Camille BESSE

Contraintes et observabilité dans les systèmes de Markov décentralisés

Thèse présentée
à la Faculté des études supérieures de l'Université Laval
dans le cadre du programme de doctorat en Informatique
pour l'obtention du grade de Philosophiæ Doctor (Ph.D.)

Département d'Informatique et de Génie Logiciel
Faculté des Sciences et de Génie
UNIVERSITÉ LAVAL
QUÉBEC

Mai 2010

À mes parents, mes frères, mes grands-parents et mes amis
pour avoir fait de moi ce que je suis aujourd'hui.

Résumé

De manière générale, les problèmes séquentiels de décisions multiagents sont très difficiles à résoudre surtout lorsque les agents n'observent pas parfaitement ni complètement l'état de leur environnement. Les modèles actuels pour représenter ces problèmes restent à ce jour très généraux et difficilement applicables dans les multiples applications possibles.

Nous proposons dans cette thèse plusieurs approches de réduction de la complexité computationnelle et en pire cas de ces modèles. Une première approche se base sur l'utilisation de contraintes sur l'espace des actions possibles que les différents agents du système peuvent entreprendre. Cette utilisation de connaissances *a priori* dans la modélisation au travers de modèles déjà connus, mais non appliqués à la prise de décision séquentielle permet une réduction significative d'un des facteurs de la complexité algorithmique.

La seconde approche consiste à restreindre les possibilités d'observations de l'agent à un ensemble similaire à l'espace d'états utilisé pour représenter son environnement. De cette manière, nous montrons que les agents peuvent converger rapidement en probabilité vers des croyances communes sans nécessairement avoir à communiquer. Dans ce contexte, nous avons également développé un algorithme permettant alors aux agents de se coordonner au moment de l'exécution lorsqu'il n'existe pas de communication explicite.

Enfin, nous avons entrepris la mise en application de telles réductions à deux problèmes. Un premier problème de patrouille multiagent est considéré et modélisé, et un second problème lié à l'apprentissage de POMDPs continus dans des cas précis de transition et d'observabilité est également abordé. Les résultats obtenus montrent que dans certains cas de problèmes de coordination, la communication -- lorsqu'elle est disponible -- est non négligeable, et que dans le cas de l'apprentissage de POMDPs, considérer le quasi-déterminisme du modèle permet l'apprentissage de converger.

Abstract

In general, multiagent sequential decision making problems are very hard to solve, especially when agents do not perfectly and completely observe the state of their environment. Current models that are used to represent these problems are still too general and hardly applicable to the multiple possible applications.

In this thesis, we propose several approaches in order to reduce computational complexity and worst-case complexity of such models. We first propose a approach based on constraint utilization on the different action spaces that agents have. This use of *a priori* knowledge in the modeling process through already known and mastered techniques of constraint optimization in the case of sequential decision making allows to significantly reduce one of the factor of the algorithmic complexity in Markov models : the branching factor through the action space.

A second approach consists in restricting the observation space of the agents to a space similar to the state space they used to represent their environment. In this way, we show that agents will thus all quickly converge in probability to some common beliefs without necessarily having to communicate. In this context we also developed an online algorithm, allowing agents to coordinate at exécution time, even when no explicit communication is available.

Finally, we start to apply these complexity reduction techniques to two sequential decision making problems. A first multiagent patrolling problem is considered and modeled. A second problem of continuous POMDP learning in specific cases of transition and observation functions is also addressed. Experimental results mainly that in coordination problems, communication -- when available -- is not an option, and that in case of POMDP learning, considering model quasi-determinism allows faster convergence of this learning.

Avant-propos

Avant d'entrer dans le vif du sujet, il est d'usage de rappeler en avant-propos à quel point écrire une thèse de doctorat est un exercice long, fastidieux et difficile. Il s'agit en effet de présenter en quelques pages le résultat de plusieurs années de travaux de recherche et d'une vie plus ou moins sociale. Là où l'ensemble du document que vous tenez entre vos mains est la digne représentation du premier point, l'avant-propos est là pour rappeler au lecteur que le second point existe tout de même et que sans cette vie plus ou moins sociale, de nombreux étudiants-chercheurs auraient viré fou.

Cette partie se concentre donc à présenter et à remercier toutes les personnes qui m'ont accompagné dans mon exil volontaire ou que j'ai rencontré, tous ceux qui sont restés derrière moi et avec qui je suis resté en contact, bref tous ceux qui ont participé à ma vie sociale ces cinq dernières années.

Je tenais tout d'abord à remercier Brahim Chaib-draa, mon directeur de recherche, pour m'avoir fait découvrir de nombreux domaines de recherche et méthodes de travail auxquelles je n'était pas encore assez rompu. Je le remercie de l'avoir su m'encourager quand il le fallait et d'avoir su donner des coups de pieds au cul aux moments où il le fallait. Il a réussi à sortir le meilleur de moi pour m'aider à mener ce projet à son terme.

Je tiens ensuite à remercier la famille proche (Isabelle, Philippe, Aurèle, Baptiste, Cécile, Pierre, Régine) pour m'avoir laissé partir à 5300km et pour m'avoir visité et/ou m'avoir soutenu pendant cette longue période. Je remercie également l'oncle, les tantes et les cousines (Françoise, Jean Marc, Claire, Lucile, Guillemine) pour leur soutien à distance et leur intérêt pour mes incompréhensibles travaux. pour finir le chapitre familial, et bien que pas de la famille réellement, je remercie Marion pour être venue me voir et je remercie Nico, mon frère de tête qui est venu me rejoindre pour faire ses études au Québec également. Finalement, je remercie Ariane et sa famille pour leur soutien pendant 3 ans, et je remercie en dernier Kati et sa famille pour la dernière ligne droite. Je vous aime tous beaucoup.

Je n'oublierai pas tous les étudiants du DAMAS qui m'ont "entertainé" pendant mon séjour à l'université. Dans l'ordre alphabétique j'aimerais remercier : Frédérick Asselin, Abdeslam Boularias, Andriy Burkov, Sébastien Chouinard, Patrick Cinq-Mars, Patrick Dallaire, Charles Desjardins, Jilles Dibangoye, Olivier Gagné, Guillaume Gras, Pierre-Luc Grégoire, Pierr Kreitmann, Maxime Lemonnier, Julien Laumônier, Jean-Samuel Marier, Pierrick Plamondon, et Stéphane Ross. J'ajouterais des mentions spéciales à Pat

et JS pour leurs discussions animées et Julien pour toutes les activités hors universités et les rencontres qu'il m'a permis de faire, notamment Martine Lapointe, Jean-Sébastien Martel, Nathalie Fontalvo et François Lapointe.

Pour mes premières soirées au Québec et mes premières rencontres hors-professionnelles, j'aimerais remercier mes (nombreux) colocs : Adriana Amador, Marie-Pierre Côté, Julie Dubois, Ouail Essaadouni, Philippe Genest, Damian Kotzev, Nicolas Lachance-Bernard et Marianne Lusignan.

Hors contexte familial et professionnel j'aimerais tout particulièrement remercier François Bégin et Yamilah Silva de l'école de danse Dos Con Dos qui m'ont permis de découvrir plusieurs personnes formidables et une seconde famille. Pour toutes ces soirées mémorables, ces voyages fantastiques et ces délires sympathiques, j'aimerais également remercier : Mélanie Alain, Mélanie Beaudoin, Chantale Blanchette, Sonia Claveau, Maxime Daoust-Hébert, Joëlle DeBlois, Marjorie DeBlois, Karim Diallo, Manon Doré, Nicolas Fortin, Christian Fournier, Dominic Genest, Stéphane Genest, Claude Hamel, Marijo Labadie, Marie-Hélène Laflèche, Catherine Leclerc, Jason Martel, Dave Massicote, Katharine Ouellet, Frédéric Poisson, Julie Poitras, Isabelle Rousseau-Nepton, Guy-Ann Roy, Carolanne Simard, Mathieu St-Amant, Marie-Josée Theriault et Mélanie Turcotte. J'aurais deux dédicaces toutes particulières à deux amis qui nous ont quitté prématurément et que je garderais pour toujours près du coeur : Mario Greendale et David-Emmanuel Bouchard.

J'aimerais ensuite remercier tout ceux qui m'ont accompagné de loin ou qui m'attendaient quand je revenais en France : Meriem Barrada, Cécile Baux, Anne Battle, Antoine Benain, Agneta Chadee, Sonia Dehimi, Céline Fouron, Jérémie Labbé, François Lienard, Stéphane Mas, Julie Maurice, Sébastien Maurice, Stéphane Plais, Delphine Pons et Marie-Laure Thomas. Une mention toute spéciale est adressée à Sébastien Giordano et plus généralement tout monde du club de Judo de Ramonville St-Agne ainsi qu'à Emmanuel Rachelson, Cédric Bonnet et tous ceux du laboratoire DCSD-ONERA qui m'ont accueilli à de nombreuses reprises dans leur laboratoire lors de mes longs séjours en terre natale.

J'aimerais enfin remercier mes rapporteurs Luc Lamontagne, Sébastien Paquet, Soumaya Cherkaoui pour avoir pris le temps de lire ce rébarbatif ouvrage et de m'avoir pointé les différentes lacunes de mes recherches tout en sachant trouver les mots justes pour me récompenser. Un remerciement spécial pour Nadir Belkhiter qui a su mettre sa touche au très protocolaire discours de félicitations du jury.

Tout ce monde là (et d'autres encore) m'ont accompagnés durant ces cinq premières années au Québec et pour cela je les en remercie beaucoup en espérant continuer à les côtoyer et à avoir du plaisir avec eux.

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

Chapitre 1

Introduction

«A human being is a deciding being.»

Man's Search for Meaning -- Viktor E. Franckl

Ce chapitre d'introduction développe le contexte général des recherches exposées dans cette thèse. La question de la prise de décision est tout d'abord expliquée et formalisée sous plusieurs de ses aspects. La discussion est ensuite restreinte autour de la prise de décisions pour des entités autonomes lorsque plusieurs issues sont possibles aux décisions et/ou lorsque l'information nécessaire à la prise de décision n'est pas complètement disponible. Plusieurs exemples de problèmes typiques sont alors présentés, permettant ainsi de comprendre dans les grandes lignes les contributions et l'organisation de cette thèse.

Chaque jour, chacun d'entre nous doit faire des choix multiples de toute nature et de toute importance : depuis le simple choix de ses repas quotidiens, jusqu'aux choix de sa future maison et du prêt hypothécaire qui va avec. Ainsi, prendre des décisions est une activité parfois complexe que nous faisons plus ou moins naturellement à chaque instant de notre existence. Dans le cadre de l'intelligence artificielle, la reproduction et l'analyse de ces situations de choix et les comportements à adopter selon les opportunités ont intéressé les chercheurs depuis plusieurs décennies. C'est ainsi qu'identifier et résoudre ces situations de choix, au travers de leurs caractéristiques, de leurs contraintes ou de leurs conséquences est devenu au fil du temps un des credos de l'intelligence artificielle moderne.

1.1 Prendre des décisions

De nos jours, l'intelligence artificielle s'attache à modéliser, formaliser et concevoir des entités logicielles autonomes et ''intelligentes'', appelées *agents*, qui interagissent avec un *environnement* au travers de *percepts* et d'*actions* avec pour objectif de se comporter de manière *satisfaisante* ou *rationnelle*.

Ces agents intelligents cherchent donc à optimiser un certain *critère de performance* en interagissant avec leur environnement. Ces interactions, qui se font au travers des capteurs et des actuateurs des agents, décomposent le processus de décision en trois étapes essentielles (cf. figure 1.1) : *percevoir, décider, agir*.

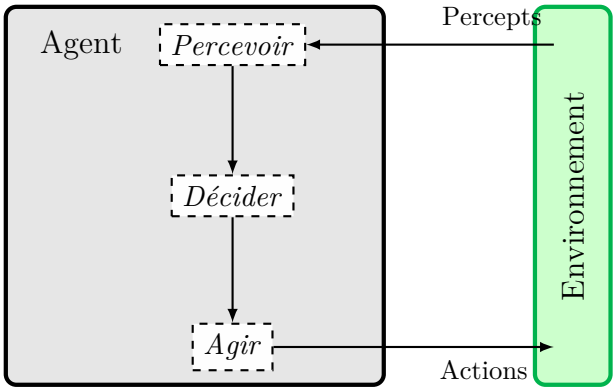


FIGURE 1.1 – Modèle d'un agent intelligent [Russell et Norvig, 2009].

