et avec une paramétrage de $\gamma = 0.95$ et de $\epsilon = 0.5$. Ensuite, des simulations ont été réalisées afin de moyenner les résultats. Nous en avons réalisés 12000 avec un horizon de 30 étapes de décision.

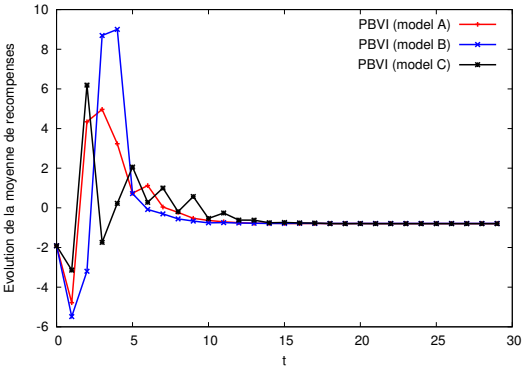
La figure 4.15 montre l'évolution de la moyenne des récompenses (équation 4.40) et de l'entropie (négative) de l'état de croyance (équation 4.39). Nous pouvons voir que pour les trois missions, l'évolution de la moyenne de l'entropie tend vers zéro (figure 4.15(a)). Dans le cas de la mission (3) (atterrissage à côté de la voiture de modèle C), l'entropie diminue moins vite . Ceci peut s'expliquer par le fait que le modèle d'observation relatif à ce modèle de voiture est moins informatif que ceux associés aux autres modèles, ce qui conduit l'agent à prendre plus d'informations, donc à réaliser plus d'actions avant d'atterrir. La table 4.2 montre le modèle d'observation utilisé, qui a été appris à partir de données réelles (cf. chapitre 3). Nous pouvons y voir que la probabilité d'observer le modèle C, étant donné que l'on l'observe effectivement ce modèle est moins importante que pour les autres modèles (63% contre 85% et même 92%).

modèle, altitude	non détecté	détecté non ident.	modèle A	modèle B	modèle C
modèle A, à 30 mètres	0.006103	0.117698	0.680907	0.060157	0.135135
modèle A, à 40 mètres	0.007340	0.067758	0.845285	0.000000	0.079616
modèle B, à 30 mètres	0.029536	0.111814	0.004219	0.848101	0.006329
modèle B, à 40 mètres	0.010526	0.044211	0.014737	0.915789	0.014737
modèle C, à 30 mètres	0.001000	0.027027	0.057432	0.280532	0.634009
modèle C, à 40 mètres	0.003589	0.022254	0.103374	0.241924	0.628859
aucune, à 30 mètres	0.882418	0.091308	0.008791	0.016484	0.001000
aucune, à 40 mètres	0.855918	0.140509	0.001000	0.001572	0.001000

TABLE 4.2 – Table avec les probabilités d'observation par modèle de voiture et altitude de vol.



(a) Moyenne de l'entropie de l'état de croyance à chaque instant t.



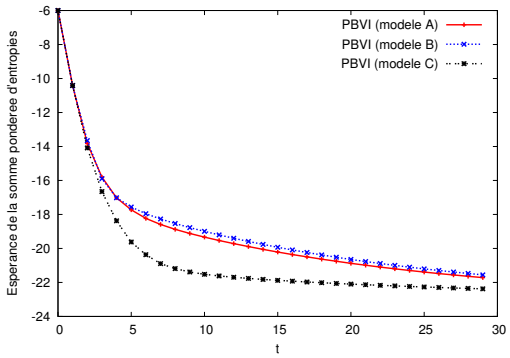
(b) Moyenne de la récompense à chaque instant t.

FIGURE 4.15 – Moyennes de l'entropie de l'état de croyance et de récompenses.

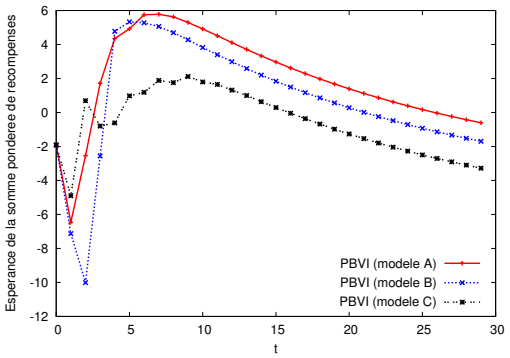
Le fait de devoir réaliser plus d'actions pour accomplir la mission où l'on cherche à atterrir à côté de la voiture de modèle C est aussi mis en évidence dans les courbes de la figure 4.16 où les espérances des sommes pondérées des récompenses (figure 4.16(b)) et des entropies de l'état de croyance (figure 4.16(a)) sont tracées en fonction du temps. La courbe relative à la somme pondérée des récompenses pour la mission (3) est située en-dessous de celles relatives aux autres missions en raison des actions supplémentaires qui sont nécessaires pour mieux distinguer les modèles. La courbe caractérisant la mission (2) (modèle B) est située sous le tracé rouge qui correspond au résultat de la mission (1) (modèle A). Ceci peut être expliqué par le fait que le modèle d'observation est plus informatif pour le modèle de véhicule B (table 4.2). L'agent à tendance à changer de zone plus tôt dans la mission, ce qui pénalise la somme pondérée pour cette mission, étant donné que le changement de zone est l'action la plus coûteuse du modèle.

L'espérance de la somme pondérée des récompenses, dont les variations dans le temps sont tracées sur la figure 4.16(b), ne converge pas vers une valeur constante. Une explication possible reside dans le fait que pour certaines simulations où la voiture recherchée n'est pas présente dans l'environnement, l'hélicoptère n'a pas forcément atterri en utilisant ainsi continument des actions de déplacement, ce qui est mis en évidence par la pente descendante des courbes.

La figure 4.21 illustre le pourcentage de missions réussies, de missions manquées, ainsi que le nombre moyen d'étapes jusqu'à l'atterrissage. Le taux de réussite est d'environ 56% (figure 4.17(a)), ce qui est en accord avec les états initiaux possibles. Autrement dit, pour ces simulations nous avons tiré l'état initial caché selon b_0. Encore plus précisément, au début de la mission, l'état de croyance suit une loi uniforme sur les 64 états possibles. Et, parmi ces



(a) Espérance de la somme pondérée d'entropies de l'état de croyance pour chaque instant t.



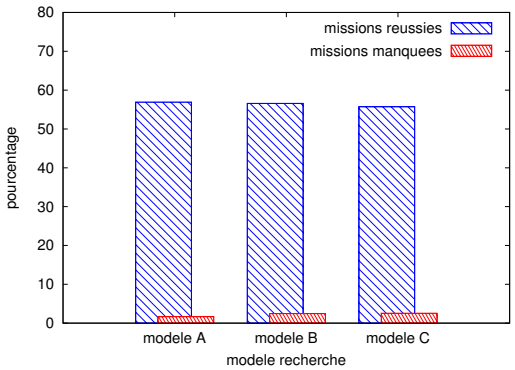
(b) Espérance de la somme pondérée de récompenses pour chaque instant t.

FIGURE 4.16 – Espérance de la somme pondérée d'entropies de l'état de croyance et de récompenses.

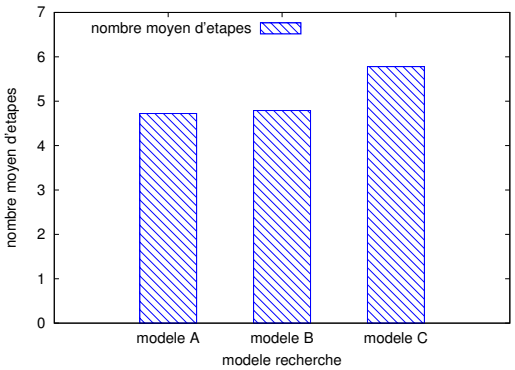
64 états, il y a 37 états pour lesquels la voiture recherchée est bien présente, ce qui conduit à une probabilité de réussite (selon état de croyance) de 58%. Le pourcentage de mission manquées se situe aux alentours de $1,5\%$ (pourcentage de fois que l'hélicoptère a atterri sur une mauvaise cible).

Nous pouvons vérifier sur la figure 4.17(b) que le nombre moyen d'étapes jusqu'à l'atterrissage pour la mission (3) est plus important que pour les autres missions, ce qui démontre que l'agent autonome a besoin d'acquérir plus d'information.

Ces résultats prouvent que pour cette application la politique obtenue avec le critère classique des POMDP gère de manière implicite une acquisition d'informations qui amène l'agent autonome à finaliser correctement sa mission, indépendamment de la nature de la cible recherchée. De plus, ces résultats démontrent aussi que le rapport choisit entre les récompenses R_l et C_l est, en quelque sorte, bien "réglé" permettent à la politique d'acquérir suffisamment d'information avant que l'agent décide ou non d'atterrir. Nous avons fixé $R_l = 50$ et $C_l = 100$, ces valeurs nous garantissent que la politique finira par y aller chercher l'information.



(a) Pourcentage de missions réussies et manquées.



(b) Nombre moyen d'étapes pour l'atterrissage.

FIGURE 4.17 – Pourcentages de missions réussies et manquées, et nombre d'étapes moyen avant l'atterrissage pour les 3 missions étudiées.

Dans la suite, nous nous intéressons au comportement que confère la politique calculée sur la base du critère mixte proposé dans le cadre de cette mission. Nous voulons vérifier si, pour ce type de mission, l'utilisation de ce type critère mixte peut favoriser convergence

plus rapide de l'entropie de l'état de croyance, ou bien, si le fait de modéliser une action de type *land*, qui oriente l'agent vers une cible en particulier, est suffisant pour permettre une évolution rapide de la croyance de l'agent.

4.2.2 Mission d'atterrissage et critère mixte

Dans cette section, nous cherchons à analyser les politiques obtenues à partir de l'emploi du critère mixte dans le cadre de la mission d'atterrissage. Pour cela, nous avons calculé une politique pour chaque valeurs de la pondération λ, et pour un ensemble d'états de croyance $\mathcal{B}$, où $\mathcal{B} = 5000$. Le temps moyen nécessaire pour le calcul de ces politiques a été d'environ 15 heures, pour un $\epsilon = 2$. Une fois encore, nous précision que la valeur de ce seuil ϵ est relativement lâche, dû à l'approximation linéaire du premier ordre. L'algorithme PBVI adapté par nous au critère mixte peut entrainer à des oscillations de la valeur des états de croyance qui appartiennent à $\mathcal{B}$ au de là de cette valeur fixée par expérience.

Il est à noter que le temps moyen nécessaire pour le calcul de la stratégie est supérieur à celui observé pour la mission d'exploration. Ceci peut être expliqué par le fait que le modèle POMDP, pour ce critère mixte, a besoin de plusieurs états terminaux. Plus précisément, comme l'entropie est directement optimisée, l'existence d'un seul état terminal pourrait engendrer à une finalisation précoce de la mission. La politique aurait tendance à atteindre au plus vite cet état terminal au fur et à mesure que la valeur de la pondération λ grandie. Ce fait a été confirmé par l'expérimentation. Pour éviter ce problème, plusieurs états terminaux ont été ajoutées au modèle.

Comme dans la section précédente, nous avons fixé le nombre de zones à 3 pour générer le modèle POMDP de la mission. De plus, dans cette mission l'hélicoptère autonome cherchera le modèle C de voiture, pour lequel le modèle d'observation est le moins favorable.

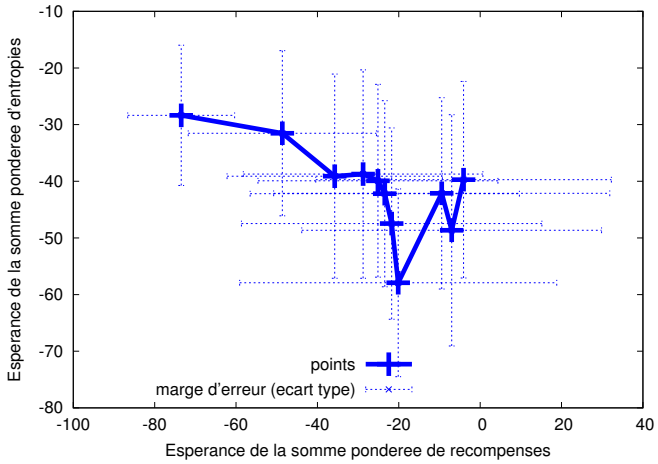
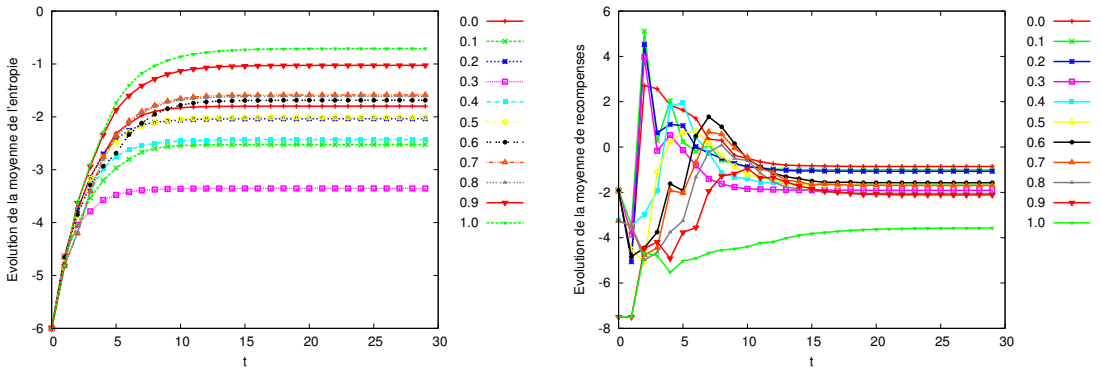


FIGURE 4.18 – Ensemble de solutions obtenues pour la mission d'atterrissage - $\lambda \in \{0, 0.1, 0.2, ..., 0.9, 1.0\}$.

.

Nous avons tracé l'ensemble des solutions à partir de 12000 simulations de différentes politiques pour un horizon composé de 30 étapes de décision, ceci est montré sur la figure 4.18. Cette fois-ci, ces solutions ne correspondent pas aux solutions non dominées. Nous vérifions que, pour cette mission, la courbe tracée à partir des solutions n'a pas une forme concave. Ceci traduit que la politique déterminée pour mener la mission vise essentiellement à réduire l'entropie de l'état de croyance afin de pouvoir identifier le modèle de voiture recherché (comme déjà été constaté dans la section 4.2.1). En d'autres termes, si l'on ajoute l'entropie au critère d'optimisation, celle-ci ne se trouve pas forcément optimisée. Ce fait



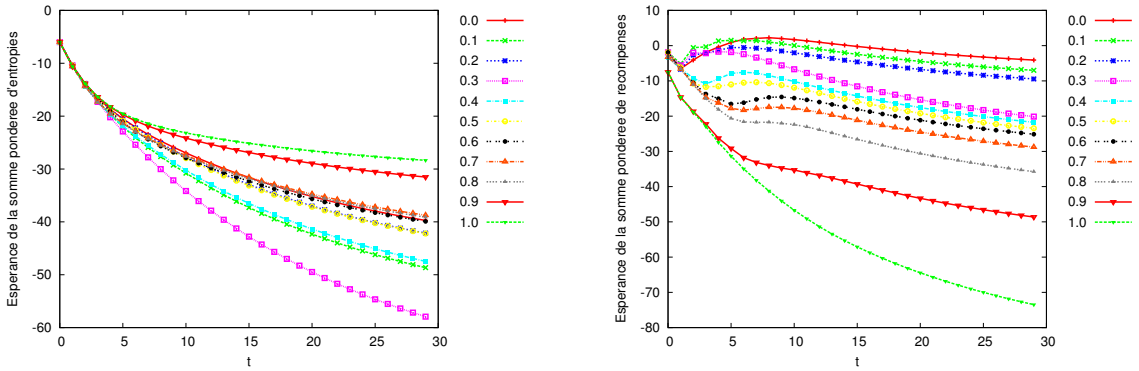
(a) Moyenne de l'entropie de l'état de croyance à chaque instant t.

(b) Moyenne de la récompense à chaque instant t.

FIGURE 4.19 – Évolution de la moyenne de l'entropie et de récompenses pour les différents valeurs de λ.

est clairement démontré pour des valeurs de λ comprises entre 0.1 et 0.5. Nous voyons que l'entropie commence à être optimisée pour des valeurs de λ supérieures à 0.5. La définition d'une valeur de la pondération λ dépendra entre autre du modèle de récompense.

L'évolution de la moyenne de l'entropie pour les différentes valeurs de λ est tracée sur la figure 4.19(a). Nous pouvons constater que pour des valeurs de λ supérieures à 0.5, l'entropie de l'état de croyance est réduite plus rapidement, conforme aux résultats de la figure 4.18, au prix de gains moyens moins élevés (voir figure 4.19(b)). L'agent a tendance à acquérir plus d'informations en "dépensant plus".



(a) Espérance de la somme pondérée d'entropies de l'état de croyance pour chaque instant t.

(b) Espérance de la somme pondérée de récompenses pour chaque instant t.

FIGURE 4.20 – Espérance de la somme pondérée d'entropies et de récompenses pour les différents valeurs de λ.

La figure 4.2.2 montre l'espérance de la somme pondérée des entropies et des récompenses obtenues lors de ces simulations. Notez que l'espérance de la somme pondérée des entropies pour $\lambda = 0$ ne se situe pas à l'une des extrémités de la figure du réseau de courbes de la figure 4.20(a). La politique obtenue dans ce cas là se rapporte à celle que l'on calculerait avec le critère classique. Ce fait démontre que donner une préférence à une cible en particulier pousse implicitement l'agent à réduire l'incertitude de son état de croyance de manière sensée. Cette courbe n'est dépassée que pour des valeurs de $\lambda > 0.5$, puisque à partir de cette valeur de λ la vitesse de convergence de l'entropie est améliorée.

Dans la suite nous comparons plus en détail les performances de différentes politiques en

ce que concerne le pourcentage de missions réussies et manquées pour la mission d'atterrissage.

4.2.3 Comparaison des politiques obtenues avec les critères classique et mixte.

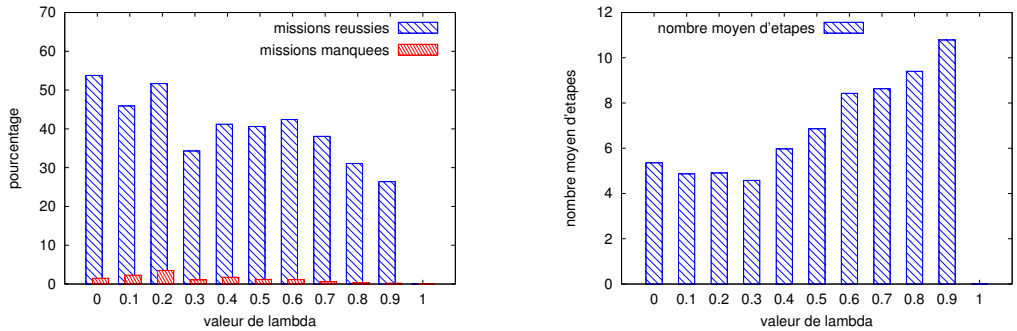
Éléments de comparaison

La comparaison des performances des différentes politiques portera sur le pourcentage de missions réussies, manquées ainsi que sur le nombre moyen d'étapes nécessaires à l'accomplissement de la mission. Les entropies et récompenses moyennes ont déjà été discutées auparavant. Le pourcentage de mission réussies correspond au nombre de fois où l'agent a décidé d'atterrir à côté de la bonne cible. A contrario, le pourcentage de mission manquées correspond au nombre de fois qu'il a choisi d'atterrir à côté d'une mauvaise cible. Le nombre moyen d'étapes pour accomplir la mission représente le nombre moyen d'actions nécessaires à l'agent d'acquérir d'informations pour aboutir à la décision finale d'atterrir.

Les résultats ont été calculés pour 12000 simulations de la politique à partir d'un état de croyance initial uniforme sur les 64 états possibles, et pour un horizon de 30 étapes de décision. La croyance initial uniforme permet de faire une comparaison en moyenne sur des situations de départ les moins spécifiques possibles.

Résultats

En reposant notre analyse sur les figures 4.21(a) et 4.21(b), nous nous apercevons que plus le poids accordé à l'entropie ($\lambda > 0.5$) est élevé, moins important est le taux de réussite de la mission. Ceci illustre directement que, dans ce cas précis, la réduction de l'entropie de l'état de croyance devient prépondérante vis-à-vis du le but de la mission. Ainsi, l'hélicoptère passera plus de temps à acquérir de l'information et mettra plus de temps à finaliser sa mission (voir figure 4.21(b)). En revanche, plus le poids associé à l'entropie est grand, plus petit est le nombre de missions manquées. L'hélicoptère autonome a tendance à *se tromper* moins de cible.



(a) Pourcentage de missions réussies et manquées.

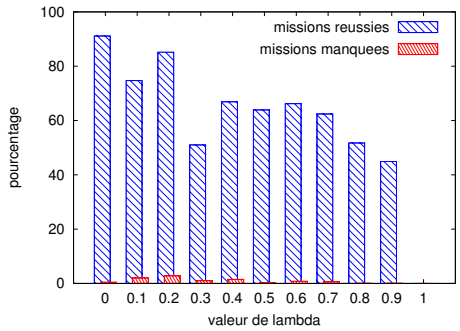
(b) Nombre moyen d'étapes pour l'atterrissage.

FIGURE 4.21 – Pourcentages de missions réussies et manquées, et nombre moyen d'étapes pour l'atterrissage pour les différentes valeurs de λ.

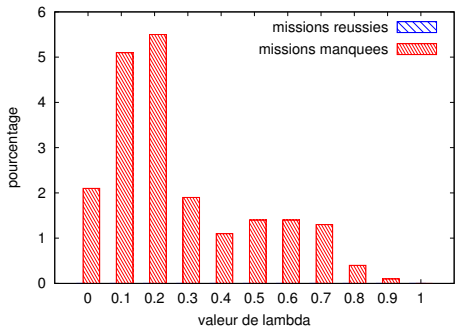
Pour corroborer ces résultats, nous avons également réalisé des simulations pour deux cas particuliers : (1) la cible recherchée se trouve dans l'environnement ; (2) la cible recherchée n'est pas présente dans l'environnement. Nous avons choisi de considérer séparément ces deux cas de figure afin de vérifier les pourcentages de missions réussies et de missions manquées

obtenus ci-dessus. Les résultats suivant ont été obtenus pour un ensemble de 1000 simulations des politiques obtenues à partir de différents λ.

Pour le cas particulier (1), dont les résultats sont illustrés dans la figure 4.22(a), l'agent est sensé réduire son incertitude et atterrir à côté de la cible recherchée dans quasiment 100% des tests. Ce fait est vérifié pour $\lambda = 0$ ce qui correspond au critère classique des POMDP. Au fur et à mesure que la valeur de λ augmente, le taux de réussite de la mission diminue, confirmant ainsi les résultats précédents, puisque l'agent passe plus de temps à observer. Plus la valeur de λ est élevée, moins la récompense attribuée à l'atterrissage est prise en compte dans le critère mixte, ce qui entraine à un surplus d'observations.



(a) Pourcentage de missions réussies et manquées pour le cas d'étude (1) où la cible recherchée se trouve dans l'environnement.



(b) Pourcentage de missions réussies et manquées pour le cas d'étude (2) où la cible recherchée n'est pas présente dans l'environnement.

FIGURE 4.22 – Pourcentages de missions réussies et manquées pour les différents cas d'étude de la mission d'atterrissage et pour les différents valeurs de λ.

Dans le cas particulier (2), l'agent autonome doit réduire son incertitude et ne pas se poser à côté d'aucune cible. Nous vérifions sur la figure 4.22(b) que plus l'agent observe moins il a tendance à manquer une mission, en particulier pour des valeurs de la pondération supérieures à 0.4.

Ceci nous permet de conclure que pour ce genre de mission, le concepteur n'a pas forcément besoin d'optimiser explicitement l'entropie dans le critère. L'action terminale d'atterrir conditionné par un niveau de préférence pour une cible en particulier, induit une réduction implicite du niveau d'incertitude de l'agent et ainsi à accomplir correctement sa mission. D'autre part, si on veut garantir un taux de missions manqués inférieur à un certain seuil ($\leq 1.5\%$), on peut, en s'appuyant sur ces résultats, modifier le rapport entre R_l et C_l.

4.3 Conclusion

Dans ce chapitre nous avons étudié deux problèmes de décision séquentielle où intervient un compromis action-perception. Nous avons montré que le critère mixte (ρPOMDP) proposé peut être un critère d'optimisation pour la mission d'exploration. Toutefois, il nous semble indispensable de mettre au point une nouvelle forme de paramétrisation de la fonction de valeur, autre que celle employant des α-vecteurs pour cette catégorie de critère non-linéaire. La paramétrisation à base d'α-vecteurs n'est pas bien adaptée à ce genre de critère non-linéaire dû à la difficulté d'approximation linéaire lors des opérations de mise à jour de la valeur des états de croyance. D'autre part, l'agrégation linéaire par l'intermédiaire d'un facteur de pondération λ entre les récompenses classiques avec l'entropie négative de l'état de croyance reste une question ouverte, étant donné que : la bonne valeur de λ dépend du

modèle de récompense/coût, et dépend du concepteur, puisque lui seul pourra définir quel point du front de Pareto doit être considéré comme optimal.

Nous avons aussi proposé une alternative au critère mixte pour la problématique de la perception active, qui consiste en un modèle pour lequel on ajouterait des buts fictifs au moyen des actions dites *report* s pour chaque état s du système, de façon à ce que l'agent décideur soit récompensé si et seulement si, il réalise l'action *report* s quand l'état s est le véritable état. Il est à noter que l'optimisation de ces actions (buts) ne dépend plus de manière directe d'une mesure de l'incertitude de l'état de croyance puisque la récompense associée peut être modélisée par les paires état-action. La politique appliquant ces actions réduira de manière implicite l'incertitude de l'état de croyance. D'autre part, il est nécessaire de rappeler qu'il y a besoin d'introduire autant d'actions *report* s qu'il y a d'états s à identifier dans le POMDP. Toutefois, avec l'aide des algorithmes de résolution basés sur la recherche heuristique nous avons vu que le nombre important d'actions n'est pas un facteur très limitant.

Ainsi, les deux approches peuvent être considérées comme comparables et équivalentes en ce qui concerne l'acquisition d'information. Nous pouvons même les considérer comme complémentaires : le comportement obtenu par l'application d'une politique résultant de l'optimisation du critère mixte nous permet de trouver le bon rapport entre R_r et C_r dans le cadre d'une modélisation POMDP classique afin de garantir le taux désiré de bonnes classifications.

En ce qui concerne le modèle de la mission d'atterrissage, nous avons vu que le modèle POMDP d'optimisation classique peut être appliqué directement et qu'il fournit des politiques sensées. Nous avons vu que l'entropie de l'état de croyance est implicitement réduite dans ce genre de mission, ce qui amène l'agent à acquérir suffisamment d'information pour bien traiter sa mission, indépendamment du modèle de cible recherché. Ceci a été démontré par le fait que, même pour un modèle d'observation moins précis, l'hélicoptère a atteint un pourcentage de réussite de mission équivalent à celui obtenu avec des modèles plus précis, moyennant quelques étapes supplémentaires de prise d'information.

D'autre part, pour ce genre de mission d'atterrissage, nous avons étudié le comportement d'une politique calculée avec le critère mixte proposé. Ceci nous permet de conclure que pour ce type de mission l'optimisation explicite de l'entropie n'est pas nécessaire. Nous avons vu que la vitesse de convergence de l'entropie de l'état de croyance ne peut être améliorée que pour des valeurs importantes de la pondération λ avec une perte non négligeable d'efficacité de la politique en ce qui concerne le pourcentage de réussite de la mission. D'autre part, l'utilisation d'un critère mixte pour ce type de mission met en évidence le fait que le rapport entre R_l et C_l doit être raffiné. Les valeurs de la pondération λ supérieures à 0.5 induisent la politique à réduire encore plus l'incertitude de l'état de croyance et par conséquent à *se tromper* moins de cible. Nous pensons que le même résultat peut être obtenu en raffinant R_l et C_l. Un *bon* "réglage" du modèle de récompense pourra ainsi garantir que le comportement de la politique respectera les différents taux exigés par l'utilisateur.

Tous ces résultats nous permettent de conclure que la structure de la fonction de récompense est déterminante par rapport à l'intérêt ou à l'utilité d'ajouter une mesure d'incertitude pour guider l'optimisation de la politique. Nous avons pu voir que les deux missions étudiées – exploration et atterrissage – ne sont pas de nature différente. Une réflexion approfondie lors de la modélisation du problème et en particulier dans le choix du modèle de récompense à utiliser est déterminante.

Il nous semble qu'un critère de type mixte peut être considéré comme une approche intuitive qui vient en complément au cadre classique des POMDP. L'utilisateur du système peut plus facilement indiquer au concepteur le taux de bonnes classifications ou le taux de missions réussies qu'il souhaite. Pour des applications robotiques, le concepteur cherche à

définir ou à garantir un certain comportement de la politique de l'agent à l'exécution. La modélisation des récompenses du POMDP classique, qui est dans la plupart de cas, faite de manière empirique peut être guidée par une comparaison avec un modèle équivalent ρPOMDP afin de définir le *bon* rapport entre les récompenses et les coûts du modèle.

Dans la suite de cette thèse nous focaliserons l'attention sur le cadre d'application de type *mission d'atterrissage*. Nous rappelons que dans la mission d'atterrissage l'agent cherche à détecter et à reconnaître un modèle particulier de voiture, parmi plusieurs modèles répertoriés dans la base de données, puis à atterrir à côté de cette cible. Cette application réaliste nous a incité à réfléchir à d'autres aspects liés à ce type de mission, comme par exemple les contraintes de sécurité liées au vol, ainsi que la question de l'optimisation en ligne de la politique, supposant que le modèle de l'environnement (nombre de zones à explorer) ne peut être connu qu'au moment du vol.

Pour apporter une réponse alternative aux questions liées à la prise en compte de ces contraintes de sûreté, nous allons proposer au chapitre 5 un nouveau modèle, appelé AC-POMDP, pour la prise en compte de contraintes sur les actions de l'agent autonome dans un cadre probabiliste et partiellement observable. Ce modèle permet de découpler les informations concernant la vérification des propriétés du système (les préconditions) de celles qui renseignent sur la présence ou l'identité des objets qui entourent l'agent autonome. Les équations d'optimisation proposées pour ce modèle permettent de ne sélectionner avec certitude que des actions réalisables lors de la mise à jour de la valeur de l'état de croyance. La structure du modèle AC-POMDP permet d'optimiser la fonction de valeur paramétrée par des α-vecteurs dont les composantes n'ont des valeurs définies que sur les états (sommets) où l'action associée à l'α-vecteur est réalisable. Contrairement au modèle POMDP classique, où la fonction de valeur est paramétrée par des α-vecteurs dont les composantes ont des valeurs définies pour tous les états (sommets), sans pouvoir tenir compte des actions non réalisables.

Chapitre 5

Prise en compte de contraintes de faisabilité d'actions dans le cadre POMDP

Ce chapitre a pour but d'étudier la prise en compte de contraintes sur la faisabilité des actions dans un cadre partiellement observable. Ces contraintes de faisabilité sont généralement modélisées par des préconditions d'action dans la planification classique. Les préconditions sont des formules booléennes qui indiquent si certaines propriétés de l'environnement sont vérifiées au moment de l'application de l'action. Dans un cadre de planification sous observabilité partielle, la vérification de ces propriétés n'est pas une tâche facile ou directe. Nous allons voir que ceci impose une acquisition supplémentaire d'information. Dans ce contexte, et contrairement à l'approche classique des POMDP, nous proposons une approche alternative, appelée AC-POMDP, qui sépare l'information concernant la vérification des propriétés du modèle (les préconditions), de celle qui renseigne sur la présence du but de mission ou sur la nature des objets de l'environnement, en deux ensembles différents d'observations. Des équations d'optimalité seront fournies pour le modèle AC-POMDP. De plus, nous montrerons qu'une approche découplée entre observations classiques et observations de faisabilité peut fournir une alternative intéressante lors du calcul de la politique, ceci à partir d'une étude théorique et expérimentale. En effet, le modèle AC-POMDP permet à un algorithme de résolution dédié de tirer profit du découplage lors de l'optimisation de la politique, tout en respectant les contraintes sur les actions.

5.1 Introduction

En planification automatique, les préconditions sont largement utilisées pour modéliser les propriétés de l'environnement indispensables pour réaliser une action. De telles préconditions sont des fonctions qui représentent le domaine de définition d'une action, i.e. l'ensemble d'états pour lesquels cette action est applicable [Ghallab *et al.*, 2004]. Une précondition sur une action est définie comme une formule booléenne, qui vérifie si une action est réalisable ou acceptable pour un état donné. Dans de nombreuses applications, la garantie du respect de ces préconditions est indispensable pour la protection de l'agent robotique contre un danger physique réel, où, par exemple, des actions considérées dangereuses doivent être évitées.

À notre connaissance, l'utilisation formelle de préconditions n'a jamais été adaptée pour des Processus Décisionnels de Markov *Partiellement Observables* (POMDPs), en dépit des besoins a priori identiques à ceux de la planification automatique des actions et pratiques

encore plus marqués : l'observabilité partielle de l'environnement de l'agent robot impose de prendre de précautions accrues quant aux conséquences à long terme en termes de sûreté des actions exécutées. La recherche sur les POMDP reste très focalisée sur les moyens d'améliorer l'efficacité des algorithmes visant à produire une politique pour le modèle POMDP standard, plutôt que sur des améliorations permettant de confronter le modèle standard aux applications. Une des nombreuses difficultés quant à l'extension du modèle POMDP standard proviennent de l'observabilité partielle : (1) la vérification des préconditions n'est pas directe quand on travaille dans des domaines partiellement observables, car l'état courant n'est pas connu précisément, étant remplacé par une distribution de probabilité sur un ensemble d'états ; (2) l'équation de Bellman doit être revue afin de n'optimiser que les actions faisables dans un état de croyance donné.

Dans les Processus Décisionnels Markoviens (MDPs), des préconditions booléennes ont été introduites de manière formelle [Younes et Littman, 2003], dans le but de produire des politiques qui contiennent seulement des actions faisables ou désirables dans chaque état atteignable. Les approches précédentes associaient une pénalité élevée aux actions indésirables, c'est-à-dire un coût proche de l'infini pour la paire (s, a), en espérant que l'optimisation du processus conduise à proscrire l'action a de la politique optimale. Si ces coûts sont *suffisamment grands*, cette astuce est équivalente à la définition de préconditions sur l'action (pour une résolution optimale).

D'autre part, les approches qui modélisent des coûts *suffisamment grands* sont aujourd'hui dépassées par des algorithmes efficaces qui évaluent directement des préconditions booléennes [Buffet et Aberdeen, 2009, Teichteil-Königsbuch *et al.*, 2010, Kolobov *et al.*, 2011], car ces algorithmes évaluent moins d'actions pour un état donné. De plus, les systèmes modernes de planification probabiliste sont basés sur des langages de haut niveau d'expressivité qui permettent d'exprimer la faisabilité des actions comme des formules logiques du premier ordre [Younes et Littman, 2003, Sanner, 2010].

Il est important de noter que, les préconditions sur les actions ne sont qu'une façon d'exprimer certaines propriétés du système, mais elles peuvent également être utilisées par les systèmes de planification pour évaluer seulement des actions dites faisables ou acceptables dans un certain état. Une discussion sur l'utilité des préconditions basées sur les différentes propriétés du modèle peut être faite ici :

Pourquoi certaines actions peuvent/doivent être considérées comme inacceptables ? Le concepteur d'un système autonome peut être intéressé par l'interdiction de certaines actions du modèle sur certains états, ce qui peut se justifier par plusieurs raisons. La faisabilité d'une action peut dépendre de :

- *la non définition* d'une action sur un état donné : par exemple si on considère le domaine *RockSample* [Smith et Simmons, 2004], permettre au robot de ramasser une pierre sur une zone où en fait il n'y a pas de pierre n'a pas de sens.
- *des limitations de la navigation* : une action peut être interdite sur un état donné en utilisant directement dans le modèle une connaissance a priori ; par exemple, dans la mission d'atterrissage modélisée dans le chapitre 3, nous savons qu'une phase de descente est nécessaire avant l'atterrissage ; nous devons donc interdire à l'agent d'atterrir si il est à une altitude de 40 mètres, car il n'est autorisé à débuter l'atterrissage automatique que s'il est à une altitude de vol inférieure à 30 mètres.
- *la sécurité* : il est impératif d'utiliser la connaissance a priori pour éviter que le robot réalise des actions qui pourraient l'endommager : par exemple, il est interdit à un robot de se déplacer vers une falaise, si il est dans une zone frontière, ou encore il est interdit à l'agent hélicoptère de changer de zone, si il est à une altitude de vol inférieure à 30 mètres, car il peut heurter un arbre ou un bâtiment à proximité.

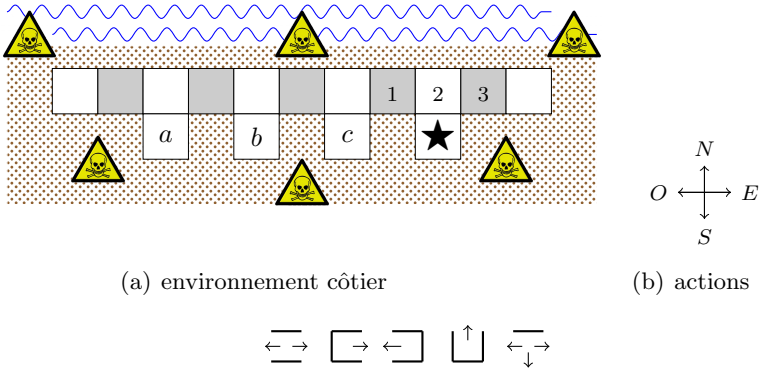
Cette discussion sur les différentes raisons et les différentes classes d'actions justifie l'utilité d'une telle modélisation. Toutefois, dans ce chapitre nous ne ferons aucune différence entre les différentes raisons d'interdiction des actions. Nous cherchons plutôt à proposer une approche générale, qui tient compte de contraintes de faisabilité ou d'acceptabilité, quelque soit l'origine ou la raison du besoin d'interdire telle ou telle action. Dans ce sens, et pour la suite, nous utiliserons à quelques nuances près les termes "faisables", "réalisables", "applicables" ou "acceptables" pour désigner les actions qu'il est souhaitable que l'agent utilise dans l'optimisation de sa stratégie, à l'exclusion des autres.

Cependant, la vérification à l'exécution des préconditions n'est immédiate que lorsque l'état du système est complètement observable. Dans des systèmes partiellement observables, la vérification de l'applicabilité d'une action n'est pas une tâche facile [Pralet *et al.*, 2010b] : l'agent ne peut pas décider quelle action est applicable parce qu'il ne peut pas connaître son état courant avec certitude.