L'accumulation de percepts permet aux agents d'*identifier* la situation courante de l'environnement, de prendre la bonne décision ou de *résoudre* le problème courant, avant d'appliquer la solution trouvée par l'exécution du bon plan d'action.

1.1.1 Identifier ...

Il n'est pas toujours clair pour l'agent de savoir quelles sont exactement toutes les informations qui pourraient lui être nécessaire à la prise d'une décision satisfaisante. L'agent peut en effet être dépourvu de certaines informations, ou pire, avoir certaines informations erronées. Prendre une décision rationnelle relativement à une mesure de performance fournie à l'agent devient alors un problème complexe où l'incertain sur la situation courante se mêle à l'estimation de la performance conséquente à la décision courante.

Ainsi, identifier la situation courante, ou l'*état* courant, de l'environnement est effectivement difficile. Dans un contexte humain par exemple, il est très rare d'avoir accès à l'ensemble de l'information nécessaire pour pouvoir prendre une décision éclairée et l'on fait souvent appel à l'expérience passée de situations ayant les mêmes caractéristiques. Dans un contexte logiciel, robotique par exemple, les capteurs actuels ne permettent pas d'identifier toujours avec exactitude la localisation du robot ou sa situation environnante. Extraire de l'information de caméras vidéo ou de capteurs odométriques est souvent difficile et induit une incertitude sur l'information extraite. Beaucoup de modèles actuels supposent que l'agent accède complètement et avec certitude à l'état de son environnement, même si en pratique sa vision n'est que partielle et/ou probabiliste.

1.1.2 ... et résoudre ...

Résoudre un problème de décision revient alors, une fois l'état du système quasiment identifié, à trouver une manière de se rendre dans une situation *satisfaisante* et/ou à *optimiser* une certaine fonction de satisfaction qui indique les ''bons'' et les ''mauvais'' comportements à adopter dans tel ou tel état. Cette fonction peut être un critère de *satisfaction* ou bien d'*optimisation* selon que l'on cherche seulement à trouver une situation satisfaisante ou bien à ordonner les solutions puis à prendre la plus satisfaisante (l'optimale). Il est également possible de ne trouver qu'une solution proche de la solution optimale, mais ''suffisamment'' satisfaisante pour l'agent. Tout cela est déterminé par la *mesure de performance* du problème généralement représentée sous la forme d'une *utilité à maximiser* pour l'agent.

Il convient alors de remarquer que certaines décisions peuvent alors mener dans le futur à des états non satisfaisants ou à des impasses qu'il faut alors éviter. L'ensemble de la *séquence* des états futurs et des décisions associées devient importante pour la prise de décision dans un état donné. Prédire les issues possibles de ces décisions devient alors primordial pour la résolution du problème. Pour le résoudre, il faut donc posséder une manière de représenter l'état de l'environnement et de l'agent, les décisions possibles ainsi que la *dynamique du système* -- qui décrit comment le système évolue sous l'influence des agents -- lorsqu'une décision est prise dans un état donné.

1.1.3 ... en présence d'incertitudes

Comme il a été évoqué plus haut, il est très fréquent que les actions envisagées par l'agent ne se déroulent pas comme prévu [Bloch, 1977], et il faut alors prévoir toutes les

issues possibles à ces actions selon leurs probabilités d'occurrence. Ainsi, plutôt que de raisonner sur une seule des issues de chacune des actions entreprises, l'agent raisonne *en espérance* -- i.e en moyenne -- sur l'ensemble des issues possibles à chaque action.

D'autre part, lorsque l'agent n'est pas certain d'avoir parfaitement identifié l'état courant de l'environnement, il doit également raisonner sur tous les états probables du système afin de s'assurer de ne pas effectuer d'action qui pourrait lui nuire selon son utilité ou sa mesure de performance. Ce genre de situations apparaît dès lors que l'agent n'a pas une vision *complète* du système. On parle alors de problème *partiellement observable*. Ce type de problème survient notamment dans les problèmes à plusieurs agents lorsqu'un agent n'a pas connaissance des états internes des autres agents tels que le niveau d'énergie, la position exacte, etc. Il peut également survenir lorsque l'agent ne possède que des capteurs bruités, ne lui permettant ainsi que de saisir des bribes de son environnement.

1.1.4 En résumé

Pour résumer, les types de problèmes qui vont être discutés dans cette thèse possèdent tous les caractéristiques suivantes :

1. Les décisions devront être prises *séquentiellement* ;
2. Ces décisions impliquant *un ou plusieurs agents*, selon le problème considéré ;
3. Ces agents seront en interaction avec un environnement *stochastique et possiblement partiellement observable* ;
4. Et ils chercheront à maximiser une mesure de performance *commune* par interaction avec l'environnement.

Il convient ici de remarquer que l'hypothèse 4 élimine les problèmes impliquant plusieurs agents agissant les uns contre les autres puisque tous les agents maximisent la même mesure de performance et tentent donc de coopérer et se coordonner.

Pour résoudre le type de problèmes décrit ci-dessus, les agents doivent avoir une *représentation* commune de l'état du système et de sa dynamique. Voyons comment formaliser cette représentation.

1.2 Formaliser les décisions

La représentation du problème doit à la fois permettre de modéliser chacun des états du système, chacune des décisions possibles, la dynamique de l'interaction entre les agents et l'environnement, et la mesure de performance associée à chaque décision dans chacun des états. Dans le cas où l'état serait partiellement observable, il convient également de spécifier les parties observables des parties qui ne le sont pas, ou du moins de modéliser comment les agents peuvent accéder à l'information sur l'état du système.

L'état du système est en pratique constitué de plusieurs composantes relatives à différentes parties de l'environnement. Par exemple, dans le cas d'un robot aspirateur, l'état du monde est généralement composé de sa position dans la pièce, du niveau de charge de sa batterie, de la présence d'objets dans la pièce, etc. Ces composantes, appelées *variables*, peuvent prendre plusieurs valeurs qui déterminent les différentes situations dans lesquelles peut se trouver l'agent ou quelles sont ses décisions possibles. De plus, il est fréquent que ces variables soient bornées ou qu'il existe des *relations* entre elles. Il est ainsi possible que certaines combinaisons de valeurs pour ces variables ne représentent pas d'état réel accessible du système (sous un meuble pour le robot aspirateur par exemple) ou que les décisions de l'agent soient conditionnées par l'état courant du système.

Formaliser les problèmes de décision en vue de les résoudre consiste donc essentiellement à modéliser une situation à base de *variables* pouvant prendre certaines valeurs ainsi que les *relations* existant entre ces variables. De plus, cette thèse utilise certains modèles qui font également appel à des relations particulières entre les variables expliquant la *dynamique* des valeurs des variables, leur *observabilité*, ainsi que la *mesure de performance* associée aux différentes combinaisons de valeurs.

1.2.1 Variables

La nature du problème de décision conditionne toujours la nature des variables qui le constitue. Par exemple, décider de la quantité d'essence à mettre dans un réservoir induit une *variable de décision continue* alors que l'allocation d'un certain nombre de ressources à une tâche induit une *variable de décision discrète*. Il est également possible que certaines variables ne soient pas contrôlables comme la position ou la charge d'un robot par exemple, et l'on parlera alors de *variables d'environnement*.

De nombreux domaines de la théorie de la décision se sont spécialisés dans certains

types de variables particuliers. Là où l'*analyse* mathématique se préoccupe des variables réelles, de leur topologie et de leurs propriétés, l'*optimisation combinatoire* s'est spécialisée dans l'étude des variables discrètes et la logique dans les variables booléennes. Cependant, la majorité des problèmes se composent souvent de plusieurs types de variables et les approches ci-dessus peinent à rester performantes. Un des domaines les plus polyvalents à ce niveau est la *satisfaction de contraintes* qui focalise essentiellement sur les *relations* qui existent entre les variables pour trouver une solution.

1.2.2 Relations

Les relations déterminent comment les variables interagissent entre elles. Il existe de la même manière que les variables, plusieurs types de relations qui dépendent des variables mises en interaction. Par exemple, dans les problèmes de satisfaction de contraintes, les relations déterminent les tuples de valeurs autorisés. En optimisation convexe, les relations sont des inéquations linéaires délimitant les valeurs possibles des variables. En inférence bayésienne, les relations indiquent les probabilités conditionnelles de certaines valeurs de variables par rapport à la valeur d'autres variables.

On peut regrouper les classes de relations en deux types principaux : les relations déterministes et les relations incertaines. Les deux premiers exemples donnés ci-avant sont des relations de type déterministe alors que la troisième est incertaine. Les relations incertaines apparaissent lorsque certaines variables ne peuvent pas être connues avec certitude (les intentions d'autres agents ou en cas d'observabilité partielle par exemple). Ce type de relation est généralement pris en compte avec des relations probabilistes, mais d'autres domaines existent comme les relations floues [González-Rodríguez *et al.*, 2006], ou les relations possibilistes [Dubois et Prade, 1988].

On différencie également trois types de relations particulières parmi toutes les relations entre variables. La première est la relation qui définit comment une variable à un instant donné interagit avec elle-même et avec les autres à un instant futur. On appelle ce type de relation la *dynamique* de la variable. La seconde relation définit la mesure de performance associant une décision et un état à une valeur immédiate acquise par l'agent. Cette relation de *récompense* indique les décisions souhaitables ou non selon l'état du système. La dernière est la relation qui détermine quelles sont les variables d'environnement dont l'agent dispose pour prendre sa décision à un instant donné, on appelle cette relation l'observabilité de l'environnement.

1.2.2.1 Dynamique

La dynamique au sens large n'explique pas seulement comment interagit une variable avec elle-même dans un instant futur, mais également comment elle interagit avec les autres et comment ses relations évoluent au cours du temps. La dynamique n'est cependant pas toujours présente dans les problèmes de prise décision. Il est possible de rencontrer des problèmes à une seule étape où aucune conséquence sur les décisions futures ne serait engendrée par la décision courante. Les domaines concernés sont l'*optimisation*, la *satisfaction de contraintes*, et l'*apprentissage statistique*.

Dans le cadre de cette thèse, qui est le cas où la décision courante influence les décisions futures, on parle de problème de décisions *séquentielles*. La dynamique des relations et des variables devient alors primordiale pour pouvoir prédire l'évolution de l'environnement et l'influence d'une décision sur les décisions suivantes. Ici les domaines sont la *planification* ou l'*apprentissage par renforcement*. La planification suppose la connaissance exacte de la dynamique de l'environnement, des variables et des relations et utilise cette connaissance pour prévoir la séquence de décisions à effectuer. L'apprentissage par renforcement, au contraire, ne suppose aucune connaissance *a priori* sur la dynamique du problème et essaie de trouver la bonne séquence de décisions par essais-erreurs directement dans l'environnement. Bien que très intéressant, l'apprentissage par renforcement ne sera pas abordé dans cette thèse et nous supposerons dans la suite du document que la dynamique complète du système dans lequel l'agent planifie est connue.

1.2.2.2 Récompense

La mesure de performance pour la résolution d'un problème donné peut généralement être spécifiée de deux manières. Soit sous la forme d'une satisfaction booléenne comme le fameux problème SAT ou sous la forme de valeur réelle à optimiser. Les domaines associés au premier sont la *satisfaction de contraintes* ou la *logique* sous toutes ses formes. Dans le second cas, on parle plutôt de *théorie de l'utilité* [Russell et Norvig, 2009]. La théorie de l'utilité décrit les bases et les fondements du raisonnement sur les préférences d'une entité intelligente. En présence d'incertitude, c'est très souvent le principe d'*utilité espérée* qui est appliqué. Ce principe stipule simplement que tout agent rationnel devrait toujours maximiser sa récompense espérée.

Le principe d'utilité espérée sera employé dans le cadre de cette thèse plutôt que la satisfaction. Il est en effet plus naturel, lorsque l'on a affaire à un problème de

décisions séquentielles, d'estimer l'utilité de chaque action dans chaque état plutôt que de simplement indiquer si une action est satisfaisante ou non. Cela permet non seulement d'estimer l'utilité espérée d'une action à une étape selon les issues possibles de cette action, mais cela permet également d'estimer l'*utilité espérée moyenne* sur l'horizon de planification et ainsi de garantir que la séquence d'actions choisie sera meilleure que toute autre séquence d'actions possible *en moyenne*.