Autrement dit, les décisions prises dans un POMDP sont définies sur l'état de croyance courant de l'agent, alors que les préconditions sont définies sur des états, c'est-à-dire le domaine d'application de l'action. Plus précisément, si l'on définit l'ensemble d'actions applicables dans l'état s_i en tant que $\mathcal{A}_{s_i}$, on voit que définir simplement des préconditions sur des états de croyance, qui sont des distributions de probabilité sur les états, sans ajout d'observations supplémentaires sur les actions faisables n'est pas suffisant car :

- si l'ensemble d'actions applicables en b est l'union des ensembles d'actions applicables en s, pour tout s tel que : $\mathcal{A}_b = \{\bigcup_s \mathcal{A}_s : \forall s | b(s) > 0\}$, des actions faisables pour certains états peuvent ne pas être applicables dans d'autres états, et ainsi on n'a pas de garantie d'éviter que l'agent applique une action inacceptable ;
- si l'ensemble d'actions applicables en b est l'intersection des ensembles d'actions applicables en s, pour tout s tel que : $\mathcal{A}_b = \{\bigcap_s \mathcal{A}_s : \forall s | b(s) > 0\}$, l'ensemble final $\mathcal{A}_b$ peut être dans certains cas un ensemble vide.

Par exemple, considérons un agent robotique autonome "garde-côtes", qui se déplace ou sol le long de falaises, comme le montre la figure 5.1(a). Cet exemple est une variation du problème *hallway* [Littman *et al.*, 1995], très répandu dans la communauté POMDP, où les murs qui entourent le robot ont été remplacés par des falaises, de sorte que l'agent risque d'en tomber. L'agent peut se retrouver dans un carré, et peut réaliser 4 actions : *nord, sud, est* et *ouest* (figure 5.1(b)). L'objectif est d'atteindre l'étoile de manière *sûre* sans tomber d'une falaise : pour les états proches d'une falaise, les actions qui peuvent mettre l'agent en péril *doivent* être absolument interdites. Les seules observations possibles renseignent sur la présence ou non du but.



(a) environnement côtier

(b) actions

(c) quelques observations possibles des configurations des falaises autour du robot.

FIGURE 5.1 – Problème POMDP d'un robot garde-côtes.

Prenons un état de croyance b uniforme sur les trois états $1, 2$ et 3 de la figure 5.1(a) : pour le cas où l'on définit l'ensemble $\mathcal{A}_b$ comme une union des ensembles des actions applicables sur les trois états, on voit bien que l'action *sud* qui amène le robot vers le but dans l'état 2, amène aussi le robot vers la falaise pour les états 1 et 3 ; et, pour le cas où l'on définit l'ensemble $\mathcal{A}_b$ par une intersection des ensembles des actions applicables sur les trois états, on voit que l'action *sud* ne fera plus partie de $\mathcal{A}_b$, ce qui empêchera l'agent d'arriver à son but.

Nous voyons intuitivement qu'il est nécessaire d'ajouter de l'information concernant la présence des falaises, comme dans l'exemple *hallway* [Littman *et al.*, 1995], où les observations renseignent sur la présence de murs (mais dans cet exemple d'une manière imprécise).

L'approche courante pouvant être mise en œuvre dans les modèles POMDP consiste à augmenter l'espace d'observations indirectement avec des actions réalisables (voir figure 5.1(c)), à pénaliser les actions inapplicables, et à rajouter une action qui permet au robot d'acquérir initialement une observation sans se déplacer. De cette façon le robot pourra mettre à jour son état de croyance, en obtenant une distribution de probabilité sur des états cohérents avec l'observation reçue, et donc en exécutant seulement des actions acceptables.

Cette approche qui consiste à encoder les actions réalisables dans les observations du modèle est suffisante pour garantir les préconditions. Toutefois, la sémantique associée aux préconditions est perdue dans ces observations. Cette sémantique est différente puisqu'elle exprime *quelles sont les actions applicables* dans l'état caché du système. Supposons que l'agent connaisse parfaitement l'état courant du système, il peut recevoir n'importe quelle observation possible, mais toutes ces observations encodent le même ensemble d'actions réalisables. Dans le contexte de notre exemple, si l'agent connaît sa position, il recevra seulement une observation du modèle (en supposant des observations déterministes). De cette façon, on peut formellement voir que l'ensemble d'observations est structuré et de la forme : $\Omega = \mathcal{O} \times \Theta$, où, dans l'état caché du système, l'agent reçoit des observations *standard* et des observations de *faisabilité* de manière déterministe.

Nous pensons que l'ensemble d'actions réalisables doit être construit plus intelligemment, en ajoutant des observations qui sont *a priori* très similaires à l'observation standard, mais avec une *sémantique différente* : ces observations informent l'agent sur l'ensemble courant des actions réalisables dans son état de croyance courant.

Pour illustrer, une fois de plus, le fait que le modèle d'observation a une forme structurée, reprenons l'exemple 5.1. Imagions que, à la place de falaises, il y ait des murs. Dans un contexte robotique, l'observation du but serait déduite à partir d'un traitement d'image, avec un bruit probabiliste qui représente les erreurs d'interprétation (même cas que pour notre modèle de détection et reconnaissance de cibles du Chapitre 3). D'autre part, la configuration des murs autour du robot peut être détectée par un laser circulaire, qui peut être considéré suffisamment précis. Donc, la forme structurée de l'ensemble d'observations $\Omega = \mathcal{O} \times \Theta$ en découle directement, où $\mathcal{O}$ représente le résultat probabiliste du traitement d'information, et Θ les observations déterministes de la détection des murs.

Comme, pour tenir compte des préconditions, il est indispensable d'ajouter de l'information, nous proposons une extension du modèle traditionnel des POMDP, dans lequel l'ensemble des observations est un produit cartésien des observations probabilistes *standard* et des observations déterministes de *faisabilité*. Ce modèle formalise un ensemble structuré d'observations, ainsi que la nouvelle fonction d'observation, et requiert une modification significative du modèle de décision, pour lequel nous fournissons un nouveau schéma d'optimisation. Ce nouveau schéma d'optimisation évalue seulement des actions qui sont applicables dans l'état caché du système. Nous appelons l'extension proposée AC-POMDP (*Action-Constrained POMDP*). Nous montrerons aussi que le modèle AC-POMDP peut être transformé en un POMDP traditionnel équivalent avec un modèle d'observation modifié et avec

de grandes pénalités associées aux paires état-action infaisables, ce qui justifie l'utilisation de cette astuce classique pour mimer les préconditions, bien que peu efficace en pratique. En ce sens, pour comparer les politiques calculées pour les AC-POMDP avec un modèle POMDP classique et équivalent nous définirons une transformation entre AC-POMDP et POMDP. Dans certains domaines l'information concernant la faisabilité d'une action est encodée de manière indirecte dans l'espace d'observation ; par exemple, dans le domaine *hallway* déjà mentionné, la configuration des murs fait partie du modèle d'observation.

Le but de l'étude effectué dans ce chapitre est d'extraire la sémantique spécifique des préconditions pour les domaines où les préconditions sont nécessaires, et d'utiliser cette information afin d'évaluer seulement des actions faisables tout en diminuant le temps consacré au calcul de la politique. Notre contribution peut être vue comme une séparation implicite de l'étape *standard* d'observation de l'état de celle d'observation des actions réalisables.

Le chapitre est organisé de la façon suivante. Tout d'abord nous discuterons de quelques travaux connexes au sujet ici traité. Ensuite, nous définirons la contrainte de faisabilité d'une action et la relation de faisabilité d'un ensemble d'actions. Ces définitions nous seront utiles lors de la formalisation du modèle étendu que nous proposons.

En second, nous formalisons ce modèle étendu de POMDP, que nous avons appelé AC-POMDP (*Action-Constrained POMDP*). Nous verrons que la relation de faisabilité nous permettra de mettre à jour l'état de croyance en fonction de l'ensemble courant d'actions réalisables. Ce qui résultera en un état de croyance pour lequel les composantes $b(s)$ non nulles correspondront aux états s qui ont le même ensemble d'actions réalisables. Ainsi, les α-vecteurs représentant la fonction de valeur seront définis sur des sous-espaces d'états différents.

Ensuite, nous présentons une transformation entre les AC-POMDP et les POMDP, qui est nécessaire pour comparer les politiques calculées par les deux approches. Cette transformation suggère que le modèle POMDP contient le modèle AC-POMDP. La conclusion est que nous pouvons transformer n'importe quel AC-POMDP en un POMDP équivalent avec des actions et des fonctions d'observation et de récompenses différentes.

Puis, nous présenterons un algorithme de type *point-based* appelé PCVI (basé sur PBVI), qui profite de la structure spécifique des AC-POMDP afin d'évaluer seulement les actions faisables. Nous prouverons que le temps d'optimisation du modèle AC-POMDP est meilleur, ce qui est mis en évidence dans nos résultats expérimentaux. Nous montrerons aussi que, pour une version relaxée du schéma d'optimisation, PCVI permet de calculer une politique en ne tenant compte que de l'ensemble probabiliste *standard* d'observations $\mathcal{O}$, en fournissant une borne inférieure sur la valeur et une politique sous-optimale avec une complexité et un temps de calcul exponentiellement plus petit. Finalement, nous présenterons et discuterons des résultats comparatifs qui démontrent l'intérêt de modéliser formellement les préconditions dans un cadre POMDP de façon à diminuer le temps de calcul pour des problèmes où certaines actions doivent être écartées avec certitude de la politique optimisée.

5.2 Préliminaires

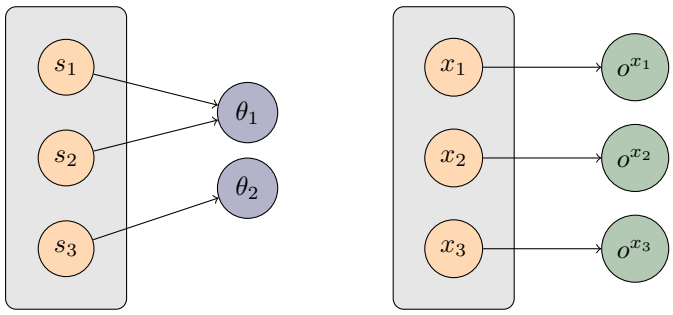
5.2.1 Travaux connexes

Des travaux récents dans le domaine de la planification non-déterministe sous observabilité partielle ont proposé de traiter de manière explicite des *relations de faisabilité*, dans le contexte de programmation sous contraintes [Pralet *et al.*, 2010b]. Ces relations représentent la faisabilité état-action par des contraintes, c'est-à-dire toutes les combinaisons implicites de paires état-action où l'action est considérée réalisable. Cette approche est limitée à des domaines non-déterministes et non-probabilistes, et donc ne s'applique pas à des modèle

probabilistes comme les POMDP. De plus, l'algorithme proposé ne suppose pas un ensemble d'observation structuré en observations *standard* et *de faisabilité*, de cette façon il ne tire pas profit de cette structure spécifique pour accélérer le calcul.

En robotique, des chercheurs ont étudié une classe particulière de POMDP, appélée *Mixed Observability MDP* - MOMDP (présenté dans le chapitre 2), dans laquelle un sous-ensemble de variables d'état est considéré comme complètement observable [Ong *et al.*, 2009, López *et al.*, 2010]. Le modèle MOMDP divise l'espace d'états S et l'espace d'observation Ω en deux parties, l'une observable, et l'autre partiellement observable : (x, y) spécifie l'espace d'état complet, x la variable observable et y celle partiellement observable, avec $|S| = |\mathcal{X}| \times |\mathcal{Y}|$, où $\mathcal{X}$ représente l'espace avec toutes les valeurs possibles de la variable x (resp. $\mathcal{Y}$ pour y), et $|\Omega| = |O^x| \times |O^y|$, avec $X = O^x$. Cette approche propose d'explorer cette structure spécifique des observations et des états de façon à réduire la dimension de l'espace de l'état de croyance (sous-espace Y), ce qui conduit à des gains significatifs au niveau du temps de calcul de la politique.

Cependant, dans notre approche, la sémantique des variables d'observation est complètement différente : nous supposons $\Omega = \mathcal{O} \times \Theta$, avec $\Theta \subseteq 2^A$ représentant les sous-ensembles d'actions réalisables d'un ensemble d'actions A. De plus, notre approche ne suppose pas d'associer une observation par état observable (*mapping* $1 \rightarrow 1$) tel que $X = O^x$. La fonction d'observation de l'ensemble Θ est une fonction surjective, c'est à dire, qu'une observation $\theta \in \Theta$ peut être la même pour différents états, par exemple, un ensemble d'actions réalisables dans un certain état s_1 peut être aussi l'ensemble d'actions réalisables d'un certain état s_2. Pour illustrer cette différence fondamentale, nous montrons sur la figure 5.2 la différence sémantique entre les deux modèles. Ainsi, nous pouvons affirmer que le modèle AC-POMDP se situe entre le modèle POMDP et le modèle MOMDP. D'ailleurs nous pensons que ces deux modèles peuvent être couplés afin d'explorer les structures spécifiques de chaque problème ; nous discuterons de cette piste de recherche dans les perspectives de cette thèse.



(a) AC-POMDP (b) MOMDP

FIGURE 5.2 – Différence entre les fonctions d'observation des modèles AC-POMDP et MOMDP.

D'autre part, dans notre approche, nous cherchons à maximiser le critère d'optimisation seulement sur l'ensemble d'actions applicables Θ (voir la section 5.3.2), contrairement au MOMDP qui maximisent sur toutes les actions du modèle. Le modèle et les équations des MOMDP et des AC-POMDP sont totalement différents. Toutefois, les deux approches exploitent la structure spécifique de l'espace d'observation pour accélérer les calculs.

Dans la suite, nous présentons des définitions qui nous seront utiles.

5.2.2 Contrainte de faisabilité d'une action

Une précondition est une formule à valeur booléenne qui est vraie (sûrement) si et seulement si une action est applicable dans un état donné. Nous notons $\mathcal{A}_s$ l'ensemble des actions réalisables dans un état s, tel que $\mathcal{A}_s \subseteq A$.

Définition 5.2.1 (Relation de faisabilité pour une action) *Une précondition basée sur un état est définie par une relation de faisabilité* $\mathbb{F}$ *qui indique si pour un état* s*, une action* a *est applicable ou non :* $\mathbb{F}(a,s) = \mathbf{1}_{a\in\mathcal{A}_s}$*, où* $\mathbf{1}_{cond}$ *est la fonction indicatrice telle que :* $\mathbf{1}_{cond}$ *vaut* 1 *si cond est vrai, ou* 0 *sinon.* $\mathbb{F}(a,s)$ *peut aussi être vue comme la probabilité* 1 *ou* 0 *de l'applicabilité d'une action* a *sachant l'état* s*, c'est-à-dire,* $\mathbb{F}(a,s) = Pr(a_t = a \in \mathcal{A}_s | s_t = s)$.

Le modèle POMDP doit être étendu de manière à prendre en compte des contraintes de sûreté dans *l'optimisation* et dans *l'exécution* de la politique. Comme nous ne connaissons pas à l'avance l'ensemble des actions réalisables qui sera reçu par l'agent en tant qu'observation $\theta \in \Theta$, nous devons tenir compte de *tous les ensembles cohérents d'actions réalisables possibles* $\{\theta_1, \cdots, \theta_n\}$, *indépendamment de l'état de croyance de l'agent*, avec dans le pire cas : $n = 2^{|A|} - 1$ combinaisons d'actions différentes, où $|A|$ est le nombre total d'actions du modèle. La propriété de cohérence vient du fait que, certains ensembles d'actions pouvant être générés par combinaisons ne seront cohérents avec aucun $\mathcal{A}_s$; pour cette raison on ne s'intéressera donc qu'aux ensembles d'actions cohérents par rapport aux états. Formellement, nous définissons un sous-ensemble $S_k \subseteq S$, tel que :

$$S_k := \{\forall s_i, s_j \in S^2, \mathcal{A}_{s_i} = \mathcal{A}_{s_j}\}$$

Ceci veut dire que le sous-ensemble S_k représente l'ensemble d'états qui ont le même ensemble d'actions réalisables. Ainsi, on peut définir un ensemble d'actions $\theta_k \in \Theta$, tel que :

$$\theta_k = \{\mathcal{A}_s | \forall s \in S_k\},$$

qui est l'ensemble d'actions applicables cohérent avec le sous-ensemble S_k. Pour notre exemple du robot garde-côtes (Fig.5.1(a)), les combinaisons cohérentes d'actions possibles, qui formeront Θ, sont : $\{est\}$, $\{est, ouest\}$, $\{est, ouest, sud\}$, $\{nord\}$, et $\{ouest\}$.

Définition 5.2.2 (Relation sur la faisabilité d'un ensemble d'actions.) *Nous définissons la fonction indicatrice jointe d'un ensemble d'actions* $\theta \in \Theta$ *tel que* $\theta \subseteq A$ *pour un état* s*, qui vaut* 1 *si et seulement si toutes les actions de l'ensemble* θ *sont réalisables en* s *:*

$$\mathbb{I}(\theta, s) = \prod_{a_i \in \theta} \mathbb{F}(a_i, s) \prod_{a_j \notin \theta} (1 - \mathbb{F}(a_j, s)) \tag{5.1}$$

Nous avons directement $\mathbb{I}(\mathcal{A}_s, s) = 1$. De plus, il est intéressant de remarquer que $\mathbb{I}(\theta, s) = Pr(\mathcal{A}_s = \theta | s)$ est la probabilité 1 ou 0 que l'ensemble des actions cohérent avec s, et conditionné sur l'état s, soit égal à l'ensemble des actions réalisables $\mathcal{A}_s$.

Par suite, nous formalisons le modèle étendu POMDP que nous proposons.

5.3 AC-POMDP

Formellement, un AC-POMDP (*Action Constrained POMDP*) est défini par un n-uplet $\langle S, (\mathcal{A}_s)_{s\in S}, \Omega, T, O, \mathbb{F}, \mathbb{I}, R, b_0, \Theta_0\rangle$, où :

- S est l'ensemble des états ;
- $(\mathcal{A}_s)_{s\in S}$ est l'ensemble des ensembles d'actions applicables pour chaque état, avec $\mathcal{A}_s$ l'ensemble d'actions applicables pour un état donné s ;
- $\Omega = \mathcal{O} \times \Theta$ est l'ensemble d'observations, tel que $\Theta \subseteq 2^A$; les observations dans $\mathcal{O}$ et dans Θ sont indépendantes sachant chaque paire état-action ;
- $T : \mathcal{S} \times A \times S \to [0, 1]$ est la fonction de transition, telle que :

$$T(s, a, s') = p(s_{t+1} = s' | s_t = s, a_t = a);$$

- $\mathbb{F} : A \times S \to \{0, 1\}$ est la relation de faisabilité d'une action :
  $\mathbb{F}(a, s) = \begin{cases} 1 & \text{si } a \in \mathcal{A}_s \\ 0 & \text{autrement} \end{cases}$
- $O : \mathcal{O} \times A \times S \to [0, 1]$ est la fonction d'observation, telle que :

$$O(o, a, s') = p(o_{t+1} = o | s_{t+1} = s', a_t = a);$$

- $\mathbb{I} : \Theta \times S \to \{0, 1\}$ est la relation de faisabilité d'un ensemble d'actions :
  $\mathbb{I}(\theta, s') = p(\theta_{t+1} = \theta \mid s_{t+1} = s') = \begin{cases} 1 & \text{si } \theta = \mathcal{A}_{s'} \\ 0 & \text{autrement} \end{cases}$
- $R : S \times A \to \mathbb{R}$ est la fonction de récompense $r(s, a)$ associée à la paire état-action ;
- b_0 est l'état de croyance initial ;
- Θ_0 est l'ensemble initial d'actions applicables, observé lors de l'exécution avant l'application de la première action.

Il y a quatre différences par rapport au modèle POMDP.

1. la relation de faisabilité d'une *action* dans un *état* est explicitement décrite ;
2. l'ensemble d'observations est un produit cartésien ;
3. la fonction d'observation est définie pour seulement la partie gauche du produit cartésien ;
4. il y a une observation initiale qui renseigne sur l'ensemble d'actions réalisables, similaire à une approche existante dans le domaine non-déterministe [Pralet *et al.*, 2010b], qui est requise pour appliquer la première action de manière sûre.

Dans la figure 5.3 nous illustrons l'AC-POMDP comme un processus stochastique contrôlé. L'action a_t exécutée à l'instant t est forcée d'appartenir à l'ensemble d'actions réalisables observées θ_t. L'observation suivante θ_{t+1} est égale à l'ensemble d'actions réalisables $A_{s_{t+1}}$ applicable sur l'état caché s_{t+1}, qui est le résultat stochastique de l'application de a_t sur s_t.

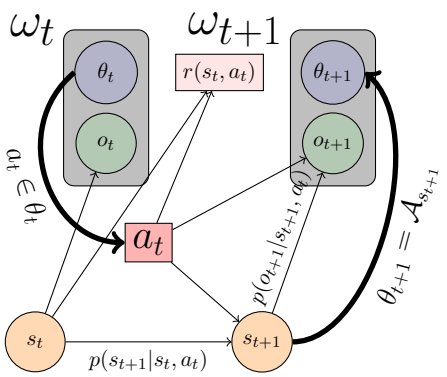


FIGURE 5.3 – Diagramme d'influence dynamique pour un AC-POMDP

Les politiques AC-POMDP sont-elles sûres ?

Contrairement au cas des POMDP, les politiques AC-POMDP sont forcées d'exécuter seulement des actions qui sont applicables dans l'état caché du système. Étant donné un historique $h_t = (\omega_0 = (o_0, \theta_0), ..., \omega_t = (o_t, \theta_t))$ d'observations à l'instant t, nous définissons l'ensemble faisable de politiques comme :

$$\Pi^{h_t} = \{\pi \in \mathcal{A}^{\Delta} : \forall\, 0 \leqslant i \leqslant t, \pi(b_i((o_0, \theta_0), \cdots, (o_i, \theta_i)) \in \theta_i\} \tag{5.2}$$

où, $b_i((o_0, \theta_0), \cdots, (o_i, \theta_i))$ est l'état de croyance résultant de la politique π en recevant les observations $((o_0, \theta_0), \cdots, (o_i, \theta_i))$. Donc, résoudre un AC-POMDP consiste à trouver une politique déterministe stationnaire π^*, telle que, pour tout $b \in \Delta$ et $\theta \in \Theta_0$:

$$\pi^*(b, \theta) \in \underset{\pi \in \Pi^{h_\infty}}{\arg\max}\, E\left[\sum_{t=0}^{+\infty} \gamma^t r(b_t, \pi(b_t)) \middle| b_0 = b, \Theta_0 = \theta\right] \tag{5.3}$$

L'idée de découpler les différentes informations spécifiques à la planification et au contrôle d'exécution afin de garantir la sûreté d'exécution des actions n'est pas nouvelle dans le domaine robotique. Par exemple, dans [Ingrand *et al.*, 2007], une approche similaire est utilisée au travers de fonctionnalités spécifiques et découplées de la planification, comme dans le cas particulier du contrôle d'exécution. Plus précisément, un composant spécifique de l'architecture robotique peut être en charge du suivi de l'état et peut inhiber la réalisation d'une action indésirable en la remplacent par une action naïve, qui n'appartient pas forcément au modèle. Ceci, dans certains cas, arrête l'exécution de la politique et conduit le superviseur à lancer une phase de replanification. Le but est d'éviter des dommages que pourrait subir le robot. Généralement, l'exécution de la politique doit être contrôlée pour garantir que le plan sera réalisé en toute sécurité. Pour cela, les systèmes autonomes sont généralement munis d'un composant responsable du contrôle d'exécution qui vérifie l'applicabilité des actions ainsi que l'évolution physique de l'engin. L'approche AC-POMDP vise ainsi à optimiser la politique de manière à tenir compte de ces contraintes en évitant des replanifications.

Dans la suite nous définirons la mise à jour de l'état de croyance dans le cadre du modèle AC-POMDP.

5.3.1 Mise à jour de l'état de croyance

Comme pour les POMDP, nous devrons calculer le nouvel état de croyance de l'agent, que nous noterons $b_a^{o,\theta}$, après l'exécution de a dans l'état de croyance b et l'observation de (o, θ).

Théorème 5.3.1 *Soit b l'état de croyance à un instant t donné, a l'action appliquée par l'agent à cet instant, et (o, θ) les observations immédiatement reçues. L'état de croyance suivant, pour tout état s' suivant possible, est :*

$$b_a^{(o,\theta)}(s') = \frac{\mathbb{I}(\theta, s') b_a^o(s')}{\sum_{s'' \in \mathcal{S}} \mathbb{I}(\theta, s'') b_a^o(s'')} \tag{5.4}$$

avec :

$$b_a^o(s') = \frac{p(o \mid s') \sum_{s \in S} p(s' \mid s, a) b(s)}{\sum_{s' \in S} p(o \mid s') \sum_{s \in S} p(s' \mid s, a) b(s)} \tag{5.5}$$

Preuve. Quel que soit l'état suivant possible s' et l'action a, nous avons :

$$\begin{aligned} b_a^{(o,\theta)}(s') &= Pr(s_{t+1}=s' | o_{t+1}=o, \theta_{t+1}=\theta, b_t=b, a_t=a) \\ &= \frac{Pr(s_{t+1}=s', o_{t+1}=o, \theta_{t+1}=\theta | b_t=b, a_t=a)}{Pr(o_{t+1}=o, \theta_{t+1}=\theta | b_t=b, a_t=a)} \\ &= \frac{Pr(\theta_{t+1}=\theta | s_{t+1}=s', b_t=b, a_t=a) Pr(o_{t+1}=o | s_{t+1}=s', b_t=b, a_t=a) Pr(s_{t+1}=s' | b_t=b, a_t=a)}{\sum_{s''\in S} Pr(\theta_{t+1}=\theta | s_{t+1}=s'', b_t=b, a_t=a) Pr(o_{t+1}=o | s_{t+1}=s'', b_t=b, a_t=a) Pr(s_{t+1}=s'' | b_t=b, a_t=a)} \end{aligned} \tag{5.6}$$

La dernière factorisation réalisée au dénominateur vient du fait que les observations o_{t+1} et θ_{t+1} sont indépendantes sachant l'état s' et l'action a. Ainsi, en utilisant cette propriété dans le dénominateur nous pouvons écrire :

$$Pr(o_{t+1}=o, \theta_{t+1}=\theta \mid b_t=b, a_t=a)$$

comme le produit :

$$\sum_{s''\in S} Pr(\theta_{t+1}=\theta | s_{t+1}=s'', b_t=b, a_t=a) Pr(o_{t+1}=o | s_{t+1}=s'', b_t=b, a_t=a) Pr(s_{t+1}=s'' | b_t=b, a_t=a)$$

Pour la suite, nous multiplions le numérateur et le dénominateur par le facteur $\frac{1}{Pr(o_{t+1}=o|b_t=b,a_t=a)}$, en faisant l'hypothèse classique (comme dans les POMDP) que le dénominateur de ce facteur est différent de zéro. On retrouve donc :

$$b_a^{(o,\theta)}(s') = \frac{Pr(\theta_{t+1}=\theta \mid s_{t+1}=s', b_t=b, a_t=a) \frac{Pr(o_{t+1}=o|s_{t+1}=s', b_t=b, a_t=a) Pr(s_{t+1}=s'|b_t=b, a_t=a)}{Pr(o_{t+1}=o|b_t=b, a_t=a)}}{\sum_{s''\in S} Pr(\theta_{t+1}=\theta \mid s_{t+1}=s', b_t=b, a_t=a) \frac{Pr(o_{t+1}=o|s_{t+1}=s'', b_t=b, a_t=a) Pr(s_{t+1}=s''|b_t=b, a_t=a)}{Pr(o_{t+1}=o|b_t=b, a_t=a)}} \tag{5.7}$$

Ainsi, on fait apparaitre $b_a^o(s')$ dans le numérateur et dans le dénominateur. De plus, on sait que $Pr(\theta_{t+1}=\theta \mid s_{t+1}=s'', b_t=b, a_t=a) = \mathbb{I}(\theta, s'')$, puisque la probabilité d'observer θ en étant en s'' correspond à la relation de faisabilité $\mathbb{I}(\theta, s'')$. On retrouve donc :

$$\begin{aligned} b_a^{o,\theta}(s') &= \frac{Pr(\theta_{t+1}=\theta \mid s_{t+1}=s') b_a^o(s')}{\sum_{s''\in\mathcal{S}} Pr(\theta_{t+1}=\theta \mid s_{t+1}=s'') b_a^o(s'')} \\ &= \frac{\mathbb{I}(\theta, s') b_a^o(s')}{\sum_{s''\in\mathcal{S}} \mathbb{I}(\theta, s'') b_a^o(s'')} \end{aligned} \tag{5.8}$$

Finalement, on a bien l'expression finale de $b_a^{(o,\theta)}(s')$. ■

L'équation 5.4 du théorème 5.3.1 montre que l'état de croyance d'un AC-POMDP se comporte comme si l'agent recevait les observations *o puis* θ, et mettait à jour son état de croyance entre les deux observations avec $b_a^o(s')$. Autrement dit, la mise à jour de l'état de croyance est équivalente à observer l'environnement, comme dans le POMDP, à mettre à jour l'état de croyance, à ensuite observer θ (ensemble d'actions réalisables), à encore mettre à jour l'état de croyance pour finalement appliquer la prochaine action.

Sur la figure 5.4, nous présentons un schéma avec les évolutions probables de l'état de croyance b suite aux actions exécutées et aux observations o et θ reçues. Les rectangles oranges indiquent les états de croyance intermédiaires.

Une autre propriété qui découle de l'équation 5.4 est que, à tout instant t, tous les états s' pour lesquels l'état de croyance courant est non nul ont le même ensemble d'actions réalisables. Cette propriété montre qu'exécuter des politiques dans le modèle AC-POMDP est cohérent avec le cadre proposé, c'est-à-dire qu'à aucun moment la politique optimisée n'exécutera d'actions infaisables dans les états cachés du système.

Formellement, nous définissons l'ensemble d'états supports d'un état de croyance par : $\sigma(b_a^{(o,\theta)}) = \{s' \in \mathcal{S} : b_a^{(o,\theta)}(s') > 0\}$, ce qui sera utilisé dans le théorème suivant.

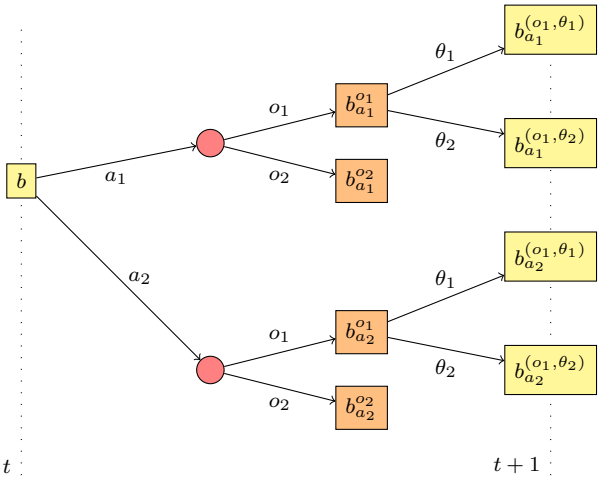


FIGURE 5.4 – Schéma de l'évolution stochastique de l'état de croyance.

Théorème 5.3.2 *Soit $b_a^{(o,\theta)}$ un état de croyance pour un instant t donné. Quelles que soient deux états s'_1 et s'_2 dans $\sigma(b_a^{(o,\theta)})$, nous avons : $\mathcal{A}_{s'_1} = \mathcal{A}_{s'_2}$.*

Preuve. (Démonstration par l'absurde). Soient a l'action exécutée à un instant t quelconque, o et θ les observations reçues à l'instant suivant $t+1$, et s'_1 et s'_2 deux états suivants possibles, tels que $s'_1, s'_2 \in \sigma(b_a^{(o,\theta)})$.

Supposons que $\mathcal{A}_{s'_1} \neq \mathcal{A}_{s'_2}$, donc par définition $\mathbb{I}(\theta, s'_1) \neq \mathbb{I}(\theta, s'_2)$. Si $\mathbb{I}(\theta, s'_1) = 1$ alors $\mathbb{I}(\theta, s'_2) = 0$, et par l'équation 5.4 on voit que $b_a^{(o,\theta)}(s'_2) = 0$. Ce qui contredit le fait que $s'_1, s'_2 \in \sigma(b_a^{(o,\theta)})$. Donc, pour que $s'_1, s'_2 \in \sigma(b_a^{(o,\theta)})$, on a nécessairement $\mathcal{A}_{s'_1} = \mathcal{A}_{s'_2}$. ■

La conséquence de ce théorème est très importante à l'exécution de la politique, ainsi que *durant son optimisation* : puisque tous les états support de l'état de croyance b ont le même ensemble d'actions réalisables, nous pouvons ainsi déduire l'ensemble d'actions réalisables de b, noté $\mathcal{A}_b$, sans aucune ambiguïté par :

$$\mathcal{A}_b := \mathcal{A}_s, \forall s \in \sigma(b).$$

De cette façon, les politiques AC-POMDP définies dans l'équation 5.3 peuvent être fonction de b, avec un abus de notation, tel que : $\pi(b) = \pi(b, \mathcal{A}_b)$. Ceci signifie que les politiques AC-POMDP et les politiques POMDP sont définies sur le même espace, ce qui nous permet de comparer les politiques optimales des AC-POMDP avec les politiques optimales des POMDP équivalents. La transformation vers un POMDP équivalent sera présentée dans la section 5.4.

Dans la suite nous allons formaliser l'équation d'optimalité des AC-POMDP, afin de pouvoir extraire de cette équation des moyens de résolution efficaces par programmation dynamique.

5.3.2 Équation d'optimalité pour le modèle AC-POMDP

Nous cherchons maintenant à définir l'équation d'optimalité des AC-POMDP. Notons que les politiques des AC-POMDP peuvent être optimisées sur des ensembles spécifiques d'actions ($\mathcal{A}_b := \mathcal{A}_s, \forall s \in \sigma(b)$). De cette façon l'opérateur *max* sera utilisé pour un nombre (généralement) réduit d'actions, c'est-à-dire $\mathcal{A}_b \subset A$, contrairement à l'approche classique des POMDP où toutes les actions du modèle doivent être évaluées. Ce théorème peut sembler évident à première vue, mais il repose sur le fait primordial que l'ensemble d'actions faisables peut être extrait d'un état de croyance donné sans ambiguïté.

Théorème 5.3.3 *L'équation d'optimalité d'un AC-POMDP est telle que :*

$$V^*(b) = \max_{a \in \mathcal{A}_b} \left[r(b,a) + \gamma \sum_{\substack{o \in \mathcal{O} \\ \theta \in \Theta}} p(o, \theta \mid a, b) V^* \left(b_a^{(o,\theta)} \right) \right] \tag{5.9}$$

avec $b_a^{(o,\theta)}$ *donné par l'équation 5.4, et :*

$$p(o, \theta \mid a, b) = \sum_{s' \in \mathcal{S}} \mathbb{I}(\theta, s') O(o, a, s') \sum_{s \in \mathcal{S}} T(s, a, s') b(s) \tag{5.10}$$

Preuve. Suivant le théorème 5.3.2, l'ensemble d'actions réalisables observé avant le calcul de l'état de croyance b, c'est-à-dire, θ_{t-1}, peut être déduit de b sans aucune ambiguïté : $\theta_{t-1} = \mathcal{A}_b = \mathcal{A}_s$, quel que soit $s \in \sigma(b)$. Cette déduction est nécessaire pour l'optimisation, puisque au moment de l'optimisation on ne connait pas à l'avance quel sera le θ observé. De plus, nous savons que les politiques optimales sont forcées d'appliquer uniquement des actions faisables : suivant l'équation 5.3, les actions gloutonnes candidates pour maximiser la fonction de valeur doivent être choisies parmi $\mathcal{A}_b$. Donc l'équation 5.9 peut être obtenue de la même façon que pour les POMDP, en considérant (o, θ) comme une observation jointe. Ainsi, l'équation 5.10 peut être obtenue en faisant dépendre θ et o de s' et a, comme pour la mise à jour de l'état de croyance, car ces deux observations sont indépendantes sachant s' et a :

$$\begin{aligned} p(o, \theta \mid a, b) &= \textstyle\sum_{s' \in \mathcal{S}} Pr(\theta \mid s', a) Pr(o \mid s', a) Pr(s' \mid a, b) \\ &= \textstyle\sum_{s' \in \mathcal{S}} \mathbb{I}(\theta, s') O(o, a, s') \sum_{s \in \mathcal{S}} T(s, a, s') b(s) \end{aligned}$$

■

Ce théorème nous permet d'utiliser la programmation dynamique pour le calcul des politiques optimales pour les AC-POMDP. Contrairement à l'artifice utilisant des pénalités "infinies", aucune pénalité artificielle et empirique n'est requise dans notre cas. L'opérateur *max* opère aussi sur un nombre réduit d'actions.

Pour comparer les politiques calculées pour les AC-POMDP avec un modèle POMDP classique et équivalent, nous avons besoin de définir la transformation entre AC-POMDP et POMDP. Dans certains domaines l'information concernant la faisabilité d'une action est encodée de manière indirecte dans l'espace d'observation ; par exemple, dans le domaine *hallway*, la configuration des murs fait partie du modèle d'observation. Ceci suggère que le modèle POMDP contient le modèle AC-POMDP. La conclusion est que nous pouvons transformer n'importe quel AC-POMDP en un POMDP équivalent, avec des actions et des fonctions d'observation et de récompense différentes.

5.4 Équivalence des politiques AC-POMDP et POMDP

Soit $\mathcal{M} = \langle S, (\mathcal{A}_s)_{s \in S}, \Omega = \mathcal{O} \times \Theta, T, O, \mathbb{I}, R, b_0, \Theta_0 \rangle$ un AC-POMDP donné. Considérons le POMDP $\widetilde{\mathcal{M}} = \langle S, \widetilde{A}, \Omega = \mathcal{O} \times \Theta, T, \widetilde{O}, \widetilde{R}, b_0, \Theta_0 \rangle = \Psi(\mathcal{M})$, où :

- $\widetilde{A} = \bigcup_{s \in S} \mathcal{A}_s$;
- $\widetilde{O} : (\mathcal{O} \times \Theta) \times \widetilde{A} \times S \to [0; 1]$ la fonction d'observation résultat de l'agrégation, telle que $\widetilde{O}((o, \theta), a, s') = O(o, a, s') \mathbb{I}(\theta, s')$;
- $\widetilde{R} : S \times \widetilde{A} \to \mathbb{R}$ la fonction de récompense modifiée, telle que $\widetilde{r}(s, a) = r(s, a)$ si $a \in A_s$, et $\widetilde{r}(s, a) = -\infty$ sinon.

A partir des équations d'optimalité des POMDP et des AC-POMDP nous pouvons démontrer qu'une politique optimale pour un POMDP $\widetilde{\mathcal{M}}$ est optimale pour l'AC-POMDP original $\mathcal{M}$. A cette fin, examinons le théorème qui suit.