1.2.2.3 Observabilité

La relation d'observabilité détermine l'information dont dispose l'agent au moment où il prend sa décision. Dans le cas le plus simple, l'agent a accès à toute l'information suffisante pour planifier et donc à l'état complet du système. Dans le cas inverse, l'agent doit raisonner sur ses incertitudes et donc maintenir une croyance sur les valeurs des variables qu'il ne connaît pas. Dans le cas où en plus les variables n'ont pas de dynamique, le domaine usuellement associé et l'inférence bayésienne ou les *réseaux bayésiens*. Dans le cas où s'ajoute en sus une relation de récompense, on parle de *diagramme de décision* et la valeur de l'information devient alors importante [Russell et Norvig, 2009].

Dans le cadre de cette thèse, plusieurs cas d'observabilité seront considérés et les différentes implications de l'observabilité sur la complexité du problème à résoudre seront étudiées. Voyons maintenant un résumé de la formalisation du type de problèmes à résoudre.

1.2.3 En résumé

En résumé, plusieurs caractéristiques peuvent être extraites des problèmes de prise de décision :

Problème épisodique/séquentiel : Une première caractéristique des problèmes concerne le degré d'influence de chacune des décisions sur les décisions à venir. Lorsque le problème est épisodique, l'agent perçoit sa situation, trouve la meilleure action à entreprendre dans cette situation et l'exécute. Une nouvelle situation est alors perçue, indépendamment de l'action entreprise. Dans le cas séquentiel, en revanche, l'action effectuée par l'agent influence le prochain état du système et les états subséquents. Il devient donc nécessaire de raisonner sur toutes les séquences futures d'états possibles pour évaluer une action convenablement. Seuls ces derniers types de problèmes seront abordés dans cette thèse.

Problème de satisfaction/d'optimisation : Qu'il soit épisodique ou séquentiel, une

solution au problème est toujours évaluée par une mesure de performance. Certains problèmes exigent seulement de trouver une solution satisfaisante en regard des relations exprimées entre les variables ou atteignant une certaine valeur minimale en utilité. D'autres exigent que la meilleure solution possible soit retournée pour être résolus. Cette thèse se concentre essentiellement sur les problèmes d'optimisation.

Environnement discret/continu : Lors d'une prise de décision, l'agent doit souvent prendre en compte des quantités aussi bien discrètes que continues. Lorsqu'il s'agit d'orienter un véhicule ou de choisir la poussée d'un moteur par exemple, l'agent doit choisir parmi une infinité de possibilités. Alors que lorsqu'il se déplace sur une grille, ou un graphe, ses différents choix sont souvent énumérables. La majorité des problèmes considérés dans cette thèse sont discrets.

Environnement contraint/libre : Il est fréquent de trouver dans la définition de problème la présence d'une connaissance à priori qui contraint la prise de décision ou les situations possibles. Ces relations spécifiques peuvent être alors utilisées pour améliorer la recherche d'une solution optimale au problème considéré.

Dynamique déterministe/stochastique : Lorsque l'action d'un agent peut éventuellement échouer ou mener à plus d'une situation possible, alors on parle de problème stochastique. Dans le cas inverse, si l'action d'un agent dans un état conduit toujours dans le même état suivant, alors l'environnement est considéré déterministe. Cette thèse est particulièrement intéressée par les problèmes stochastiques bien que certains résultats présentés ne s'appliquent qu'aux problèmes déterministes.

Observabilité complète/partielle : L'observabilité de l'environnement est un des axes majeurs de cette thèse. En effet, la complexité d'un problème est très souvent rattachée à la capacité que l'agent a à déterminer précisément dans quel état is se trouve. L'observabilité complète est définie dès lors que l'agent à accès à toute l'information nécessaire pour prendre sa décision. Dans le cas inverse, où certaines parties de l'état ne sont pas observées ou le sont avec un certain bruit, l'agent doit ainsi raisonner sur une croyance sur les valeurs que peuvent prendre ces parties d'état.

Un ou plusieurs agents : La dernière caractéristique du problème concerne le nombre d'agents dans l'environnement. Dans cette thèse, nous ne considérerons pas le cas ou plusieurs agents travaillent les uns *contre* les autres, néanmoins, nous serons amenés à considérer des problèmes où plusieurs agents devront coopérer en vue d'attendre un objectif commun. La première partie de cette thèse focalise plus sur les problèmes monoagents alors que la seconde s'oriente vers les problèmes de décisions distribuées. La principale difficulté associée à la distribution de la décision est essentiellement due au fait que chacun des agents doit raisonner sur

toutes les actions possibles des autres agents augmentant considérablement le nombre d'options possibles.

Ainsi, les problèmes de décisions séquentielles considérés dans cette thèse peuvent être représentés comme à la figure 1.2 et se formalisent à l'aide de *variables d'environnement* S (les cercles), de *variables de décisions* A (les carrés), de *relations de la dynamique* (flèches pleines), de *relations d'observation* (flèches hachées) et de *relation d'utilité* r (doubles flèches vers les losanges). À chaque étape de temps t, chaque variable d'environnement est non contrôlable par opposition aux variables de décisions qui le sont et dont les valeurs sont choisies par l'agent. Dans la figure 1.2, la dynamique indique que chaque variable S^t au temps suivant dépend de la variable S^{t-1} et de la décision A^{t-1} au temps précédent. La récompense relie l'état courant S^t et la décision courante A^t à une valeur réelle r^t. La relation d'observabilité spécifie qu'à chaque instant de décision A^t, la variable d'environnement de l'étape précédente S^{t-1} est connue. L'objectif de l'agent est alors de maximiser la récompense espérée obtenue sur les T étapes de décisions :

$$\max \mathrm{E}\left[\sum_{t=1}^{T} r^t \middle| \mathrm{S}_0\right]$$

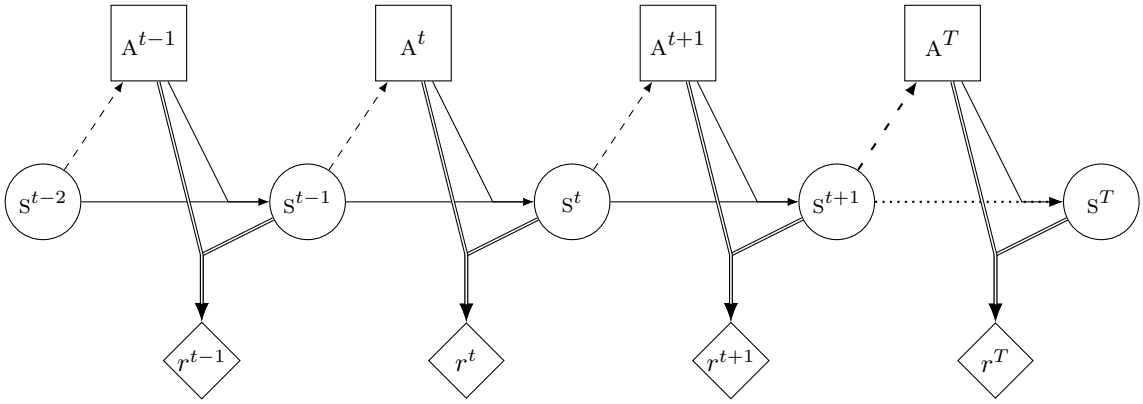


FIGURE 1.2 – Représentation graphique d'un problème de décisions séquentielles sous incertain.

Voyons maintenant quelques exemples de problèmes s'appliquant à la description ci-dessus et pour lesquels nous avons proposé des solutions originales.

1.3 Exemples de problèmes

Nous présenterons tout d'abord le problème qui a occupé la première partie de cette thèse avant de se focaliser sur les problèmes multiagents qui ont occupé la seconde partie.

1.3.1 Problème monoagent

Le premier problème auquel nous nous sommes intéressés est un problème d'allocation de ressources dans un environnement incertain. Ce problème concerne un bâtiment de marine devant se défendre en présence de menaces ennemies comme présenté par la figure 1.3. Le bateau est équipé d'un certain nombre d'armes possédant différentes portées pour se défendre de missile air-mer ou mer-mer arrivant à des vitesses différentes.

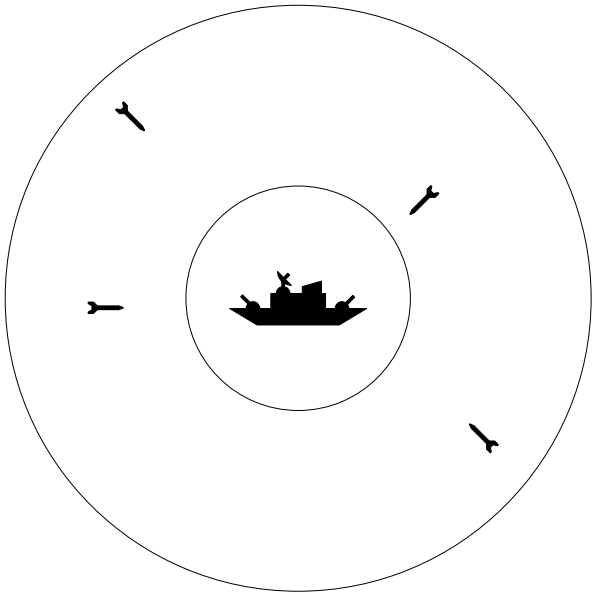


FIGURE 1.3 – NEREUS : a Naval Environment for Resource Engagement in Unpredictable Situations.

Pour percevoir les missiles s'approchant, le navire possède un radar disposant d'une portée de 3 km que nous supposerons parfait. Il possède également deux STIRs (radars dédiés au guidage) permettant d'attacher aux menaces ennemies des missiles longue portée ou une mitrailleuse lourde de portée moyenne. Enfin, un système de mitrailleuse légère à courte portée peut également être activé si un missile adverse entre dans son champ d'action.

Nous considérons dans ce problème que la seule incertitude possible porte sur l'efficacité des armes, à savoir la probabilité que chacune détruise la menace ciblée. Ainsi, bien que discutable, l'hypothèse est faite que le navire perçoit parfaitement le nombre et la position des missiles adverses.

Si l'on considère ce problème comme un problème stochastique de prise de décisions séquentielles, il est possible de modéliser au niveau d'un état les missiles adverses présents et les armes disponibles pour les engager et modéliser la dynamique de ces objets selon

leurs capacités et les probabilités que chaque arme a de détruire une menace. Le but est d'optimiser la probabilité de survie du navire face à une attaque complexe pouvant comporter jusqu'à une dizaine de missiles.

Une formalisation mathématique et une solution à ce problème sont présentés dans le chapitre 3 de cette thèse. La solution proposée est essentiellement basée sur une combinaison des modèles de planification classique sous incertitude avec une modélisation efficace des contraintes du problème, permettant un gain substantiel en terme de temps de calcul de la séquence de décision optimale.

1.3.2 Problèmes multiagents

Il a ensuite été suggéré d'étudier le cas ou plusieurs de ces navires auraient à communiquer pour se défendre conjointement d'une attaque. Cependant, au vu de l'état actuel de la littérature multiagent pour résoudre les problèmes de planification décentralisée avec communication, et du peu de résultats de modélisation du domaine, nous nous sommes plutôt consacrés à modéliser la communication dans des problèmes de coordination de la littérature plutôt que le problème de défense maritime. Voici quelques exemples de ces problèmes.

1.3.2.1 Patrouille de véhicules aériens autonomes

Le problème de la patrouille est un très vieux problème qui prend ses racines dans les problèmes de la théorie des graphes, notamment celui de la découverte d'un circuit hamiltonien [Hamilton, 1858]. Dans son expression la plus simple, il consiste à trouver un circuit hamiltonien sur un graphe de telle sorte à optimiser la mise à jour de l'information qui serait disponible à chaque nœud. Néanmoins, si certains endroits doivent être mis à jour plus fréquemment (par exemple si certains endroits sont plus ''sensibles'' ou ''à risque'' que d'autres), alors il est possible que le circuit hamiltonien ne soit plus la solution optimale et plusieurs agents aériens autonomes doivent alors se coordonner pour conjointement maintenir une connaissance optimale de l'état courant de l'environnement.

Dans ce contexte, les agents n'accèdent pas à chaque instant à l'état complet de l'environnement, mais seulement au nœud sur lequel ils se trouvent. Il leur faut donc maintenir une croyance sur la ''fraîcheur'' de l'information perçue dans les nœuds précédemment visités en considérant par exemple un taux d'obsolescence de l'information sur chaque sommet.