Théorème 5.4.1 *Soit $\mathcal{M}$ un modèle AC-POMDP et $\widetilde{\mathcal{M}} = \Psi(\mathcal{M})$ son POMDP transformé. Toute politique optimale pour $\widetilde{\mathcal{M}}$ est aussi optimale pour $\mathcal{M}$.*

Preuve. Soit π^* une politique optimale pour $\mathcal{M}$. Selon l'équation de Bellman (équation 2.9 du chapitre 2), nous savons que :

$$\pi^*(b) \in \underset{a \in \widetilde{\mathcal{A}}}{\operatorname{argmax}} \left\{ \tilde{r}(b,a) + \gamma \sum_{\substack{o \in \mathcal{O} \\ \theta \in \Theta}} \tilde{p}((o,\theta) \mid a, b) V^{\pi^*}(b_a^{(o,\theta)}) \right\}$$

avec $\tilde{r}(b,a) = \sum_{s \in S} b(s) \widetilde{r}(s,a)$ et :

$$\begin{aligned} \tilde{p}((o,\theta) \mid a, b) &= \sum_{s' \in \mathcal{S}} \widetilde{O}((o,\theta), a, s') \sum_{s \in \mathcal{S}} T(s,a,s') b(s) \\ &= \sum_{s' \in \mathcal{S}} O(o,a,s') \mathbb{I}(\theta, s') \sum_{s \in \mathcal{S}} T(s,a,s') b(s) \\ &= p(o, \theta \mid a, b) \end{aligned}$$

comme défini antérieurement dans l'équation 5.10.

De plus, pour $a \notin \mathcal{A}_b, \tilde{r}(b,a) = -\infty$; de cette façon, suivant la définition de $\mathcal{A}_b$, il existe un état $s \in \sigma(b)$, tel que $b(s) > 0$, pour lequel $a \notin \mathcal{A}_s$, et donc $\widetilde{r}(s,a) = -\infty$. Par conséquent, la valeur maximale résultante de l'opérateur *max* est nécessairement obtenue pour une action $a^* \in \mathcal{A}_b$. Et, finalement, pour tout $a \in \mathcal{A}_b$ et pour tout état $s \in \sigma(b)$, $\widetilde{r}(s,a) = r(s,a)$, et $\tilde{r}(b,a) = r(b,a)$, nous avons :

$$\pi^*(b) \in \underset{a \in \mathcal{A}_b}{\operatorname{argmax}} \left\{ r(b,a) + \gamma \sum_{\substack{o \in \mathcal{O} \\ \theta \in \Theta}} p((o,\theta) \mid a, b) V^{\pi^*}(b_a^{(o,\theta)}) \right\}$$

ce qui implique que V^{π^*} est solution de l'équation d'optimalité des AC-POMDPs selon l'équation 5.9. ■

Cette équivalence repose sur l'hypothèse communément admise que l'on sait choisir une récompense égale à $-\infty$, tout du moins une valeur qui correspond à la borne inférieure des valeurs des états.

5.4.1 Comparaison des complexités de résolution

Dans le but d'évaluer si il est plus intéressant de résoudre directement l'AC-POMDP basé sur l'équation 5.9 que résoudre le POMDP équivalant basé sur l'équation 2.9 du chapitre 2, nous avons besoin de comparer la complexité des deux approches. Pour cette étude comparative nous nous sommes appuyée sur les algorithmes dits *point-based*, tels que : PBVI [Pineau *et al.*, 2003], Perseus [Spaan et Vlassis, 2004], HSVI [Smith et Simmons, 2005], SARSOP [Kurniawati *et al.*, 2008], qui sont très répandus dans la communauté. Nous tenons à rappeler que la fonction de valeur est convexe et linéaire par morceaux, et qu'elle peut être paramétrée par un ensemble d'α-vecteurs, conformément à ce qui a été présenté dans le chapitre 2.

Comme pour les méthodes de résolution exactes [Smallwood et Sondik, 1971], les algorithmes *point-based* calculent les ensembles de projections des α-vecteurs pour chaque action a et observation (o,θ), tels que :

$$\Gamma^{a,(o,\theta)} \leftarrow \alpha^{a,(o,\theta)}(s) = \gamma \sum_{s' \in S} p(s'|s,a) \tilde{\mathcal{O}}((o,\theta), a, s') \alpha_i(s'), \ \forall \alpha_i \in V' \tag{5.11}$$

Ensuite, ces algorithmes calculent l'ensemble Γ_b^a des α-vecteurs pour tout $a \in \tilde{A}$, selon :

$$\Gamma_b^a \leftarrow \Gamma^{a,*} + \gamma \sum_o \underset{\alpha_i^{a,(o,\theta)} \in \Gamma^{a,(o,\theta)}}{\arg\max} \; b \cdot \alpha_i^{a,(o,\theta)} \tag{5.12}$$

avec le revenu immédiat $\Gamma^{a,*} \leftarrow \alpha^{a,*}(s) = r(s,a)$, en utilisant un sous-ensemble $\mathcal{B}$ constitué de points b de l'espace des états de croyance obtenus à partir des simulations stochastiques de la dynamique du système (PBVI, Perseus), ou à partir de recherches heuristiques dans l'espace d'états de croyance (HSVI, SARSOP).

La mise à jour de la fonction de valeur est ainsi réalisée par l'opération de *backup* :

$$V \leftarrow backup(b) = \underset{\alpha_b^a \in \Gamma_b^a, a \in \tilde{A}}{\arg\max} \; b \cdot \alpha_b^a, \; \forall b \in \mathcal{B} \tag{5.13}$$

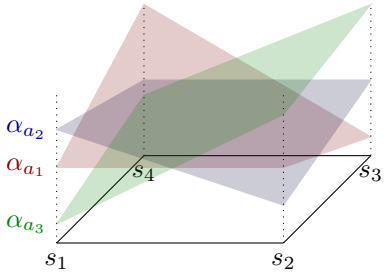
Calculer la mise à jour de la valeur nécessite de générer $|\tilde{A}||\mathcal{O}||\Theta||V'|$ projections de la fonction de valeur à l'instant de temps précédent, sachant que la taille de la fonction de valeur $|V'|$ est bornée par $|\mathcal{B}|$. $|V|$ peut être plus petit que $|\mathcal{B}|$, puisque des états de croyance $b \in \mathcal{B}$ peuvent avoir le même α-vecteur dominant. Donc, le temps nécessaire pour la mise à jour de la fonction de valeur V est : $|S||\tilde{A}||\mathcal{O}||\Theta||V'||\mathcal{B}|$ [Pineau *et al.*, 2003].

En revanche, nous mettons en évidence que calculer la mise à jour de la valeur pour un AC-POMDP implique, d'une part une maximisation sur un sous-ensemble d'actions tel que $\mathcal{A}_b \subset A$, et d'autre part que des projections $\Gamma^{a,(o,\theta)}$ soient calculées seulement pour les α-vecteurs dont l'action associée est cohérente avec θ, telles que $a_\alpha \in \theta$. Si nous appliquons les mêmes opérations que précédemment directement sur le modèle AC-POMDP, on voit avec l'aide de l'équation 5.9 que la mise à jour de la valeur pour un état de croyance b donné, sera évaluée sur un plus petit nombre d'α-vecteurs, parce que nous allons maximiser la valeur sur un sous-ensemble d'actions applicables $\mathcal{A}_b \in A$. Pour chaque $b \in \mathcal{B}$, le temps nécessaire au calcul de l'opération de mise à jour de la valeur $V(b)$, appelée *backup*, est réduit, dans le pire cas, à : $|S||\mathcal{A}_b||\mathcal{O}||\Theta||V'||\mathcal{B}|$. De cette façon, le temps pour l'opération complète pour tout $b \in \mathcal{B}$ est : $\sum_{b \in \mathcal{B}} |S||\mathcal{A}_b||\mathcal{O}||\Theta||V'|$ dans le pire cas.

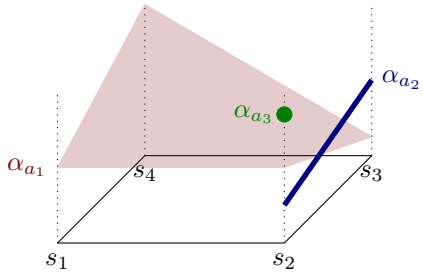
D'autre part, lors du calcul des projections $\Gamma^{a,(o,\theta)}$, il est nécessaire de tenir compte du fait que chaque $\alpha \in V'$, a une action a associée. Et donc, pour un θ donné, nous allons projeter les α-vecteurs dont l'action associée a appartient à l'ensemble d'actions réalisables θ. Ceci est dû au fait que l'état de croyance futur $b_a^{(o,\theta)}$, qui sera une distribution de probabilité sur des états dont l'ensemble d'actions réalisables est θ (cf. théorème 5.3.2), aura sa valeur maximisée pour un α-vecteur de V' dont l'action associée appartient nécessairement à l'ensemble θ.

En s'appuyant sur ces arguments, nous présumons que le gain minimum dû au fait qu'on optimise directement le modèle AC-POMDP à la place du POMDP équivalent vaut : $\frac{|\cup_{s \in \mathcal{S}} \mathcal{A}_s|}{\sum_{b \in \mathcal{B}} |\mathcal{A}_b|}$. Sachant que pour plusieurs problèmes, $\mathcal{A}_b$ peut être petit, puisqu'il contient les actions qui sont appliquées pour tout s tel que $b(s) > 0$ en filtrant plusieurs actions, contrairement au cas classique où $\cup_{s \in \mathcal{S}} \mathcal{A}_s$ qui contient toutes les actions du modèle, sera possiblement (beaucoup) plus grand. De plus, comme les projections $\Gamma^{a,(o,\theta)}$ sont calculées pour les α-vecteurs cohérents avec θ, moins de vecteurs seront pris en compte. Ceci nous amène à conclure que le gain de temps de calcul dans le pire cas sera linéaire, et polynomial dans le cas général. Ceci sera mis en évidence dans nos résultats expérimentaux, présentés dans la section 5.6.

D'autre part, notons que le modèle AC-POMDP n'attribue pas valeur $r(s,a) \rightarrow \mathbb{R}$ à la paire état-action pour laquelle l'action a n'est pas réalisable dans l'état s, puisque le modèle AC-POMDP ne tient simplement pas compte de cette paire, c'est-à-dire que la paire n'a pas de raison d'exister. En termes d'α-vecteur, nous avons besoin définir la valeur de ces actions sur l'espace des états pour lesquels cette action est réalisable. Ainsi, le α-vecteur d'une action



(a) Espace de définition des α-vecteurs dans les POMDP.



(b) Espace de définition des α-vecteurs dans les AC-POMDP. Notez que : a_1 est réalisable sur les 4 états ; a_2 est réalisable en s_1 et s_4 ; et a_3 n'est réalisable qu'en s_2.

FIGURE 5.5 – Schéma de l'espace de définition des α-vecteurs pour un POMDP et un AC-POMDP à 4 états, pour lesquels l'action a associée au α-vecteur n'est pas réalisable sur tous les états. Notez que la représentation utilisée est abusive, étant donné qu'une fonction de valeur du POMDP sur 4 états nécessite d'un représentation à 4 dimensions.

a n'a sa valeur définie que sur les sommets s où cette action est définie. L'espace de définition des α-vecteurs du modèle AC-POMDP est illustré sur la figure 5.5. Notons que contrairement au modèle MOMDP, qui définit un sous-espace $Y \subset S$ de variables partiellement observables, sur lequel tous les α-vecteurs de la fonction de valeur sont définis, le modèle AC-POMDP ne définit pas de sous-espace : l'espace d'état est le même que pour le modèle POMDP, mais un α-vecteur n'a sa valeur définie que pour les états où réaliser l'action associée à cet α-vecteur a un sens. De cette façon, les différents α-vecteurs qui constituent la fonction de valeur des AC-POMDP ont des supports différents.

Dans la prochaine section, nous formalisons notre algorithme PCVI (*PreCondition Value Iteration*), dédié à une résolution directe du modèle AC-POMDP. PCVI a été conçu pour mettre en évidence un gain de temps de calcul dû à la structure particulière du modèle AC-POMDP, discuté dans cette sous-section, ainsi que le respect des contraintes de faisabilité sur les actions du modèle à certains états.

5.5 PCVI : PreConditions Value Iteration

Dans le but de valider notre approche, nous avons implémenté PCVI – *PreCondition Value Iteration*, un algorithme *point-based* qui est une adaptation de l'algorithme d'itérations sur la valeur PBVI à des contraintes de faisabilité d'actions. PCVI travaille sur un ensemble fini $\mathcal{B} = \{b_0, ..., b_n\}$ d'états de croyance et utilise la nouvelle mise à jour de l'état de croyance de l'équation 5.4 ainsi que l'opérateur de mise à jour de la valeur de l'équation 5.9. Ces opérations nous permettent de prendre en compte les préconditions sur les actions directement à partir des différents ensembles d'actions réalisables $\Theta = \{\theta_1, ..., \theta_m\}$. PCVI, comme PBVI, explore l'espace d'états de croyance par des trajectoires stochastiques. Toutefois, il existe quatre différences majeures par rapport à PBVI :

- Les projections $\Gamma^{a,(o,\theta)}$ sont calculées pour les paires (o, θ), où o représente l'observation standard et θ l'observation sur l'ensemble d'actions réalisables, contrairement à PBVI qui ne fait aucune différence entre elles ;
- PBVI calcule les projections $\Gamma^{a,(o,\theta)}$ pour tout α-vecteur en V'. PCVI, au contraire, calcule les $\Gamma^{a,(o,\theta)}$ pour tout α-vecteur de V' tel que $a_\alpha \in \theta$, c'est-à-dire qu'il ne tient compte que des α-vecteurs de V' cohérents avec θ. De cette façon, moins de α-vecteurs seront projetés.

– La mise à jour de la fonction de valeur V pour un état de croyance donné est évaluée seulement pour les actions dans $\mathcal{A}_b$, contrairement à PBVI qui évalue toutes les actions du modèle.
– Un α-vecteur n'a sa valeur définie que sur les états (sommets s) où l'action associée est réalisable.

Afin de formaliser les opérations de projection et de mise à jour de la valeur pour un état de croyance donné, nous détaillons dans la suite l'opérateur de mise à jour de la valeur (*backup*) utilisé par PCVI.

Comme pour les méthodes de résolution exactes [Smallwood et Sondik, 1971], l'algorithme PCVI calcule les ensembles de projections des α-vecteurs pour chaque action a et chaque paire d'observations (o, θ), tels que :

$$\Gamma^{a,(o,\theta)} \leftarrow \alpha^{a,(o,\theta)}(s) = \gamma \sum_{s' \in S} p(s'|s,a)\mathbb{F}(a,s)p(o|,a,s')\mathbb{I}(\theta,s')\alpha_i(s'),\ \forall \alpha_i \in V' \text{avec } a_{\alpha_i} \in \theta \tag{5.14}$$

où $\mathbb{F}(a,s)$ représente la relation de faisabilité d'une action dans un état. Ceci est nécessaire pour empêcher l'algorithme d'associer une valeur à une action a qui n'est pas réalisable dans un état s, ce qui n'aurait pas de sens.

Ensuite, en utilisant un sous ensemble $\mathcal{B}$ constitué des points b de l'espace des états de croyance obtenus à partir des simulations stochastiques de la dynamique du système comme pour le cas des algorithmes PBVI et Perseus, PCVI calcule l'ensemble Γ_b^a des α-vecteurs pour tout $a \in \mathcal{A}_b$, selon :

$$\Gamma_b^a \leftarrow \Gamma^{a,*} + \gamma \sum_o \underset{\alpha_i^{a,(o,\theta)} \in \Gamma^{a,(o,\theta)}}{\arg\max} b \cdot \alpha_i^{a,(o,\theta)} \tag{5.15}$$

avec le revenu immédiat $\Gamma^{a,*} \leftarrow \alpha^{a,*}(s) = r(s,a)\mathbb{F}(a,s)$, construit de telle façon qu'aucune récompense ne soit associée aux états pour lesquels l'action a n'est pas réalisable. Ainsi, les α-vecteurs générés pour une action a ont leur valeur définie sur les états de S, pour lequel cette action est réalisable.

La mise à jour de la fonction de valeur est donc réalisée par l'opération de *backup*, telle que :

$$V \leftarrow backup(b) = \underset{\alpha_b^a \in \Gamma_b^a, a \in \mathcal{A}_b}{\arg\max}\ b \cdot \alpha_b^a,\ \forall b \in \mathcal{B} \tag{5.16}$$

Comme déjà mentionné (voir figure 5.5), dans le modèle AC-POMDP, la relation de faisabilité $\mathbb{F}(a,s)$ est définie. Donc, pour une paire (s,a) si l'action a est réalisable, $\mathbb{F}(a,s) = 1$, sinon, $\mathbb{F}(a,s) = 0$. Le modèle AC-POMDP n'attribue pas de valeur $r(s,a) \rightarrow \mathbb{R}$ à cette paire état-action, puisque le modèle AC-POMDP ne tient pas compte de cette paire. Pour cela, nous avons besoin de filtrer la valeur de ces états. Ceci est fait par la multiplication $r(s,a)\mathbb{F}(a,s)$. Cette multiplication joue le rôle d'un filtre en laissant passer la valeur des états s pour lesquels a est réalisable et en mettant à zéro la valeur des états pour lesquels a n'est pas réalisable. Ceci est une façon de définir la valeur de ces actions uniquement sur les états pour lesquels cette action est réalisable. Ainsi, le α-vecteur d'une action a n'a sa valeur définie que sur les sommets s où cette action est définie.

Notons que l'opération arg max de l'équation 5.16 pour un $b \in \mathcal{B}$ ne tient compte que des α-vecteurs dont les actions associées appartiennent à $\mathcal{A}_b$. De plus, nous savons que les composantes, c'est-à-dire les états supports de b, tels que $b(s) > 0$, appartiennent à $\sigma(b)$. Comme $\mathcal{A}_b$ est construit selon $\mathcal{A}_b := \mathcal{A}_s, \forall s \in \sigma(b)$, lors du produit scalaire, les composantes de b qui ont une valeur plus grande que zéro ont aussi une valeur définie dans les α-vecteurs évalués. Ceci nous permet de mettre à zéro les composantes $\alpha(s)$ tels que $\mathbb{F}(s,a_\alpha) = 0$. Ce masque se rapproche des masques de type BDD utilisés dans les approches symboliques à

base des ADD [Feng et et al., 2002, Feng *et al.*, 2002]. De plus, si on n'utilise pas le masque $F(s, a_\alpha) = 0$, une action non réalisable en s pourrait avoir une valeur non nulle incorrecte qui pourrait être évaluée et comparée aux autres.

Nous tenons à remarquer, que nous nous sommes aussi intéressée à une version relaxée de l'équation d'optimalité utilisée par PCVI qui permet, à la manière des MOMDP, de découpler les observations standard des observations sur les faisabilité d'actions et à réduire de façon exponentielle en nombre d'observation $|\Theta|$ le temps de calcul. Cette version relaxée sera présenté plus tard.

Dans la prochaine section, nous détaillerons le pseudo-code de la version *non-relaxé* de notre algorithme PCVI qui s'appuie sur les opérations de projection et de mise à jour de la valeur que nous venons de présenter, et qui nous appelons PCVI1.

5.5.1 Algorithme PCVI1

L'algorithme 9 formalise le pseudo-code de PCVI. Comme déjà mentionné, PCVI est un algorithme basé sur l'itération approchée de la valeur. Pour cela il est nécessaire de générer les états de croyance qui formeront l'ensemble $\mathcal{B}$.

PCVI initialise l'ensemble $\mathcal{B}$ au début de la résolution (ligne 3), à partir de b_0 et de Θ_0 selon équation 5.8. Notons qu'au moment de l'optimisation PCVI ne connaît pas à l'avance Θ_0, puisque Θ_0 est la première observation reçue avant la réalisation de la première action. Pour cela, il est nécessaire de tenir compte de tout Θ_0 possible par rapport à b_0. Nous avons besoin de vérifier quels sont les états s pour lesquels $b_0(s) > 0$, afin d'identifier les Θ_0 possibles. La mise à jour avec l'équation 5.8 est ainsi réalisée pour tout Θ_0 possible, et chaque $b_0^{\Theta_0}$ généré est intégré à $\mathcal{B}$.

Ensuite, les projections $\Gamma^{a,(o,\theta)}$ de la fonction de valeur V_{k-1} sont calculées (ligne 9) suivant l'opérateur décrit dans l'équation 5.14. Notons qu'à ce moment, les projections seront générées pour toutes les actions du modèle. Nous avons séparé le calcul des projections de la boucle de mise à jour de la valeur pour les états de croyance, afin de les calculer une fois pour toutes (comme dans PBVI).

Ensuite, pour tout $b \in \mathcal{B}$ l'opérateur de mise à jour de la valeur est appliqué (lignes 11 et 12), suivant les équations 5.15 et 5.16. Une fois les opérations de mise à jour réalisées, l'ensemble $\mathcal{B}$ est étendu.

L'ensemble $\mathcal{B}$ est étendu comme pour PBVI : pour chaque point $b \in \mathcal{B}$, un état s est tiré suivant $b(s)$, puis pour chaque action $a \in \mathcal{A}_b$ ($\mathcal{A}_b$ construit selon $\mathcal{A}_b := \mathcal{A}_s, \forall s \in \sigma(b)$), une paire d'observations est tirée suivant $\mathbb{I}(\theta, s')$, $p(o|s')$ et $p(s'|s, a)$, ainsi un ensemble $\{b_{a_0}, ..., b_{a_j}\}$ est créé. L'étape suivante consiste à mesurer la distance euclidienne des points $\{b_{a_0}, ..., b_{a_j}\}$ par rapport à tout $b \in \mathcal{B}$, et le point le plus distant de tous les $b \in \mathcal{B}$ est choisi et intégré à $\mathcal{B}$.

L'algorithme PCVI s'arrête dès que K itérations sont réalisées ou quand la différence maximale entre les valeurs actuelle et précédente pour tout $b \in \mathcal{B}$ est plus petite qu'un seuil défini par l'utilisateur.

Comme déjà mentionné, le masque $\mathbb{F}(a, s)$, utilisé dans la ligne 9 permet de filtrer les valeurs des vecteurs pour chaque action a lors de projections de valeurs futurs. Ainsi, nous évitons de répercuter une valeur non nulle d'une composante du α-vecteur projeté dans l'instant présent sur les états où l'action a n'est pas réalisable. Ceci garantit qu'une valeur non nulle incorrecte puisse être évaluée et comparée aux autres.

Notons que la fonction de valeur obtenue contient tous les α-vecteur, sans faire la différence entre les α-vecteurs dont les composantes ont des valeurs définies sur des états différents. Le choix du bon α-vecteur à évaluer dans les opérations des lignes 9, 11 et 12 dépend de θ (ligne 9) et de $\mathcal{A}_b$ (lignes 11 et 12). Contrairement à ce qui est fait dans le cadre MOMDP, dans l'algorithme 7 présenté au chapitre 2, la sélection du bon α-vecteur défini sur le sous-espace

Algorithme 9: PCVI1

entrée : POMDP, K
sortie : fonction de valeur V

$k \leftarrow 0$;
Initialiser $V_{k=0} \leftarrow \emptyset$;
Initialiser $\mathcal{B}$ avec b_0, Θ_0 ;
repeat
  $k \leftarrow k+1$;
  $V_k \leftarrow \emptyset$;
  if $V_{k-1} \neq \emptyset$ **then**
    for $a \in A$ **and** $(o,\theta) \in \mathcal{O} \times \Theta$ **do**
      $\Gamma^{a,(o,\theta)} \leftarrow \alpha_i^{a,(o,\theta)}(s) = \gamma \sum_{s' \in \mathcal{S}} p(s'|a,s')\mathbb{F}(a,s)p(o|a,s')\mathbb{I}(\theta,s')\alpha'_i(s'), \forall \alpha'_i \in V_{k-1}$ avec $a_{\alpha'_i} \in \theta$;
  for $b \in \mathcal{B}$ **do**
    $\Gamma_b^a \leftarrow \Gamma^{a,*} + \sum_{\substack{o \in \mathcal{O} \\ \theta \in \Theta}} \operatorname{argmax}_{\alpha \in \Gamma^{a,(o,\theta)}} (\alpha \cdot b), \forall a \in \mathcal{A}_b$, avec $\mathcal{A}_b := \mathcal{A}_s, \forall s \in \sigma(b)$;
    $V_k \leftarrow \operatorname{argmax}_{\Gamma_b^a, \forall a \in \mathcal{A}_b} (\Gamma_b^a \cdot b)$;
  Étendre $\mathcal{B}$ comme dans l'algorithme PBVI [Pineau *et al.*, 2003] ;
until $k < K$ ou $||\max_{\alpha_k \in V_k} \alpha_k \cdot b - \max_{\alpha_{k-1} \in V_{k-1}} \alpha_{k-1} \cdot b|| < \epsilon, \forall b \in \mathcal{B}$;

Y, est faite en choisissant d'abord le sous-ensemble de α-vecteurs correspondant à la variable visible $x \in X$, puisque la fonction de valeur sur l'espace complet d'états est paramétrée par un ensemble de $\Gamma_{\mathcal{Y}}(x)$, c'est-à-dire $\Gamma = \{\Gamma_{\mathcal{Y}}(x) | x \in \mathcal{X}\}$.

Nous nous sommes aussi intéressée à une version relaxée de l'équation d'optimalité utilisée par PCVI qui permet, à la manière des MOMDP, de découpler les observations standard des observations sur les faisabilité d'actions. Cette version relaxée, que nous avons appelé PCVI2, est basée sur une borne inférieure de la valeur pour un état de croyance donné. Nous verrons dans la section 5.6 que cette version relaxée sous-optimale permet d'accélérer encore plus le temps de calcul (gain exponentiel en la taille de Θ) avec des récompenses espérées correctes dans certains domaines.

5.5.2 Algorithme PCVI2

Afin de mettre en évidence la construction de la borne inférieure de la valeur pour un état de croyance b donné utilisée dans une version relaxée de PCVI, l'équation 5.9, pour un instant n, peut être réécrite telle que :

$$V_{n+1}(b) = \max_{a \in \mathcal{A}_b} \left[r(b,a) + \gamma \sum_{o \in \mathcal{O}} \sum_{\theta \in \Theta} p(o,\theta|a,b) V_n(b_a^{(o,\theta)}) \right] \tag{5.17}$$

En développant cette équation en fonction de $V_n(b_a^{(o,\theta)})$ paramétrée par des α-vecteurs nous avons :

$$V_{n+1}(b) = \max_{a \in \mathcal{A}_b} \left[\sum_{s \in S} r(s,a)b(s) + \gamma \sum_{o \in \mathcal{O}} \sum_{\theta \in \Theta} p(o,\theta|a,b) \max_{\alpha_n \in V_n} \sum_{s' \in S} b_a^{(o,\theta)}(s')\alpha_n(s') \right] \tag{5.18}$$

En utilisant l'équation 5.4, dans l'équation 5.18 :

$$\begin{aligned}
V_{n+1}(b) &= \max_{a\in\mathcal{A}_b}\left[\sum_{s\in S} r(s,a)b(s) + \gamma\sum_{o\in\mathcal{O}}\sum_{\theta\in\Theta} p(o,\theta|a,b)\max_{\alpha_n\in V_n}\sum_{s'\in S}\frac{\mathbb{I}(\theta,s')p(o|s',a)\sum_{s\in S}p(s'|s,a)b(s)\alpha_n(s')}{p(o,\theta|a,b)}\right]\\
&= \max_{a\in\mathcal{A}_b}\left[\sum_{s\in S} r(s,a)b(s) + \gamma\sum_{o\in\mathcal{O}}\sum_{\theta\in\Theta}\max_{\alpha_n\in V_n}\sum_{s'\in S}\mathbb{I}(\theta,s')p(o|s',a)\sum_{s\in S}p(s'|s,a)b(s)\alpha_n(s')\right]\\
&\geq \max_{a\in\mathcal{A}_b}\left[\sum_{s\in S} r(s,a)b(s) + \gamma\sum_{o\in\mathcal{O}}\max_{\alpha_n\in V_n}\sum_{\theta\in\Theta}\sum_{s'\in S}\mathbb{I}(\theta,s')p(o|s',a)\sum_{s\in S}p(s'|s,a)b(s)\alpha_n(s')\right]\\
&= \max_{a\in\mathcal{A}_b}\left[\sum_{s\in S} r(s,a)b(s) + \gamma\sum_{o\in\mathcal{O}}\max_{\alpha_n\in V_n}\sum_{s'\in S}\sum_{\theta\in\Theta}\mathbb{I}(\theta,s')p(o|s',a)\sum_{s\in S}p(s'|s,a)b(s)\alpha_n(s')\right],
\end{aligned}$$

sachant que, $\sum_{\theta\in\Theta}\mathbb{I}(\theta,s')=1$, quel que soit $s'\in S$, nous avons donc :

$$V_{n+1}(b) \geq \max_{a\in\mathcal{A}_b}\left[\sum_{s\in S} r(s,a)b(s) + \gamma\sum_{o\in\mathcal{O}}\max_{\alpha_n\in V_n}\sum_{s'\in S}p(o|s',a)\sum_{s\in S}p(s'|s,a)b(s)\alpha_n(s')\right] \quad (5.19)$$

$$= \max_{a\in\mathcal{A}_b}\left[\sum_{s\in S} r(s,a)b(s) + \gamma\sum_{o\in\mathcal{O}}\max_{\alpha_n\in V_n}\sum_{s\in S}b(s)\sum_{s'\in S}p(o|s',a)p(s'|s,a)\alpha_n(s')\right] \quad (5.20)$$

Cette mise à jour sous-optimale de la valeur pour un état de croyance donné nous permet de calculer des projections non plus pour les couples (o,θ), mais seulement pour les différents $o\in\mathcal{O}$ comme dans le cas classique des POMDP :

$$\Gamma^{a,o} \leftarrow \alpha^{a,o}(s) = \gamma\sum_{s'\in\mathcal{S}} p(s'|a,s')\mathbb{F}(a,s)p(o|a,s')\alpha'_i(s'),\ \forall\alpha_n\in V_n \quad (5.21)$$

$$\Gamma^a_b \leftarrow \Gamma^{a,*} + \sum_{o\in\mathcal{O}}\operatorname*{argmax}_{\alpha\in\Gamma^{a,o}}(\alpha\cdot b), \forall a\in\mathcal{A}_b \quad (5.22)$$

$$V_{n+1} \leftarrow \operatorname*{argmax}_{\Gamma^a_b,\forall a\in\mathcal{A}_b}(\Gamma^a_b\cdot b) \quad (5.23)$$

avec $\Gamma^{a,*}\leftarrow\alpha^{a,*}(s)=r(s,a)\mathbb{F}(a,s)$.

Étant donné que l'ensemble $\mathcal{O}\ll\mathcal{O}\times\Theta$, le nombre de projections calculées est divisé par $|\Theta|$. Ainsi, le temps de calcul sera réduit significativement par l'utilisation de ce calcul approché. Le pseudo-code de PCVI2 est quasiment le même que PCVI1. Les différences sont telles que :

1. les projections de la ligne 9 de l'algorithme 9 sont calculées suivant l'équation 5.21 ;
2. la mise à jour de la valeur réalisée dans les lignes 11 et 12 de l'algorithme 9 sera faite selon les équations 5.22 et 5.23.

Ainsi, le nombre d'itérations de la boucle *for* est divisé par $|\Theta|$ ainsi que la taille de la fonction de valeur.

Dans la prochaine section, nous présentons nos résultats expérimentaux pour cette nouvelle approche de résolution qui découple les différentes informations des observations. Nous cherchons à démontrer que les politiques AC-POMDP obtenues pour cette nouvelle approche sont fondées et équivalentes aux politiques des POMDP transformées. Nous mettrons en évidence l'efficacité des algorithmes PCVI1 et PCVI2 dans des problèmes avec des contraintes sur les actions modélisées par des préconditions.

5.6 Évaluation Expérimentale

Protocole expérimental

Pour la vérification de l'approche nous avons adapté des problèmes de la littérature en ajoutant des contraintes de faisabilité sur les actions. Ces contraintes sont exprimées en tant que préconditions d'actions. Nous rappelons que les actions sont applicables pour un sous-ensemble de l'espace d'état, et que l'ensemble des actions réalisables est observé en plus des autres observations.

Nous avons étudié quatre critères de performance :

1. La taille de la fonction de valeur par rapport au nombre d'α-vecteurs qu'elle contient ;
2. L'évolution de l'erreur de Bellman pendant le calcul de la politique ;
3. Le temps de planification jusqu'à convergence (ϵ donné pour un ensemble $\mathcal{B}$ donné) ;
4. L'espérance de la somme pondérée des récompenses obtenues lors de l'évaluation de la politique ;

Tous les problèmes ont été modélisés en tant que AC-POMDP, et résolus par les deux versions de notre algorithme PCVI. Pour pouvoir comparer nos résultats avec d'autres algorithmes de la littérature, comme par exemple PBVI et HSVI, les mêmes problèmes ont été transformés vers le modèle POMDP équivalent. Pendant l'évaluation de la politique, notre simulateur examine aussi le respect des contraintes sur les actions des politiques obtenues pour les différents algorithmes, étant donné que le simulateur connaît l'état caché du système.

Modélisation d'une contrainte sur une action

Nous rappelons que le format Cassandra [1] est un format classique répondu de description de problèmes POMDP, il a été proposé par [Cassandra, 1998] lors de sa thèse. Ainsi, pour que la faisabilité des actions puisse être prise en compte par le modèle AC-POMDP, nous avons ajouté dans la spécification des fichiers POMDP en format Cassandra une fonction du type :

$$\texttt{P} : < \texttt{action} > : < \texttt{state} > \ \ \texttt{bool value}$$

qui dit au *parser* si une action est faisable ou non dans un état donné. De cette façon, nous disposons en plus dans le modèle d'une matrice de faisabilité de taille $|A| \times |S|$.

À partir de l'information sur la faisabilité d'une action, notre algorithme construit les différents ensembles d'actions réalisables du modèle grâce à la définition 5.2.2, c'est-à-dire l'ensemble Θ du modèle AC-POMDP. Notons que, les autres observations stochastiques qui forment l'ensemble $\mathcal{O}$ sont décrites de manière classique dans les fichiers POMDP au format Cassandra :

$$\texttt{O} : < \texttt{action} > : < \texttt{final} - \texttt{state} > : < \texttt{observation} > \ \ \texttt{value}$$

Pour le modèle POMDP translaté, qui ne tient pas compte de Θ, la description de toutes les observations de l'ensemble $\tilde{\mathcal{O}}$ se fait selon le format Cassandra. Nous tenons à signaler que pour le modèle transformé, les récompenses de type $\tilde{r}(s,a)$ doivent aussi être décrites dans le fichier de spécification du problème. Les pénalités associées à des actions dites inacceptables doivent respecter la définition de la fonction de récompense du modèle POMDP équivalent : $\widetilde{R} : S \times \widetilde{A} \to \mathbb{R}$, telle que $\widetilde{r}(s,a) = r(s,a)$ si $a \in A_s$, et $\widetilde{r}(s,a) = -\infty$ sinon. La valeur

1. disponible en `http://www.pomdp.org/pomdp/code/pomdp-file-spec.shtml`

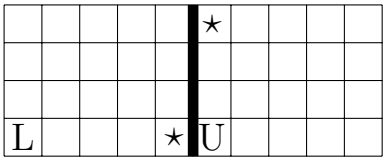


FIGURE 5.6 – Environnement de navigation *maze4x5x2*.

$\widetilde{r}(s,a) = -\infty$ est approchée par une valeur inférieure à la borne inférieure $\underline{R}$ de la fonction de valeur optimale. La valeur de la borne inférieure $\underline{R}$ peut être calculée par :

$$\underline{R} = \max_{a \in A} \sum_{t=0}^{\infty} \gamma^t r(s,a) = \frac{\max_{a \in A} \min_{s \in S} r(s,a)}{1-\gamma} \tag{5.24}$$

Ce même type d'approximation est utilisé par l'algorithme HSVI [Smith et Simmons, 2004], pour initialiser la borne inférieure de la fonction de valeur.

Les problèmes classiques qui seront présentés dans la suite démontrent le potentiel de l'approche. Ces problèmes se sont avérés intéressants car le modèle d'observation encode des observations qui peuvent être interprétées comme ensemble d'actions réalisables, comme par exemple, la présence de murs ou d'obstacles autour du robot. Ceci est le cas des problèmes *maze* et *hallway*, ce dernier étant celui qui a motivé la discussion au début de ce chapitre.

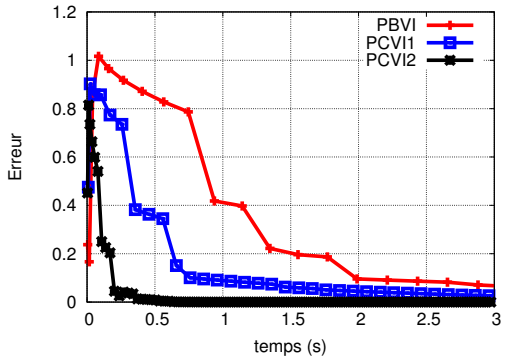
5.6.1 Le domaine : *maze*

Le domaine *maze* consiste en un problème de navigation. Nous présenterons particulièrement le problème *maze4x5x2* [Cassandra, 2005], où le robot doit naviguer dans une grille à deux étages. Le but est d'atteindre un état précis qui se situe dans différents endroits de la grille suivant l'étage auquel le robot se trouve, en lui rapportant une récompense de 1. La figure 5.6 illustre l'environnement de navigation du robot. Les actions de déplacement sont : *nord, sud, est, ouest*, et n'ont aucun coût associé. Les observations sont constituées par deux ensembles distincts : $\mathcal{O}$ qui concerne les observations des repères du premier (L) et deuxième étage (U) plus la présence ou non du but ($\star$), au total 4 ; et Θ qui concerne la présence des murs autour du robot, dont le nombre total dépend des différentes configurations pouvant être observées par le robot, dans ce cas un total de 9. Initialement le robot ne sait pas dans quelle case de la grille il se trouve, ceci est modélisé par une distribution uniforme sur tous les états de la grille sauf le but.

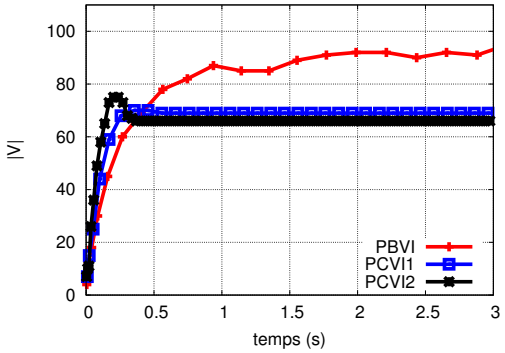
Pour le modèle POMDP classique, l'ensemble d'observations est constitué par le produit cartésien $\mathcal{O} \times \Theta$. Pour les paires état-action où le robot entrera en collision avec un mur, une pénalité de -10 est associée, sachant que $\underline{R} > -10$ ($\underline{R}$ donné par l'équation 5.24). De plus, une action supplémentaire *ne rien faire* est nécessaire, pour que le robot puisse recevoir une première information sur la présence ou pas de murs autour de lui. Pour le modèle AC-POMDP, pour les paires état-action où le robot entrera en collision avec un mur, la relation de faisabilité $\mathbb{F}(a,s)$ est explicitement mise à *false*, et aucune pénalité n'est associée. Comme dans le modèle AC-POMDP on débute par la réception de Θ_0, aucune action supplémentaire n'est nécessaire.

Dans la figure 5.7 nous montrons la vitesse de convergence pour PCVI1, PCVI2 et PBVI ainsi que l'évolution de la taille de la fonction de valeur, avec un temps limite de 3 secondes.

La figure 5.7(a) montre que la vitesse de convergence de l'erreur de Bellman est plus rapide pour les deux versions de PCVI. A noter que nous parlons d'une convergence vers une valeur plus petite que ϵ et pour un ensemble $\mathcal{B}$. La figure 5.7(b) montre aussi que le nombre d'α-vecteurs qui constitue la fonction de valeur est plus petit pour les deux versions



(a) Évolution de l'erreur de Bellman avec temps limite de planification de 3 secondes.



(b) Taille de la fonction de valeur en nombre d'α-vecteurs.

FIGURE 5.7 – Évaluation expérimentale pour le problème de navigation *maze4x5x2*.

de PCVI. Ces résultats confirment l'analyse théorique de la complexité de PCVI1 présentée dans la section 5.4.1, qui suggère que les ensembles d'actions applicables $\mathcal{A}_b$ pour les états de croyance $b \in \mathcal{B}$ sont plus petits, ce qui mène à moins d'évaluations lors de la mise à jour de la valeur pour tout $b \in \mathcal{B}$. Le facteur de gain théorique calculé par $\frac{|\cup_{s\in\mathcal{S}}\mathcal{A}_s|}{\sum_{b\in\mathcal{B}}|\mathcal{A}_b|}$, peut être vérifié par ces résultats. Prenons par exemple $\epsilon = 0.2$: nous savons que PBVI atteint cette valeur au bout de 1.5s, et que PCVI1 et PCVI2 au bout de 0.65s et 0.35s respectivement. Le facteur de gain expérimental de PCVI1 est d'au moins de 2 fois, si nous utilisons le facteur de gain théorique, pour ce modèle à 5 actions (POMDP) et à 9 ensembles différents d'actions réalisables avec une moyenne de $3,5$ actions par ensemble, nous avons $1.4 < 2$. Le gain réel (expérimental) est supérieur au gain théorique.

algorithme	E[R]	n.e.b	%but
PBVI	0.749257	7.945	100
PCVI1	0.784295	7.059	100
PCVI2	0.786018	6.842	100

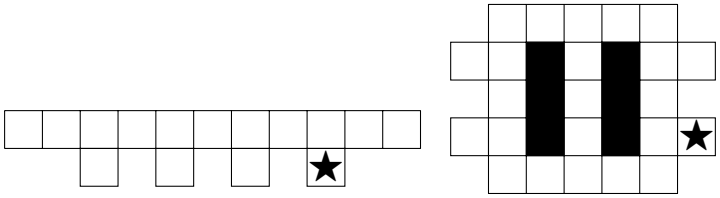
TABLE 5.1 – Résumé de performance des algorithmes pour le problème *maze*$4x5x2$.

La table 5.1 résume l'espérance de récompenses accumulées (notée $E[R]$) obtenues lors de 1000 simulations de la politique. Nous pouvons conclure que les politiques obtenues pour les deux versions de PCVI sont fondées et équivalentes à celle de PBVI. La politique de PBVI a un nombre moyen d'étapes pour atteindre le but (n.e.b) légèrement supérieur à celui de PCVI1 et PCVI2, ce qui est dû au fait que pour le modèle POMDP classique nous avons besoin d'ajouter une action supplémentaire qui n'est utilisée qu'à la première étape de décision, afin de renseigner le robot sur la configuration des murs qui l'entoure initialement.

5.6.2 Le domaine *hallway*

Au début de ce chapitre nous avons motivé cette étude en nous appuyant la discussion sur le domaine *hallway*, où nous avons remplacé les murs par des falaises dans le but de mettre en évidence l'importance d'éviter des actions qui peuvent endommager le robot qui évolue dans un environnement présentant des obstacles dangereux.

Le domaine *hallway* a été proposé par Littman dans [Littman *et al.*, 1995]. Ces problèmes encodent dans la fonction d'observation la présence des murs autour du robot, ce qui revient à observer les actions faisables. Par contre, dans la version originale de ce problème, le robot perçoit de manière imprécise la configuration de murs qui l'entoure. Pour la mise en évidence de notre approche, nous avons modifié le modèle d'observation : le robot observe de manière



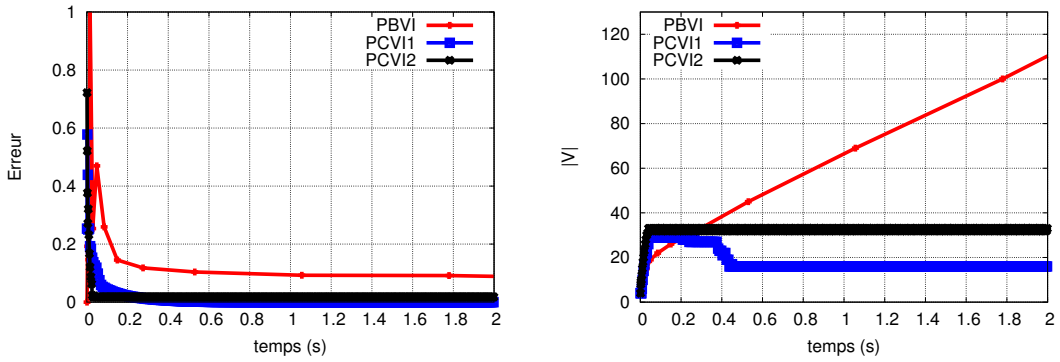
(a) hallway (b) hallway2

FIGURE 5.8 – L'environnement du domaine *hallway*.

précise la configuration des murs qui l'entoure, pour permettre au robot de choisir des actions sûres.

Dans la figure 5.8 nous illustrons les deux versions de ce problème, appelées *hallway* et *hallway2*. Les actions sont les mêmes que pour le domaine *maze*. Les ensembles d'observation obéissent au même découplage : $\mathcal{O}$ est constitué pas deux symboles qui informent sur la présence ou non du but, et Θ représente les différentes configurations de murs observées. Les actions de déplacement n'ont pas de coût, sauf celles qui amènent le robot à une collision dans la cas du modèle POMDP classique, avec une pénalité fixé à 10, où on a $\underline{R} > -10$. Pour le modèle AC-POMDP ceci est modélisé directement dans la relation de faisabilité $\mathbb{F}(a, s)$. Arriver à la case où le but se trouve rapporte une récompense de 1. A noter que cette structure d'évitement des murs se rapproche de la structure du problème de la mission d'atterrissage (problème d'éviter d'atterrir à partir d'une altitude de vol supérieure à 40 mètres, ou atterrir dans une zone considérée comme atterrissable), d'où notre intérêt pour l'étude de ce problème particulier.

Sur les figures 5.9 et 5.10 nous présentons une fois de plus la vitesse de convergence de l'erreur de Bellman, et la taille de la fonction de valeur en nombre d'α-vecteurs pour un temps de planification de 2 secondes.

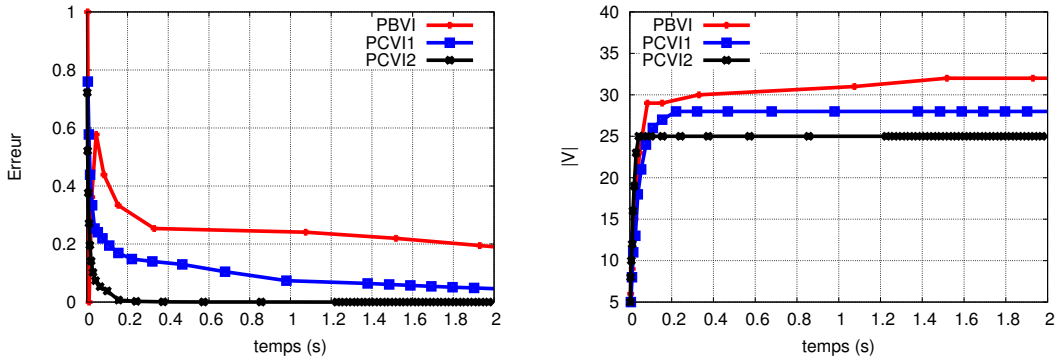


(a) Évolution de l'erreur de Bellman avec un temps limite de planification de 2 secondes.

(b) Taille de la fonction de valeur en nombre de α-vecteurs.

FIGURE 5.9 – Évaluation expérimentale pour le problème de navigation *hallway*.

Ces résultats nous permettent d'arriver aux mêmes conclusions que pour le domaine *maze* : les deux versions de PCVI convergent plus vite, et le nombre d'α-vecteurs est aussi plus petit. Ceci est mis en évidence dans le cas *hallway* (figure 5.9) : PBVI, au bout de 2 secondes de calcul, a tendance à encore augmenter le nombre d'α-vecteurs de la fonction de valeur. Contrairement aux deux versions de PCVI qui, au bout de 0.6 secondes, avaient déjà convergé en précision et en nombre de α-vecteurs. Toutefois, dans *hallway*2 PBVI atteint un seuil constant pour la taille de la fonction de valeur, nous pensons que cette différence de comportement vient de la différence de configuration des deux instances : *hallway*, modélisant un couloir étroit avec une fonction de transition d'état probabiliste amène la politique à



(a) Évolution de l'erreur de Bellman avec un temps limite de planification de 2 secondes.

(b) Taille de la fonction de valeur en nombre d'α-vecteurs.

FIGURE 5.10 – Évaluation expérimentale pour le problème de navigation *hallway2*.

considérer et à générer plus de α-vecteurs que pour *hallway*2.

Dans la table 5.2 nous présentons un résumé de la performance des trois algorithmes ($E[R]$, n.e.b, %but). Nous pouvons conclure une fois de plus que les politiques obtenues pour les deux version de PCVI sont fondées et équivalentes à celle de PBVI. De plus, on voit que le temps de calcul nécessaire aux différentes versions de PCVI est très faible comparé à celui de PBVI. Ceci nous permet de démontrer, une fois encore, l'avantage de résoudre directement le modèle AC-POMDP au lieu de son POMDP équivalent. Comme pour le domaine *maze*, la politique de PBVI a un nombre moyen d'étapes légèrement supérieur à celui de PCVI1 et PCVI2, ce qui est dû au fait que pour le POMDP classique nous avons besoin de rajouter une action supplémentaire qui n'est utilisée qu'à la première étape de décision.

algorithme	E[R]	n.e.b	%but
PBVI	0.617846	10.706	100
PCVI1	0.658238	9.64	100
PCVI2	0.642681	10.33	100

algorithme	E[R]	n.e.b	%but
PBVI	0.672065	9.012	100
PCVI1	0.707178	8.009	100
PCVI2	0.709952	7.923	100

TABLE 5.2 – Résumé de performance des algorithmes pour *hallway* et *hallway*2.

5.6.3 Le domaine *grid*

Nous avons aussi testé notre approche sur une classe de problèmes de navigation dans des grilles. Ces différentes grilles ont été générées aléatoirement. Ce domaine, qui sera ici appelé *grid*, est une extension des problèmes *maze* et *hallway*. Dans le domaine *grid* certaines cellules de la grille sont des obstacles qui peuvent endommager le robot. Ces obstacles peuvent être observés comme des murs qui entourent le robot. Les états sont les cellules libres de la grille. Les actions de déplacement sont : *nord*, *sud*, *est* et *ouest*, et elles ont un coût de -0.01. Pour le POMDP équivalent, il y a une action supplémentaire de type *rester sur place*, nécessaire lorsque le robot part d'un état de croyance uniforme sur toutes les positions possibles (excepté le but). L'ensemble d'observations $\mathcal{O}$ a deux symboles qui indiquent la présence ou non du but. L'ensemble Θ est formé par les différents ensembles d'actions réalisables, qui peuvent être parfaitement observés, en utilisant par exemple un capteur considéré suffisamment précis (détecteur d'obstacles de type laser circulaire par exemple). Notons que pour le POMDP équivalent l'ensemble d'observations $\tilde{\mathcal{O}}$ est le produit cartésien des deux ensembles $\mathcal{O}$ et Θ. De plus, pour le modèle POMDP équivalent, la fonction de récompense doit encoder le fait que se déplacer vers un obstacle peut endommager le robot : pour de telles paires état-action un coût de 100 est ajouté, sachant que $\underline{R} \gg -100$.

Un exemple de grille générée est montrée sur la figure 5.11. Le but est d'arriver à l'étoile qui est placée aléatoirement dans une cellule libre, et atteindre cet état rapporte une récompense de 10.

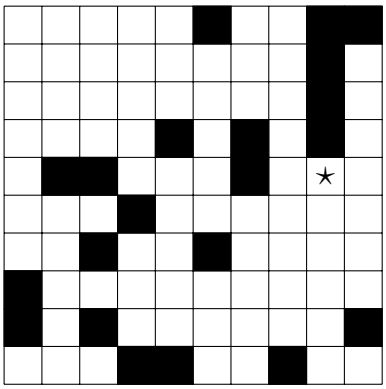


FIGURE 5.11 – Exemple de grille de taille 10×10 générée aléatoirement.

Nous présentons la figure 5.12 le temps de planification (échelle logarithmique), la taille de la fonction de valeur obtenue à la convergence ($\epsilon = 0.5$), et l'espérance des récompenses accumulées lors de la simulation de la politique. Nous comparons aussi nos résultats avec HSVI, un algorithme de type $point - based$ qui s'est montré très efficace dans plusieurs domaines de la littérature. Les deux versions de PCVI ont été aussi évaluées.

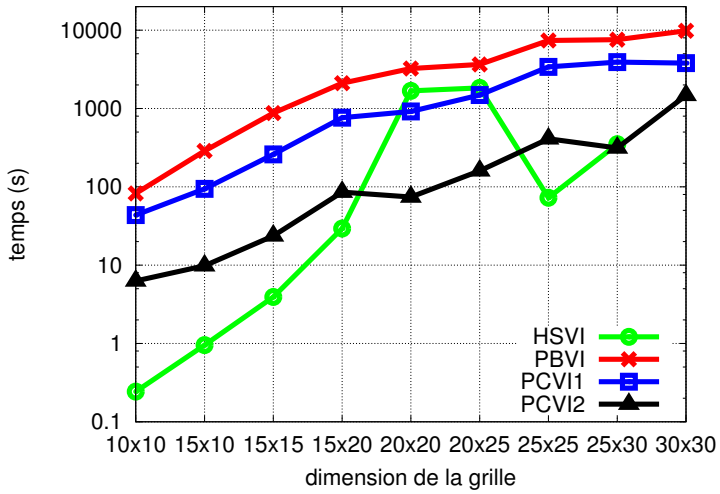
Comme PCVI1 et PCVI2 sont des algorithmes inspirés par PBVI, nous pourrions nous attendre à ce que HSVI ait une performance meilleure en ce qui concerne le temps de calcul et les récompenses accumulées. Mais, nous voyons que, pour certaines dimensions de la grille, les temps de planification de PCVI1 et PCVI2 sont meilleurs que celui de HSVI, en particulier pour la dimension 20×25, où les récompenses de PCVI1 et HSVI sont équivalentes (figures 5.12(a) et 5.12(b)). PCVI2, étant sous-optimal, conduit à une récompense inférieure.

De plus, pour certaines dimensions de la grille, la politique de HSVI est très mauvaise, accumulant des récompenses négatives assez importantes, ce qui veut dire que l'heuristique utilisée par HSVI conduit l'agent à heurter un obstacle. Nous croyons que, pour HSVI, qui est un algorithme de résolution *sous-optimal* basé sur la recherche heuristique, la pénalité de 100 associée aux paires état-action non faisables n'est pas suffisante. Dans ce cas, la pénalité même en étant inférieure à $\underline{R}$ ne garantit pas d'empêcher de réaliser des actions considérées non faisables.

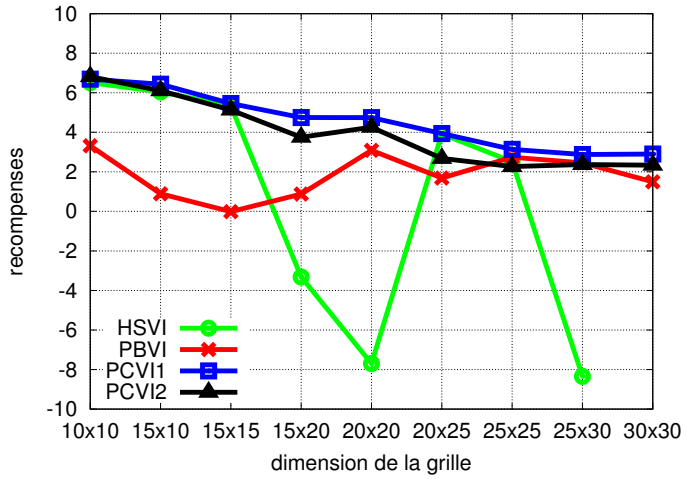
Nous pouvons en déduire que l'attribution d'une pénalité d'infaisabilité de paire état-action pour le POMDP classique dépend de l'algorithme de résolution utilisé s'il n'est pas prouvé optimal. Toutefois, PBVI n'utilise à aucun moment des actions non réalisables. Mais ce résultat démontre qu'un réglage de coût de type potentiomètre (convenablement théorique mais tout de même empirique) n'est pas très convenable, surtout pour les algorithmes sous-optimaux, et renforce notre argument qui considère que l'infaisabilité d'une paire état-action est un critère booléen (et non numérique) qui doit être explicitement décrit dans le modèle avec une interprétation sémantique claire.

Nous tenons à souligner que la politique obtenue par PCVI1 ou PCVI2 n'utilise à aucun moment des actions interdites, indépendamment du coût qu'on aurait pu associer à ces actions. Pour les deux versions de PCVI, aucune valeur n'est attribuée aux paires état-action indésirables ; autrement dit, la valeur de ces états n'intervient pas dans la définition de l'α-vecteur.

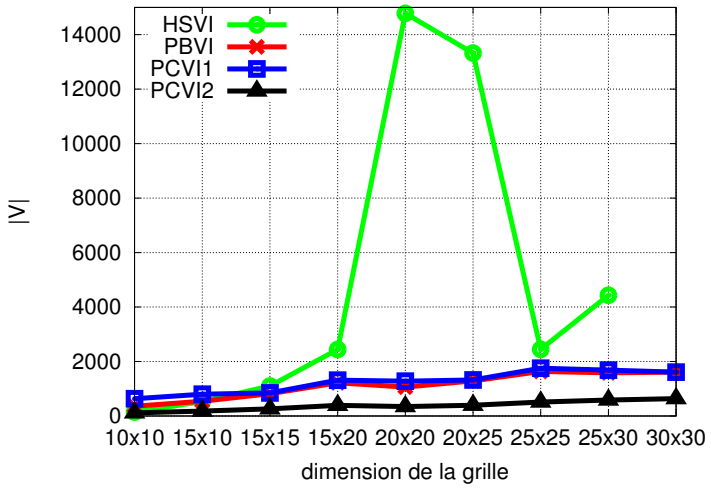
La figure 5.12(a) utilise une échelle logarithmique afin d'illustrer le rapport de gain de temps de planification ; nous démontrons, une fois encore, l'intérêt d'optimiser directement le



(a) Temps de planification en secondes pour $\epsilon = 0.5$.



(b) L'espérance de récompenses accumulées.



(c) Taille de la fonction de valeur en nombre d'α-vecteurs.

FIGURE 5.12 – Évaluation expérimentale pour des problèmes de navigation dans une grille.

modèle AC-POMDP, puisque le temps de calcul de la politique est considérablement réduit ; à noter que le gain est environ d'un facteur 2 pour PCVI1 et 20 pour PCVI2, tout en gardant un très bon niveau de performance (récompenses accumulées) comparé à la politique obtenue par PBVI ou HSVI. La taille de la fonction de valeur de PCVI1 est équivalente à celle de PBVI, et pour PCVI2 la taille est beaucoup plus petite, ce qui peut être expliqué par le fait que les projections de PCVI2 concernent seulement l'ensemble des observations $\mathcal{O}$, en générant moins d'α-vecteurs.

5.6.4 Le domaine *RockSample*

Le domaine *RockSample* a été proposé par [Smith et Simmons, 2004]. Ce domaine traite un problème de navigation d'un robot d'exploration martien dont l'objectif est d'observer la nature des pierres, de les ramasser si ces pierres sont "bonnes", et ensuite d'atteindre un état terminal. Le robot se déplace dans un environnement de type grille et réalise des actions de mesure pour recueillir des informations concernant les pierres. La figure 5.13 illustre l'environnement de navigation ainsi que la position a priori connue des pierres.

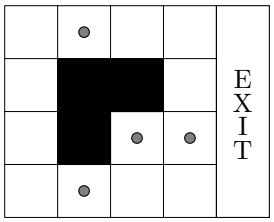


FIGURE 5.13 – Exemple d'environnement de navigation du domaine *RockSample*.

Nous avons modifié l'environnement de navigation en ajoutant des trous dans la grille, qui modélisent des zones dangereuses du terrain où le robot ne doit pas aller. Par exemple, une zone couverte de sable peut provoquer l'immobilisation temporaire ou même la mise hors service du robot. Ces zones sont représentées par les rectangles en noir sur la figure 5.13.

Nous avons testé notre approche sur trois instances différentes de ce domaine. Dans *RockSample* 4×4 le robot navigue dans une grille de taille 4×4 et peut ramasser 4 pierres, dans *RockSample* 5×5, la grille a une taille de 5×5 et le robot peut ramasser 5 pierres, et ainsi de suite. Les actions de déplacement sont les mêmes que pour les domaines précédents et n'ont aucun coût, par contre ces déplacements sont déterministes.

De plus, dans *RockSample*, l'agent dispose d'une action de mesure par pierre ainsi que d'une action de prise. La précision de la mesure de la nature d'une pierre est liée à la distance du robot à la pierre qu'il cherche à observer. Plus le robot est proche d'une pierre plus il a de bonnes chances de percevoir la vraie nature de cette pierre. Nous pouvons voir qu'une fois de plus, le modèle d'observation du problème est naturellement découplé : L'ensemble $\mathcal{O}$ concerne les observations probabilistes de la nature des pierres, et l'ensemble Θ concerne la topographie du terrain entourant le robot.

Ramasser une pierre dite "bonne" rapporte au robot une récompense de 10, ramasser une pierre mauvaise le pénalise de 10. Atteindre l'état final, représenté par *exit* dans la figure 5.13, lui rapporte une récompense de 10. Pour modéliser le fait que le robot ne doit pas aller vers des zones dangereuses ou sortir du terrain ces mouvements sont pénalisés de 100 ($\underline{R} > -100$) dans le cadre POMDP classique. Cette pénalité n'est pas prise en compte dans le modèle AC-POMDP.

La figure 5.14 présente le temps de planification en échelle logarithmique, la taille de la fonction de valeur obtenue à la convergence ($\epsilon = 0.7$), et la récompense accumulée lors de la simulation de la politique. Nous comparons aussi les résultats obtenus avec les deux

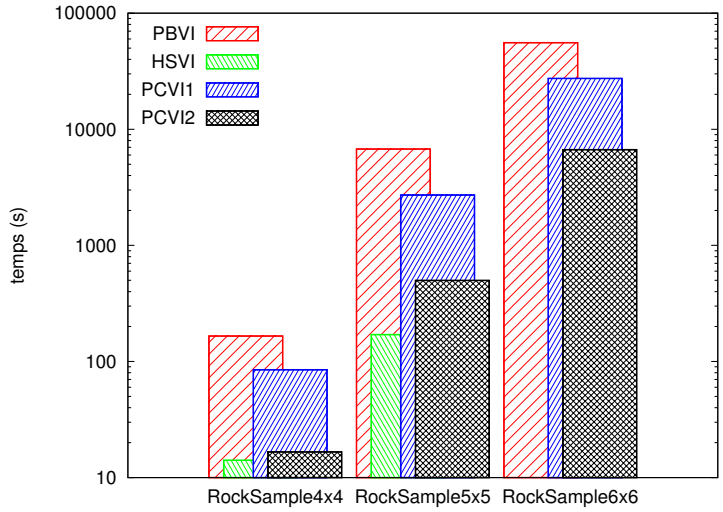
versions de PCVI avec HSVI. Ces politiques ont été calculées pour un état de croyance initial uniforme, c'est-à-dire qu'au début de la mission le robot ne connaît pas sa position dans la grille ni la nature des pierres qui sont présentes dans l'environnement. Il doit se déplacer en toute sécurité en ramassant des pierres si elles sont bonnes.

Nous pouvons voir sur la figure 5.14(a) que le temps de planification nécessaire à PBVI et aux deux versions de PCVI croît exponentiellement avec la taille du problème. HSVI est largement plus performant que PBVI, PCVI1 et PCVI2 pour *RockSample* 4×4 et *RockSample* 5×5. Par contre HSVI ne fournit pas de politique pour *RockSample* 6×6, car pour cette instance du problème HSVI atteint la limite de mémoire virtuelle allouée aux expérimentations (1.8Go).

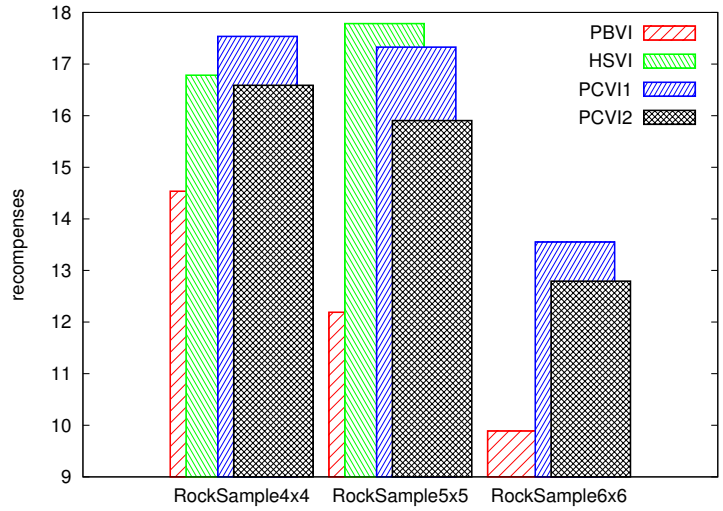
Toutefois, nous tenons à faire remarquer que PCVI1, atteint quasiment la même valeur de la politique (récompenses) que HSVI pour *RockSample* 4×4 et *RockSample* 5×5, tout en utilisant un nombre inférieur d'α-vecteurs dans la fonction de valeur optimisée. Ceci montre que notre approche est capable de fournir des politiques fondées. A noter que PBVI a une performance très pauvre, ce qui peut être expliqué par le fait que, pour certaines simulations de la politique, qui a été optimisée pour un ensemble $|\mathcal{B}| = 3000$, il a retrouvé des états de croyance pour lesquels une action n'a pas été optimisée, ce qui veut dire que la politique obtenue ne se généralise pas bien à des états de croyance qui n'appartiennent pas à $\mathcal{B}$.

La comparaison de temps de calcul des politiques entre PCVI et HSVI n'est pas tout à fait équitable, puisque PCVI (inspiré par PBVI) ne réalise aucune recherche heuristique dans l'espace d'états de croyance contrairement à HSVI. Les algorithmes basés sur la recherche heuristique se sont avérés d'être beaucoup plus efficaces que les algorithmes basés sur la recherche stochastique [Smith et Simmons, 2005, Shani *et al.*, 2007, Kurniawati *et al.*, 2008]. Ainsi, nous pensons que le temps de planification des deux versions de PCVI est comparable uniquement à celui de PBVI. Il serait par ailleurs tout à fait possible de concevoir une version AC-POMDP de HSVI, afin d'obtenir le même type de gains que pour PBVI. Cette piste de recherche sera discutée dans le chapitre 7.

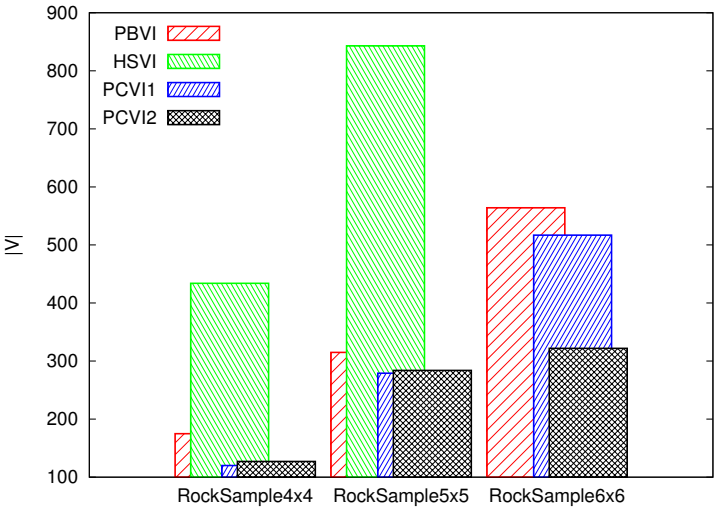
Concernant le nombre d'α-vecteurs qui constituent la fonction de valeur obtenue pour les deux versions de PCVI, nous voyons qu'il est inférieur à celui de PBVI et très inférieur à celui de HSVI. Ceci peut être expliqué en s'appuyant sur les résultats théoriques de la section 5.4.1 : résoudre directement le modèle AC-POMDP conduit à moins d'évaluations ($\max_{a \in \mathcal{A}_b}$ à la place de $\max_{a \in A}$) et à moins de projections $\Gamma^{a,(o,\theta)}$, ce qui peut, pour certains domaines, représenter moins d'α-vecteurs dominants dans la fonction de valeur. Nous tenons à remarquer que les α-vecteurs occupent chacun moins de place en mémoire, puisqu'ils ont des valeurs définies sur moins d'états.



(a) Temps de planification en secondes pour un $\epsilon = 0.7$.



(b) L'espérance de récompenses accumulées.



(c) Taille de la fonction de valeur en nombre d'α-vecteurs.

FIGURE 5.14 – Évaluation expérimentale pour des problèmes de navigation dans une grille.

5.6.5 La mission de détection et reconnaissance de cibles

Nous nous sommes à bien sûr intéressée à tester notre approche sur la mission d'atterrissage présentée dans le chapitre 3. Nous rappelons que dans cette mission l'agent hélicoptère cherche à détecter et à reconnaître un modèle de voiture en particulier parmi ceux contenus dans la base de données et à atterrir à côté de cette cible. Cette mission est particulièrement intéressante puisque certaines actions de l'agent ne sont applicables que sur un sous-ensemble d'états, dû à des contraintes de sécurité liées au vol, comme par exemple le risque d'atterrissage dans une zone dangereuse, ou aux contraintes sur les altitudes pour les différentes phases de vol. Nous supposons ici que l'agent doit explorer 3 zones. Chaque zone peut soit ne pas contenir de voiture, soit contenir un des 3 modèles de voiture de la base de données. L'agent cherchera le modèle C dans l'environnement.

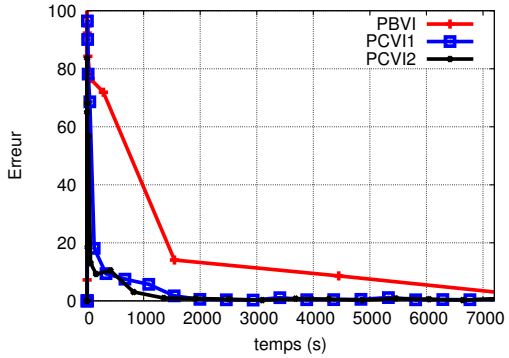
Les contraintes liées aux actions de l'agent l'hélicoptère sont décrites dans la suite :

- L'agent hélicoptère ne peut atterrir que si la zone est considérée comme sûre. Pour cela, des variables d'état booléennes doivent être ajoutées au modèle, ces variables indiquent si l'on peut ou non atterrir dans la zone concernée ;
- L'agent hélicoptère peut changer de zone seulement à une altitude de 40 mètres, pour des questions de sécurité liées à la zone d'expérimentation, comme la présence d'obstacles (des bâtiments, ou des arbres) ;
- L'agent hélicoptère peut changer d'angle de vue seulement à une altitude de 30 mètres ; le changement d'angle de vue pour une altitude de 40 mètres nécessiterait un rayon de 40 mètres au tour de la zone, ce qui n'est pas réalisable sur le terrain de test ;
- L'agent hélicoptère ne peut atterrir qu'à partir de l'altitude de 30 mètres, ce qui suppose le besoin d'une phase d'approche.

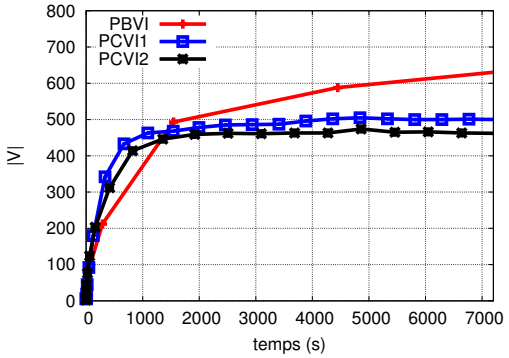
Pour tenir compte des contraintes supplémentaires, le modèle d'observation du POMDP doit être étendu. Il est nécessaire pour l'agent d'observer s'il est possible d'atterrir dans la zone où il se trouve (détection d'obstacles), et il est aussi nécessaire pour l'agent d'observer l'altitude de vol. Ces symboles d'observation, pour le modèle AC-POMDP, sont encodés dans l'ensemble Θ. L'espace d'état des deux modèles est agrandi, en raison de l'ajout des variables d'états booléennes qui caractérisent le fait que l'on puisse ou non atterrir dans les différentes zones. Nous avons au total 3073 états. La fonction de récompense attribue une récompense de 100 si l'agent atterrit près de la bonne cible et de -100 sinon. Les actions de déplacement ont un coût proportionnel à la distance parcoure. A noter que ces coûts de déplacement sont dans un intervalle de $[-1; -15]$. Ainsi pour le modèle POMDP équivalent la pénalité associée à des actions infaisables a été fixée à 100 ($\underline{R} > -100$). Dans ce problème, les actions sont déterministes, sauf le changement d'altitude. Ceci est dû au fait que la différence de 10 mètres entre les deux altitudes du modèle n'est pas très grande, ce qui peut poser un problème lors de l'envoi de la consigne de déplacement au contrôleur d'exécution, qui peut interpréter que l'hélicoptère est déjà arrivé à la destination, selon le rayon de précision lié au déplacement élémentaire *go_to* (interprétation de l'action changer d'altitude).

Dans les figures 5.15(a) et 5.15(b) nous présentons les résultats obtenus lors de l'optimisation de la politique pour la mission d'atterrissage du domaine de détection et reconnaissance de cibles pour un ensemble $\mathcal{B}$ d'états de croyance de taille 3000, et pour un temps de calcul fixé à 7200 secondes. Nous pouvons vérifier que les vitesses de convergence des deux versions de PCVI sont plus importantes que celle de PBVI, et que le nombre d'α-vecteurs qui constituent la fonction de valeur de PCVI1 et PCVI2 est moins important. Ces résultats permettent de mettre en évidence notre étude théorique de complexité, puisque moins d'actions sont évaluées en réalisant ainsi plus d'itérations sur la valeur des états de croyance qui constituent $\mathcal{B}$. Les itérations sont représentées par les points sur les courbes de la figure 5.15. Le nombre d'α-vecteurs découle directement de ces mêmes raisons, puisque comme moins de projections $\Gamma^{a,(o,\theta)}$ sont générées pour PCVI1, ce qui représente moins d'α-vecteurs

dominants dans la fonction de valeur.



(a) Évolution de l'erreur de Bellman avec temps limite de planification de 7200 secondes.



(b) Taille de la fonction de valeur en nombre d'α-vecteurs.

FIGURE 5.15 – Évaluation expérimentale pour le problème de détection et reconnaissance de cible.

Nous avons réalisé 30000 simulations avec un horizon de 50 étapes de décision pour chacune des politiques obtenues, en piochant l'état initial selon l'état de croyance initial b_0. Notez que l'état de croyance initial ne suppose aucune connaissance a priori sur la présence au non de cibles dans les différentes zones, et sur le fait que les zones soient considérées comme atterrissables. Ceci est modélisé par une croyance initial uniforme sur 512 états. En regardant la valeur, le nombre moyen d'étapes pour attendre le but et le pourcentage de missions réussies obtenus pour ces politiques (conformément le tableau 5.3), nous voyons que la politique obtenue par PCVI1 a une valeur légèrement supérieure à celle obtenu par PBVI, ce qui confirme le fait que les politiques obtenues pour le modèle AC-POMDP sont équivalentes à celle du POMDP. De plus, PCVI1 a un pourcentage légèrement plus haut de missions réussies, en démontrant que la politique obtenue par PCVI1 est fondée. D'autre part, on vérifie que PCVI2 fournit effectivement une borne inférieure de la valeur avec un taux de réussite inférieur.

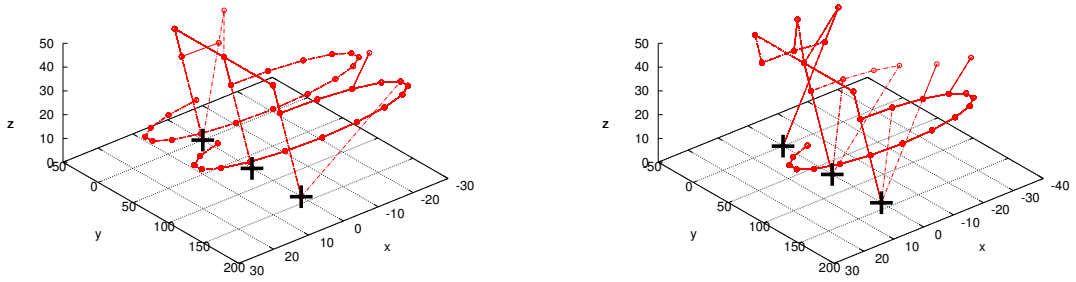
algorithme	E[R]	n.e.b (min,max)	%but
PBVI	-8.0249	5.11948 (3, 19)	30.8
PCVI1	-7.9429	5.46384 (3, 22)	31.8
PCVI2	-17.679	3.78463 (3, 18)	12.9

TABLE 5.3 – Résumé de performance des algorithmes pour le problème de détection et de reconnaissance de cibles.

Dans la figure 5.6.5 nous montrons des différentes trajectoires obtenues en simulant les politiques de PBVI et PCVI1. Ces trajectoires mettent en évidence le respect des différentes contraintes sur l'altitude de vol. On vérifie que les changements de zone s'effectuent à une altitude de 40 mètres et que les actions de changement d'angle de vue et d'atterrissage s'effectuent à partir d'un altitude de 30 mètres. La contrainte concernant le fait qu'une zone soit considérée comme atterrissable n'est pas montrée dans ces figures.

5.7 Conclusion

Dans ce chapitre nous avons étudié une sous-classe de problèmes de décision séquentielle sous observabilité partielle, pour lesquels l'observation de l'agent est constituée en partie de



(a) Trajectoires de l'hélicoptère autonome pour 100 simulations de la politique de PBVI.

(b) Trajectoires de l'hélicoptère autonome pour 100 simulations de la politique de PCVI1.

FIGURE 5.16 – Trajectoires de l'hélicoptère autonome. Les trois croix en noire représentent le centre des différentes zones.

symboles qui représentent l'ensemble d'actions réalisables dans l'état caché du système. Nous avons vu que cette information ne supprime pas l'observabilité partielle de l'état, puisque plusieurs états peuvent avoir le même ensemble d'actions réalisables.