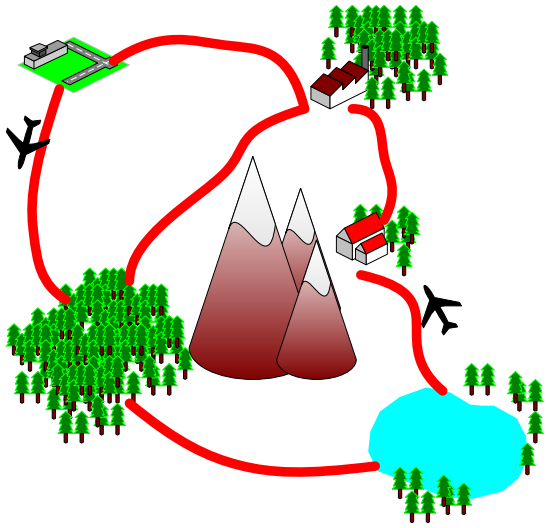


FIGURE 1.4 – Un exemple de problème de patrouille à deux agents.

Du point de vue de la modélisation stochastique, le graphe à patrouiller et la position des agents sur ce graphe constituent l'état du système complet. Néanmoins, les agents n'ont pas accès à cet état complet puisque nous faisons l'hypothèse qu'ils ne savent pas si oui ou non leurs capteurs ont pu transmettre une information pertinente sur les sommets qu'ils viennent de patrouiller. Dès lors, les agents maintiennent une croyance sur l'état de ''fraîcheur'' de l'information et cherche à la maximiser étant donné une pondération initiale des sommets selon leur importance.

1.3.2.2 Expédition MARS rovers

Initialement proposé par Smith et Simmons [2004] pour un seul agent, *MultiAgent Rock Sample* (MARS) décrit comment plusieurs robots doivent interagir et communiquer pour échantillonner de la manière la plus efficace possible un ensemble de roches ayant potentiellement une valeur scientifique (cf. figure 1.5).

On suppose que les agents connaissent leur position exacte dans l'environnement et que leurs déplacements sont déterministes. L'incertitude réside sur la valeur scientifique des roches à échantillonner. Pour cela, un scanneur leur indique à chaque déplacement la qualité des roches avoisinantes, avec une certaine confiance. Plus le robot est proche de la roche et plus le scanner est précis quant à la qualité de celle-ci. L'objectif est donc d'échantillonner toutes les roches ayant une valeur scientifique puis de quitter l'environnement au plus vite par le côté droit de la grille.

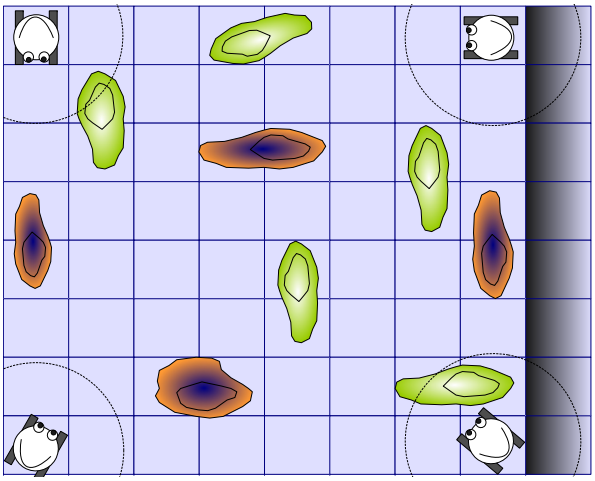


FIGURE 1.5 – Un exemple de problème d'expédition d'échantillonnage de roche à 4 agents.

L'état de ce problème se modélise aisément par la position des agents et la valeur scientifique (partiellement observable) des roches. La dynamique quant à elle, est extrêmement simple puisque quasiment déterministe. Il convient enfin de maximiser le nombre de bonnes roches échantillonnées tout en minimisant le temps passé à le faire.

1.3.2.3 Les déménageurs

Ce problème a été proposé par Seuken et Zilberstein [2007b] et décrit le comportement de deux robots déménageurs qui doivent ranger des boites dans un environnement de type grille. Il y'a 3 boites dans l'environnement, deux petites et une grande. La grande nécessite que les deux agents se coordonnent pour pouvoir la pousser. Le but est d'amener une boite (et une seule) dans la zone but (en haut de la grille, voir figure 1.6). L'optimal étant bien sur d'apporter la grosse boite dans la zone but.

Les agents n'ont ici qu'une vague idée de leur position de départ (ils savent seulement qu'ils sont à côté d'un mur). De plus, ils n'observent l'autre agent, un mur ou une boite que lorsque l'objet est exactement en face d'eux. Cette observabilité très limitée rend le problème particulièrement difficile lorsqu'en plus aucune communication n'est autorisée.

L'état de ce problème contient la position des agents et des boites et la dynamique est simple. Chaque agent peut avancer dans la direction qui lui fait face ou peut faire un quart de tour sur place. Les petites boites peuvent être poussées dans toutes les directions par un agent, mais la grosse boite nécessite la poussée coordonnée des deux agents pour être déplacée. Le problème initial stipule que l'environnement est réinitialisé

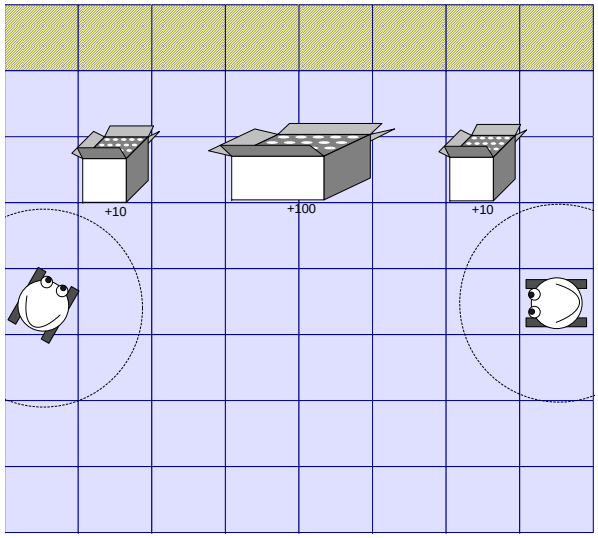


FIGURE 1.6 – Un exemple de problème de déménageurs.

dès qu'une boite atteint la zone en haut de la salle, mais on peut aussi essayer de trouver une stratégie qui cherche simplement à minimiser le nombre d'étapes permettant à la grosse boite d'être poussée en haut.

1.4 Contributions de la thèse

Cette thèse propose plusieurs nouvelles contributions au domaine de la prise de décisions séquentielles sous incertitude. La figure 1.7 représente graphiquement la topologie des contributions effectuées vis-à-vis des modèles existants. Parmi ceux-ci, nous avons représenté quatre modèles majeurs de la prise de décisions séquentielles sous incertitude qui seront plus amplement détaillés au chapitre suivant.

Le processus décisionnel de Markov (MDP) concerne un agent évoluant dans un environnement stochastique complètement observable. Plus complexe et plus général, le MDP partiellement observable (POMDP) ajoute un degré d'incertitude en n'autorisant la perception de l'état qu'au travers d'observations -- généralement bruitées -- de celui-ci. Le POMDP multiagent, quant à lui, étend le POMDP au contexte à plusieurs agents en faisant l'hypothèse que ces agents communiquent librement. La suppression de cette dernière hypothèse mène au modèle le plus complexe, le POMDP décentralisé (DEC-POMDP).

Cette thèse contribue à plusieurs niveaux de cette hiérarchie :

Au niveau des mdps :

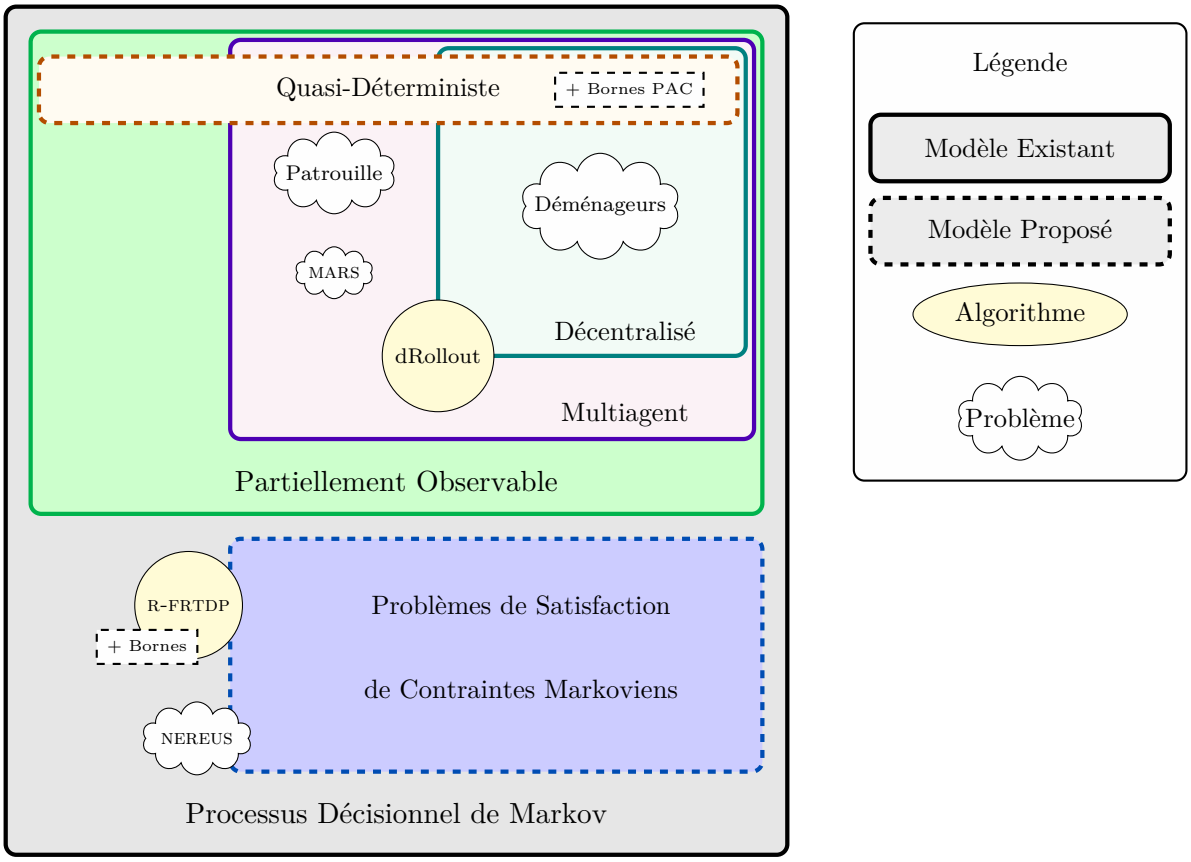


FIGURE 1.7 – Représentation hiérarchique des contributions de la thèse.

- Nous avons proposé, en collaboration avec Pierrick Plamondon, un algorithme de résolution des problèmes d'allocation de ressources sous incertitude (R-FRTDP), ainsi que des bornes serrées pour ce type de problème ;
- Nous avons également proposé un modèle prenant en compte une connaissance à priori des contraintes sur les décisions possibles ainsi que l'évolution de l'algorithme précédent pour ce nouveau modèle. Des résultats théoriques et expérimentaux sont fournis et discutés dans ce contexte dans le chapitre 3 ;

Au niveau des pomdps :

- Nous avons proposé un nouveau modèle appelé quasi-déterministe présenté dans le chapitre 4 ;
- Des modèles contraints d'observations pour ce modèle sont également proposés, permettant la dérivation de nouveaux résultats de complexité associés à une borne probablement approximativement correcte (PAC) ;

Au niveau des mpomdps :

- Dans la deuxième partie du chapitre 4, nous avons proposé l'extension du modèle quasi-déterministe au contexte multiagent et nous l'avons ensuite appliqué aux problèmes de coordination d'agents de patrouille et d'exploration MARS ;

– Pour cela, un algorithme de mélange de politiques locales a été proposé en collaboration avec Jean-Samuel Marier pour résoudre les deux problèmes précédents ;

Au niveau des dec-pomdps :

– Nous avons proposé un algorithme d'approximation décentralisé (dRollout) ainsi que son application au problème des déménageurs. Différents résultats sont présentés dans le chapitre 5 selon certaines hypothèses de mise à jour des états de croyance ;
– Le modèle quasi-déterministe est également détaillé dans le cas décentralisé, des résultats théoriques sont dérivés et l'application à un nouveau problème jouet décentralisé sous communication non fiable démontre de son efficacité expérimentale.