Nous avons formellement défini cette sous-classe en tant que AC-POMDP, pour laquelle nous avons fourni des équations d'optimalité, de mise à jour de l'état de croyance et une transformation vers le modèle POMDP équivalent afin de pouvoir comparer l'approche proposée avec le modèle classiquement utilisé. Cette transformation équivalente repose sur l'hypothèse communément admise que l'on peut choisir des récompenses équivalentes à des valeurs $-\infty$ dans le modèle POMDP. À cette récompense est généralement associée une valeur qui correspond à la borne inférieure des valeurs des états.

Nous avons proposé un algorithme appelé PCVI, qui optimise directement le modèle AC-POMDP en tirant directement profit des propriétés du modèle telles que l'évaluation d'un sous-ensemble d'actions pour un état de croyance donné ($\max_{a \in \mathcal{A}_b}$) et le calcul de projections $\Gamma^{a,(o,\theta)}$ seulement pour les α-vecteurs de V' qui sont cohérents avec l'ensemble θ observé. De plus, les α-vecteur ont des valeurs définies seulement sur les états où l'action associée au α-vecteur est applicable. Ce qui nous permet de tirer profit des vecteurs creux lors du calcul. Nous avons démontré l'avantage, en ce qui concerne le temps de calcul, d'optimiser le modèle AC-POMDP au lieu de son POMDP équivalent, tout en garantissant le respect des contraintes sur les actions.

Toutefois, une question se pose : est-il utile de modéliser des contraintes sur des actions par des relations de faisabilité ou par des préconditions ou est-ce seulement une convenance ou un moyen algorithmique ? Nous pensons que les observations concernant les ensembles d'actions réalisables doivent être clairement séparés dans le modèle afin de garder la riche sémantique des ces observations. Nous avons aussi démontré l'intérêt d'un telle modélisation en ce qui concerne le gain d'efficacité en temps de calcul. Dans le cadre POMDP mono-agent, notre modèle permet de garder l'interprétation sémantique des préconditions et montre finalement que la façon *classique* de modéliser le problème n'est pas la plus efficace en comparaison de notre algorithme PCVI qui exploite directement la structure spécifique des AC-POMDP. De plus, dans le cadre multi-agent, l'artifice d'associer une pénalité assez importante aux actions indésirables n'est pas équivalent [Pralet *et al.*, 2010a] à l'approche que nous avons proposé. Par exemple, dans un jeu à somme nulle et à deux joueurs A et B où les deux joueurs utilisent une même fonction de récompense, cette modélisation n'a pas de sens. Le joueur A cherchant à maximiser son critère et le joueur B cherchant à le minimiser : le joueur B aura tout intérêt à choisir des actions inacceptables.

En conclusion, nous croyons que les AC-POMDP s'avèrent très utiles en robotique mobile, où les symboles d'observation concernant la faisabilité des actions sont typiquement obtenus par des capteurs spécifiques. Cette information souvent découplée de la planification est généralement traitée directement par le contrôleur d'exécution. Avec le modèle AC-POMDP, nous pouvons intégrer ce type d'information dans le modèle en gardant sa sémantique naturelle, et, avec le schéma d'optimisation proposé, nous pouvons exploiter la structure du modèle afin d'accélérer certaines opérations dans le calcul de la politique.

Dans le chapitre suivant nous traiterons un problème différent, mais cependant complémentaire, qui consiste à résoudre *en ligne*, en respectant les contraintes de temps de réponse du planificateur, le problème de planification sous incertitude et observabilité partielle. Nous proposerons un cadre original d'optimisation (sous-optimal) durant l'exécution de la politique qui anticipe l'évolution probabiliste des états d'exécution en planifiant constamment pendant l'exécution des actions. Ce cadre diffère des approches de résolution en ligne utilisées en robotique.

Chapitre 6

Utilisation des POMDP sous contraintes temporelles : optimisation anticipée et exécution en parallèle de politiques POMDP

Nous proposons dans ce chapitre un cadre algorithmique d'optimisation anticipée et d'exécution en parallèle de politiques POMDP, appelé AMPLE (*Anytime Meta PLannEr*). Ce cadre contrôle un sous-planificateur POMDP pour optimiser et exécuter en parallèle une politique sous contraintes de temps associées à la durée des actions, en raisonnant sur les états d'exécution futurs possibles du système robotique. Cette approche diffère de l'approche classique d'optimisation en ligne, qui optimise l'action pour chaque état de croyance courant pendant une durée de temps incompressible fixée à l'avance. Nous présenterons le cadre proposé à l'aide des résultats obtenus lors d'un vol expérimental réalisé avec le drone hélicoptère, dans lequel nous avons embarqué ce cadre algorithmique au sein d'une architecture de décision robotique. Nous montrerons aussi, à partir de résultats expérimentaux obtenus par des simulations réalistes, que le choix de l'algorithme sous-jacent peut être déterminant vis-à-vis de la réactivité demandée au planificateur. Nous verrons aussi que l'approche proposée est une alternative à l'approche classique, puisqu'elle permet de finaliser la mission dans un temps plus court en moyenne. Ainsi, nous montrons que des techniques d'intelligence artificielle comme les POMDP peuvent être appliquées avec succès pour contrôler automatiquement des actions de perception et de décision pour des missions complexes en temps contraint.

Pour motiver l'étude menée dans ce chapitre, nous nous intéresserons à la problématique de détection et reconnaissance de cibles par un hélicoptère autonome déjà présenté dans le chapitre 3, qui est un domaine de recherche actif [Wang *et al.*, 2012], dû à la mise en place croissante de systèmes de drones dans des missions civiles et militaires. Dans ce genre de mission, la stratégie de décision de haut niveau est généralement donnée par une règle écrite à la main (survoler une zone donnée, atterrir, prendre une image, etc), qui dépend des événements stochastiques (détection d'une cible dans une zone donnée, cible reconnue), qui peuvent survenir lors de l'exécution de la règle de décision.

Concevoir des systèmes robotiques qui sont capables de construire automatiquement leur plan pour atteindre leurs objectifs de mission à long-terme sous incertitude et observabilité

partielle constitue un véritable défi, en raison des différentes incertitudes considérées et à la complexité de résolution issue des différents méthodes de décision séquentielle en environnement probabiliste et partialement observable [Sabbadin *et al.*, 2007, Ross et Chaib-Draa, 2007, Smith et Simmons, 2005, Bonet et Geffner, 2009]. Dans ce contexte, en utilisant les POMDP des politiques peuvent être optimisées et construites automatiquement [Smallwood et Sondik, 1973]. Toutefois, la plupart des systèmes robotiques qui utilisent les POMDP comme modèle formel, supposent que le problème à résoudre est défini avant le début de la mission [Taha *et al.*, 2011, Spaan, 2008, Bai *et al.*, 2011, Schesvold *et al.*, 2003]. Dans les cas où le problème n'est pas connu à l'avance, la politique est calculée en ligne avec un temps très réduit pour l'optimisation [Eidenberger *et al.*, 2009, Miller *et al.*, 2009].

La première contrainte concernant la résolution de notre application vient du fait que si on considère une mission de détection et de reconnaissance [Wang *et al.*, 2012] comme un problème de décision séquentielle dans l'incertain, certains paramètres du problème sont inconnus avant le vol. En effet, le nombre de zones qui composent l'environnement, et la position des cibles dans ces zones, sont des exemples de paramètres généralement inconnus avant le vol et qui doivent être automatiquement extraits au début de la mission (par exemple, par des techniques de traitement d'image conformément au chapitre 3) afin de définir le problème de décision séquentielle à optimiser.

Dans la suite, nous définissons l'application robotique, en lien avec le modèle présenté dans le chapitre 3, qui a motivé le travail de recherche de ce chapitre.

6.1 Définition de l'application robotique

Dans ce chapitre nous étudions la mission de détection, reconnaissance et atterrissage, présentée dans le chapitre 3 section 3.3.2, qui se rapporte à une problématique de décision séquentielle sous incertitude et observabilité partielle de détection et de reconnaissance de cible par un hélicoptère autonome. Cette mission est modélisée sous forme d'un POMDP défini en ligne, une fois que le nombre de zones à explorer a été analysé en ligne, en utilisant des techniques de traitement d'image. Le nombre de cibles, ainsi que leurs identités ne sont pas connus de l'hélicoptère autonome. Nous rappelons que le but de la mission est de trouver une cible particulière, parmi celles connues de la base de données et d'atterrir à côté de cette cible.

Cette application robotique est stimulante et originale pour deux raisons principales :

- la mission de détection et reconnaissance de cibles est considérée comme un problème de planification séquentielle à *long terme*, avec des actions à la fois de perception (changement d'angle de vue, changement d'altitude, changement de zone) et d'aboutissement de mission (atterrissage) ; Dans le cadre général de la perception active, les applications existantes optimisent plutôt une mesure de l'incertitude de l'état de croyance à court terme [Eidenberger *et al.*, 2009].
- le POMDP est résolu en ligne pendant le vol, en tenant compte des contraintes de temps requises par la durée de la mission et de l'anticipation des états futurs possibles du système robotique.

Mener entièrement de façon automatique une mission de ce type requiert plusieurs briques, techniques et théoriques : la modélisation duale du traitement d'image et de la décision, la résolution (interprétation d'image et optimisation de la politique) et le contrôle de l'exécution du plan. En ce qui concerne la modélisation, nous avons vu dans le Chapitre 3 que l'apprentissage d'un modèle d'observation à partir de données réelles est un facteur très important si l'on veut traiter le problème de façon réaliste. Pour cela, nous avons appris le modèle d'observation pour cette application robotique à partir d'une étude statistique des images collectées lors de campagnes de prise de vue. Ce modèle appris est adapté à cette application, puisqu'il

ne suppose pas un nombre fixé de zones. Il se base seulement sur le fait qu'une cible est ou non dans une zone particulière sachant la zone et l'altitude de vol de l'agent hélicoptère. Toutefois, le modèle ne peut pas être considéré comme parfaitement réaliste puisque l'apprentissage dépend des conditions de prise de vue (ensoleillement, ombre, etc). Comme déjà discuté dans le chapitre 3 nous ne supposons pas d'inférence en ligne du modèle d'observation, ceci a été appris hors ligne à partir des données acquises dans des vols expérimentaux préalables. A noter que l'attention de ce travail de thèse a porté sur la faisabilité de méthodes de décision séquentielle à *long terme* pour des applications robotiques d'identification et reconnaissance de cibles en environnement incertain et pas sur le développement de nouvelles techniques de traitement d'image.

6.1.1 Génération en ligne du problème à résoudre

Comme dans notre application robotique nous supposons que le nombre de zones à explorer est inconnu au début de mission, il était nécessaire de mettre en place, en complément du cadre de planification et d'exécution en parallèle de la politique, une fonction responsable de la génération automatique du modèle POMDP à résoudre. Cette fonction suppose qu'une phase initiale de balayage du terrain est effectuée en début de mission, afin d'extraire les zones d'intérêt, au moyen d'un traitement d'image. Une fois que le nombre et les coordonnées des zones ont été obtenus, et que le modèle de voiture recherché a été fixé, le problème POMDP est généré. Cette génération automatique, implémentée en C++, est un service disponible dans le composant superviseur de la mission (l'architecture embarquée sera présentée dans la section suivante). Dans la suite, nous détaillerons la génération automatique du problème POMDP à résoudre.

Conformément le chapitre 3, nous disposons de variables d'état qui dépendent : du nombre de zones N_z, d'altitudes de vol N_h et de modèles connus de la base de données N_{models}. Nous avons donc :

- z, avec N_z valeurs possibles, qui indique la position de l'hélicoptère autonome ;
- h, avec N_h valeurs possibles, qui indique l'altitude de vol de l'hélicoptère autonome ;
- $Id_{Ta_{z_1}}$ (respectivement $Id_{Ta_{z_2}}$, $Id_{Ta_{z_3}}$, etc), avec $N_{models}+1$ valeurs possibles, qui indique l'identité ou l'absence d'une cible dans la zone 1 (respectivement dans la zone 2, dans la zone 3, etc.)

Nous rappelons que la fonction de transition d'états est considérée comme déterministe. Pour chaque variable d'état, nous définissons sa matrice de transition selon l'action. Par exemple, pour $N_z = 3$ et pour l'action $\mathbf{go\_to}(z_1)$, la matrice de transition d'état $T_z^{z_1}$ de la variable d'état z sera donnée par :

$$T_z^{z_1} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Pour les autres variables, une matrice de transition égale à la matrice identité est définie. Ainsi la construction de la matrice de transition d'état complète pour l'action $\mathbf{go\_to}(z_1)$ est donné par le produit de Kronecker entre les matrices de transition des variables d'état :

$$T_{go\_to(z_1)} = T_z^{z_1} \bigotimes I_h \bigotimes I_{Id_{Ta_{z_1}}} \bigotimes I_{Id_{Ta_{z_2}}} \bigotimes I_{Id_{Ta_{z_3}}}$$

Pour tenir compte de l'état terminal, qui est atteint lors que l'action d'atterrissage est réalisée, on ajoute une ligne et une colonne à cette matrice avec une composante égale à 1 dans la diagonale. Ceci parce qu'une fois que l'on est dans l'état terminal, l'action $\mathbf{go\_to}(z_1)$ n'a aucun effet (cf. chapitre 3).

La même procédure est utilisée pour la génération de la fonction de récompense. Par exemple, pour la génération de la fonction de récompense de l'action **go_to(z_1)**, la distance entre le centre des zones est calculée et la matrice distance entre zones D_z est construite. Ceci est possible parce que les coordonnées de zones sont des données d'entrée pour la génération automatique. La matrice coût associée au changement de zone C_{z,z_1} est donc obtenue par :

$$C_{z,z_1}(i,j) = \begin{cases} \frac{D_z(i,j)}{10}, & \text{si } i \neq j \\ 100, & \text{sinon} \end{cases}$$

où la valeur de 100 modélise le fait que si on est dans une zone i on ne veut pas aller vers cette même zone i (le point de rendez-vous d'entrée dans une zone est fixé, voir figure 3.1(a) du chapitre 3). Puis, le produit élément par élément entre $T_z^{z_1}$ et C_{z,z_1} est réalisé, et la matrice $R_s^{s'}(a)$ est obtenue par le produit de Kronecker. Pour obtenir la fonction $r(s,a)$ on somme les colonnes de $R_s^{s'}(a)$, pour chaque s. Le même calcul est utilisé pour les action de changement d'altitude. Pour l'action de changement d'angle de vue, un coût constant est considéré $C_{view} = \frac{H_m \phi}{10}$, proportionnel à l'altitude moyenne de vol H_m (par exemple 35 mètres), et à l'arc de cercle parcouru par l'agent hélicoptère, qui dépend de la valeur de l'angle ϕ fixé par le concepteur (par exemple $\phi = 10$ dégrées), conformément la définition de l'action **change_view** donnée dans la section 3.2.2 du chapitre 3 (voir figure 3.1(b) du chapitre 3). Pour toutes les actions de déplacement (sauf atterrissage), un coût constant $C_{proc} = 0.5$ est aussi ajouté à chaque état, qui modélise le coût du traitement d'image engendré une fois que l'action est exécutée.

La génération de la fonction d'observation, qui est la même pour toutes les actions du modèle, est faite à partir d'une méthode récursive. Cette méthode identifie pour chaque état s la valeur des variables d'état au moyen d'une division euclidienne. A partir des valeurs des variables d'état z, h et Id_{Ta_z}, la table de probabilité apprise est consultée et les probabilités de toutes les observations sont copiées en tant que ligne de la matrice O_s^o. Par exemple, si l'hélicoptère est dans la zone z_1, à une altitude h_1, et que $Id_{Ta_{z_1}}$ = modèle A, la première ligne de la table 4.2 (chapitre 4) est copiée dans la matrice O_s^o. Dans cette même procédure récursive, la fonction de récompense associée à l'action d'atterrissage est générée, puisque dans cette fonction l'on peut tester les valeurs des variables z et Id_{Ta_z}, conformément la définition de la fonction de récompense de l'action **land**, donnée dans la section 3.3.2 du chapitre 3. La récompense attribuée à l'atterrissage proche de la cible recherchée est fixée à 50, et une pénalité de 100 est associée à une atterrissage manquée.

Le résultat de la génération automatique est un fichier texte où le modèle POMDP est décrit selon le format standard Cassandra [1].

Maintenant que nous avons décrit la génération automatique du problème à résoudre, nous présentons l'architecture Orocos, qui est l'architecture embarquée sur les drones de l'Onera. Nous focaliserons l'attention sur les différents composants créés pour la mission de détection et de reconnaissance de cibles.

6.1.2 Présentation de l'architecture Orocos implémenté pour l'application robotique

L'architecture embarquée sur l'hélicoptère autonome est une architecture Orocos [2] [Soetens et Bruyninckx, 2005]. Orocos est une librairie pour la robotique implémenté en C++. Elle fournit un cadre de développement temps réel et modulaire. Orocos est basé sur des composants qui facilitent le développement des applications robotiques, en reposant sur la séparation des fonctions : calcul, communication, configuration et coordination.

1. disponible en `http://www.pomdp.org/pomdp/code/pomdp-file-spec.shtml`
2. `http://www.orocos.org/`

Orocos permet de créer des composants dédiés à différentes tâches, par exemple pour notre application robotique nous avons un composant pour le traitement d'image, un composant pour la planification, un composant pour la navigation, un composant pour la supervision, etc. Orocos garantit des communications en temps-réel et sécurisées entre ces composants. Dans l'architecture, chaque composant possède une interface standardisée, qui décrit ses ports de données et les services qu'il fournit (voir figure 6.1), et peut réagir à des événements ou traiter des requêtes, ainsi qu'exécuter des scripts en temps réel.

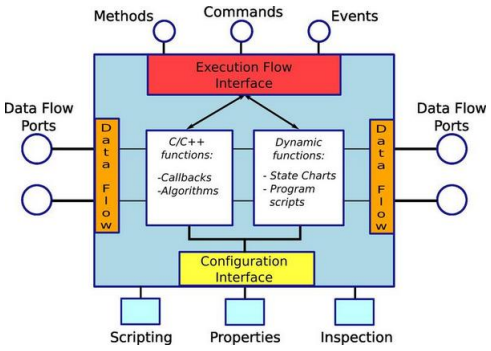


FIGURE 6.1 – Schéma d'un composant Orocos (source `http://www.orocos.org/rtt`).

Pour cette mission plusieurs composants ont été crées. Dans la figure 6.2 nous montrons le schéma général des composants Orocos de la mission test. Le composant *Camera* est responsable de la prise d'image en temps-réel. Le composant *Obc* est le composant responsable de la navigation de notre hélicoptère autonome. Les composants *Detection* et *Matching* sont responsables des phases de détection et classification, respectivement, de l'algorithme de traitement d'image utilisé, et qui a été déjà présenté dans le Chapitre 3. Le composant *lua* est le composant qui implémente l'interface avec l'opérateur. Le composant *AmpleOptimize* contrôle le sous-planificateur. Ces composants communiquent en temps-réel avec le composant *AmpleExecute* qui est responsable de la supervision de la mission (création du problème POMDP à résoudre, exécution de la politique calculée en ligne).

La communication particulière entre les composants *AmpleOptimize* et *AmpleExecute* forment le cadre d'optimisation et exécution en parallèle proposé dans ce chapitre. Ainsi, dans la suite ce chapitre nous allons focaliser notre attention sur le cadre d'optimisation anticipée et d'exécution en parallèle, appelé AMPLE (*Anytime Meta PLannEr*), que nous avons développé pour résoudre en ligne des problèmes POMDP sous contraintes de durée de mission. Chacun de ces deux composants sera détaille.

6.2 Cadre d'optimisation anticipée et d'exécution en parallèle – AMPLE

Les problèmes POMDP complexes et de grande taille peuvent rarement être optimisés en ligne à cause de l'absence de moyens de calcul suffisants. De plus, le problème à résoudre n'est pas toujours connu à l'avance. Par exemple, notre mission de détection et reconnaissance de cibles dépend du nombre de zones considérées qui sont automatiquement extraites de l'environnement par un traitement d'image en vol au tout début de la mission. De telles applications nécessitent une plateforme robotique efficace pour la résolution et exécution parallèles de politiques POMDP de sorte à réussir la mission dans le temps imparti.

Notre contribution se ramène à l'extension du cadre d'optimisation en parallèle de l'exécution de politique proposé par [Teichteil-Konigsbuch *et al.*, 2011], auparavant limité à la planification déterministe ou de type MDP, aux modèles de type POMDP soumis à des contraintes de temps.

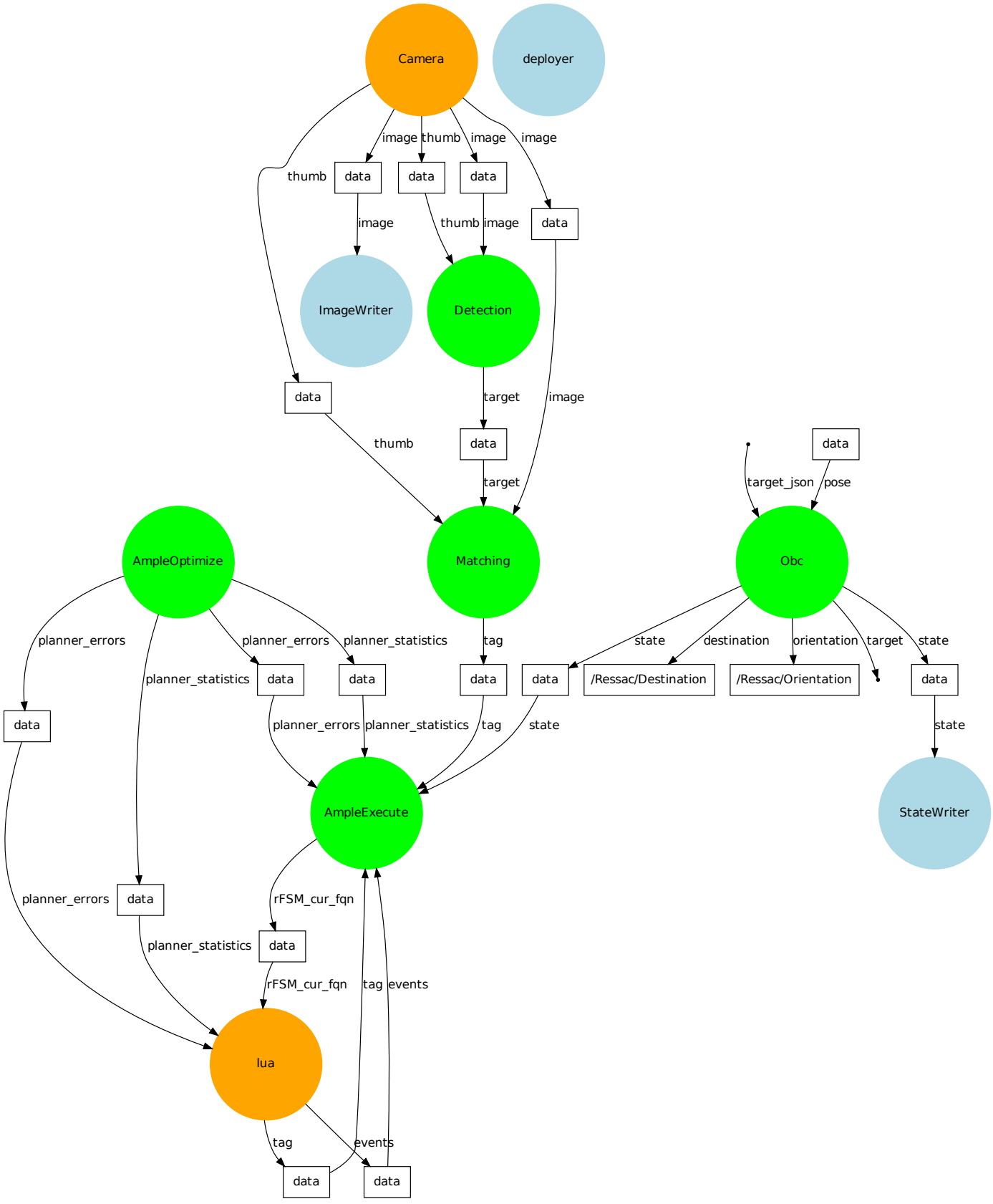
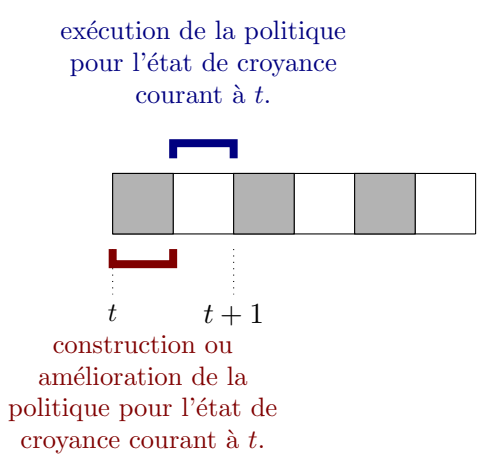
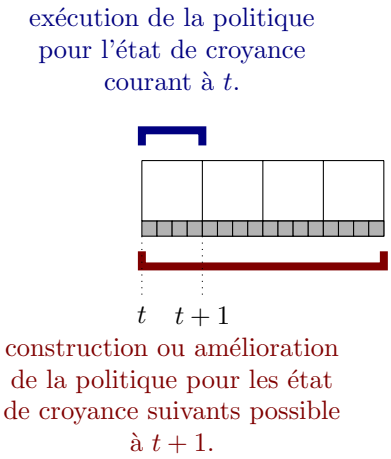


FIGURE 6.2 – Schéma des composants Orocos créés pour la mission robotique “détection et reconnaissance de cibles”.



(a) Approche classique d'optimisation en ligne.



(b) Approche proposée pour l'optimisation en ligne (AMPLE).

FIGURE 6.3 – Schéma représentatif des différences entre les approches de résolution en ligne. Les zones grises représentent le processus d'optimisation.

Notre extension est un méta-planificateur, appelé AMPLE (*Anytime Meta PLannEr*), qui repose sur les algorithmes standard de planification des POMDP, tels que PBVI, HSVI [Smith et Simmons, 2005], RTDP-bel [Bonet et Geffner, 2009], AEMS [Ross et Chaib-Draa, 2007]. Notre cadre permet a priori à n'importe quel algorithme de résolution de POMDP, anytime ou non, d'être utilisé en ligne sous contraintes d'exécution temporelles. Dans ce contexte, l'on peut profiter des spécificités de chaque algorithme.

Dans AMPLE, ces algorithmes sont appelés à partir d'éventuels états d'exécution futurs, lors de l'exécution de l'action optimisée courante pour l'état courant d'exécution, en anticipation de l'évolution du système probabiliste et de son environnement. Notre approche diffère de l'approche classique d'optimisation en ligne qui optimise à chaque fois l'action pour l'état d'exécution courant *avant de pouvoir l'exécuter*. Pour clarifier nos arguments, nous illustrons dans la figure 6.2 la différence de notre approche par rapport à l'approche classique d'optimisation en ligne.

L'idée principale repose sur le fait que nous utilisons deux *threads* indépendants et concurrents : l'un dédié à la planification d'action, qui s'exécute sur le *thread* responsable de l'optimisation, et l'autre dédié à l'exécution des actions, qui s'exécute sur le *thread* responsable de l'exécution de la politique, et qui communique directement avec le reste des composants du système robotique. Ces deux composants communiquent par des requêtes. La figure 6.2 présente le schéma de communication entre les composants d'exécution et de planification.

Dans la suite nous définissons les deux requêtes : de planification et d'action qui sont le coeur de l'approche.

Définition 6.2.1 (Requête de planification) *Une requête de planification est un appel envoyé par le composant d'exécution au composant de planification, qui contient l'état de croyance pour lequel on doit optimiser une action b, le temps alloué à la tâche de planification Δ, et l'algorithme à utiliser alg, ainsi que ses paramètres $param$. Formellement :*

$$\mathcal{R}_p = \langle b, \Delta, alg, param \rangle$$

Définition 6.2.2 (Requête d'action) *Une requête d'action est un appel envoyé par le composant d'exécution au composant de planification, qui contient l'état de croyance courant b pour lequel on récupère l'action optimisée a^*. Si aucune action n'est disponible, une*

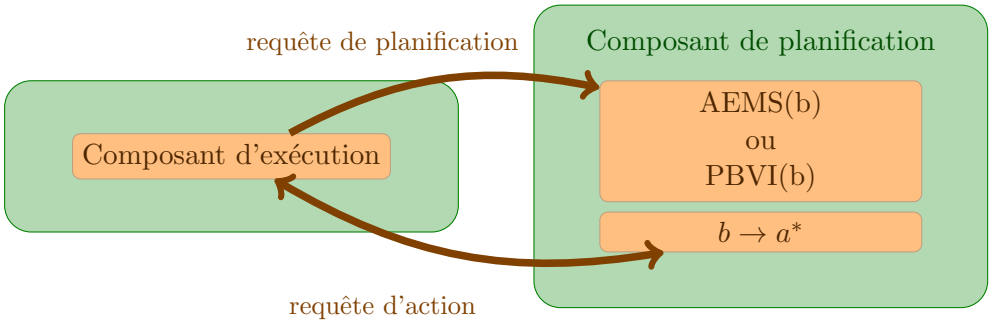


FIGURE 6.4 – Schéma de communication entre les composants d'exécution et de planification. La flèche supérieure représente une requête de planification pour un état de croyance donné. La flèche inférieure représente une requête d'action pour un état de croyance donné. Ces requêtes sont administrés en parallèle par le composant de planification.

action par défaut est envoyée au composant d'exécution.

$$a^* \leftarrow \mathcal{R}_a = \langle b \rangle$$

Il se peut qu'aucune action optimisée ne soit disponible quand le contrôle d'exécution le requière, dans ce cas, l'absence d'action optimisée peut être résolu via une politique par défaut qui dépend de l'application. La politique par défaut peut être générée au tout début de la mission de différentes façons : soit par une politique experte paramétrique calculée hors ligne en préparation de mission dont les paramètres sont adaptés en ligne à la problématique réelle, soit par une politique sous-optimale calculée très rapidement en ligne grâce à des heuristiques avant le calcul de la politique optimale. Le second cas contrairement au premier, nécessite la génération du problème POMDP. Des heuristiques simples mais complètes pour le calcul de politiques pour les POMDP, comme par exemple une approximation QMDP proposée par [Littman *et al.*, 1995], peuvent être rapidement générées.

Grâce à la réactivité, assurée par les actions par défaut, notre approche est une approche strictement *anytime*. Dans le sens que, pendant l'exécution sous contraintes de temps de la politique, elle garantit de retourner une action applicable dans l'état de croyance courant, dans un laps de temps précis, c'est-à-dire au moment où le composant responsable de l'exécution dans l'architecture robotique le demande.

D'autre part, dans des applications réelles, les utilisateurs exigent que certaines garanties de base soient respectées par les systèmes autonomes. Par exemple, dans le cas de notre hélicoptère autonome, l'opérateur exige que la politique exécutée ne puisse jamais mettre l'agent hélicoptère en danger, ce qui peut arriver dans des situations liées par exemple à une panne de carburant. Un autre danger peut survenir en raison de l'absence d'action optimisée pour un état courant d'exécution, si le processus d'optimisation n'a pas encore calculé une action réalisable dans cet état. Pour ces raisons, il est indispensable que le composant de planification puisse fournir immédiatement une action pertinente et applicable à exécuter lorsqu'il est interrogé par le composant d'exécution en fonction de l'état d'exécution courant.

Enfin, la complexité du problème POMDP à résoudre ne permet pas à une majorité de planificateurs de fournir une politique, même sous-optimale dans le temps imparti pour la durée de la mission.

En nous appuyant sur les définitions de requête de planification et de requête d'action, nous présentons maintenant la description détaillée du cadre d'optimisation et d'exécution en parallèle, en débutant par la présentation du composant de planification.

6.2.1 Composant générique de planification adaptée à des algorithmes de l'état de l'art des POMDP

Comme nous l'avons déjà mentionné, notre contribution se ramène à l'extension du cadre d'optimisation en parallèle de l'exécution de politique proposé par [Teichteil-Konigsbuch *et al.*, 2011], auparavant limité à la planification déterministe ou de type MDP, aux modèles de type POMDP soumis à des contraintes de temps. Ce cadre propose un composant de planification générique basé sur des *templates* C++. Ce composant de planification a été conçu pour répondre à des besoins spécifiques et de hauts niveaux comme : *la réactivité* qui garantit qu'une action sera retournée au composant d'exécution quand il le demande ; *la généricité* qui assure que différents algorithmes de planification pour des domaines variés peuvent être adaptés au cadre d'optimisation et d'exécution en parallèle ; et *la validation* qui permet de vérifier formellement la validité de la politique par défaut ainsi que celle qui est optimisée.

Sur la figure 6.5, la machine à états du composant générique de planification de [Teichteil-Konigsbuch *et al.*, 2011] est montrée. Cette machine à états définit les différents états d'exécution du composant de planification (différents des états du problème de planification), et les transitions entre ces états en fonction des requêtes provenant des autres composants de l'architecture. En interne, ces transitions sont définies par des méthodes dans des classes *templates* C++ qui doivent être implémentées selon l'utilisation que l'on veut faire du composant (planification déterministe, ou stochastique, ou partiellement observable, ou temporelle, etc).

Dans notre cas, chaque méthode à été implémentée de façon à résoudre des problèmes POMDP avec des algorithmes de l'état de l'art de ce domaine, tels que PBVI [Pineau *et al.*, 2003] ou AEMS [Ross et Chaib-Draa, 2007]. PBVI est un algorithme *anytime*[3], qui réalise des itérations approchées sur la valeur pour des états de croyance b qui appartiennent à l'ensemble $\mathcal{B}$. Cet ensemble est construit par des simulations dynamiques et stochastiques du POMDP. AEMS est un algorithme de résolution en ligne, basé sur des simulations en profondeur de la dynamique du POMDP guidées par des heuristiques. Les heuristiques amènent l'algorithme à mettre à jour la valeur des états de croyance qui réduisent au plus l'erreur entre les bornes supérieure et inférieure de la valeur à l'état de croyance courant (racine de l'arbre d'états de croyance). Ces algorithmes ont été présentés dans le chapitre 2, si bien que nous ne nous attarderons pas à les rappeler ici. Nous tenons tout de même à noter que la version de AEMS que nous avons implémentée correspond à la version AEMS2 de [Ross et Chaib-Draa, 2007], qui utilise un choix glouton d'actions (basé sur la borne supérieure) pour la recherche en profondeur.

L'état initial du composant est l'état *Waiting Problem* (voir figure 6.5). Dans cet état d'exécution, le composant est initialisé et charge le problème POMDP à résoudre. Une fois que le problème a été chargé, le composant de planification peut recevoir des requêtes de planification par la méthode *add_plan_request* (possiblement en parallèle). Les requêtes de planification peuvent être annulées avec la méthode *remove_plan_request*. Si l'état du composant est *Planning* ou *Problem Solved*, il est toujours possible de récupérer à tout moment et en parallèle : les actions applicables (*get_actions*) ; l'action planifiée pour un état de croyance donné (*get_action* qui traduit la requête d'action) ; ou encore, les effets (*get_effects*) d'une action sachant un état de croyance donné. Dans le cadre POMDP, nous considérons les observations comme effets des actions. Le composant de planification peut être arrêté à tout

3. PBVI est considéré par ses auteurs comme *anytime* puisqu'à chaque itération il fournit une approximation de la fonction de valeur. Mais cela ne signifie en aucune manière que PBVI peut fournir une action optimisée dans l'état de croyance courant à *n'importe quel* instant, contrairement à notre approche. AMPLE couplé à PBVI permet en quelque sorte de rendre PBVI réellement *anytime*, avec l'aide ponctuelle d'une politique par défaut.

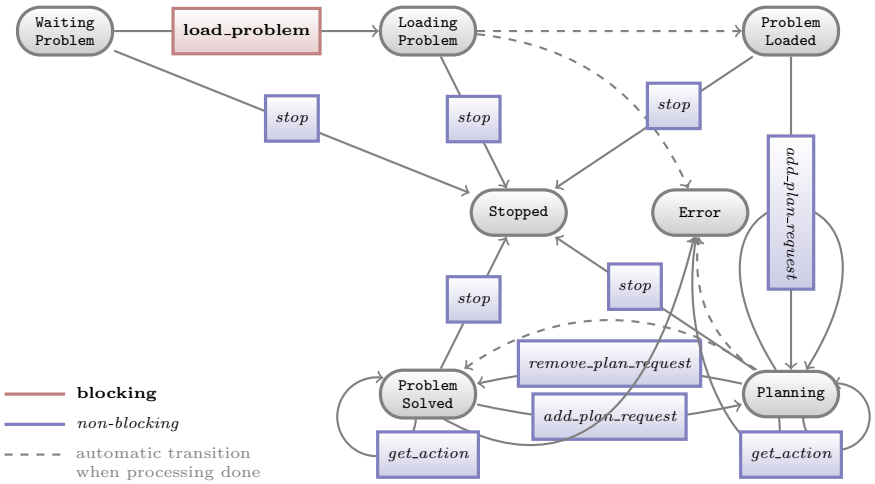


FIGURE 6.5 – Machine à états du composant générique de planification proposé par [Teichteil-Konigsbuch *et al.*, 2011]

moment par une requête de type *stop*.

Notons qu'aucune requête est bloquante, c'est-à-dire que toutes les requêtes retournent un booléen qui indique si la requête a été prise en compte. Si des calculs sont nécessaires, par exemple au moment du chargement d'un problème, ou au moment de la planification pour un état de croyance donné, la machine à états entre dans un état qui indique ce que le composant est en train de faire (*Loading Problem* ou *Planning*). Dès que le calcul est terminé, la machine à états bascule automatiquement vers un autre état, indiquant que la tâche est finie (par exemple *Problem Loaded* ou *Problem Solved*). Ces états d'exécution sont lus par les autres composants de l'architecture robotique, notamment le composant de supervision qui est le chef d'orchestre de la mission et qui coordonne l'exécution de tous les composants (planification, traitement d'image, navigation, etc.).

Le composant de planification est toujours réceptif en ce qui concerne les différentes requêtes, sauf quand il est en train de charger un problème à résoudre. Pour les autres cas, le composant est capable de gérer des requêtes et de résoudre le problème de planification en parallèle. Ceci requiert une protection et une recopie partielle de la politique qui a été calculée ou qui est en train d'être calculée à l'écriture et à la lecture. Il est important de noter que la politique locale est constamment mise à jour au fur et à mesure que des requêtes de planification sont traitées. Cette protection est gérée par des *mutex*. Ce mécanisme complexe est en fait un algorithme en soi qui est détaille dans [Teichteil-Konigsbuch *et al.*, 2011].

Dans la suite, nous présentons le composant d'exécution qui nous permet de tirer profit du composant générique de planification adapté à des algorithmes de résolution des POMDP. Pour notre mission, nous optimisons en anticipation des politiques en même temps que nous les exécutons. Le composant de planification est suffisamment flexible pour permettre des utilisations bien différentes (par exemple, cadre classique d'optimisation puis d'exécution).

6.2.2 Fonctionnement du cadre d'optimisation anticipée et d'exécution en parallèle : AMPLE

Nous détaillons ici comment AMPLE gère les requêtes de planification et d'exécution en parallèle :

1. Initialement, AMPLE envoie une requête de planification au composant de planification pour un état de croyance initial b à l'aide de l'algorithme de résolution de POMDP

sous-jacent, pendant un temps fixé et généralement très court par rapport à la durée de la mission. On appellera cette phase, qui permet d'obtenir une première action optimisée à exécuter, de *bootstrap*.