1.5 Organisation de la thèse

À partir de cette topologie, nous avons organisé cette thèse comme suit :

Le Chapitre 2 présente la base des processus décisionnels de Markov monoagents et multiagents. Il introduit également quelques algorithmes basiques de la littérature et certains résultats choisis de complexité pour ces modèles ;

Le Chapitre 3 présente les contributions apportées aux processus décisionnels de Markov à un seul agent et sous observabilité complète dans le cadre du problème d'allocation de ressources présenté à la section 1.3.1. À cette fin, il introduit le problème d'allocation d'armes à des cibles puis les problèmes de satisfaction de contraintes. Le modèle proposé est ensuite détaillé puis analysé théoriquement. Finalement, un algorithme adapté pour ce modèle, ainsi que de nouvelles bornes utilisées comme heuristiques, sont présentés et analysés empiriquement ;

Le Chapitre 4 présente le nouveau modèle quasi-déterministe pour certains problèmes partiellement observables monoagents et multiagents. Une borne PAC sur l'horizon maximal de planification est dérivée sous certaines conditions permettant l'obtention de résultats de complexité intéressants. Ces modèles sont ensuite utilisés pour la représentation des problèmes formalisés à la section 1.3.2. Quelques résultats expérimentaux sont également présentés ;

Le Chapitre 5 présente l'algorithme de résolution approchée en ligne dans le cas décentralisé basé sur un algorithme de type Rollout [Bertsekas, 2000]. Une étude empirique de la mise à jour de l'état de croyance est également réalisée sur le problème des déménageurs ;

Le Chapitre 6 conclut cette thèse et présente quelques travaux futurs.

Chapitre 2

Bases sur les processus décisionnels de Markov

«Toute connaissance dégénère en probabilité.»

David Hume

Ce chapitre introduit les bases théoriques des modèles probabilistes, c'est à dire le formalisme mathématique permettant la description de problèmes de décisions séquentielles et stochastiques appelés Processus Décisionnels de Markov *(en anglais,* Markov Decision Processes *ou* MDP*). Dans une première partie, les modèles fondamentaux centralisés sont présentés à la manière de Szer [2006] avant de présenter ceux destinés aux problèmes impliquant plusieurs agents comme l'ont fait Seuken et Zilberstein [2005].*

2.1 Modèles pour un agent

Les MDPs ont été introduits et popularisés à la fin des années cinquante par Bellman [1957] et Howard [1960]. Ce formalisme se base sur un modèle à la fois très général et assez abstrait pour pouvoir être appliqué à des problèmes variés, mais cependant restreint de par ses hypothèses sur le caractère totalement observable de l'environnement et sur le contrôle disponible strictement centralisé.

2.1.1 Processus décisionnel de Markov (MDP)

Nous présentons ici rapidement le formalisme MDP ainsi que quelques algorithmes fondamentaux permettant leur résolution. Notons qu'une introduction exhaustive des MDPs peut être trouvée dans le livre de Puterman [1994].

2.1.1.1 Formalisme

Définition 1. *(MDP) Un* MDP *est défini par un tuple* $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$*, où :*

- $\mathcal{S}$ *est un ensemble fini d'*états $s \in \mathcal{S}$ *;*
- $\mathcal{A}$ *est un ensemble fini d'*actions $a \in \mathcal{A}$ *;*
- $\mathcal{T}(s, a, s') : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ *est la* probabilité de transition *du système de l'état* s *vers l'état* s' *après l'exécution de l'action* a *;*
- $\mathcal{R}(s, a) : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ *est la* récompense *produite par le système lorsque l'action* a *est exécutée dans l'état* s*.*

La propriété fondamentale de ce formalisme est *la propriété de Markov*. Elle garantit que la probabilité de transition d'un état s vers un état s' sous l'effet d'une action a est indépendante du passé :

$$\Pr(s_{t+1} = s' | s_0, a_0, s_1, a_1, ..., s_t, a_t) = \Pr(s_{t+1} = s' | s_t, a_t)$$

Par conséquent, l'état courant et la dernière action constituent toujours une information suffisante pour la prédiction de l'évolution du système.

Une représentation graphique d'un MDP est donnée par la figure 2.1. Les cercles représen-tent les variables non contrôlables (les états), les carrés représentent les variables de décisions (les actions) et les losanges les récompenses reçues. Un lien entre deux variables induit une table relationnelle (par exemple de probabilité) qui associe à chaque couple de valeur possible pour ces variables sa probabilité de vraisemblance. Les liens en tirets indiquent la connaissance disponible par l'agent au moment de prendre la décision (dans l'exemple de la figure 2.1, l'agent connaît l'état du système).

Exécuter un MDP revient à observer en boucle l'état courant s du système, à choisir une action a à effectuer puis à observer la transition de l'état s vers le nouvel état s'. On appelle politique $\pi : \mathcal{S} \times \mathbb{N} \mapsto \mathcal{A}$ une loi de décision qui détermine à chaque instant t et pour chaque état s, l'action a qu'il convient d'effectuer.

Un *problème de décision de Markov* est alors défini comme un MDP muni d'un

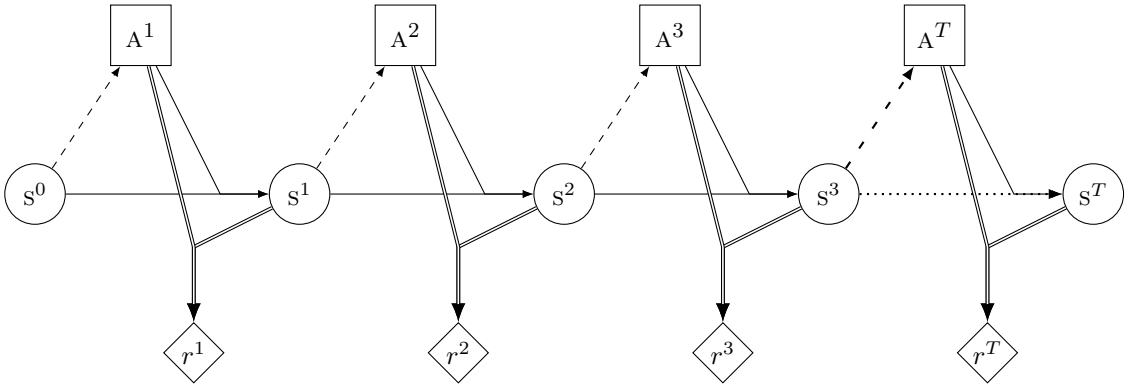


FIGURE 2.1 – Représentation graphique d'un MDP.

critère de performance. Ce critère précise la façon dont on doit évaluer la quantité d'une politique pour un MDP donné. Pour des problèmes à *horizon fini* T, un des critères les plus utilisés est l'espérance de la somme des récompenses accumulées à partir d'un certain état initial s_0 :

$$\mathrm{E}\left[\sum_{t=0}^{T-1}\mathcal{R}(s_t,a_t)\middle|s_0\right]$$

Pour des problèmes à *horizon infini*, utiliser le même critère pose le problème de l'accumulation non bornée des récompenses. On introduit alors en général un *facteur d'escompte* γ :

$$\mathrm{E}\left[\sum_{t=0}^{\infty}\gamma^t\mathcal{R}(s_t,a_t)\middle|s_0\right] \qquad \text{avec} \qquad 0\leqslant\gamma<1$$

On peut ensuite évaluer la valeur d'une politique en établissant sa *fonction de valeur* V. C'est une fonction qui retourne pour chaque état la valeur du critère de performance :

$$V_\pi(s)=\mathrm{E}\left[\sum_{t=0}^{T-1}\mathcal{R}(s_t,\pi(s_t))\middle|s_0\right] \qquad \text{pour l'horizon fini} \tag{2.1}$$

$$V_\pi(s)=\mathrm{E}\left[\sum_{t=0}^{\infty}\gamma^t\mathcal{R}(s_t,\pi(s_t))\middle|s_0\right] \qquad \text{pour l'horizon infini} \tag{2.2}$$

Une fois le critère de performance fixé, on peut alors s'intéresser à la politique qui optimise l'évaluation de ce critère, c'est à dire à trouver la politique qui maximise la fonction de valeur pour un état initial donné s_0. La recherche de cette politique optimale π^* constitue l'enjeu principal du problème de décision de Markov :

$$\pi^*=\arg\max_\pi V_\pi(s_0)$$

Ce problème a été montré comme appartenant à la classe des problèmes polynômiaux les plus difficiles (P-complet) par Papadimitriou et Tsisiklis [1987]. En général, pour un même MDP, il existe plusieurs politiques optimales. Dans ce qui suit, nous allons nous limiter à des algorithmes permettant d'en calculer une.

2.1.1.2 Résolution par planification

Puterman [1994] a montré que la politique optimale pour des MDPs à horizon fini et pour le critère de performance considéré ci-dessus est déterministe, indépendante du passé, mais non stationnaire. Une politique optimale choisira donc toujours la même action dans la même configuration du système, mais cette action sera dépendante de l'instant d'exécution. Nous noterons $\pi = \{\pi_T, \pi_{T-1}, ..., \pi_1\}$ une telle politique et $\pi_t(s) = a$ l'action à effectuer dans l'état s à l'instant t. Établir la fonction de valeur d'une politique peut se faire directement en déterminant explicitement les espérances de récompense accumulée pour chaque instant t :

$$V_{\pi_t}(s) = \mathcal{R}(s, \pi_t(s)) + \sum_{s' \in \mathcal{S}} \mathcal{T}(s, \pi_t(s), s') V_{\pi_{t-1}}(s')$$

Puisque le calcul de la fonction de valeur au moment t nécessite la connaissance de celle au moment $t-1$, on commence par l'étape finale, pour ensuite retropropager les valeurs pour le calcul des espérances au moment précédent. L'algorithme 1 est une des manières d'évaluer une politique.

Algorithm 1 (EVALMDP) Évaluation de la politique à horizon fini.

requiert: un MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ et une politique π_T
retourne: La fonction de valeur V pour la politique π_T

1: $V_{\pi_0} \leftarrow 0$, $(\forall s \in \mathcal{S})$
2: **pour** $t = 1$ **à** T **faire**
3: **pour tout** $s \in \mathcal{S}$ **faire**
4: $V_{\pi_t}(s) = \mathcal{R}(s, \pi_t(s)) + \sum_{s' \in \mathcal{S}} \mathcal{T}(s, \pi_t(s), s') V_{\pi_{t-1}}(s')$
5: **fin pour**
6: **fin pour**

Il est ainsi possible de déterminer une politique optimale en utilisant l'algorithme 1 sur l'ensemble des politiques possible pour garder ensuite le meilleur choix. Il s'agit alors d'une recherche exhaustive dans l'espace des politiques. Cependant, Bellman [1957] a introduit une approche plus efficace pour déterminer une politique optimale. En effet, son *principe d'optimalité* stipule que la politique optimale $\pi^* = \{\pi^*_T, \pi^*_{T-1}, \ldots, \pi^*_1\}$ pour l'horizon T contient forcément une politique optimale pour l'horizon $T-1$, à savoir la

politique $\pi^* = \{\pi^*_{T-1}, \ldots, \pi^*_1\}$. De manière plus générale, elle contient des sous-politiques optimales pour tout horizon $t \leqslant T$. Le principe garantit donc que le calcul d'une politique optimale pour un horizon T peut utiliser la solution pour l'horizon $T-1$ du même MDP. Il suffit alors de trouver la fonction de décision optimale pour l'horizon 1, à savoir π^*_1. Trouver une politique optimale pour un horizon T exige que ce calcul soit opéré T fois, ce qui signifie qu'un nombre total de $T \cdot |\mathcal{A}|^{|\mathcal{S}|}$ politiques doit être considéré. Comparativement aux $|\mathcal{S}|^{|\mathcal{A}|^T}$ politiques possibles à évaluer initialement, cette technique consistant à réutiliser des solutions intermédiaires constitue le cœur de ce qu'on appelle la *Programmation Dynamique* (DP).

Algorithm 2 (DP) Programmation Dynamique pour les MDPs. [Bellman et Dreyfus, 1959]

requiert: un MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ et un horizon T
retourne: une politique optimale π^*_T pour l'horizon T

$V_{\pi_0} \leftarrow 0$, $(\forall s \in \mathcal{S})$
pour $t = 1$ **à** T **faire**
  pour tout $s \in \mathcal{S}$ **faire**
    $\pi_t(s) \leftarrow \arg\max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s') V_{\pi_{t-1}}(s') \right]$
    $V_{\pi_t}(s) = \mathcal{R}(s, \pi(s_t)) + \sum_{s' \in \mathcal{S}} \mathcal{T}(s, \pi_t(s), s') V_{\pi_{t-1}}(s')$
  fin pour
fin pour

La programmation dynamique, décrite dans l'algorithme 2, permet en outre de calculer la politique optimale d'un MDP à horizon infini puisque le principe peut y être étendu directement, avec la seule contrainte que le nombre de décisions soit possiblement infiniment grand. Heureusement, on peut montrer que pour le critère choisi pour l'horizon infini (avec le facteur d'escompte γ de l'équation 2.2), il suffit de se limiter à des politiques déterministes, markoviennes et stationnaires [Puterman, 1994]. Le fait que la politique optimale soit stationnaire signifie que toutes les fonctions de décisions sont identiques $\pi^*_1 = \pi^*_2 = \cdots = \pi^*_\infty = \pi^*$. Il n'existe ainsi qu'une seule fonction de valeur qui peut être trouvée itérativement.