2. Ensuite, le composant de planification reçoit une requête d'action pour laquelle il retourne au composant d'exécution l'action optimisée pour b.
3. Le temps estimé d'exécution de l'action en cours d'exécution est retournée. Ensuite, le composant d'exécution envoie au composant de planification des requêtes de planification pour certains états de croyance futurs possibles en répartissant le temps de planification selon le temps d'exécution estimé de l'action courante.
4. Une fois l'action courante exécutée, une observation est reçue et l'état de croyance est mis à jour, ce qui amène à un nouvel état de croyance b' pour lequel l'action courante optimale est renvoyée par le composant de planification au composant d'exécution via une requête d'action.

À noter que notre cadre d'optimisation anticipée et d'exécution en parallèle propose un algorithme de planification en continu qui prend pleinement en charge les incertitudes probabilistes : il construit plusieurs morceaux de politique pour différents états probables d'exécution futurs. Toutefois, de part sa nature, cet algorithme n'a aucune garantie d'optimalité. Dans cette approche, nous nous préoccupons de réussir au mieux la mission dans le temps alloué à cette mission, au détriment des garanties d'optimalité. Pour les applications robotiques qui nous intéressent on ne peut pas faire mieux, nous choisissons de sacrifier l'optimalité afin de réussir la mission dans le temps imparti.

Choix des états de croyance suivants possibles et du temps de planification associé

Chaque requête de planification est définie par l'état de croyance pour lequel on planifie et par le temps alloué à cette tâche de planification. De cette façon, les requêtes de planification pour les états de croyance futurs, nécessitent le calcul des ces états de croyance futurs. Pour cela, nous demandons au composant de planification quels sont les effets probabilistes de l'action en cours d'exécution.

Nous considérons les observations comme effets d'une action. Et, à partir des observations probables liées à la réalisation d'une action, nous calculons les états de croyance suivants par observation à partir de la règle de Bayes (équation 2.6 du chapitre 2). Toutefois, le temps que nous allouerons à la tâche de planification doit aussi être défini.

La définition de ce temps peut être réalisée de différentes manières :

- nous pourrions fixer un temps égal de planification pour chaque état de croyance suivant ;
- nous pourrions allouer tout le temps de planification à l'état de croyance le plus probable ;
- nous pourrions donner la priorité à l'état de croyance pour lequel l'estimation de la valeur est la plus incertaine ;
- ou encore, nous pourrions allouer un temps de calcul à chaque état de croyance suivant de manière proportionnelle à sa probabilité d'occurrence.

Nous avons choisi de définir le temps de calcul d'un état de croyance suivant de manière proportionnelle à sa probabilité d'occurrence. En effet, il nous semble cohérent d'attribuer une priorité, sous forme temporelle, proportionnelle à la probabilité de la prochaine observation. Ce calcul est fait avec l'aide de la probabilité $p(o|a,b) = \sum_{s' \in S} p(o|s',a) \sum_{s \in S} p(s'|s,a) b(s)$. Notons que ce calcul dépend de l'état de croyance courant et des fonctions de transition et d'observation du modèle.

Dans la prochaine section, nous présentons le pseudo-code de notre algorithme AMPLE. Comme déjà mentionné, nous le considérons comme continu puisqu'il construit plusieurs

Algorithme 10: Cadre d'optimisation anticipé et d'exécution en parallèle – AMPLE

```
1  b ← état de croyance initial;
2  Envoie d'une requête_de_planification(b,temps de bootstrap, algorithme) ;
3  Attendre la fin du temps de bootstrap ;
4  while requête d'arrêt d'AMPLE = false do
5      a ← requête_d'action(b);
6      Exécution de l'action a;
7      Δ_a ← estimation de la durée de l'action a;
8      O ← obtention des effets probabilistes de l'action a (observations possibles) ;
9      for 0 < i < |O| do
10         b' ← calcul du prochain état de croyance par la règle de Bayes ;
11         Δ' ∼ Δ_a * p(o|a, b) ;
12         Envoie d'une requête_de_planification(b', Δ', algorithme);
13     Attendre que l'exécution de l'action a soit finie;
14     Réception de l'observation o;
15     Retirer toutes les requêtes de planification;
16     b ← mise à jour de l'état de croyance courant avec a et o ;
```

morceaux de politique pour les différents états de croyance probables et futurs, suivant la méthodologie précédemment présentée.

6.3 Mise en oeuvre du cadre d'optimisation anticipée et d'exécution en parallèle

Notons qu'AMPLE est conduit par un superviseur de mission (voir *AmpleExecute* figure 6.2) qui peut être mis en place différemment selon l'architecture utilisée. Nous pensons que le cadre d'optimisation anticipée et d'exécution en parallèle est un cadre général, et qui offre une approche alternative et originale pour résoudre de problèmes en ligne, en particulier des problèmes POMDP. Dans ce but, nous présentons par la suite le pseudo-code de notre algorithme AMPLE et après la mise en place du superviseur de la mission robotique étudié dans ce chapitre.

6.3.1 Algorithme

Dans l'algorithme 10 nous présentons le pseudo-code de notre approche. Initialement, le composant d'exécution envoie au composant de planification une première requête de planification pour l'état de croyance initial (ligne 2). Nous avons appelé cette phase la phase de *bootstrap*. Après le temps de *bootstrap*, le composant d'exécution envoie une requête d'action au composant de planification, dans le but de récupérer l'action optimisée pour l'état de croyance courant (ligne 5). Ensuite, pendant l'exécution de l'action courante, le composant d'exécution anticipe l'évolution stochastique de l'environnement, par le calcul des prochains états de croyance, et envoie au composant de planification des requêtes de planification pour chaque état de croyance futur avec un temps alloué proportionnel à $p(o|a, b)$ (lignes 7 à 13). Une fois l'action exécutée, le composant d'exécution met à jour l'état de croyance courant avec l'observation reçue (ligne 16), et renvoie une requête d'action pour le nouvel état de croyance. Et ainsi de suite jusqu'à l'arrêt d'AMPLE à la fin de la mission.

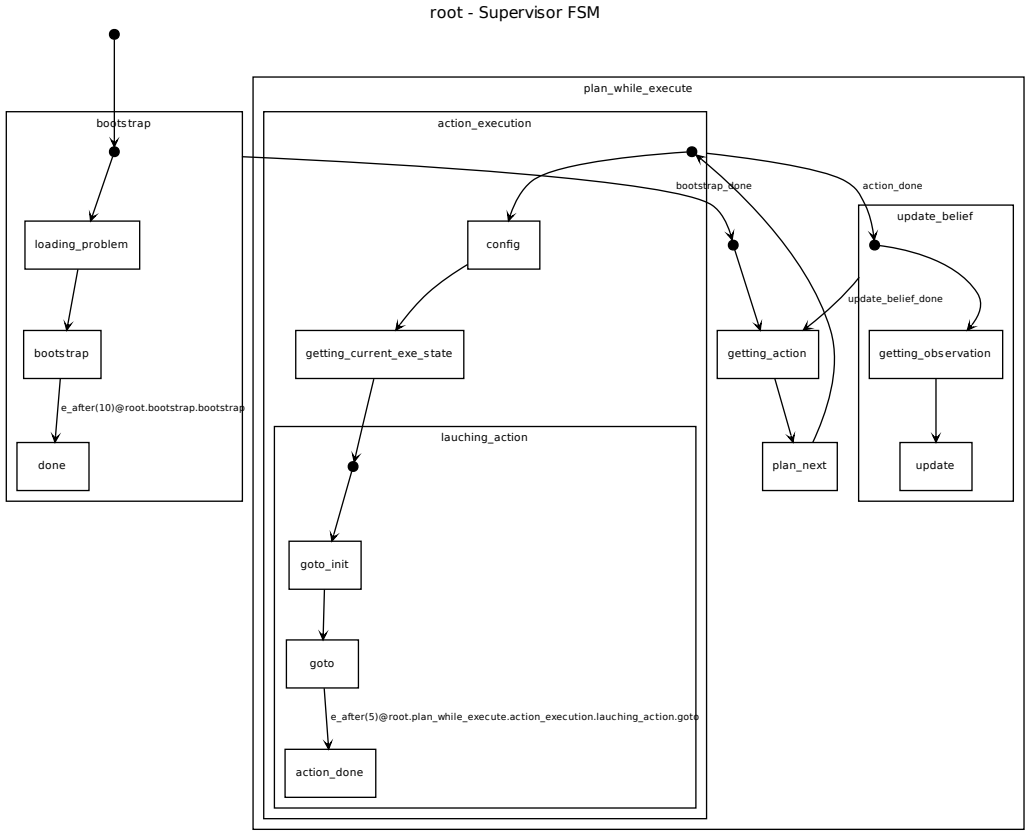


FIGURE 6.6 – Superviseur de mission : machine à états du composant *AmpleExecute*.

6.3.2 Mise en œuvre du superviseur pour la mission

Pour le contrôle d'exécution de la mission robotique il a été nécessaire de mettre en place un superviseur de mission. Le superviseur est le composant de l'architecture Orocos qui assure la coordination des différents composants de la mission. Le composant de supervision (composant *AmpleExecute* dans la figure 6.2), a été implémenté dans le language script lua [4]. Ce composant est une machine à états qui gère la coordination entre tous les composants nécessaires à la mission.

La figure 6.6 montre le schéma général de la machine à état du composant *AmpleExecute*. Le fonctionnement du superviseur est expliqué par la suite.

Après une phase d'initialisation (non montrée dans la figure 6.6), où la génération automatique du problème est effectuée, la machine à état bascule vers l'état *bootstrap* : le problème POMDP est chargé et la phase de *bootstrap* est lancée. Ensuite, une fois le temps de *bootstrap* écoulé, la machine à états bascule vers l'état *plan_while_execute*. Cet état gère l'envoie des requêtes d'action et de planification. Une requête d'action est d'abord envoyé au composant *AmpleOptimize*, puis, une fois l'action récupérée, des requêtes de planification pour les prochains états d'exécution sont aussi envoyées au composant *AmpleOptimize*. Dès que l'action courante est récupérée, la machine à états bascule vers une sous-machine qui gère l'envoie des `go_to` à la navigation, soit des *way-points* au composant *Obc*. Une fois que l'exécution de l'action est terminée, la mise à jour de l'état de croyance courant est faite à partir de l'observation obtenue du composant *Matching*. Ensuite une nouvelle requête d'action est envoyée à *AmpleOptimize* et ainsi de suite.

Ce composant superviseur nous permet donc de mettre en place notre cadre d'optimisa-

4. http ://www.orocos.org/wiki/orocos/toolchain/luacookbook

tion en parallèle de l'exécution, ainsi que les communications nécessaires entre les différents composants de navigation et de traitement d'information.

Dans la prochaine section, nous présentons les résultats obtenus lors du vol expérimental. Et après, nous présentons les résultats obtenus par des simulations réalistes, où nous avons utilisé l'architecture fonctionnelle embarquée et les algorithmes utilisés à bord de notre hélicoptère autonome.

6.4 Vol expérimental : mission de détection et de reconnaissance de cibles par un drone hélicoptère autonome

Pendant nous travaux de thèse, nous avons effectué un vol expérimental avec l'hélicoptère autonome Yamaha Rmax de l'Onera (montré sur la figure 6.7). Nous présenterons dans cette section les résultats obtenus pendant ce vol, où nous avons embarquée notre cadre d'optimisation anticipée et exécution en parallèle. Nous allons par la suite décrire la configuration de l'environnement lors de ce vol.

FIGURE 6.7 – Photo de l'hélicoptère Yamaha Rmax de l'Onera adapté au vol autonome.

La configuration expérimentale suit le schéma présenté sur la figure 6.8. Nous considérons 2 altitudes de vol (30 et 40 mètres), 3 modèles de voitures et 3 zones, ce qui nous ramène à 385 états. Les zones sont alignées, et la distance entre les zones est de 70 mètres. Le but est d'atterrir à côté du modèle de voiture C, par contre le nombre et la nature de cibles effectivement dans la scène est inconnu de l'hélicoptère autonome au début de la mission, ceci est modélisé par une croyance initiale uniforme sur les possibles états initiaux, conforme :

$$b_0 = \{z = 1, h = 30, Id_{Ta_{z_1}} = inconnu, Id_{Ta_{z_2}} = inconnu, Id_{Ta_{z_3}} = inconnu\}.$$

ainsi, b_0 correspond à une distribution de probabilité uniforme sur les 64 états initiaux possibles.

Notons que l'état de croyance initial uniforme modélise la méconnaissance subjective de l'agent par rapport à l'état caché du système. Cet état de croyance subjectif n'est pas une approximation fréquentielle des états initiaux possibles. Ce choix résulte du fait que l'hélicoptère autonome ne connaît rien sur l'environnement initialement. Mais est-ce que l'équiprobabilité est satisfaisante pour représenter la méconnaissance totale ? Nous pensons qu'on ne peut pas faire mieux avec une modèle POMDP.

Des voitures ont été mises en place dans le terrain. La configuration, c'est-à-dire l'état caché du système était le suivant : aucune voiture dans la zone 1, une voiture de modèle A dans la zone 2 et une voiture de modèle C dans la zone 3. Le temps de *booststrap* a été fixé à 30 secondes, la durée de chaque action est représentée par une distribution uniforme

sur $[T^a_{min}, T^a_{max}]$, avec $T^a_{min} = 8s$ et $T^a_{max} = 10s$. Ces valeurs nous sont nécessaires pour que AMPLE puisse estimer la durée d'exécution des actions. La politique par défaut utilisée est le résultat de l'approximation QMDP [Littman *et al.*, 1995], calculée au début de la mission. L'algorithme de résolution de POMDP utilisé lors de cette expérimentation a été l'algorithme AEMS [Ross et Chaib-Draa, 2007].

Nous avons choisi d'implémenter notre cadre d'optimisation en parallèle de l'exécution avec l'algorithme AEMS [5] parce que nous avons pu constater lors de simulations (présentées dans la prochaine section) qu'AEMS est particulièrement adapté à notre approche. Avec AEMS, nous pouvons arrêter le calcul d'une action pour un état de croyance donné quand l'on veut, contrairement à PBVI par exemple, qui ne peut être arrêté que à la fin d'une mise à jour de la valeur pour l'ensemble complet $\mathcal{B}$. Nous tenons, tout de même, à attirer l'attention sur le fait que notre but n'est pas d'améliorer des algorithmes existants, mais de les incorporer dans un cadre plus flexible, qui traite des requêtes de planification de manière réactive comme toutes les autres fonctionnalités embarquées sur l'hélicoptère autonome. Ceci nous semble primordial, afin de permettre à des techniques d'intelligence artificielle, telles que les POMDP, d'être utilisées dans des applications robotiques réelles. En effet, il existera toujours des applications suffisamment complexes qui ne pourront pas être résolues dans le temps imparti de la mission avec les meilleures méthodes existantes.

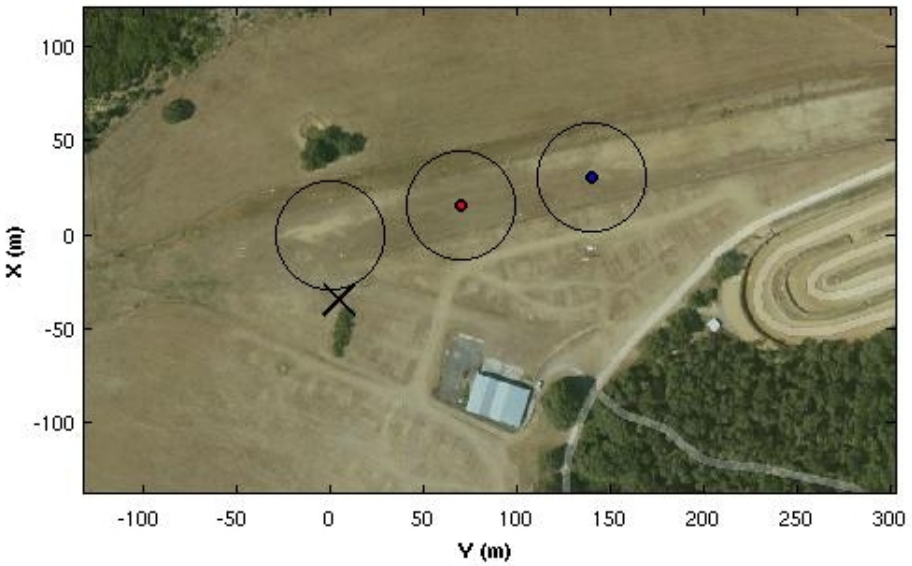


FIGURE 6.8 – Schéma représentatif de l'état caché sur le terrain pour le vol expérimental : la croix représente la position initial de l'agent hélicoptère, le point rouge représente la position de la cible de modèle A, et le point bleu, la position de la cible de modèle C (cible recherchée).

Comme décrit dans la section 6.1.1, lors de la génération automatique du problème POMDP à résoudre, la fonction de récompense du modèle est basée sur la distance entre le centre des zones. Les zones sont supposées alignées sur la piste du terrain d'essai, et la distance entre deux zones est de 70 mètres. Le changement d'angle de vue, est fait par de pas de 10 dégrées (voir angle ϕ dans la modélisation de l'action *change view* dans le chapitre 3).

La architecture Orocos utilisé pour ce vol a été montrée dans la section 6.1.2, cette

5. AEMS a les bornes inférieure et supérieure initialisées respectivement par une approximation par politique myope et par une approximation de type QMDP (cf. section 2.3 du chapitre 2) dans notre implémentation.

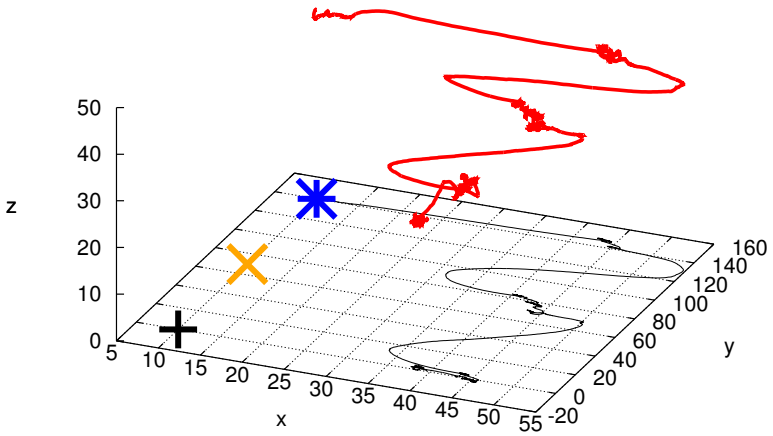


FIGURE 6.9 – Trajectoire de l'agent hélicoptère pendant le vol expérimental (en rouge). La croix noire représente la zone 1, la croix orange la zone 2, et l'étoile bleu la zone 3, zone dans laquelle la cible se trouve. La fine ligne en noire montre la projection au niveau du sol de la trajectoire réalisée pendant le vol : la cible correct a donc pu être trouvé par l'hélicoptère.

architecture nous permet de générer automatiquement le problème POMDP à résoudre, une fois que le nombre de zones et le modèle de cible recherché ont été définis. Elle nous permet aussi via le superviseur de mission (*AmpleExecute* dans la figure 6.2) de mettre en place le cadre d'optimisation anticipée et d'exécution en parallèle développé.

6.4.1 Résultats

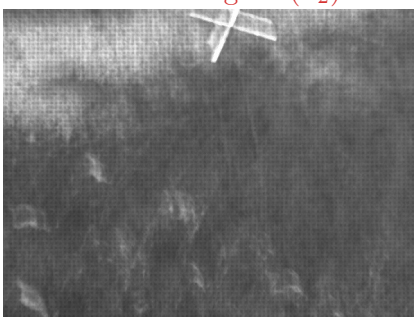
La trajectoire qui a été effectuée par l'hélicoptère autonome est illustrée sur la figure 6.9. Les coordonnées sont exprimées dans le repère local du terrain d'essai de l'agent hélicoptère.

La figure 6.10 montre la séquence d'actions réalisée par l'agent hélicoptère autonome lors du vol expérimental. Initialement, l'agent hélicoptère est dans la zone 1 à une altitude de vol de 30 mètres. Dans les deux premières étapes, l'agent observe qu'aucune cible n'est dans la zone 1, puis choisit d'aller vers la zone 2. Après trois étapes de décision, l'agent observe que la cible dans la zone 2 est celle de modèle A, il décide donc d'aller vers la zone 3. Après 2 observations supplémentaires, l'agent hélicoptère croit (à raison) que la cible dans la zone 3 est le modèle C, qui est le modèle recherché, et décide donc d'atterrir, terminant, ainsi, correctement sa mission. Notons que la trajectoire de l'agent, montrée sur la figure 6.9, ne se termine pas à la hauteur du sol : ceci est dû au fait que, pour ce vol expérimental, l'agent hélicoptère n'a pas atterri, en réalité pour des raison de sécurité dues à l'expérimentation, mais il a en fait survolé la cible (action *land*). L'atterrissage automatique est en effet risquée.

Cet exemple impliquait peu d'états, mais nous attirons l'attention sur le fait que la réussite et l'intérêt d'une telle mission de robotique autonome repose avant tout sur l'expressivité de la fonction d'observation et non nécessairement sur la taille du problème à résoudre. Dans de telles applications robotiques, la précision du modèle d'observation liée à l'algorithme de traitement d'image et la réactivité des requêtes de planification sont primordiales, comme montré dans ce chapitre. Nous pensons que notre approche supportera le passage à l'échelle, puisque si l'algorithme sous-jacent ne fournit pas d'action optimisée dans le temps, des actions seront toujours retournées en temps et en heure, s'il le faut des actions par défaut. Donc, dans notre approche comme dans beaucoup d'approches de planification en ligne, le passage à l'échelle est aussi impacté par la durée des actions.

Dans la prochaine section nous présentons les résultats obtenus par des simulations

$t = 0 : a =$ go_to(h_2)



$o =$ détecté non identifié.

$t = 1 : a =$ change_view

$o =$ non détecté

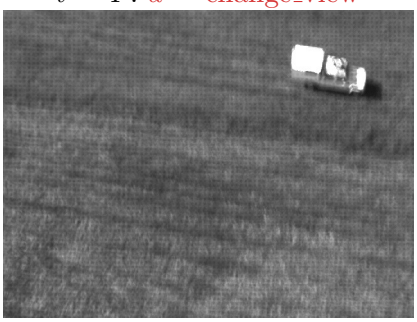
$t = 2 : a =$ goto_zone(z_2)

$o =$ modèle A

$t = 3 : a =$ change_view

$o =$ modèle A

$t = 4 : a =$ change_view

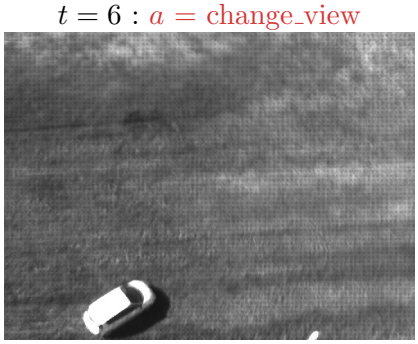


$o =$ modèle A

$t = 5 : a =$ go_to(z_3)

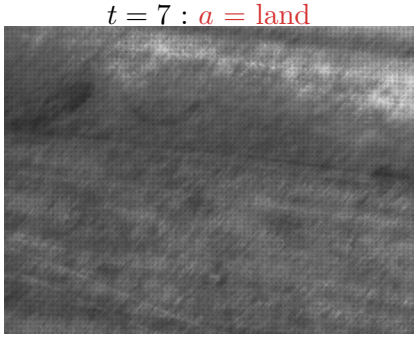
$o =$ modèle C

$t = 6 : a =$ change_view



$o =$ modèle C

$t = 7 : a =$ land



$o =$ non détecté

FIGURE 6.10 – Séquence d'étapes de décision. Chaque image représente l'image d'entrée reçu par l'algorithme de traitement d'image après réalisation d'action. Les observations représentent les réponses successives de la classification donnée par le traitement d'image.

réalistes, où nous avons utilisé l'architecture fonctionnelle embarquée et les algorithmes utilisés à bord de notre hélicoptère autonome. Ces simulations ont été mènes afin de : (1) comparer des différents algorithmes sous-jacents de résolution de POMDP, et (2) démontrer l'intérêt de l'approche alternative que nous proposons pour l'optimisation anticipée et l'exécution en parallèle vis-à-vis de l'approche classique de résolution en ligne.

6.5 Évaluation par simulation de l'approche d'optimisation anticipée et d'exécution en parallèle

Pour l'évaluation de l'approche en simulation, nous avons réalisé des simulations réalistes et complètes (*hardware in the loop*), c'est-à-dire en utilisant l'architecture fonctionnelle embarquée et les algorithmes utilisés à bord de notre hélicoptère autonome (Yamaha Rmax adapté au vol autonome, voir figure 6.7). Ces simulations réalistes sont très importantes et constituent une étape indispensable. Elles nous permettent d'évaluer l'approche de planification en ligne intégrée à l'architecture fonctionnelle réellement embarquée sur le drone hélicoptère, et de déboguer le fonctionnent du système entier. Ainsi, nous pouvons nous assurer du bon fonctionnement de la procédure avant un vol expérimental. De plus, les vols réels coûtent chers et leur mise en place peu être longue.

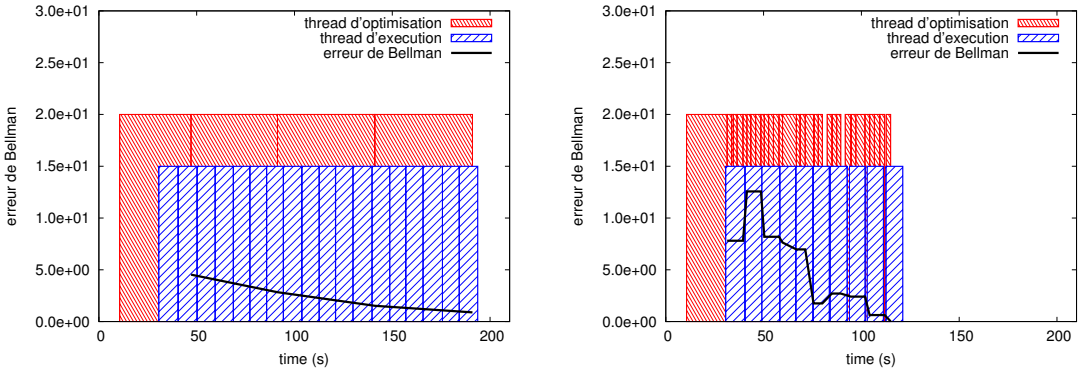
Comme nous l'avons mentionné, et montré lors de l'expérimentation en vol, nos observations sont caractérisées par les probables sorties du traitement d'image basé sur l'algorithme de [Saux et Sanfourche, 2011]. Le traitement d'image dans l'architecture est géré par deux composants *Detection* et *Matching*, eux-mêmes gérées par le composant *AmpleExecute*. Lors de nos simulations réalistes, *AmpleExecute* lance le traitement d'image dès qu'une action est exécutée. Ainsi, le simulateur, qui connaît l'état du système, prend une image de la base de données (qui correspond à l'état courant) et envoie aux composants de traitement d'image *Detection* puis *Matching*, le dernier retournant l'observation.

Suite à cette introduction sur le simulateur réaliste que nous avons mis en place, nous présentons une analyse approfondie des résultats obtenus lors de ces simulations.

Comparaison entre deux algorithmes sous-jacents : PBVI et AEMS

Tout d'abord nous cherchons à analyser l'impact de l'algorithme de résolution de POMDP sous-jacent utilisé. Pour cela, nous comparons deux algorithmes de l'état de l'art : PBVI [Pineau *et al.*, 2003] et AEMS [Ross et Chaib-Draa, 2007]. PBVI, qui réalise des itérations approchées sur la valeur pour des états de croyance b qui appartiennent à l'ensemble $\mathcal{B}$, à chaque itération fournit une approximation de la fonction de valeur. Cet ensemble est construit par des simulations dynamiques et stochastiques du POMDP, ainsi nous pourrions espérer que PBVI soit adaptée à l'application étant donné que plusieurs états de croyance, qui pourraient être potentiellement retrouvés lors de l'exécution de la politique, appartiennent à $\mathcal{B}$. AEMS est un algorithme de résolution en ligne, basé sur des simulations en profondeur de la dynamique du POMDP guidées par des heuristiques. AEMS nous semble particulièrement adapté à notre approche, parce que nous pouvons arrêter le calcul d'une action pour un état de croyance donné quand l'on veut.

Nous pensons que le cadre AMPLE proposé est une option raisonnable pour ce genre d'application robotique (section 6.1), car : (1) le nombre de zones est découvert pendant le vol, rendant impossible la résolution du problème avant vol ; et, (2) les algorithmes de résolution utilisés pour le POMDP – PBVI et AEMS – ne convergent pas dans la durée limite de la mission. PBVI et AEMS ont besoin de plus d'une heure de calcul pour converger selon le précision demandée ($\epsilon = 0.5$ par exemple), sachant la puissance du processeur embarqué sur les drones (Intel Core Duo 2, 1,5GHz et 4G de mémoire RAM).



(a) Frise chronologique pour PBVI.

(b) Frise chronologique pour AEMS.

FIGURE 6.11 – Frise chronologique pour les implémentations de PBVI et AEMS dans le cadre AMPLE en partant de b_0.

Pour ces simulations, nous avons considéré la même instanciation du problème que celle du vol expérimental (voir figure 6.8). De plus, pour cette analyse par simulation réaliste, nous avons considéré le même état initial caché que celui de l'expérimentation en vol (section 6.4). Notons que pour une évaluation exhaustive, il faudrait tirer les états initiaux selon b_0. Nous avons choisi de fixer l'état initial pour : (1) évaluer la politique optimisée en ligne vis-à-vis seulement de la dynamique probabiliste des observations pouvant être obtenues lors d'un vol réel ; (2) évaluer le comportement de la politique en partant d'une distribution de probabilité uniforme, qui encode la méconnaissance subjective de l'agent hélicoptère par rapport à la nature de l'environnement statique qui l'entoure. L'état initial de l'environnement n'est pas connu de l'hélicoptère autonome, mais en réalité il est fixé (selon expérimentation) et non tiré aléatoirement.

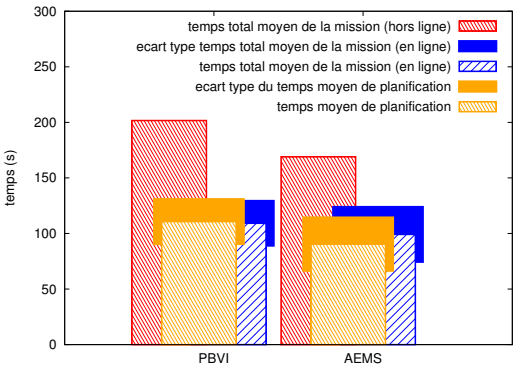
Pour ces simulations, la durée de chaque action est représentée par une distribution uniforme sur $[T^a_{min}, T^a_{max}]$, avec $T^a_{min} = 8s$ et $T^a_{max} = 10s$. Le temps de *bootstrap* est de 20 secondes, et la politique par défaut utilisée est le résultat de l'approximation QMDP [Littman *et al.*, 1995], calculée au début de la mission. A noter que le calcul de la politique par défaut prend moins de 10 seconds.

La figure 6.11 montre les frises chronologiques de temps pour les processus d'exécution et optimisation d'AMPLE pour une simulation. Ces diagrammes temporels représentent les périodes de temps où la politique a été optimisée (*thread* d'optimisation) ou exécutée (*thread* d'exécution) – les deux en parallèle –, ainsi que l'évolution de l'erreur de Bellman pendant la durée de la mission. Après une phase d'initialisation (*bootstrap*), où seulement le *thread* d'optimisation est actif, nous pouvons noter que le processus d'optimisation se poursuit pendant quelque temps par des petites phases d'optimisation qui sont encore effectuées lorsque des nouvelles requêtes de planification sont envoyées au composant de planification ; En effet, la politique n'a pas été entièrement optimisée dans l'état de croyance concerné pendant les étapes d'optimisation précédentes.

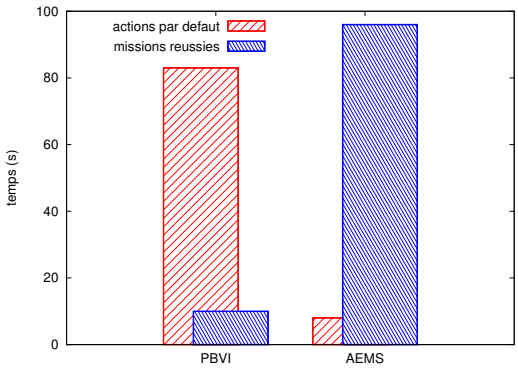
L'évolution de l'erreur de Bellman, qui est rapportée par le planificateur à chaque requête de planification lors de l'optimisation, montre la convergence temporelle du processus d'optimisation (éventuellement vers une politique sous-optimale). Dans la figure 6.11(a) la fonction de valeur ne converge pas pour l'ensemble complet des états de croyance concernés (300 dans cette simulation), contrairement à 6.11(b) où le processus d'optimisation a convergé pour l'état de croyance glissant. La conclusion est qu'AEMS est plus réactif que PBVI, du fait qu'il a suffisamment de temps pour optimiser les états futurs possibles de croyance pendant

l'exécution de l'action courante. Nous pouvons remarquer que pour AEMS les processus d'exécution et d'optimisation s'arrêtent dû au fait que l'agent hélicoptère a finalisé sa mission.

La figure 6.12 montre des résultats relatifs au temps de planification et au pourcentage de missions réussies, pour les 2 sous-planificateurs PBVI et AEMS utilisés dans le cadre d'optimisation en parallèle de l'exécution - AMPLE : le temps total moyen de la mission en ligne représente le temps jusqu'à la fin de la mission, c'est-à-dire jusqu'à l'atterrissage, puisque cette moyenne a été calculée uniquement sur les missions réussies ; le temps moyen de planification représente le temps consacré à l'optimisation, qui est très proche du temps total de mission pour l'algorithme PBVI, puisqu'il ne converge pas avant la fin de la mission. Ces moyennes ont été calculées à partir de 50 tests pour chaque instance du problème. Chaque essai constituait une mission complète : optimisation et exécution en parallèle à d'aucune connaissance préalable. De plus, nous montrons également le temps d'une mission dite hors ligne, pour laquelle la politique serait d'abord optimisée pendant un certain temps, encore pendant le vol après l'extraction des zones de l'environnement, et puis finalement exécutée.



(a) Temps de moyen de planification



(b) Pourcentages de réussite de mission et d'actions par défaut.

FIGURE 6.12 – Résultats moyennés pour les implémentations de PBVI et d'AEMS dans le cadre AMPLE partant de b_0.

En examinant les pourcentages de réussite de mission et d'actions par défaut utilisées, nous pouvons remarquer que AMPLE permet de globalement mener à bien la mission, même avec un algorithme pas très performant comme PBVI. AEMS contrôlé par AMPLE a une bonne performance, contrairement à PBVI qui a une performance très pauvre dû au fait qu'il est moins réactif qu'AEMS. Dans le cas de PBVI, qui ne fournit pas d'action optimisée pour de nombreux états de croyance dans le temps imparti, la politique par défaut est quasiment utilisée tout le temps. La politique par défaut, étant une politique heuristique de type QMDP, calculée au début de la mission, fournit une action de très mauvaise qualité pour ce problème.

La figure 6.13 présente l'évolution temporelle de l'espérance de la somme pondérée des récompenses cumulées pendant les 50 tests d'exécution de la politique, calculées par :

$$V^{\pi}(s_t) = \frac{1}{50} \sum_{50} \left[\sum_{k=0}^{t} \gamma^k r(s_k, \pi_k) \middle| b_0, s_k \right].$$

A noter que le simulateur utilise la connaissance de l'environnement (soit s_t, et tous les s_k), pour attribuer les récompenses pendant la simulation. De plus, cette équation nous permet de montrer la moyenne des récompenses accumulées à partir de l'instant zéro jusqu'à l'instant t. Pour PBVI les récompenses moyennes recueillies au cours des exécutions de politique tendent

à être moins importantes que pour AEMS. La performance de PBVI se dégrade dans le temps, ceci est dû au fait de l'utilisation des actions par défaut, qui sont fréquemment utilisées par PBVI et qui ne sont pas optimales pour les états de croyance dans lesquels elles ont été appliquées.

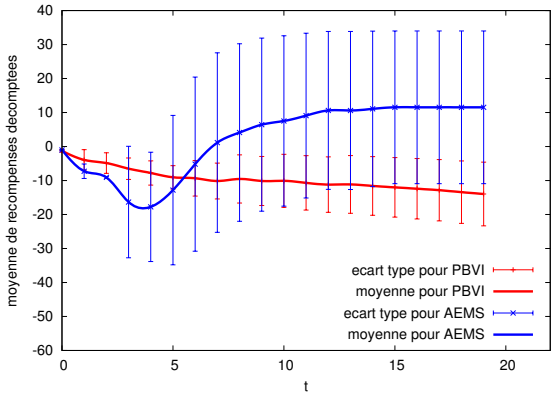


FIGURE 6.13 – Espérance de la somme pondérée des récompenses cumulées pendant les 50 tests pour les implémentations de PBVI et d'AEMS dans le cadre d'optimisation anticipée et d'exécution en parallèle AMPLE en partant de b_0.

Pour enrichir l'analyse et démontrer l'intérêt de l'approche proposée, nous avons aussi comparé le cadre d'optimisation anticipée et exécution en parallèle avec le cadre classique d'optimisation en ligne sur la base de l'algorithme AEMS. L'approche classique (montrée sur la figure 6.3(a)), qui consiste à entrelacer l'optimisation et l'exécution ne planifie pas directement pour les états de croyance futurs, c'est-à-dire qu'elle planifie à chaque fois pour l'état de croyance courant pendant un certain temps puis exécute la meilleure l'action courante.

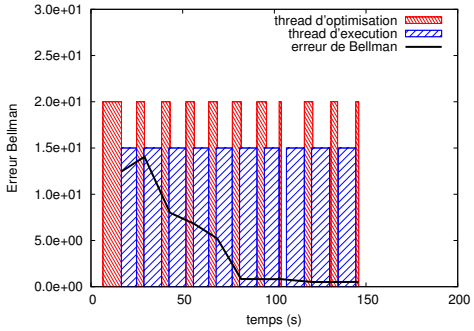
Comparaison de la version classique d'optimisation en ligne *versus* AMPLE pour l'algorithme AEMS

Déroulement de la planification. Nous comparons d'abord le déroulement temporel de la planification et de l'exécution de ces deux approches pour une réalisation de la mission d'un exemple type, pour lequel la cible recherchée se trouve dans la zone la plus éloignée.

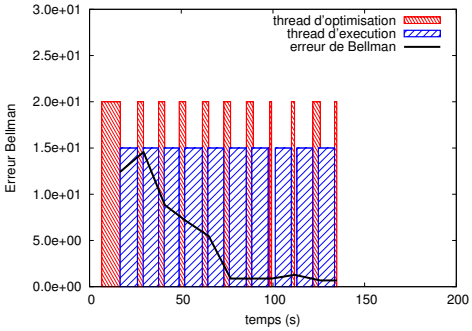
Dans la figure 6.14 nous cherchons à mettre en évidence les avantages de notre approche comparée à l'approche classique d'optimisation en ligne pour différentes durées de planification (4, 3 et 2 seconds) sur la base d'AEMS. Nous rappelons que l'approche classique entrelace la planification et l'exécution de la politique, c'est-à-dire que le planificateur planifie pendant un certain temps avant d'exécuter ensuite la meilleure action courante. Dans cette figure, l'abscisse représente le temps en secondes, les barres rouges et bleues représentent respectivement les portions de temps des différents *threads* d'optimisation et d'exécution, la ligne en noir représente l'évolution de l'erreur de Bellman. Les deux *threads* tournent en parallèle.

Supposons que nous voulions résoudre la mission en moins de 120 seconds, sachant que pour cette simulation la cible recherchée se trouve dans la zone la plus éloignée (comme montré sur la figure 6.8). Nous pouvons facilement remarquer que le temps total de mission augmente avec le temps alloué à la planification pour l'utilisation classique d'AEMS (AEMS avec 4 et 3 secondes), dépassant le temps limite de mission. Il est à noter que la valeur de 120 seconds veut pousser les deux approches à la limite et illustrer les différents comportements dans ce cas. En effet les missions effectuées sur le terrain de vol ont une durée plus longue.

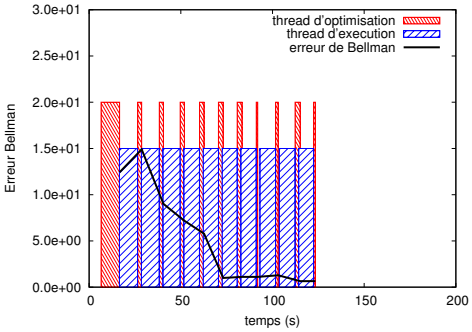
Encore sur la figure 6.14, les barres rouges successives montrent que l'approche classique planifie dans chaque état de croyance courant (état d'exécution). Notre approche, au



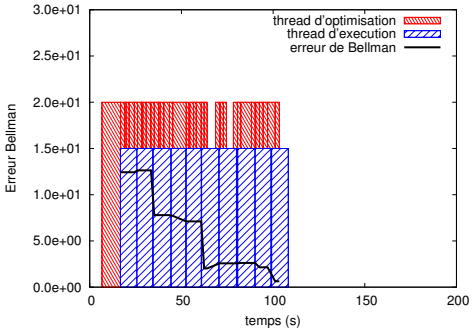
(a) Frise de temps pour AEMS dans l'approche classique avec temps de planification fixé à 4 secondes.



(b) Frise de temps pour AEMS dans l'approche classique avec temps de planification fixé à 3 secondes.



(c) Frise de temps pour AEMS dans l'approche classique avec temps de planification fixé à 2 secondes.



(d) Frise de temps pour AEMS dans le cadre AMPLE d'optimisation en parallèle de l'exécution.

FIGURE 6.14 – Cadre d'optimisation en ligne et exécution en parallèle *versus* cadre classique d'optimisation en ligne, qui entrelace optimisation et exécution.

contraire, optimise en permanence pour les états de croyance futurs (états d'exécution futurs possibles), de sorte que des actions optimisées sont toujours immédiatement disponibles dans chaque état de croyance (état d'exécution) courant. La durée de mission est donc plus petite dans le cadre AMPLE que dans le cadre classique d'optimisation en ligne.

Temps moyen de mission, taux de succès et actions par défaut. Nous avons aussi comparé l'approche classique d'optimisation en ligne avec l'approche proposée dans ce chapitre pour d'autres états que nous jugeons intéressants dans cette application robotique. Ces états suggèrent d'autres cas d'études, où les cibles sont dans des zones différentes.

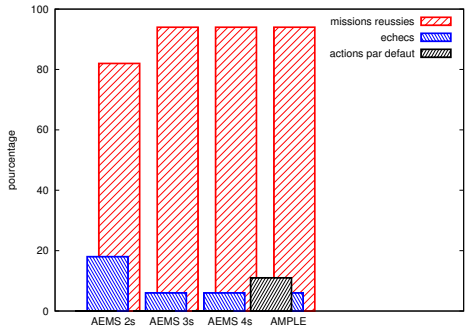
Nous montrons des résultats moyens obtenus à partir de 50 simulations avec notre simulateur réaliste. Le problème à résoudre respecte la configuration des zones montrée dans la figure 6.8. Par contre, la cible recherchée (modèle C) et les autres cibles présentes ou non dans la scène changent de zone entre les 3 cas d'étude :

1. La zone 1 contient la cible de modèle A, la zone 2 contient la cible de modèle C, et la zone 3 ne contient aucune cible ;
2. La zone 1 contient la cible de modèle B, la zone 2 ne contient aucune cible, et la zone 3 contient la cible de modèle C ;
3. La zone 1 contient la cible de modèle C, la zone 2 la cible de modèle B, et la zone 3 aucune cible.

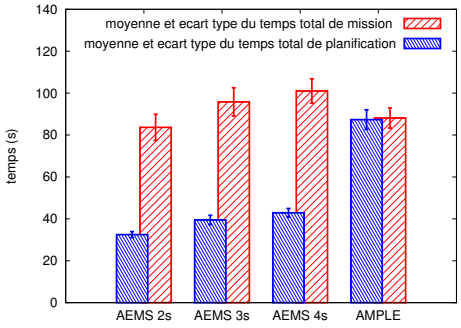
Pour chaque cas d'étude, nous avons fixé le temps limite de mission à 180 secondes, et le temps de *bootstrap* à 20 secondes. Notez que l'on cherche toujours à trouver la cible de

modèle C. La méconnaissance subjective de l'agent hélicoptère par rapport à la nature de l'environnement statique qui l'entoure est modélisée par une distribution uniforme sur les 64 états initiaux possibles.

Dans les figures 6.15,6.16 et 6.17 nous présentons les pourcentages de missions réussies, d'échecs et d'actions par défaut utilisées, ainsi que le temps moyen de mission et de planification pour différents temps de planification, pour l'approche classique d'optimisation en ligne avec l'algorithme AEMS (2, 3 et 4 seconds), et du cadre d'optimisation en parallèle de l'exécution AMPLE proposé dans ce chapitre sur la base d'AEMS.



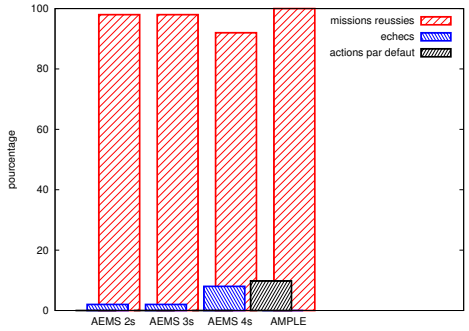
(a) Pourcentages de missions réussies, d'échecs et d'actions par défaut pour le cas d'étude 1.



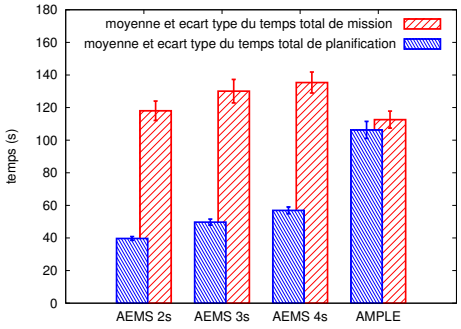
(b) Temps moyen de mission et de planification pour le cas d'étude 1.

FIGURE 6.15 – Comparaison entre l'approche classique d'optimisation en ligne et cadre de planification en parallèle de l'exécution pour le cas d'étude 1.

Pour le cas d'étude 1, le temps moyen d'AEMS dans le cadre AMPLE est seulement légèrement supérieur à AEMS 2s (83.6s pour AEMS 2s contre 88.11s pour AMPLE). Par contre, le pourcentage de missions réussies avec le cadre AMPLE sur la base d'AEMS est supérieur à celui de AEMS 2s. Le pourcentage d'actions par défaut est de 11%. Il est à noter que les actions par défaut garantissent la réactivité du composant de planification pour les cas où une action optimisée n'est pas disponible au moment exact où le composant d'exécution la lui demande. Dans cette étude, l'utilisation des actions par défaut ne dégradent pas le niveau d'efficacité de l'approche (pourcentage de missions réussies).



(a) Pourcentages de missions réussies, d'échecs et d'actions par défaut pour le cas d'étude 2.

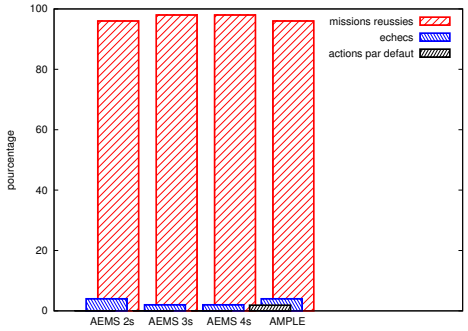


(b) Temps moyen de mission et de planification pour le cas d'étude 2.

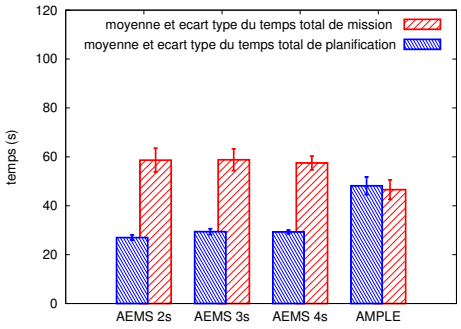
FIGURE 6.16 – Comparaison entre l'approche classique d'optimisation en ligne et cadre de planification en parallèle de l'exécution pour le cas d'étude 2.

Pour le cas d'étude 2, le temps moyen du cadre AMPLE est inférieur à AEMS 2s, 3s et 4s. De plus, le pourcentage de missions réussies avec le cadre AMPLE sur la base d'AEMS (100%) est aussi supérieur aux autres approches. Le pourcentage d'actions par défaut est de 9%. Comme pour le cas précédent, il est à noter que l'utilisation des actions par défaut ne

dégradent pas la performance de l'approche (pourcentage de missions réussies).



(a) Pourcentages de missions réussies, d'échecs et d'actions par défaut pour le cas d'étude 3.



(b) Temps moyen de mission et de planification pour le cas d'étude 3.

FIGURE 6.17 – Comparaison entre l'approche classique d'optimisation en ligne et cadre de planification en parallèle de l'exécution pour le cas d'étude 3.

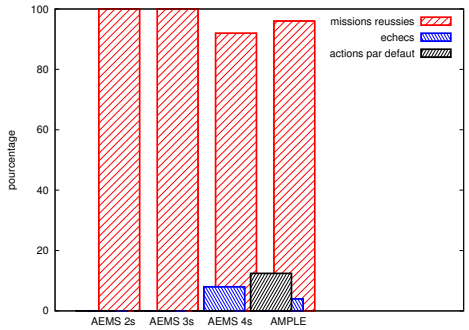
Pour le cas d'étude 3, le temps moyen du cadre AMPLE est encore inférieur à AEMS pour 2, 3 et 4 secondes. Pour ce cas d'étude, nous rappelons que la cible recherchée se trouve dans la zone explorée initialement, ce qui explique un temps moyen de mission plus court que pour les cas d'études précédents. Le pourcentage de missions réussies avec le cadre AMPLE sur la base d'AEMS est équivalent aux autres approches. De plus, le pourcentage d'actions par défaut est relativement faible (9%). Là encore, comme pour les cas précédents, nous notons que les actions par défaut ne dégradent pas le niveau d'efficacité de l'approche (pourcentage de missions réussies).

Nous pouvons conclure que sur ces 3 cas d'étude l'approche de planification en parallèle de l'exécution présente en moyenne un temps de mission plus court. En réalité le gain de temps n'est pas très important, puisqu'il se ramène à environ la somme des temps de planification (soit 2, 3 ou 4 secondes) utilisés par AEMS dans le cadre classique d'optimisation en ligne. Dans notre approche d'optimisation en parallèle de l'exécution, les actions optimisées sont immédiatement disponibles dans chaque état de croyance (état d'exécution) courant. Donc cette phase de planification n'est plus nécessaire et le gain se ramène à cette différence de temps. Toutefois, ces résultats confirment le fait que la durée de mission est plus petite dans le cadre d'optimisation pendant exécution que dans le cadre classique d'optimisation en ligne.

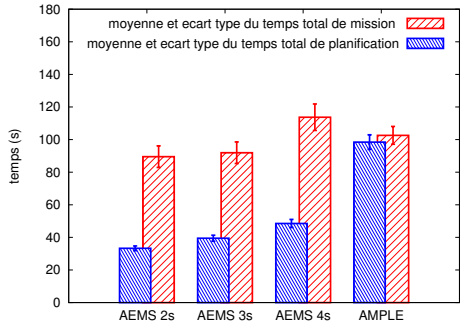
Sur la figure 6.18 nous montrons les résultats obtenus avec 50 simulations pour le même état initial caché que celui de l'expérimentation en vol (section 6.4).

Ces résultats montrent en revanche que le pourcentage de missions réussies est légèrement inférieur pour le cadre AMPLE, et que le temps moyen de mission d'AMPLE est seulement inférieur à celui de AEMS 4s. L'explication réside dans l'utilisation des actions par défaut (presque 20%), car la politique par défaut utilisée est le résultat de l'approximation QMDP [Littman *et al.*, 1995] calculée au début de la mission. Étant donné que nous avons besoin d'utiliser une politique par défaut afin d'assurer la réactivité du composant de planification, on pourrait considérer une autre politique par défaut, par exemple paramétrée par le nombre de zones et écrite à la main.

Exigences de la politique par défaut. Nous ouvrons la discussion sur l'utilité de cette politique par défaut. Il nous semble primordial que la politique par défaut soit une politique (sous-)optimale qui puisse garantir une bonne performance de l'agent. Nous avons démontré que pour certains cas, une approximation de type QMDP ne peut pas garantir cette efficacité. Nous avons choisi l'approximation QMDP puisqu'elle peut être calculée très rapidement après génération du problème POMDP. Comme le problème n'est pas connu à l'avance, l'utilisation



(a) Pourcentages de missions réussies, d'échecs et d'actions par défaut pour l'état initial caché de l'expérimentation en vol.



(b) Temps moyen de mission et de planification pour l'état initial caché de l'expérimentation en vol.

FIGURE 6.18 – Comparaison entre l'approche classique d'optimisation en ligne et cadre de planification en parallèle de l'exécution pour l'état initial caché de l'expérimentation en vol.

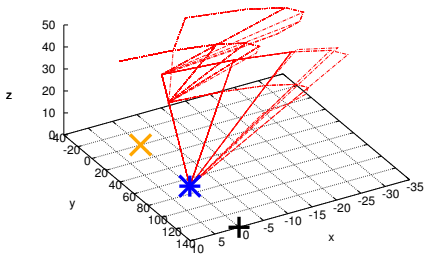
d'une politique par défaut calculée hors ligne n'est donc pas possible. Nous croyons que l'utilisation d'une politique paramétrique experte dédiée à la mission serait une meilleure option. Dans ce sens, nous gardons la philosophie du cadre d'optimisation en parallèle de l'exécution.

Il est à noter qu'un lien avec ce qui a été développé dans les chapitres 4 et 5 peut être ici fait. On pourrait exiger de cette politique par défaut un comportement qui donnerait la priorité à la prise d'information, étant donné que pour ce genre d'application l'obtention d'un certain niveau de certitude est indispensable pour le bon déroulement de la mission. On pourrait aussi exiger de cette politique des actions *sûres*, c'est-à-dire que la politique par défaut ne devrait jamais fournir des actions inacceptables (par exemple, changer de zone à une altitude de vol trop faible). Ces exigences peuvent être prises en compte lors de la définition d'une politique paramétrique, qui peut être aussi validé par des méthodes formelles.

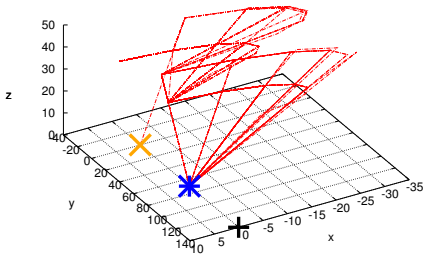
D'autre part, si le concepteur du système autonome relâche la contrainte de réactivité du composant de planification, on pourrait ainsi admettre de planifier pendant quelques secondes pour les cas où aucune action n'est disponible pour l'état de croyance courant (état d'exécution), tout en planifiant pour les états de croyance futurs possibles pendant l'exécution des actions. Nous croyons que dans ce cas, nous pourrions coupler les avantages des deux approches, tout en utilisant le temps d'exécution d'une action pour la planification des états de croyance futurs et en utilisant une politique par défaut qui puisse garantir une certaine efficacité.

Comportement à l'exécution. À titre illustratif, sur la figure 6.19 nous présentons quelques exemples de trajectoires développées par l'hélicoptère lors des simulations réalistes pour l'approche classique d'optimisation en ligne avec l'algorithme AEMS 2s et pour l'approche proposée AMPLE sous la base d'AEMS pour les 3 cas d'étude présentés précédemment.

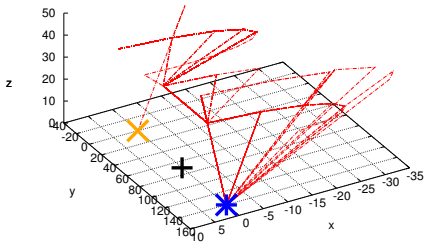
Nous cherchons simplement, sur cette figure, à illustrer le fait que les stratégies sont légèrement différentes. Lors de l'exécution, nous observons que les trajectoires diffèrent selon les approches, d'une part parce que les politiques sont différentes à cause de l'utilisation ou non des actions par défaut et à cause au temps de planification nécessaire à chaque approche. D'autre part, les trajectoires changent selon les événements stochastiques, c'est-à-dire suivant les différentes observations obtenues par l'hélicoptère. On voit qu'il effectuera des profils de vol différents à des altitudes de vol variées en prenant plus ou moins de temps pour acquérir des informations sur l'identité des cibles.



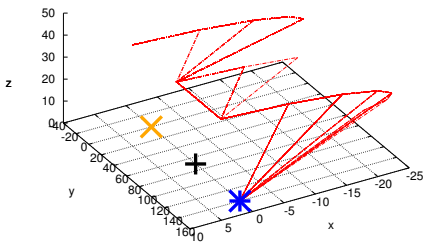
(a) Trajectoires de l'agent hélicoptère pendant des simulations réalistes (en rouge) pour AEMS 2s pour le cas d'étude 1. La croix orange représente la zone 1, l'étoile bleu la zone 2, zone dans laquelle la cible se trouve et la croix noire la zone 3.



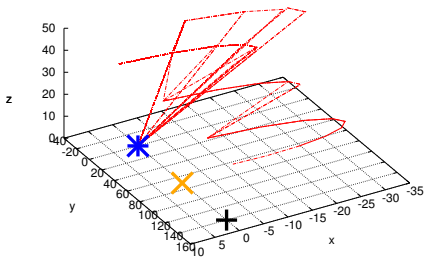
(b) Trajectoires de l'agent hélicoptère pendant des simulations réalistes (en rouge) pour AMPLE pour le cas d'étude 1. La croix orange représente la zone 1, l'étoile bleu la zone 2, zone dans laquelle la cible se trouve et la croix noire la zone 3.



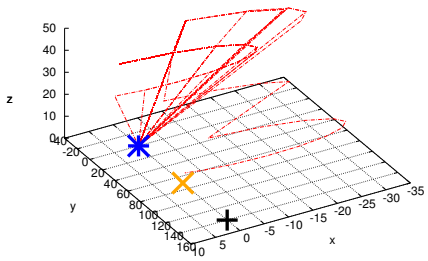
(c) Trajectoires de l'agent hélicoptère pendant des simulations réalistes (en rouge) pour AEMS 2s pour le cas d'étude 1. La croix orange représente la zone 1, la croix noire la zone 2, l'étoile bleu la zone 3, zone dans laquelle la cible se trouve.



(d) Trajectoires de l'agent hélicoptère pendant des simulations réalistes (en rouge) pour AMPLE pour le cas d'étude 1. La croix orange représente la zone 1, la croix noire la zone 2, l'étoile bleu la zone 3, zone dans laquelle la cible se trouve.



(e) Trajectoires de l'agent hélicoptère pendant des simulations réalistes (en rouge) pour AEMS 2s pour le cas d'étude 1. L'étoile bleu la zone 1, zone dans laquelle la cible se trouve, la croix orange la zone 2 et la croix noire la zone 3.



(f) Trajectoires de l'agent hélicoptère pendant des simulations réalistes (en rouge) pour AMPLE pour le cas d'étude 1.L'étoile bleu la zone 1, zone dans laquelle la cible se trouve, la croix orange la zone 2 et la croix noire la zone 3.

FIGURE 6.19 – Trajectoires de l'agent hélicoptère pendant quelques simulations réalistes pour l'approche classique d'optimisation en ligne avec l'algorithme AEMS 2s et pour l'approche proposée AMPLE pour les 3 cas d'étude.

6.6 Conclusion

A notre connaissance, ce chapitre présente l'une des premières implémentations d'une mission de détection et de reconnaissance de cibles basée sur l'approche POMDP pour un drone hélicoptère autonome. Le coeur de notre contribution repose sur la proposition des moyens algorithmiques pratiques pour optimiser et exécuter des politiques POMDP en parallèle sous des contraintes de temps, ce qui est nécessaire dans le cas où le problème POMDP est généré pendant le vol. Quatre communications ont été publiées sur la base des résultats de ce chapitre [Carvalho Chanel *et al.*, 2012a, Carvalho Chanel *et al.*, 2012b, Carvalho Chanel *et al.*, 2012c, Carvalho Chanel *et al.*, 2013], cette dernière à apparaître.

Nous avons analysé les résultats des expérimentations réalisées avec des simulations réalistes et complètes (*hardware in the loop*) et basées sur des données réelles. Ces simulations nous ont permis de démontrer que le choix de l'algorithme sous-jacent utilisé reste décisif pour que les techniques de planification POMDP puissent s'attaquer à des missions de robotique aérienne assez complexes. Nous avons aussi démontré que le cadre proposé d'optimisation en parallèle de l'exécution permet en moyenne d'obtenir un temps de mission plus court que l'approche classique d'optimisation, qui entrelace les phases d'optimisation et d'exécution. Mais, ceci à condition que la politique par défaut puisse garantir un certain niveau d'efficacité. Des recherches devraient donc être poursuivies sur l'étude de cette politique par défaut. En particulier, des pites peuvent être soulévées pour tenir compte de ce que nous avons proposé dans les chapitres 4 et 5. La politique par défaut pourrait garantir un comportement qui donnerait priorité à la prise d'information, étant donné que pour ce genre d'application l'obtention d'un certain niveau de certitude est indispensable pour le bon déroulement de la mission. Ou encore, cette politique par défaut pourrait garantir des actions *sûres*, notamment par sa validation formelle pour satisfaire les exigences de sûreté imposées par les agences de régulation aérienne.

De plus, nous avons embarqué nos composants de décision à bord d'un drone hélicoptère autonome, un Yamaha Rmax de l'Onera. Ceci a demandé la mise en place de la génération automatique du problème POMDP à résoudre, ainsi que d'un superviseur de mission. Ce travail nous a permis d'acquérir de nombreuses connaissances sur les architectures robotiques, ainsi que sur le fonctionnement du superviseur de mission chargé d'assurer la coordination entre les différents composants de l'architecture, indispensables au bon déroulement de la mission. Nous avons aussi enrichi la communauté robotique avec des moyens de décision autonome pour les architectures robotiques.

De nombreuses améliorations peuvent être envisagées pour de futures recherches, à savoir : (i) l'analyse de l'impact de différentes politiques par défaut, ainsi qu'une étude sur le couplage de l'approche classique d'optimisation en ligne (pour les actions par défaut) avec l'approche d'optimisation en parallèle de l'exécution (planification pour les états de croyance futurs) ; (ii) l'adaptation des algorithmes de résolution en ligne pour la prise en compte des contraintes de sécurité imposées par les agences civiles aéronautiques lors de l'optimisation de la stratégie (modèle AC-POMDP proposé dans le chapitre 5) ainsi que la validation formelle des processus stochastiques engendrés par les politiques ; (iii) la prise en compte lors du calcul de la stratégie des incertitudes liées au modèle d'observation appris à partir d'une base de données réelles, afin d'obtenir une politique robuste aux incertitudes ; (iv) l'utilisation du modèle MOMDP [Ong *et al.*, 2009, López *et al.*, 2010] qui permettrait de factoriser l'état de croyance en séparant la partie observable de la partie cachée, ainsi que l'adaptation des algorithmes de résolution en ligne de POMDP pour le modèle MOMDP.

Chapitre 7

Conclusion

7.1 Conclusion générale

Dans ce mémoire de thèse nous avons abordé le problème de planification de perception et de mission, appliqué à la détection et à la reconnaissance de cibles. Cette problématique nous place dans un cadre probabiliste de planification d'actions et en situation d'observabilité partielle de l'environnement, dû à la méconnaissance de l'agent décideur et à la nature des sorties obtenues du traitement d'information.

Ainsi, l'objectif de cette thèse a été de modéliser et de résoudre un problème de planification de perception et de mission pour un drone hélicoptère qui évolue dans un environnement incertain et partiellement observable. Dans notre démarche, nous avons été amenés à considérer l'acquisition d'information non seulement comme une fin mais aussi un moyen pour renseigner efficacement la politique d'action permettant d'atteindre les objectifs de la mission : il s'agit de décider pour percevoir et de percevoir pour décider. Nous avons comparé différentes approches de modélisation, et nous avons aussi développé et mis en œuvre des schémas algorithmiques de résolution adéquats. À partir d'une analyse du compromis perception-action, nous avons abouti à une approche comparative permettant d'obtenir des politiques objectivement acceptables d'un point de vue pratique en ce qui concerne les seuils d'identification de l'état caché pour aboutir au but de la mission.

Nous avons alors été confrontés à la problématique de la prise en compte des contraintes liées à la physique ou à la sûreté du système dans son environnement lors de l'exécution des actions. Il faut garantir que la politique d'action résultant de la planification ne puisse pas engendrer des actions dangereuses ou interdites du fait de la méconnaissance de l'état courant, ou d'une mauvaise appréciation des conséquences possibles de ces actions. Nous avons proposé une approche de modélisation et développé un schéma algorithmique de résolution efficace permettant de générer des politiques d'action sûres à l'exécution, ce qui est essentiel dans des applications aéronautiques, et typiquement exigé de tout système aérien en vue d'obtenir des autorisations de vol.

Nous avons enfin été obligée de prendre en compte le fait que le problème de planification à résoudre ne peut être connu qu'en ligne, c'est-à-dire une fois en vol. En effet, le nombre de cibles ou le nombre de zones qui composent l'environnement sont des exemples de paramètres généralement inconnus avant le vol devant être automatiquement extraits au début de la mission. Le problème de décision séquentielle dans l'incertain et sous observabilité partielle est alors généré en vol puis optimisé. Nous avons ainsi proposé des moyens algorithmiques pratiques pour optimiser et exécuter des politiques en parallèle sous des contraintes de temps.

Après avoir apporté les réponses correspondant à ces trois parties, nous pensons qu'il est nécessaire de garantir des solutions interprétables et sûres à la fois pour la solution

du compromis perception-action et pour le respect des contraintes et exigences de sûreté à l'exécution. Nos contributions fournissent un cadre d'étude, d'analyse et de conception de tels systèmes, et il resterait à rassembler ces différentes composantes dans un cadre formel unifié, ce que nous devons laisser pour les perspectives à l'issue de cette thèse.

Dans la suite, nous détaillerons les différentes contributions apportées par cette thèse.

7.2 Contributions

7.2.1 Modélisation (cf. chapitre 3)

Dans le chapitre 3, nous avons pu modéliser le problème dual de perception-décision en tant que POMDP, car ce modèle propose un schéma général applicable à notre problématique, et optimal pour les tâches de perception et de décision à long terme. Une attention spéciale a été donnée à l'apprentissage (hors ligne) du modèle d'observation du POMDP à partir de données réelles de l'environnement, obtenues lors de campagnes de prise d'images de voitures (nos cibles) par le drone hélicoptère. Ces données nous ont permis d'obtenir un modèle statistique et réaliste de l'application. Deux scénarios de mission ont été ainsi proposés : une mission d'exploration et une mission d'atterrissage. La première consiste à identifier l'état caché de l'environnement dans lequel l'agent autonome évolue tout en tenant compte des coûts liés aux déplacements de l'agent (perception active). La deuxième mission consiste à identifier une cible en particulier parmi les différentes cibles connues de la base de données afin d'atterrir à proximité de la cible recherchée (perception active et satisfaction de but symbolique).

7.2.2 Étude du compromis perception-action (cf. chapitre 4)

Dans le chapitre 4, nous avons mis en évidence une étude du compromis entre la prise d'information et la décision dans les deux contextes applicatifs décrits précédemment. L'intérêt d'étudier la politique d'action résultant de l'optimisation d'un critère mixte d'optimisation (ρPOMDP) a été illustré pour des problèmes de diagnostic ou d'identification de l'état caché de l'environnement (mission d'exploration du chapitre 3). Le critère mixte proposé couple le critère d'optimisation classique des POMDP avec un critère d'optimisation très utilisé par la communauté de perception active basé sur l'entropie de l'état de croyance de l'agent. Nous avons implémenté cette approche en modifiant un algorithme issu de l'état de l'art du domaine des POMDP, en tenant compte du fait que la fonction de valeur résultante de ce couplage n'est plus linéaire.

Nous avons donc proposé un schéma algorithmique de résolution et d'optimisation de la politique pour ce critère mixte non-linéaire. En ce qui concerne l'optimisation de la politique pour le critère non-linéaire, nous pensons qu'il serait souhaitable de changer la forme de paramétrisation de la fonction de valeur : la paramétrisation à base d'α-vecteurs n'est pas bien adaptée à ce genre de critère non-linéaire dû à la difficulté d'approximation linéaire lors des opérations de mise à jour de la valeur des états de croyance. D'autre part, la détermination de la bonne valeur de la pondération λ dépend du modèle de récompense/coût et du concepteur, puisque lui seul pourra définir quel point du front de Pareto doit être considéré comme optimal. Nous avons aussi démontré qu'il est possible de modéliser ce problème de perception active par un modèle classique de POMDP où l'on rajoute des buts fictifs par des actions supplémentaires dites *report* s pour chaque état s du système, de façon à ce que l'agent décideur soit récompensé si et seulement s'il réalise l'action *report* s quand l'état s est le véritable état. Sur la base du schéma algorithmique proposé nous avons pu mener à bien

des comparaisons entre ces deux approches du point de vue de la politique d'acquisition d'information.

Notre proposition est une contribution originale à trois titres différents :

- nous avons proposé, développé et expérimenté pour le critère mixte non-linéaire un algorithme de résolution de type *point-based* qui permet une optimisation générique de ce type de critère non-linéaire ;
- nous avons proposé une modélisation POMDP classique équivalente (au modèle avec critère mixte) par ajout de récompenses classiques supplémentaires sur chaque état devant être correctement identifié, récompenses obtenues par les actions terminales supplémentaires de type *report s* traduisant explicitement des buts d'identification de l'état de l'environnement (qui sont traités par la mesure d'incertitude de l'état de croyance dans le modèle avec critère mixte) ;
- nous avons réalisé une évaluation comparative objective entre ces deux modèles par simulation de la politique obtenue sur une même base de récompenses reçues à l'exécution.

Les résultats comparatifs nous permettent de conclure que la fonction de récompense utilisée dans le modèle est le facteur décisif pour l'obtention d'une politique pertinente. Étant donné que l'utilisation d'un critère mixte non-linéaire est possible d'un point de vue algorithmique, il est possible d'évaluer et de comparer les politiques obtenues par un critère mixte avec celle obtenue en choisissant des récompenses pour le modèle classique (ajout d'actions). Il ressort de la comparaison que le critère mixte permet de définir de manière intuitive (par l'utilisateur) des taux acceptables de bonnes et de mauvaises classifications. Ainsi, il est possible de revenir vers une modélisation classique POMDP par ajout des buts fictifs en ayant des éléments objectifs de comparaison pour fixer le *bon* rapport entre les récompenses et les coûts du modèle. Plus précisément, le choix de ce *bon* rapport est déterminant vis-à-vis des taux intuitifs de bonnes classifications que l'utilisateur souhaite obtenir au moment de l'exécution de la politique. Ainsi, en pratique, les deux approches (modélisation par critère mixte ou modélisation par POMDP classique équivalent) peuvent être utilisées en parfaite complémentarité de façon à permettre : de caractériser une politique correspondant aux taux acceptables des bonnes et mauvaises classifications, de déterminer les bonnes valeurs des coûts et des récompenses, et de garantir la convergence vers une politique ϵ-optimale de la politique obtenue.

En outre, les travaux effectués dans ce chapitre nous ont aussi permis d'évaluer l'applicabilité des POMDP dans le cadre d'une mission d'atterrissage. Nous avons vu que l'incertitude de l'état de croyance, mesurée par l'entropie de l'état de croyance, est intrinsèquement réduite avec le critère classique dans ce genre de mission en amenant l'agent à acquérir suffisamment d'information pour bien la mener à terme, indépendamment du modèle de cible recherchée. Ceci a été démontré par le fait que, même pour un modèle d'observation moins précis, l'agent hélicoptère a atteint un pourcentage de réussite de mission équivalent à celui obtenu avec des modèles plus précis, au détriment de quelques étapes supplémentaires de prise d'information. De plus, notre étude du comportement d'une politique calculée avec le critère mixte proposé nous a amené à conclure que, pour ce type de mission, l'optimisation explicite d'une mesure d'incertitude de l'état de croyance n'est clairement pas nécessaire. Nous avons vu que la vitesse de convergence de l'incertitude de l'état de croyance, mesurée par l'entropie de l'état de croyance, ne peut être améliorée que pour des valeurs importantes de la pondération λ, avec une perte non négligeable d'efficacité de la politique en ce qui concerne le pourcentage de réussite de la mission. Toutefois, les différentes politiques obtenues par le critère mixte mettent en évidence que si l'on souhaite "garantir" que le taux de missions manquées est inférieur à un certain seuil, le rapport entre les récompenses et coûts du modèle de récompense doit être raffiné.

Les résultats obtenus nous permettent de conclure que la structure de la fonction de

récompense est déterminante par rapport à l'intérêt ou à l'utilité d'ajouter une mesure d'incertitude pour guider l'optimisation de la politique. Les deux missions étudiées – exploration et atterrissage – ne sont pas de nature différente. Une réflexion approfondie lors de la modélisation du problème et en particulier dans le choix du modèle de récompense à utiliser est déterminante. Nous montrons dans cette thèse que si l'on ajoute au modèle des actions terminales (actions de classification ou d'atterrissage) avec des récompenses classiques sur les états à identifier correctement pour bien réaliser la mission (états buts fictifs), il n'est plus nécessaire d'utiliser un critère mixte (basé sur des mesures de récompense et d'incertitude), *y compris* pour des problèmes de perception active pure. Un critère de type mixte peut être considéré comme une approche intuitive qui vient en complément du cadre classique des POMDP. L'utilisateur du système peut plus facilement indiquer au concepteur le taux de bonnes classifications ou le taux de missions réussies qu'il souhaite. Ainsi, la modélisation des récompenses du POMDP classique, qui est faite dans la plupart de cas, de manière empirique, peut être guidée par une comparaison objective avec un modèle équivalent ρPOMDP afin de définir le *bon* rapport entre les récompenses et les coûts du modèle.

7.2.3 POMDP et précondition d'action (cf. chapitre 5)

Dans le chapitre 5, nous avons proposé une approche qui nous permet de prendre en compte dans un cadre d'observabilité partielle certaines contraintes physiques et de sûreté liées à la mission du drone hélicoptère. Ces contraintes sont par exemple : atterrir seulement si une zone est "atterrissable" même si la cible recherchée se retrouve dans cette zone ; ou garantir que le changement de zone se fera à une altitude de vol donnée. Cette approche nous a amené à proposer une extension du modèle POMDP, appelée AC-POMDP, pour laquelle l'observation de l'agent est constituée pour une partie de l'observation classique de l'état de l'environnement et pour une autre partie de symboles qui représentent l'ensemble des actions réalisables dans le domaine des états possibles du système. Ainsi, le modèle AC-POMDP nous conduit à un nouvel algorithme qui exploite explicitement la séparation entre l'information concernant la vérification des propriétés du modèle (les préconditions), et celle qui renseigne sur la présence du but de mission ou sur la nature des objets de l'environnement. Une propriété fondamentale des AC-POMDP, qui permet de réduire significativement la complexité, est que les α-vecteurs sont définis chacun sur des sous-espaces de croyance différents et réduits. Ceci permet de n'évaluer qu'un sous-ensemble d'actions pour un état de croyance donné, et de ne calculer des projections que pour les α-vecteurs de la fonction de valeur qui sont cohérents avec l'ensemble d'actions réalisables. Il est à noter que le processus de décision est toujours partiellement observable, puisque plusieurs états peuvent avoir le même ensemble d'actions réalisables. Nous avons fourni des équations d'optimalité, de mise à jour de l'état de croyance et une transformation vers le modèle POMDP équivalent afin de pouvoir comparer l'approche proposée avec le modèle classiquement utilisé. Cette transformation équivalente repose sur l'hypothèse communément admise que l'on peut choisir des récompenses équivalentes à des valeurs $-\infty$ dans le modèle POMDP, tout du moins à une valeur qui correspond à la borne inférieure des valeurs des états.

Notre contribution dans ce chapitre porte aussi sur l'étude théorique et sur des expérimentations réalisées grâce à l'adaptation d'un algorithme de l'état de l'art du domaine POMDP. Notre algorithme PCVI1 permet de tirer profit des propriétés du modèle telles que l'évaluation d'un sous-ensemble d'actions pour un état de croyance donné (opération $\max_{a \in \mathcal{A}_b}$), et le calcul des projections $\Gamma^{a,(o,\theta)}$ seulement pour les α-vecteurs de V qui sont cohérents avec l'ensemble θ. Les α-vecteurs de la fonction de valeur n'ont leurs valeurs définies que sur les états où l'action associée est réalisable. Ceci nous ramène à une fonction de valeur paramétrée par des α-vecteurs définis chacun sur différents sous-espaces de croyances différents

et réduits. Par ailleurs, l'algorithme PCVI2 utilise une borne inférieure de la fonction de valeur optimale, et qui donne lieu à une optimisation exponentiellement plus rapide.

Nos résultats expérimentaux démontrent qu'il est possible d'explorer le découplage optimal des différentes observations lors de l'optimisation de la politique, en réduisant le temps de calcul de manière significative. Nous obtenons bien plus rapidement des politiques fondées et équivalentes à celles du modèle classique POMDP équivalent, tout en respectant les contraintes physiques et de sûreté modélisées par des préconditions sémantiquement fondées et ne dépendant d'aucun réglage arbitraire des coûts et récompenses du modèle. Dans le cadre POMDP mono-agent, notre modèle permet de garder l'interprétation sémantique des préconditions et montre finalement que la façon *classique* de modéliser le problème n'est pas la plus efficace en comparaison de notre algorithme PCVI qui exploite directement la structure spécifique des AC-POMDP.