L'algorithme par itération de valeur peut être interprété comme l'application répétée d'un opérateur à la fonction de valeur. Cet opérateur est souvent appelé *opérateur de Bellman* H, et est défini de la manière suivante :

$$H(V)(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s') V(s') \right] \tag{2.3}$$

Il a été montré que l'opérateur de Bellman est une contraction par rapport à la norme max [Bertsekas et Tsitsiklis, 1996], et que la fonction de valeur optimale V^* est son

unique point fixe :

$$V^* = H(V^*) \text{ ou encore } V^*(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s') V^*(s') \right] \tag{2.4}$$

Cette équation est généralement appelée l'*Équation d'optimalité de Bellman*.

L'algorithme 3, appelé *itération de valeur* (VI), détermine donc une approximation de cette fonction de valeur optimale V^*. Une politique optimale peut ensuite être déduite de la fonction de valeur par simple choix de la meilleure action :

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s') V^*(s') \right] \tag{2.5}$$

Algorithm 3 (VI) Itération de Valeur pour les MDPs. [Russell et Norvig, 2009]

requiert: un MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ et une borne d'erreur ε
retourne: une ε-approximation de la fonction de valeur optimale

$n \leftarrow 0$, $V_{\pi_0} \leftarrow 0$, $(\forall s \in \mathcal{S})$
répéter
  pour tout $s \in \mathcal{S}$ **faire**
    $V^{n+1}(s) \leftarrow \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s') V^n(s') \right]$
  fin pour
  $n \leftarrow n + 1$
jusqu'à $||V^n - V^{n-1}|| \leqslant \varepsilon$

Ainsi, lorsque le modèle MDP est connu, c'est à dire lorsque les paramètres $\mathcal{T}$ et $\mathcal{R}$ sont disponibles à l'agent, celui-ci peut planifier et déterminer sa politique optimale via les équations (2.3) à (2.5). Dans le cas où seulement $\mathcal{R}$ est connue, l'agent doit alors faire des essais/erreurs dans le but d'*apprendre* le modèle de l'environnement.

2.1.1.3 Résolution par apprentissage

Le paragraphe précédent introduisait les techniques de planification dites *hors-ligne*, où le modèle exact du MDP, i.e. sa fonction de transition $\mathcal{T}$ et sa fonction de récompense $\mathcal{R}$, sont connus. L'exécution de la politique se faisait alors indépendamment du processus de planification, c'est à dire qu'on planifie, et une fois la planification terminée, on exécute la politique. L'apprentissage d'une politique est un processus *en-ligne*, i.e. que l'agent doit en même temps apprendre à connaître l'environnement et déterminer une politique optimale pour le contrôler. La boucle de contrôle est alors la suivante. l'agent

perçoit l'état courant s, il choisit une action a, le système fait la transition vers un nouvel état s' et l'agent reçoit une récompense $\mathcal{R}(s,a)$ qui lui indique l'utilité locale de cette transition. Le problème est alors de déterminer une politique optimale pendant que l'exploration et l'estimation de l'environnement est encore en cours.

Un des algorithmes les plus répandus d'apprentissage dans les MDPs est celui du *Q-Learning*, introduit par Watkins et Dayan [1992]. Le Q-Learning est la version en-ligne de l'algorithme d'itération de la valeur. Il est fondé sur l'évaluation des couples état-action, et non directement sur la fonction de valeur. La fonction $Q(s,a)$ représente la valeur espérée lorsque l'action a est exécutée dans l'état s et qu'une politique optimale est suivie à partir de l'état suivant s'. Dès lors, la fonction Q et la fonction de valeur V sont liées par la relation :

$$V(s) = \max_{a \in \mathcal{A}} Q(s,a)$$

Watkins [1989] a pu démontrer que la mise à jour 2.6, effectuée après chaque exécution d'une action, garantit la convergence vers la fonction Q optimale, sous réserve que chaque couple état-action soit visité un nombre infini de fois et que le facteur d'apprentissage α vérifie certaines propriétés basiques ($0 \leqslant \alpha, \sum_t \alpha_t = \infty, \sum_t \alpha_t^2 < \infty$) :

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha \left[\mathcal{R}(s,a) + \gamma \max_{a' \in \mathcal{A}} Q(s',a')\right] \tag{2.6}$$

L'algorithme 4 décrit le fonctionnement du Q-Learning.

Algorithm 4 Q-Learning pour les MDPs. [Watkins et Dayan, 1992]

requiert: un MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$
retourne: une approximation de la fonction Q optimale

1: $Q(s,a) \leftarrow 0$, $(\forall s \in \mathcal{S},\ \forall a \in \mathcal{A})$
2: **boucler**
3: Executer une action a
4: $Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha \left[\mathcal{R}(s,a) + \gamma \max_{a' \in \mathcal{A}} Q(s',a')\right]$
5: **fin boucler**

Cette approche est néanmoins limitée dès que le nombre d'états $|\mathcal{S}|$ devient trop consé-quent. En effet, il est nécessaire de maintenir une valeur $Q(s,a)$ pour chaque paire état-action, ce qui peut s'avérer très coûteux en mémoire en pratique. Dans cette situation, l'idée est d'utiliser de l'information supplémentaire *a priori* sur la *structure* de l'espace d'état permettant d'induire des relations entre les états et d'approximer directement la fonction Q selon les différentes caractéristiques des états.

2.1.2 MDP factorisé

La définition précédente des MDPs est basée sur une description relativement simple de l'état du système et suffit très bien lorsque le nombre d'état est petit. Cependant, dans de nombreux domaines, les états sont souvent structurés d'une manière ou d'une autre. Par exemple, si la position d'un robot est modélisée dans l'état du système, alors une notion de proximité des états devrait pouvoir être exploitée pendant la phase de calcul de la politique. Des états proches ou similaires devraient conduire à avoir des valeurs similaires pour des actions similaires. Le formalisme original des MDPs ne permets pas de modéliser ce type de proximité ou similarité.

Les MDPs *factorisés* (FMDP) constituent une description alternative pour les MDPs. Dans un MDP factorisé, l'état n'est pas une entité atomique, mais est constitué de facteurs ou caractéristiques. Ces facteurs peuvent être donnés *a priori*, ou peuvent être appris [Abbeel *et al.*, 2006; Degris *et al.*, 2006; Strehl *et al.*, 2007]. La fonction de transition et la représentation de la politique exploitent également cette structure comme l'a montré Boutilier *et al.* [2000]. Revenons à l'exemple du robot ; l'espace d'état de celui-ci est fondamentalement factorisé si nous utilisons les coordonnées cartésiennes par exemple. il existe deux facteurs, la coordonnée x et la coordonnée y, qui constituent l'état $s = (x, y)$.

Plus formellement :

Définition 2. *(*MDP factorisé*) Un* MDP *est défini par un tuple* $\langle \mathcal{X}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$*, où :*
- $\mathcal{X} = \{\mathrm{X}_1, \ldots, \mathrm{X}_m\}$ *est un ensemble fini de* variables d'états *où chaque* X_i *prend ses valeurs sur un domaine*
- $\mathcal{D}_{\mathrm{X}_i}$*. On a donc* $\mathcal{S} = \bigtimes_{i=1}^{m} \mathcal{D}_{\mathrm{X}_i}$ *;*
- $\mathcal{A}$ *est un ensemble fini d'*actions $a \in \mathcal{A}$ *;*
- $\mathcal{T}$ *est une* fonction de transition factorisée *;*
- $\mathcal{R}(s, a)$ *est une* fonction de récompense factorisée.

Dans ce contexte, un état d'un MDP devient alors simplement une assignation des variables d'état X. Comme il y a possiblement un nombre exponentiel d'assignations des variables, le nombre d'états du MDP est exponentiel en le nombre de variables d'état. Les processus décisionnels de Markov représentent usuellement leur fonction de transition comme une fonction de $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$. Seulement, il est possible à la place de représenter l'influence d'une assignation de variables sur une assignation de ces mêmes variables à un temps avancé par un réseau bayésien dynamique (DBN) [Dean et Kanazawa, 1990]. Ce modèle a la propriété d'identifier les dépendances conditionnelles entre les variables d'état et ainsi de définir une représentation généralement plus compacte et

plus structurée. Des algorithmes permettant de calculer des politiques structurées ont également été proposées [Boutilier, 1999].

Il convient cependant de remarquer que prendre avantage des représentations compactes est parfois difficile. Une des difficultés provient de la représentation compacte de politique. Koller et Parr [1999] ont par exemple trouvé des cas où même si le domaine peut se représenter très efficacement comme une fonction de variables d'état, la fonction de valeur reste tout de même très difficile à représenter. Dearden et Boutilier [1997] ont proposé une méthode qui s'attaque à ce problème en restreignant l'espace des valeurs autorisées aux règles de décisions conjonctives sur de petits ensembles de variables d'état. Des travaux sur la factorisation ont également été réalisés dans les systèmes multiagents par Guestrin [2003] qui a d'ailleurs proposé un algorithme basé sur l'élimination de variable [Dechter, 2003, chap. 13.3.3] pour résoudre les problèmes de coordination multiagents. Nous reviendrons sur les représentations factorisées et graphiques dans les chapitres suivants pour l'expression de contraintes sur les domaines des variables d'états ou pour les relations causales qui existent au niveau des différentes fonctions du modèle.

2.1.3 MDP partiellement observable (POMDP)

La politique de décision dans un MDP est toujours basée sur l'état réel du système. Malheureusement, il existe des cas où cet état n'est pas accessible. Tout ce dont dispose le contrôleur lors de l'exécution est un signal ou une observation bruitée, à l'aide duquel il peut au mieux essayer d'inférer la vraie configuration du système. De tels systèmes doivent être alors modélisés à l'aide de Processus de Markov Partiellement Observables (POMDP) [Cassandra *et al.*, 1994].

2.1.3.1 Formalisme

Définition 3. *(POMDP) Un* POMDP *est défini par un tuple* $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O} \rangle$*, où :*

- $\mathcal{S}$ *est un ensemble fini d'*états $s \in \mathcal{S}$ *;*
- $\mathcal{A}$ *est un ensemble fini d'*actions $a \in \mathcal{A}$ *;*
- $\mathcal{T}(s, a, s') : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ *est la* probabilité de transition *du système de l'état* s *vers l'état* s' *après l'exécution de l'action* a *;*
- $\mathcal{R}(s, a) : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ *est la* récompense *produite par le système lorsque l'action* a *est exécutée dans l'état* s*,*
- Ω *est un ensemble fini d'observations* $o \in \Omega$*,*

– $\mathcal{O}(s, a, o, s') : \mathcal{S} \times \mathcal{A} \times \Omega \times \mathcal{S} \mapsto [0, 1]$ *est la probabilité l'observation* o *survienne lors de la transition du système de l'état* s *vers l'état* s' *sous l'effet de l'action* a.

Le processus restreint à $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ *est appelé* MDP *sous-jacent du* POMDP.

Par comparaison à la figure 2.1, l'agent n'a accès à l'état du système qu'au travers des observations. Ceci est représenté par la figure 2.2.

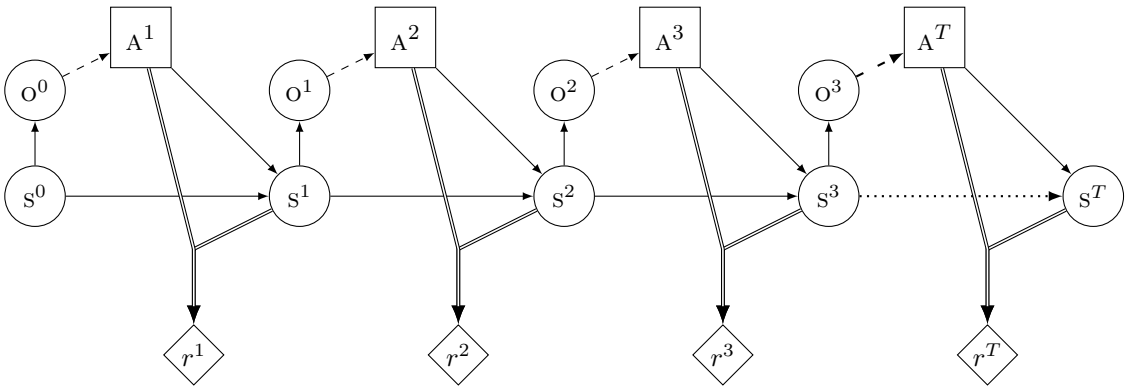


FIGURE 2.2 – Représentation graphique d'un POMDP sans mémoire.