7.2.4 Planification anticipée et exécution en parallèle des politiques partielles (cf. chapitre 6)

Dans le chapitre 6, nous avons présenté l'une des premières implémentations en temps réel, sur système robotique aérien avec validation en vol expérimental, d'une mission de détection et de reconnaissance de cibles basée sur l'approche POMDP pour un drone hélicoptère autonome (mission d'atterrissage). Nous avons embarqué nos composants de décision à bord d'un drone hélicoptère autonome, un Yamaha Rmax de l'Onera. Ceci a nécessité la mise en place de la génération automatique du problème POMDP à résoudre, puis la conception d'un cadre algorithmique permettant d'optimiser en anticipation et d'exécuter en parallèle la politique afin de concevoir un superviseur de mission chargé d'assurer la coordination entre les différents composants de l'architecture, indispensables au bon déroulement de la mission.

Le cœur de notre contribution repose sur la proposition, le développement et la mise en œuvre pratique d'un schéma algorithmique, baptisé AMPLE, pour optimiser et exécuter en parallèle des politiques POMDP sous contraintes de temps de réponse du planificateur. L'optimisation de la politique nécessite l'utilisation d'un solveur POMDP sous-jacent (par exemple PBVI ou AEMS), capable d'optimiser localement des bouts de politique. Des expérimentations réalistes de type "*hardware in the loop*" nous ont permis de démontrer que le choix de l'algorithme sous-jacent utilisé reste décisif pour que les techniques de planification POMDP puissent s'attaquer à des missions de robotique aérienne assez complexes. Nous avons aussi démontré que le cadre proposé d'optimisation en parallèle de l'exécution permet en moyenne d'obtenir un temps de mission plus court et un pourcentage de réussite de mission plus élevé (dans certains cas) pour des missions très contraintes en temps, que l'approche classique d'optimisation, qui entrelace les phases d'optimisation et d'exécution. Cependant, notre schéma algorithmique AMPLE repose sur la définition d'une politique par défaut, par exemple une politique paramétrique et définie en laboratoire en préparation de la mission, permettant de garantir un certain niveau d'exigence en termes d'efficacité et de sûreté.

Nous sommes ainsi ramenée à la question de garantir le comportement de la politique d'action par rapport aux objectifs de la mission, en termes de temps de réponse du planificateur, aussi bien qu'en termes de conformité à la sûreté ou à la sécurité, ou en termes de respect d'un certain seuil de confiance dans l'état de croyance avant décision (taux de bonne et mauvaise classification par exemple).

Nous avons ainsi apporté par nos travaux des contributions en termes de modélisation et de schémas algorithmiques pour la décision et la perception autonomes adaptables aux architectures robotiques et applicables dans des situations exigeantes en termes de garantie

d'efficacité, sûreté et fiabilité. Il nous reste à proposer des pistes pour poursuivre ces travaux, peut être vers un cadre formel de perception-action plus unifié.

7.3 Perspectives

Dans la suite, nous présentons plusieurs perspectives de recherche envisagées pour le prolongement de nos travaux :

- les deux premières propositions nous semblent directement dans le prolongement de nos travaux dans la perspective d'unifier les approches et modèles pour la décision séquentielle en environnement incertain et partiellement observable pour des applications robotiques (notamment aériennes), la seconde proposition étant la plus ambitieuse ;
- les suivantes nous semblent des suites logiques de nos travaux pour compléter les développements algorithmiques ainsi que les comparaisons réalisées dans notre thèse : il s'agirait d'améliorer une des approches proposées dans la thèse.

7.3.1 Vers un modèle et des outils unifiés

Analyse de l'impact de différentes politiques par défaut dans le cadre d'optimisation en parallèle de l'exécution

Dans le chapitre 6, nous avons vu que la politique par défaut utilisée est importante pour la performance globale des algorithmes à chaque fois que le temps de réponse du composant de planification n'est pas suffisant pour le problème de décision en ligne, c'est-à-dire quand la réactivité du composant de planification ne peut pas être assurée. Nous avons utilisé l'approximation QMDP lors de nos expérimentations, car elle peut être calculée rapidement en début de mission. Nos résultats démontrent que l'utilisation d'une politique par défaut heuristique n'est pas toujours convenable.

Dans ce contexte, une piste de recherche est l'évaluation de l'impact de différentes politiques par défaut dans le cadre proposé. Il nous semble primordial que la politique par défaut soit une politique (sous-)optimale qui puisse garantir une bonne performance de l'agent. Nous croyons que l'utilisation d'une politique paramétrique experte, par exemple dédiée à la mission, serait une meilleure option pour garantir l'efficacité de la politique par défaut nécessaire au cadre d'optimisation en parallèle de l'exécution. En particulier, des pistes peuvent être soulevées pour tenir compte de nos propositions et solutions des chapitres 4 et 5. La politique par défaut pourrait ainsi garantir un comportement qui donnerait la priorité à la prise d'information, étant donné que pour ce genre d'application l'obtention d'un certain niveau de certitude est indispensable pour le bon déroulement de la mission. Ou encore, cette politique par défaut pourrait garantir des actions *sûres* ou non dangereuses, c'est-à-dire des actions qui respectent des préconditions ou des règles d'interdiction garantes des exigences de sûreté et de sécurité des opérations. Ces différentes exigences seraient ainsi satisfaites lors de la construction de la stratégie par défaut, puis maintenues au cours de l'exécution des actions et de l'optimisation en ligne de la stratégie. Elles peuvent aussi être soumises à vérification et validation formelle pour satisfaire les exigences de sûreté imposées par les agences de régulation aérienne.

Nouvelle modélisation pour la méconnaissance de l'agent et pour la prise en compte de l'imprécision de modèles de transition et d'observation lors du calcul de la politique

Tout au long de cette thèse, nous avons modélisé la méconnaissance subjective de l'agent hélicoptère par rapport à la nature de l'environnement statique qui l'entoure par une distri-

bution de probabilité uniforme sur les états initiaux possibles. Il est à noter qu'on ne peut pas faire mieux en utilisant le modèle POMDP, ou un modèle probabiliste (Bayésien), pour modéliser une méconnaissance subjective [Fabiani, 1994, Bouyssou *et al.*, 2009]. Il est admis que nous devons interpréter la croyance initiale de l'agent comme une fréquence d'initialisation du monde par l'environnement ou comme un pari sur l'état caché du système. Mais, nous attirons l'attention sur le fait que la politique optimisée n'est exécutée qu'une seule fois (voir très peu de fois dans le monde réel). Ceci contredit l'hypothèse d'existence d'une fréquence d'initialisation. D'un point de vue pratique, il est acceptable d'initialiser la croyance sous forme d'une distribution uniforme : on admet ne pas vouloir prendre de risque et donc on donne le même crédit à tout état initial possible. De plus, les probabilités des modèles de transition et d'observation sont en général déterminées par des apprentissages statistiques. Nous obtenons ainsi des fonctions *fréquentistes* qui seront "confondues" avec une croyance initiale *subjective*.

Un autre point concerne l'apprentissage de modèle d'observation hors ligne que nous avons réalisé dans le chapitre 3 pour l'obtention du modèle d'observation du POMDP. Lors de l'apprentissage par étude statistique des sorties de l'algorithme, nous avons constaté qu'il était possible de définir des intervalles de confiance liés aux probabilités apprises. Malheureusement, le modèle POMDP n'utilise pas cette information puisqu'il suppose un modèle d'observation parfait. Peu de travaux se sont intéressés à ce problème dans le cadre POMDP. Ceci est dû principalement au fait que, si les fonctions de transition et d'observation ne sont plus parfaites, la mise à jour de l'état de croyance ne peut plus être réalisée facilement, tout comme les projections des α-vecteurs lors de la mise à jour de la fonction de valeur. Nous pensons que cette problématique est une piste de recherche prometteuse. En effet, dans le cadre MDP, des travaux tels que [Givan *et al.*, 2000, Buffet et Aberdeen, 2005] ont proposé des techniques pour obtenir des politiques dites robustes aux imprécisions du modèle. Par contre, dans le cadre POMDP, quelques pistes ont été développées toutefois sans résultat convaincant pour une problématique réelle.

Dans ce contexte, le sujet de thèse de N. Drougard, en première année de doctorat à l'Onera, porte sur l'étude et la proposition d'un modèle fin capable de tenir compte des incertitudes (fréquentistes, imprécises et subjectives) concernant tant la croyance initiale que la dynamique du système, et ce afin d'améliorer les propriétés des politiques décisionnelles obtenues.

Cette seconde proposition de perspective de recherche nous semble la plus ambitieuse afin de tenter d'unifier les approches et modèles pour la perception et la prise de décision en contexte incertain et partiellement observable dans des applications robotiques.

7.3.2 Vers des améliorations des outils algorithmiques

Nouvelle paramétrisation de la fonction de valeur pour le calcul de politiques obtenues par des critères non-linéaires

Dans le chapitre 4, nous avons proposé un critère mixte pour l'optimisation de politiques POMDP. Nous avons implémenté cette approche en modifiant un algorithme de l'état de l'art (PBVI) du domaine des POMDP. Cette adaptation a été très délicate, puisque la fonction de valeur associée à ce critère mixte n'est pas linéaire, contrairement à celle utilisé dans PBVI, qui est linéaire par morceaux et paramétrée par des α-vecteurs. Pour réaliser cette adaptation, nous avons utilisé des approximations linéaires du premier ordre. Ces approximations linéaires engendrent des erreurs dans l'approximation de la valeur des états de croyance, en limitant fortement l'approche. Ceci a été également observé dans la thèse récente de M. Araya López [Araya López, 2013].

Nous pensons qu'en paramétrant la fonction de valeur autrement que par des α-vecteurs,

c'est-à-dire par un autre type de fonction plus générale (non nécessairement linéaire), il serait possible d'explorer d'autres types de critères, en particulier des critères non-linéaires de façon à ne plus reposer sur des approximations linéaires du premier ordre. Ce type de critère non-linéaire peut s'avérer utile pour des applications purement épistémiques, où l'utilisation du critère mixte non-linéaire permet dans certains cas de gagner en efficacité algorithmique par rapport à une modélisation POMDP classique avec ajout d'actions terminales de classification.

Couplage du modèle AC-POMDP avec le modèle MOMDP

Nous avons vu dans le chapitre 5 que le modèle AC-POMDP peut s'avérer très utile en robotique mobile, où le type de symbole d'observation concernant la faisabilité d'une action est typiquement obtenu par des capteurs spécifiques. Cette information, souvent découplée de la planification, est généralement traitée directement par le contrôleur d'exécution. Avec le modèle AC-POMDP, nous pouvons intégrer ce type d'information directement dans le modèle de décision en gardant sa sémantique naturelle. De plus, avec le schéma d'optimisation proposé, nous pouvons exploiter la structure du modèle afin d'accélérer certaines opérations dans le calcul de la politique.

Le modèle MOMDP explore aussi certaines propriétés structurelles du modèle afin de séparer les variables observables de celles partiellement observables. Cette nouvelle structure permet de définir l'espace d'état de croyance de manière plus compacte, et d'accélérer le calcul de la politique.

Dans le chapitre 5, nous avons souligné que le modèle AC-POMDP diffère du modèle MOMDP dans un aspect fondamental. Dans le modèle AC-POMDP, la fonction d'observation de l'ensemble Θ est une fonction surjective, c'est-à-dire qu'une observation $\theta \in \Theta$ peut être la même pour différents états. Au contraire, dans le modèle MOMDP, la fonction d'observation $\mathcal{O}^x$ est une fonction injective, l'observation $o^x \in \mathcal{O}^x$ étant égale à la valeur de la variable visible $o^x = x$ (cf. la section 2.5.2 du chapitre 2). Toutefois, nous pensons qu'un couplage des deux modèles peut être possible. Le modèle AC-POMDP, qui est plus général que le modèle MOMDP, peut aussi explorer le fait que certaines variables d'état sont visibles ; toutefois, la vérification des préconditions peut dépendre des variables d'états qui sont complètement observables et aussi des variables d'état qui ne le sont pas. Ainsi, si la vérification des préconditions doit se faire sur des variables partiellement observables, l'état de croyance du modèle dépendra aussi de ces variables. Mais cette piste de recherche intuitive nécessite une étude plus approfondie des équations d'optimisation.

Adaptation des algorithmes de recherche heuristique pour les AC-POMDP

L'évaluation du modèle AC-POMDP que nous avons menée dans le chapitre 5 implémente un algorithme de résolution de type *point-based* basé sur une recherche stochastique (PCVI). Toutefois, nous pensons que les algorithmes de recherche heuristique de résolution hors ligne et en ligne peuvent aussi être adaptés au modèle AC-POMDP. Cette piste de recherche requiert une étude approfondie des heuristiques qui guident la recherche dans l'espace d'état de croyance. Nous pensons que la version relaxée de l'algorithme PCVI, appelé PCVI2 et développée dans le chapitre 5, qui repose sur une borne inférieure de la valeur des états de croyance, peut proposer une piste pour une approximation de la borne inférieure des algorithmes de recherche heuristique. L'obtention d'une borne supérieure *étroite* nous semble par contre plus compliquée. L'approximation QMDP par exemple se base sur le MDP sous-jacent, et dans ce cas la vérification de l'applicabilité d'une action est directe ; toutefois, dans le cadre partiellement observable, les vérifications ne sont pas si évidentes (cf. discussion de la section 5.1).

Bibliographie

[Akmal Butt et Maragos, 1998] Akmal Butt, M. et Maragos, P. (1998). Optimum design of chamfer distance transforms. *IEEE Transactions on Image Processing*, 7(10):1477–1484.

[Araya López, 2013] Araya López, M. (2013). *Des algorithmes presque optimaux pour les problèmes de décision séquentielle à des fins de collecte d'information.* Thèse de doctorat, Université de Lorraine.

[Araya-López *et al.*, 2010] Araya-López, M., Buffet, O., Thomas, V. et Charpillet, F. (2010). A POMDP Extension with Belief-dependent Rewards. *Advances in Neural Information Processing Systems*, 23.

[Bahar *et al.*, 1997] Bahar, R., Frohm, E., Gaona, C., Hachtel, G., Macii, E., Pardo, A. et Somenzi, F. (1997). Algebric decision diagrams and their applications. *Formal methods in system design*, 10(2):171–206.

[Bai *et al.*, 2011] Bai, H., Hsu, D., Kochenderfer, M. et Lee, W. S. (2011). Unmanned Aircraft Collision Avoidance using Continuous-State POMDPs. *In Proceedings of Robotics : Science and Systems*, Los Angeles, CA, USA.

[Barto *et al.*, 1995] Barto, A., Bradtke, S. et Singh, S. (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1):81–138.

[Bass, 1967] Bass, J. (1967). *Éléments de calcul des probabilités téorique et appliqué.* Masson.

[Bertsekas et Castanon, 1999] Bertsekas, D. et Castanon, D. (1999). Rollout algorithms for stochastic scheduling problems. *Journal of Heuristics*, 5(1):89–108.

[Bonet et Geffner, 2003] Bonet, B. et Geffner, H. (2003). Labeled rtdp : Improving the convergence of real-time dynamic programming. *In Proc. ICAPS*, volume 3.

[Bonet et Geffner, 2009] Bonet, B. et Geffner, H. (2009). Solving POMDPs : RTDP-bel vs. point-based algorithms. *In Proceedings of the 21st International Joint Conference on Artifical intelligence (IJCAI)*, page 1641–1646, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Bonet et Gefner, 1998] Bonet, B. et Gefner, H. (1998). Solving large pomdps using real time dynamic programming. *In AAAI Fall Symposium on POMDPs.*

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

[Bouyssou *et al.*, 2009] Bouyssou, D., Dubois, D., Pirlot, M. et Prade, H. (2009). Decision-making process (cf. chapitre 3). *ISTE, London, UK & Wiley, Hoboken, NJ USA.*

[Bradski et Kaehler, 2008] Bradski, G. et Kaehler, A. (2008). *Learning OpenCV : Computer vision with the OpenCV library.* O'Reilly Media, Incorporated.

[Buffet et Aberdeen, 2005] Buffet, O. et Aberdeen, D. (2005). Robust planning with (L)RTDP. *In Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI)*.

[Buffet et Aberdeen, 2009] Buffet, O. et Aberdeen, D. (2009). The factored policy-gradient planner. *Artificial Intelligence*, 173(5):722–747.

[Burgard *et al.*, 1997] Burgard, W., Fox, D. et Thrun, S. (1997). Active mobile robot localization. *In Proc. of International Joint Conference on Artificial Intelligence (IJCAI)*. Morgan Kaufmann.

[Candido et Hutchinson, 2011] Candido, S. et Hutchinson, S. (2011). Minimum uncertainty robot navigation using information-guided POMDP planning. *In IEEE International Conference on Robotics and Automation (ICRA)*, pages 6102–6108.

[Carvalho Chanel *et al.*, 2010a] Carvalho Chanel, C., Farges, J., Teichteil-Königsbuch, F. et G.Infantes (2010a). POMDP solving : what rewards do you really expect at execution ? *In Proc. of the 5th Starting AI Researchers' Symposium*.

[Carvalho Chanel *et al.*, 2010b] Carvalho Chanel, C., Farges, J.-L., Teichteil-Königsbuch, F. et Infantes, G. (2010b). Optimisation de pomdp : quelles récompenses sont réellement attendues à l'exécution de la politique ? *In 5èmes Journées Francophones Planification, Décision, et Apprentissage pour la conduite de systèmes (JFPDA)*.

[Carvalho Chanel *et al.*, 2010c] Carvalho Chanel, C., Farges, J.-L., Teichteil-Königsbuch, F. et Infantes, G. (2010c). Optimisation des processus décisionnels de markov partiellement observables avec prise en compte explicite du gain d'information. *In 17ème congrès francophone AFRIF-AFIA Reconnaissance des Formes et Intelligence Artificielle*.

[Carvalho Chanel *et al.*, 2011a] Carvalho Chanel, C., Teichteil-Königsbuch, F., Infantes, G. et Fabiani, P. (2011a). Modeling action feasibility in pomdps with boolean-valued preconditions. *In IJCAI Workshop on Decision Making in Partially Observable, Uncertain Worlds : Exploring Insights from Multiple Communities*.

[Carvalho Chanel *et al.*, 2011b] Carvalho Chanel, C., Teichteil-Königsbuch, F., Infantes, G. et Fabiani, P. (2011b). Modélisation de la faisabilité d'action dans le pomdp avec des préconditions booléennes. *In 6èmes Journées Francophones Planification, Décision, et Apprentissage pour la conduite de systèmes (JFPDA)*.

[Carvalho Chanel *et al.*, 2012a] Carvalho Chanel, C., Teichteil-Königsbuch, F. et Lesire, C. (2012a). Détection et reconnaissance de cibles en ligne pour des UAV autonomes avec un modèle de type POMDP. *In 7èmes Journées Francophones Planification, Décision, et Apprentissage pour la conduite de systèmes (JFPDA)*.

[Carvalho Chanel *et al.*, 2012b] Carvalho Chanel, C., Teichteil-Königsbuch, F. et Lesire, C. (2012b). Planning for perception and perceiving for decision : POMDP-like online target detection and recognition for autonomous UAVs. *In Scheduling and Planning Applications woRKshop (SPARK) of ICAPS*.

[Carvalho Chanel *et al.*, 2012c] Carvalho Chanel, C., Teichteil-Königsbuch, F. et Lesire, C. (2012c). POMDP-based online target detection and recognition for autonomous UAVs. *In 20th European Conference on Artificial Intelligence (ECAI). Including Prestigious Applications of Artificial Intelligence (PAIS) and System Demonstrations Track*, volume 242 de *Frontiers in Artificial Intelligence and Applications*, pages 955–960. IOS Press.

[Carvalho Chanel *et al.*, 2013] Carvalho Chanel, C., Teichteil-Königsbuch, F. et Lesire, C. (2013). Multi-target detection and recognition by UAVs using online POMDPs. *In Proceedings of the Twenty-Seventh Conference on Artificial Intelligence (AAAI), July 14–18, Bellevue, Washington, USA (to appear). AAAI Press*.

[Cassandra, 1998] Cassandra, A. (1998). *Exact and approximate algorithms for partially observable Markov decision processes*. Thèse de doctorat, Brown University Providence, RI, USA.

[Cassandra, 2005] Cassandra, A. R. (2003-2005). POMDP's homepage. `http://www.pomdp.org/pomdp/index.shtml`.

[Cover et Thomas, 2006] Cover, T. et Thomas, J. (2006). *Elements of information theory*. Wiley-interscience.

[Davis et Putnam, 1960] Davis, M. et Putnam, H. (1960). A computing procedure for quantification theory. *Journal of the ACM (JACM)*, 7(3):201–215.

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

[Deb, 2001] Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons Hoboken, NJ.

[Defretin *et al.*, 2010] Defretin, J., Herbin, S., Le Besnerais, G. et Vayatis, N. (2010). Adaptive planification in active 3d object recognition for many classes of objects. *In Workshop "Towards Closing the Loop : Active Learning for Robotics", RSS Robotics : Science and Systems Conference*.

[Deguchi et Ohtsu, 2006] Deguchi, K. et Ohtsu, H. (2006). An information theoretic approach for active and effective object recognitions. *In Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 2, pages 622–622. IEEE.

[Deinzer *et al.*, 2003] Deinzer, F., Denzler, J. et Niemann, H. (2003). Viewpoint selection-planning optimal sequences of views for object recognition. *Lecture notes in computer science*, pages 65–73.

[Deinzer *et al.*, 2006] Deinzer, F., Derichs, C., Niemann, H. et Denzler, J. (2006). Integrated viewpoint fusion and viewpoint selection for optimal object recognition. *In Proceedings of the British Machine Vision Conference*.

[Denzler et Brown, 2002] Denzler, J. et Brown, C. (2002). Information theoretic sensor data selection for active object recognition and state estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(2):145–157.

[Deutsch *et al.*, 2004] Deutsch, B., Zobel, M., Denzler, J. et Niemann, H. (2004). Multi-step entropy based sensor control for visual object tracking. *Pattern Recognition*, pages 359–366.

[Djian *et al.*, 1995] Djian, D., Probert, P. et Rives, P. (1995). Active sensing using bayes nets. *In Proc. of Int. Conf. on Advanced Robotics, ICAR'95*, pages 895–902.

[Dutta Roy *et al.*, 2004] Dutta Roy, S., Chaudhury, S. et Banerjee, S. (2004). Active recognition through next view planning : a survey. *Pattern Recognition*, 37(3):429–446.

[Eidenberger *et al.*, 2008] Eidenberger, R., Grundmann, T., Feiten, W. et Zoellner, R. (2008). Fast parametric viewpoint estimation for active object detection. *In Proc. of the IEEE International Conference on Multisensor of Fusion and Integration for Intelligent Systems (MFI 2008), Seoul, Korea*.

[Eidenberger *et al.*, 2009] Eidenberger, R., Grundmann, T. et Zoellner, R. (2009). Probabilistic action planning for active scene modeling in continuous high-dimensional domains. *In IEEE International Conference on Robotics and Automation (ICRA)*, pages 2412–2417.

[Eidenberger et Scharinger, 2010] Eidenberger, R. et Scharinger, J. (2010). Active perception and scene modeling by planning with probabilistic 6d object poses. *In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1036–1043.

[Fabiani, 1994] Fabiani, P. (1994). A new approach in temporal representation of belief for autonomous observation and surveillance systems. *In European Conference on Artificial Intelligence (ECAI)*, pages 391–395.

[Feng et al., 2002] Feng, Z. et et al. (2002). Symbolic lao* search for factored markov decision processes. *In In proceeding of the AIPS-02 Workshop on planning via model checking*, pages 49–53.

[Feng *et al.*, 2002] Feng, Z., Hansen, E. et Zilberstein, S. (2002). Symbolic generalization for on-line planning. *In Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 209–216. Morgan Kaufmann Publishers Inc.

[Filliat et Meyer, 2000] Filliat, D. et Meyer, J.-A. (2000). Active perception and map learning for robot navigation. *In Proc. of the Sixth Int. Conf. on Simulation of Adaptive Behavior*, pages 246–255. MIT Press.

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

[Givan *et al.*, 2000] Givan, R., Leach, S. et Dean, T. (2000). Bounded-parameter markov decision processes. *Artificial Intelligence*, 122(1-2):71–109.

[Hansen et Zilberstein, 2001] Hansen, E. A. et Zilberstein, S. (2001). LAO * : A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129(1-2):35–62.

[Hansson et Jonsson, 1994] Hansson, H. et Jonsson, B. (1994). A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535.

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

[Howard et Matheson, 2005] Howard, R. et Matheson, J. (2005). Influence diagrams. *Decision Analysis*, 2(3):127–143.

[Ingrand *et al.*, 2007] Ingrand, F., Lacroix, S., Lemai-Chenevier, S. et Py, F. (2007). Decisional autonomy of planetary rovers. *Journal of Field Robotics*, 24(7):559–580.

[Julier *et al.*, 2000] Julier, S., Uhlmann, J. et Durrant-Whyte, H. (2000). A new method for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Transactions on Automatic Control*, 45(3):477–482.

[Kaelbling *et al.*, 1998] Kaelbling, L., Littman, M. et Cassandra, A. (1998). Planning and acting in partially observable stochastic domains. *AIJ*, 101(1-2).

[Kolobov *et al.*, 2011] Kolobov, A., Mausam, Weld, D. S. et Geffner, H. (2011). Heuristic search for generalized stochastic shortest path MDPs. *In ICAPS*.

[Kurniawati *et al.*, 2008] Kurniawati, H., Hsu, D. et Lee, W. (2008). SARSOP : Efficient point-based POMDP planning by approximating optimally reachable belief spaces. *In Proc. RSS*.

[Littman, 1994] Littman, M. (1994). The witness algorithm : Solving partially observable markov decision processes. *Brown University, Providence, RI*.

[Littman, 1996] Littman, M. (1996). *Algorithms for sequential decision making*. Thèse de doctorat, Brown University.

[Littman, 1997] Littman, M. (1997). A. Cassandra and N. Zhang. Incremental pruning : A simple, fast, exact algorithm for partially observable Markov decision processes. *In Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI)*.

[Littman *et al.*, 1995] Littman, M., Cassandra, A. et Pack Kaelbling, L. (1995). Learning policies for partially observable environments : Scaling up. *In International Conference on Machine Learning*, pages 362–370.

[Littman *et al.*, 2001] Littman, M., Majercik, S. et Pitassi, T. (2001). Stochastic boolean satisfiability. *Journal of Automated Reasoning*, 27(3):251–296.

[López *et al.*, 2010] López, M. A., Thomas, V., Buffet, O. et Charpillet, F. (2010). A closer look at MOMDPs. *In Proceedings of the Twenty-Second IEEE International Conference on Tools with Artificial Intelligence (ICTAI-10)*.

[Lovejoy, 1991] Lovejoy, W. (1991). Computationally feasible bounds for partially observed Markov decision processes. *Operations research*, pages 162–175.

[Marchand et Chaumette, 1997] Marchand, E. et Chaumette, F. (1997). A bayes nets-based prediction/verification scheme for active visual reconstruction. *Computer Vision—ACCV'98*, pages 648–655.

[McAllester et Singh, 1999] McAllester, D. et Singh, S. (1999). Approximate planning for factored pomdps using belief state simplification. *In Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 409–416. Morgan Kaufmann Publishers Inc.

[Mihaylova *et al.*, 2002] Mihaylova, L., Lefebvre, T., Bruyninckx, H., Gadeyne, K. et Schutter, J. D. (2002). Active sensing for robotics – a survey. *In 5th Intl Conf. On Numerical Methods and Applications*, pages 316–324.

[Miller *et al.*, 2009] Miller, S. A., Harris, Z. A. et Chong, E. K. P. (2009). A POMDP framework for coordinated guidance of autonomous UAVs for multitarget tracking. *EURASIP J. Adv. Signal Process*, pages 2 :1–2 :17.

[Monahan, 1982] Monahan, G. (1982). A survey of partially observable Markov decision processes : Theory, models, and algorithms. *Management Science*, pages 1–16.

[Murphy, 2002] Murphy, K. (2002). *Dynamic Bayesian networks : representation, inference and learning*. Thèse de doctorat, University of california.

[Ni et Liu, 2008] Ni, Y. et Liu, Z. (2008). Bounded-parameter partially observable markov decision processes. *In Proc. of the 18th International Conference on Automated Planning and Scheduling (ICAPS*, pages 240–247.

[Nilsson, 1980] Nilsson, N. (1980). Principles of artificial intelligence, tioga pub. *Co., Palo Alto, CA*.

[Ong *et al.*, 2009] Ong, S. C., Png, S. W., David et Lee, H. W. S. (2009). POMDPs for robotic tasks with mixed observability. *In Proceedings of Robotics : Science and Systems (RSS)*.

[Paletta et Pinz, 2000] Paletta, L. et Pinz, A. (2000). Active object recognition by view integration and reinforcement learning. *Robotics and Autonomous Systems*, 31:71–86.

[Papadimitriou et Tsitsiklis, 1987] Papadimitriou, C. et Tsitsiklis, J. (1987). The complexity of Markov decision processes. *Mathematics of OR*, 12(3):441–450.

[Paquet *et al.*, 2006] Paquet, S., Chaib-draa, B. et Ross, S. (2006). Hybrid POMDP algorithms. *In Proceedings of The Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains (MSDM)*, pages 133–147.

[Pineau *et al.*, 2003] Pineau, J., Gordon, G. et Thrun, S. (2003). Point-based value iteration : An anytime algorithm for POMDPs. *In Proc. of International Joint Conference on Artificial Intelligence (IJCAI)*.

[Poupart, 2005] Poupart, P. (2005). *Exploiting structure to efficiently solve large scale partially observable Markov decision processes*. Thèse de doctorat, University of Toronto.

[Poupart *et al.*, 2011] Poupart, P., Kim, K. et Kim, D. (2011). Closing the gap : Improved bounds on optimal pomdp solutions. *In International Conference on Automated Planning and Scheduling (ICAPS)*.

[Pralet *et al.*, 2010a] Pralet, C., Schiex, T. et Verfaillie, G. (2010a). *Sequential Decision-Making Problems : Representation and Solution*, volume 1. ISTE Ltd and Jhon Wiley & Sons, Inc.

[Pralet et Verfaillie, 2008] Pralet, C. et Verfaillie, G. (2008). Using constraint networks on timelines to model and solve planning and scheduling problems. *In Proc. ICAPS*.

[Pralet *et al.*, 2010b] Pralet, C., Verfaillie, G., Lemaître, M. et Infantes, G. (2010b). Constraint-based controller synthesis in non-deterministic and partially observable domains. *ECAI 2010 - 19th European Conference on Artificial Intelligence, Lisbon, Portugal, August 16-20*, pages 681–686.

[Puterman, 1994] Puterman, M. (1994). *Markov decision processes : discrete stochastic dynamic programming*. John Wiley & Sons, Inc. New York, NY, USA.

[Ross et Chaib-Draa, 2007] Ross, S. et Chaib-Draa, B. (2007). AEMS : An anytime online search algorithm for approximate policy refinement in large POMDPs. *In Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2592–2598.

[Ross *et al.*, 2008a] Ross, S., Pineau, J. et Chaib-draa, B. (2008a). Theoretical analysis of heuristic search methods for online pomdps. *Advances in neural information processing systems*, 20:1216.

[Ross *et al.*, 2011] Ross, S., Pineau, J., Chaib-draa, B. et Kreitmann, P. (2011). A bayesian approach for learning and planning in partially observable markov decision processes. *Journal of Machine Learning Research*, 12:1729–1770.

[Ross *et al.*, 2008b] Ross, S., Pineau, J., Paquet, S. et Chaib-Draa, B. (2008b). Online planning algorithms for pomdps. *Journal of Artificial Intelligence Research*, 32(1):663–704.

[Sabbadin *et al.*, 2007] Sabbadin, R., Lang, J. et Ravoanjanahary, N. (2007). Purely epistemic markov decision processes. *In Proceedings of the 22nd national conference on Artificial intelligence - Volume 2*, page 1057–1062. AAAI Press.

[Sanner, 2010] Sanner, S. (2010). Relational Dynamic Influence Diagram Language (RDDL) : Language Description. Tutorial disponible sur http ://users.cecs.anu.edu.au/∼ssanner/IPPC_2011/RDDL.pdf.

[Satia et Lave, 1973] Satia, J. et Lave, R. (1973). Markovian decision processes with probabilistic observation of states. *Management Science*, 20(1):1–13.

[Saux et Sanfourche, 2011] Saux, B. et Sanfourche, M. (2011). Robust vehicle categorization from aerial images by 3d-template matching and multiple classifier system. *In 7th International Symposium on Image and Signal Processing and Analysis (ISPA)*, pages 466–470.

[Schesvold *et al.*, 2003] Schesvold, D., Tang, J., Ahmed, B., Altenburg, K. et Nygard, K. (2003). POMDP planning for high level UAV decisions : Search vs. strike. *In Proceedings of the 16th International Conference on Computer Applications in Industry and Enineering*.

[Shani *et al.*, 2007] Shani, G., Brafman, R. et Shimony, S. (2007). Forward search value iteration for pomdps. *In Proc. International Joint Conference on Artificial Intelligence (IJCAI)*.

[Shani *et al.*, 2008] Shani, G., Poupart, P., Brafman, R. et Shimony, S. (2008). Efficient add operations for point-based algorithms. *In The International Conference on Automated Planning and Scheduling (ICAPS)*.

[Sigaud et Buffet, 2008] Sigaud, O. et Buffet, O. (2008). *Processus Décisionnels de Markov en intelligence artificielle*, volume 1. Lavoisier and Hermes Sciences.

[Sim *et al.*, 2008a] Sim, H., Kim, K., Kim, J., Chang, D. et Koo, M. (2008a). Symbolic heuristic search value iteration for factored pomdps. *In Proc. Nat. Conf. on Artificial Intelligence*, pages 1088–1093.

[Sim *et al.*, 2008b] Sim, H., Kim, K., Kim, J., Chang, D. et Koo, M. (2008b). Symbolic heuristic search value iteration for factored pomdps. *In Proc. of the 23th AAAI Conference on Artificial Intelligence*, pages 1088–1093.

[Simmons et Koenig, 1995] Simmons, R. et Koenig, S. (1995). Probabilistic robot navigation in partially observable environments. *In International Joint Conference on Artificial Intelligence (IJCAI)*, volume 14, pages 1080–1087.

[Smallwood et Sondik, 1971] Smallwood, R. et Sondik, E. (1971). The optimal control of partially observable Markov processes. *Stanford University, Stanford, California, Eletronics Labs.*

[Smallwood et Sondik, 1973] Smallwood, R. et Sondik, E. (1973). The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, pages 1071–1088.

[Smith et Simmons, 2004] Smith, T. et Simmons, R. (2004). Heuristic search value iteration for POMDPs. *In Proc. UAI.*

[Smith et Simmons, 2005] Smith, T. et Simmons, R. (2005). Point-based POMDP algorithms : Improved analysis and implementation. *In Proc. UAI.*

[Soetens et Bruyninckx, 2005] Soetens, P. et Bruyninckx, H. (2005). Realtime hybrid task-based control for robots and machine tools. *In International Conference on Robotics and Automation (ICRA)*, Barcelona, Spain.

[Spaan, 2008] Spaan, M. (2008). Cooperative Active Perception using POMDPs. *Association for the Advancement of Artificial Intelligence - AAAI.*

[Spaan et Lima, 2009] Spaan, M. et Lima, P. (2009). A decision-theoretic approach to dynamic sensor selection in camera networks. *In Int. Conf. on Automated Planning and Scheduling*, pages 279–304.

[Spaan et Vlassis, 2004] Spaan, M. et Vlassis, N. (2004). A point-based POMDP algorithm for robot planning. *In IEEE International Conference on Robotics and Automation (ICRA).*

[Spaan et Vlassis, 2005] Spaan, M. et Vlassis, N. (2005). Perseus : Randomized point-based value iteration for POMDPs. *JAIR*, 24:195–220.

[Sridharan et Stone, 2005] Sridharan, M. et Stone, P. (2005). Real-time vision on a mobile robot platform. *In International Conference on Intelligent Robots and Systems (IROS 2005)*, pages 2148–2153. IEEE.

[Sridharan *et al.*, 2008] Sridharan, M., Wyatt, J. et Dearden, R. (2008). HiPPo : Hierarchical POMDPs for Planning Information Processing and Sensing Actions on a Robot. *In Proc. of ICAPS.*

[Taha *et al.*, 2011] Taha, T., Miró, J. V. et Dissanayake, G. (2011). A POMDP framework for modelling human interaction with assistive robots. *In IEEE International Conference on Robotics and Automation (ICRA)*, pages 544–549. IEEE.

[Tarim *et al.*, 2006] Tarim, S., Manandhar, S. et Walsh, T. (2006). Stochastic constraint programming : A scenario-based approach. *Constraints*, 11(1):53–80.

[Teichteil-Königsbuch, 2012] Teichteil-Königsbuch, F. (2012). Path-constrained markov decision processes : bridging the gap between probabilistic model-checking and decision-theoretic planning. *In ECAI*, pages 744–749.

[Teichteil-Königsbuch *et al.*, 2010] Teichteil-Königsbuch, F., Kuter, U. et Infantes, G. (2010). Incremental plan aggregation for generating policies in MDPs. *In Proc. AAMAS*, pages 1231–1238.

[Teichteil-Konigsbuch *et al.*, 2011] Teichteil-Konigsbuch, F., Lesire, C. et Infantes, G. (2011). A generic framework for anytime execution-driven planning in robotics. *In IEEE International Conference on Robotics and Automation (ICRA)*, pages 299–304.

[Thrun *et al.*, 2005] Thrun, S., Burgard, W., et Fox, D. (2005). *Probabilistic Robotics*. MIT Press, Cambridge, MA.

[Walsh, 2002] Walsh, T. (2002). Stochastic constraint programming. *In ECAI*, volume 2, pages 111–115.

[Wang *et al.*, 2012] Wang, J., Zhang, Y., Lu, J. et Xu, W. (2012). A Framework for Moving Target Detection, Recognition and Tracking in UAV Videos. *In* Luo, J., éditeur : *Affective Computing and Intelligent Interaction*, volume 137 de *Advances in Intelligent and Soft Computing*, pages 69–76. Springer Berlin / Heidelberg.

[Washington, 1997] Washington, R. (1997). Bi-pomdp : Bounded, incremental partially-observable markov-model planning. *Recent Advances in AI Planning*, pages 440–451.

[Williams *et al.*, 2005] Williams, J., Poupart, P. et Young, S. (2005). Factored partially observable Markov decision processes for dialogue management. *In 4th Workshop on Knowledge and Reasoning in Practical Dialog Systems, International Joint Conference on Artificial Intelligence (IJCAI)*.

[Younes et Littman, 2003] Younes, H. et Littman, M. (2003). PPDDL1.0 : An extension to PDDL for expressing planning domains with probabilistic effects. *In In Proc. of ICAPS*.

[Yu *et al.*, 2009] Yu, S., Krishnapuram, B., Rosales, R. et Rao, R. (2009). Active sensing. *In In proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 5 of JMLR, W&CP 5.

[Zhang et Zhang, 2001] Zhang, N. L. et Zhang, W. (2001). Speeding up the convergence of value iteration in partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 14:29–51.