A la différence de l'état courant, qui lui est une information suffisante pour le contrôle optimal d'un MDP, l'observation courante ne vérifie pas nécessairement la propriété de Markov dans le cas d'un POMDP. Ceci implique que les méthodes de résolution et les équations d'optimalité qui sont valables pour les états d'un MDP ne peuvent pas être directement appliquées aux observations d'un POMDP. Il est toutefois possible de se ramener à une autre description du système où la propriété de Markov est à nouveau vérifiée. On peut en effet constater que la probabilité de se trouver dans un état s au moment t de l'exécution dépend uniquement de la distribution de probabilités sur les états au moment précédent. Pour cela nous posons $\boldsymbol{b}_t(s)$ la probabilité de se trouver dans l'état s au moment t de l'exécution :

$$\boldsymbol{b}_t(s) = \Pr(s_t = s | s_0, a_0, o_0, \ldots, a_{t-1}, o_{t-1})$$

Le vecteur $\boldsymbol{b}$

$$\boldsymbol{b} = [\boldsymbol{b}_t(s_1), \boldsymbol{b}_t(s_2), \ldots, \boldsymbol{b}_t(s_{|\mathcal{S}|})]^\top$$

est souvent appelé *état de croyance* ou *belief state*. Sa mise à jour après l'exécution d'une action a et la perception d'une observation o peut être dérivée de la formule de Bayes :

$$\boldsymbol{b}_{t+1}(s') = \frac{\sum_{s \in \mathcal{S}} \boldsymbol{b}_t(s) \Big[\mathcal{T}(s, a_t, s') \mathcal{O}(s, a_t, o_t, s') \Big]}{\Pr(o_t | \boldsymbol{b}, a_t)} \tag{2.7}$$

où $\Pr(o_t|\boldsymbol{b}, a_t)$ est le facteur de normalisation :

$$\Pr(o_t|\boldsymbol{b}, a_t) = \sum_{s\in\mathcal{S}} \sum_{s'\in\mathcal{S}} \boldsymbol{b}_t(s)\Big[\mathcal{T}(s, a_t, s')\mathcal{O}(s, a_t, o_t, s')\Big]$$

En utilisant cette formulation de l'état de croyance, la figure 2.2 devient :

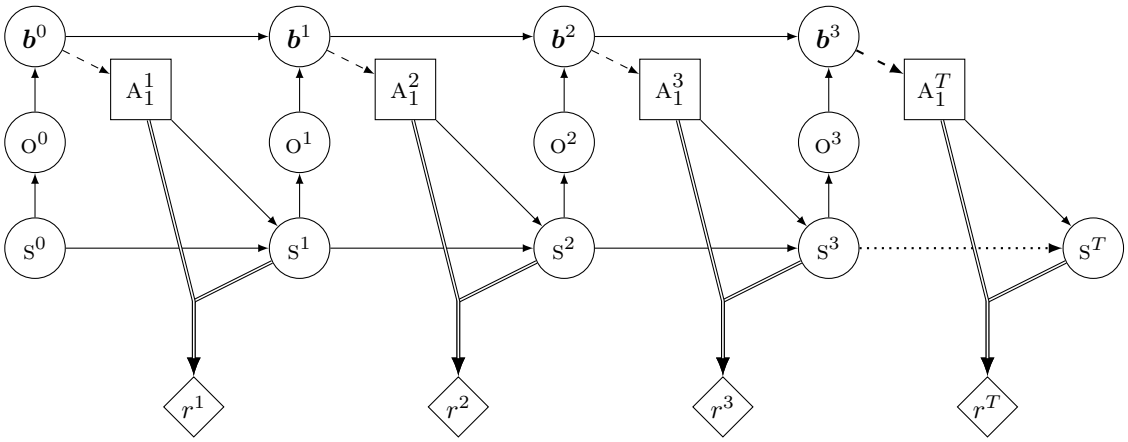


FIGURE 2.3 – Représentation graphique d'un POMDP avec état de croyance.

L'équation (2.7) montre au travers de la figure 2.3 que la distribution sur les états de croyance est une information markovienne. Il a été montré par Astrom [1965] que l'état de croyance constitue une information suffisante pour représenter le passé du système. il est donc possible de considérer un POMDP comme un MDP à états continus, un *belief state* MDP. L'obstacle majeur à sa résolution réside dans la continuité de son espace d'états. Un opérateur de type VI devrait donc être appliqué une infinité de fois pour établir une fonction de valeur. Il a cependant été montré par Smallwood et Sondik [1973] que la fonction de valeur à l'itération i peut toujours être représentée par un ensemble fini de paramètres. En effet, si V^i est une fonction de valeur convexe et linéaire par morceaux, alors la fonction V^{i+1} est aussi convexe et linéaire par morceaux. On peut montrer en particulier que la fonction de valeur à horizon 1 est forcément convexe et linéaire par morceaux. Ceci garantit, avec la propriété précédente, que la fonction de valeur pour un horizon fini est toujours représentable par des moyens finis. Le théorème de Smallwood et Sondik [1973] représente le fondement essentiel de la résolution de POMDPs :

Théorème 1. (Linéarité par morceaux et convexité de la fonction de valeur). *Si V^i représente une fonction de valeur convexe et linéaire par morceaux, alors l'opérateur de Bellman H (2.3) conserve cette propriété et la fonction de valeur $V^{i+1} = H(V^i)$ est aussi convexe et linéaire par morceaux.*

Nous avons constaté que l'observation courante ne vérifie plus la propriété de Markov dans le cas POMDP. A la différence du MDP, une politique optimale pour un POMDP dépend donc de manière générale de l'historique. Elle peut être représentée sous forme d'arbre de décision q, encore appelé *arbre de politique* [Kaelbling *et al.*, 1998]. Les nœuds de l'arbre représentent les actions à effectuer, et le parcours de l'arbre est choisi en fonction des observations reçues. Nous appelons $\alpha(q)$ l'action associée à la racine de l'arbre et $q(o)$ le sous-arbre associé à l'observation o. Un exemple est donné en figure 2.4.

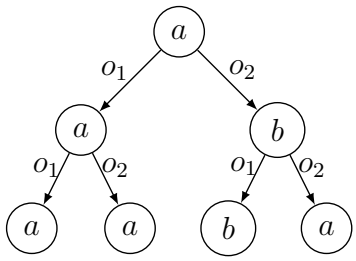


FIGURE 2.4 – Exemple d'arbre de politique à horizon trois, pour un problème à deux observations o_1 et o_2, et à deux actions a et b.

2.1.3.2 Résolution exacte des POMDP par programmation dynamique

On peut toujours représenter une politique q comme une vecteur à $\mathcal{S}$ dimensions, appelé $\boldsymbol{\alpha}$-vecteur $\boldsymbol{\alpha} = \langle V_q(s_1), \ldots, V_q(s_{|\mathcal{S}|})\rangle$. Il représente la valeur de la politique à la frontière de l'état de croyance. Dès lors, toute valeur d'un état de croyance intermédiaire peut être obtenu par interpolation linéaire :

$$V_q(\boldsymbol{b}) = \sum_{s\in\mathcal{S}} \boldsymbol{b}(s)V_q(s)$$

Étant donné un ensemble de politiques Q, la valeur optimale pour chaque état de croyance $\boldsymbol{b}$ peut être déterminée par la maximisation sur l'ensemble des politiques :

$$V(\boldsymbol{b}) = \max_{q\in Q} V_q(\boldsymbol{b}) = \max_{q\in Q} \sum_{s\in\mathcal{S}} \boldsymbol{b}(s)V_q(s)$$

Selon Smallwood et Sondik [1973], l'ensemble des politiques, et par extension l'ensemble des $\boldsymbol{\alpha}$-vecteurs, nécessaires pour représenter la politique optimale est toujours fini. Un exemple d'une fonction de valeur est donné par la figure 2.5. Pour chaque état de croyance $\boldsymbol{b}$, il existe une politique optimale, et un $\boldsymbol{\alpha}$-vecteur optimal qui domine tous les autres. Il est donc possible d'identifier des $\boldsymbol{\alpha}$-vecteurs qui sont dominés sur l'ensemble de l'état de croyance et de les éliminer sans affecter la politique optimale. Formellement, un $\boldsymbol{\alpha}$-vecteur est dominé, si la politique q associée vérifie :

$$\forall \boldsymbol{b}, \exists \tilde{q} \neq q \text{ tel que } V_q(\boldsymbol{b}) \leqslant V_{\tilde{q}}(\boldsymbol{b})$$

Un exemple de fonction de valeur avec deux $\boldsymbol{\alpha}$-vecteurs utiles et un $\boldsymbol{\alpha}$-vecteur dominé est donné par la figure 2.6.

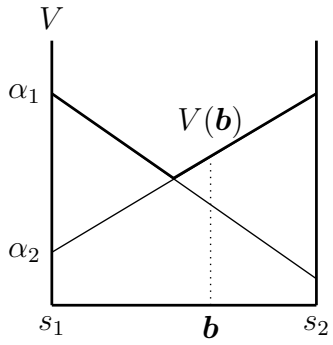


FIGURE 2.5 – Fonction de valeur d'un POMDP à deux états s_1 et s_2, représentée par deux vecteurs $\boldsymbol{\alpha}_1$ et $\boldsymbol{\alpha}_2$. L'action associée au vecteur $\boldsymbol{\alpha}_2$ est l'action optimale dans l'état de croyance $\boldsymbol{b}$.

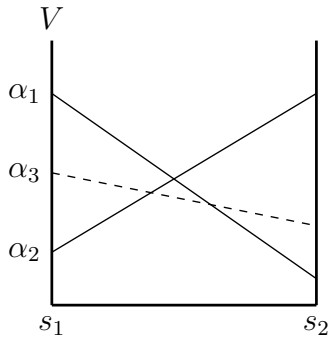


FIGURE 2.6 – Fonction de valeur d'un POMDP à deux états s_1 et s_2, représentée par trois vecteurs $\boldsymbol{\alpha}_1$, $\boldsymbol{\alpha}_2$ et $\boldsymbol{\alpha}_3$. Le vecteur $\boldsymbol{\alpha}_3$ est entièrement dominé et peut être éliminé sans affecter la politique optimale.

Nous avons vu qu'il existe une correspondance entre l'ensemble des vecteurs $\boldsymbol{\alpha} \in \Gamma$ qui constituent la fonction de valeur à l'horizon t, et l'ensemble des arbres de politiques $q \in Q$ de profondeur t. Calculer la fonction de valeur optimale par programmation dynamique est donc équivalent à construire l'ensemble des arbres de politiques associés. De la même manière que dans les MDPs, on procède incrémentalement horizon par horizon.

Pour cela, supposons l'existence d'une politique optimale à l'horizon t, représentée par un ensemble d'$\boldsymbol{\alpha}$-vecteurs Γ^t, ou de manière équivalente par un ensemble de politiques Q^t. Au début de l'algorithme, cet ensemble est constitué des politiques à horizon 1, soit l'ensemble des actions $\mathcal{A}$. Construire l'ensemble des politiques à l'horizon $(t+1)$ correspond à d'abord énumérer toutes les politiques possibles pour cet horizon. Pour

construire un nouvel arbre de politique q^{t+1}, on choisit une action $a \in \mathcal{A}$ qu'on place à la racine, puis on ajoute $|\Omega|$ politiques $q_i^t \in Q^t$ associée à chaque observation o_i. Il y a donc un total de $|\mathcal{A}||Q^t|^{|\Omega|}$ politiques à générer dans ce processus que nous appelons la *génération exhaustive de politiques* (Algorithme 5) :

Algorithm 5 (ExGen) Génération exhaustive de politiques.

requiert: un ensemble de politique Q^t pour l'horizon t
retourne: un ensemble de politique Q^{t+1}pour l'horizon $t+1$

fonction ExGen(Q^t)
  pour tout action $a \in \mathcal{A}$ **faire**
    pour tout selection possible de Ω politiques q_i^t parmi Q^t avec $1 \leqslant i \leqslant |\Omega|$ **faire**
      Construire une politique q^{t+1} telle que.
      $\alpha(q^{t+1}) \leftarrow a$ ▷ Place a comme racine
      $q^{t+1}(o_i) \leftarrow q_i^t,\ \forall o_i \in \Omega$ ▷ Choisit un sous-arbre pour chaque observation
      $Q^{t+1} \leftarrow Q^{t+1} \bigcup \{q^{t+1}\}$ ▷ Ajoute la politique à l'ensemble résultat
    fin pour
  fin pour
fin fonction

La deuxième étape consiste à éliminer les politiques complètement dominées du nouvel ensemble de politiques Q^{t+1}. Cette étape implique généralement de la programmation linéaire comme le montre l'algorithme 6. L'approche générale de programmation dynamique dans les POMDPs est donnée par l'algorithme 7. Il est important de noter que la construction des arbres de politiques se fait de *bas en haut*, i.e. que les premières politiques à être générées sont les dernières à être exécutées. À chaque itération, un nouvel ensemble de politiques est généré, et chacun de ces arbres de politique contient les politiques de l'itération précédente comme sous-arbre. Le principal problème de cet algorithme provient de la génération exhaustive des politiques avant l'élimination des politiques dominées. Des techniques un peu plus avancées permettent l'entrelacement de la génération et de l'élimination :

Algorithm 6 Programme linéaire pour l'identification des politiques dominées. [Monahan, 1982]

Maximiser	ε	
sous contraintes	$V_{\tilde{q}}(\boldsymbol{b}) - V_q(\boldsymbol{b}) \geqslant \varepsilon$	$\forall \tilde{q} \neq q$
avec	$\sum_{s \in \mathcal{S}} \boldsymbol{b}(s) = 1,\ \boldsymbol{b}(s) \geqslant 0,$	$\forall s \in \mathcal{S}$

Les formalismes MDP et POMDP constituent les bases théoriques fondamentales

Algorithm 7 Programmation Dynamique pour les POMDPs. [Cassandra *et al.*, 1994]

requiert: un POMDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O} \rangle$ et un horizon T
retourne: un ensemble de politiques optimales pour l'horizon T

```
Q^1 ← A
pour t = 2 to T faire
    pour tout s ∈ S faire
        Q^t ← ExGen(Q^t)                ▷ Génération exhaustive des politiques
        Éliminer chaque q ∈ Q^t si ∀b, ∃q̃ ≠ q telle que V_q(b) ⩽ V_q̃(b)
    fin pour
fin pour
```

de notre travail et présentent des notions telles que celle de la fonction de valeur, de politique ou d'état de croyance. Ces modèles ont été montrés comme étant des modèles très généraux et très expressifs pour quasiment tout -- ou presque -- type de problème stochastique impliquant un seul agent. Nous allons maintenant voir les différentes extensions de ces modèles au cas ou plusieurs agents évoluent dans un même environnement.

2.2 Modèles pour plusieurs agents

Dans cette section, nous présentons les extensions naturelles des formalismes précédents aux problèmes multiagents où la prise de décision est distribuée. Ceci implique que plusieurs agents évoluent *en même temps dans le même environnement*. Chaque agent choisit son action en fonction d'une politique locale, mais l'évolution du système dépend de l'*action jointe* de tous les agents. Nous allons présenter les différents modèles les plus utilisés à ce jour en commençant par les modèles se rapprochant le plus des MDPs et POMDPs.

2.2.1 MDP multiagent (MMDP)

Le modèle multiagent le plus simple est l'extension naturelle des MDPs à l'univers multiagent et a été introduit par Boutilier [1999]. Ce modèle fait l'hypothèse que l'état du monde est parfaitement connu à tout instant. Formellement,

Définition 4. *(MMDP) Un* MMDP *est défini par un tuple* $\langle \alpha, \mathcal{S}, \{\mathcal{A}_i\}_{i \in \alpha}, \mathcal{T}, \mathcal{R} \rangle$*, où :*
- α *est un ensemble fini d'*agents $i \in \alpha, 1 \leqslant i \leqslant n$ *;*

- $\mathcal{S}$ *est un ensemble fini d'*états $s \in \mathcal{S}$ *;*
- $\mathcal{A}_i$ *est un ensemble fini d'*actions *pour l'agent* i $a_i \in \mathcal{A}_i$ *;*
- $\mathcal{T}(s, a_1, \ldots, a_n, s') : \mathcal{S} \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_n \times \mathcal{S} \mapsto [0, 1]$ *est la* probabilité de transition *du système de l'état* s *vers l'état* s' *après l'exécution de l'action jointe* $\langle a_1, \ldots, a_n, \rangle$ *;*
- $\mathcal{R}(s, a_1, \ldots, a_n) : \mathcal{S} \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_n \mapsto \mathbb{R}$ *est la* récompense *produite par le système lorsque l'action jointe* $\langle a_1, \ldots, a_n, \rangle$ *est exécutée dans l'état* s*.*

Ce modèle est strictement équivalent à un MDP *joint* où $\mathcal{A} = \bigtimes_{i \in \alpha} \mathcal{A}_i$.

Graphiquement, ce modèle se représente comme montré par la figure 2.7 pour le cas à deux agents. Chacun des agents connaît l'état à chaque étape de décision et influence l'état suivant. Les récompenses ont été omises pour des raisons de clarté.

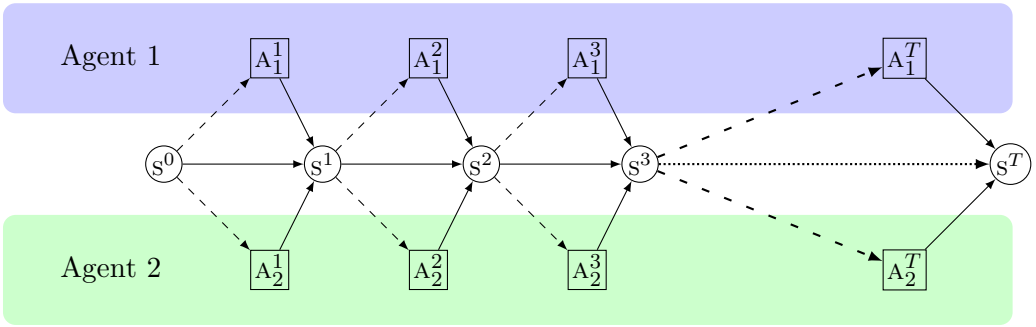


FIGURE 2.7 – Problème de décision de Markov multiagent (MMDP).

La solution que l'on souhaite trouver pour un problème modélisé avec un MMDP consiste à trouver un comportement pour chacun des agents qui correspond à un comportement globalement optimal. On peut ainsi définir ce que l'on appelle une politique individuelle et une politique jointe :

Définition 5. *(*Politique individuelle*) Une politique individuelle pour l'agent* i*,* q_i*, est une association de l'état* s *du système à une action* $a \in \mathcal{A}_i$ *de l'agent* i*.*

Définition 6. *(*Politique Jointe*) Une politique jointe,* $\boldsymbol{q} = \langle q_1, \ldots, q_n \rangle$*, est un tuple de politiques locales, une pour chaque agent.*

Tous les algorithmes centralisés pour calculer une politique jointe optimale sont alors aussi efficaces pour ce genre de modèle que pour le MDP joint correspondant. Cependant, Il peut arriver que pour un problème considéré, il n'existe pas une *unique* solution optimale, et si l'on souhaite exécuter la politique de manière distribuée, alors peut survenir un problème de *coordination* entre les agents. Par exemple, alors que chaque

agent peut choisir un politique individuelle induite par une politique jointe optimale, il n'y a aucune garantie que chacun des agents choisisse la *même politique jointe optimale*, et donc que l'action jointe soit optimale comme dans l'exemple donné par la figure 2.8.

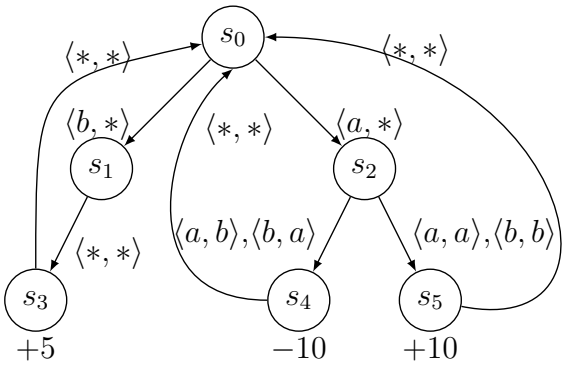


FIGURE 2.8 – Exemple de problème de coordination dans un MMDP à 2 agents [Boutilier, 1999].

2.2.2 MDP décentralisé partiellement observable (DEC-POMDP)

Dans le cas où l'environnement est seulement partiellement observable, Bernstein *et al.* [2000] ont proposé en 2000 un formalisme pour le contrôle décentralisé de plusieurs agents. Ils ont ensuite étendu le modèle d'observation en 2002 [Bernstein *et al.*, 2002] pour arriver à la formalisation suivante.

Définition 7. *(*DEC-POMDP*) Un* DEC-POMDP *est défini par un tuple* $\langle \alpha, \mathcal{S}, \{\mathcal{A}_i\}_{i\in\alpha}, \mathcal{T}, \{\Omega_i\}_{i\in\alpha}, \mathcal{O}, \mathcal{R}, T \rangle$*, où :*

- α *est un ensemble fini d'*agents $i \in \alpha, 1 \leqslant i \leqslant n$ *;*
- $\mathcal{S}$ *est un ensemble fini d'*états $s \in \mathcal{S}$ *avec un état initial* s_0 *;*
- $\mathcal{A}_i$ *est un ensemble fini d'*actions *pour l'agent* i $a_i \in \mathcal{A}_i$ *et* $\boldsymbol{\mathcal{A}} = \bigtimes_{i\in\alpha} \mathcal{A}_i$ *est l'ensemble des actions jointes et où* $\boldsymbol{a} = \langle a_1, \ldots, a_n \rangle \in \boldsymbol{\mathcal{A}}$ *est une action jointe ;*
- $\mathcal{T}(s, \boldsymbol{a}, s') : \mathcal{S} \times \boldsymbol{\mathcal{A}} \times \mathcal{S} \mapsto [0, 1]$ *est la* probabilité de transition *du système de l'état* s *vers l'état* s' *après l'exécution de l'action jointe* $\boldsymbol{a}$ *;*
- Ω_i *est un ensemble fini d'*observations *pour l'agent* i *et* $\boldsymbol{\Omega} = \bigtimes_{i\in\alpha} \Omega_i$ *est l'ensemble des observations jointes et où* $\boldsymbol{o} = \langle o_1, \ldots, o_n \rangle \in \boldsymbol{\Omega}$ *est une observation jointe ;*
- $\mathcal{O}(s, \boldsymbol{a}, \boldsymbol{o}, s') : \mathcal{S} \times \boldsymbol{\mathcal{A}} \times \boldsymbol{\Omega} \times \mathcal{S} \mapsto [0, 1]$ *est la* probabilité d'observation *d'avoir l'observation jointe* $\boldsymbol{o}$ *après transition du système de l'état* s *vers l'état* s' *par l'exécution de l'action jointe* $\boldsymbol{a}$ *;*
- $\mathcal{R}(s, \boldsymbol{a}) : \mathcal{S} \times \boldsymbol{\mathcal{A}} \mapsto \mathbb{R}$ *est la* récompense *produite par le système lorsque l'action jointe* $\boldsymbol{a}$ *est exécutée dans l'état* s *;*
- T *est l'horizon de planification.*

Le processus qui considère les actions jointes comme actions atomiques, et qui ne permet donc pas une exécution décentralisée, est appelé POMDP *sous-jacent au* DEC-POMDP.

L'exemple 1 présente un problème jouet avec 4 états, 2 actions et 2 observations que l'on peut modéliser avec un DEC-POMDP.

Exemple 1. MultiAgent Broadcast Channel *(*MABC*) : Le problème de partage de communication [Hansen et al., 2004] modélise deux agents communiquant par un canal à bande passante limitée. Chacun des agents doit envoyer un ensemble de messages à l'autre agent via ce canal, mais un seul message peut passer à la fois sur le canal ou autrement une collision se produit. Les agents ont pour but de maximiser le nombre de messages échangés. A chaque étape de temps, les agents doivent décider de communiquer ou non un message. Les deux agents reçoivent une récompense de 1 lorsqu'un message et transmis, et 0 si une collision est produite ou si aucun message n'est transmis. À la fin de chaque étape de temps, les agents reçoivent une observation parfaite sur le contenu de leur propre* buffer *et une observation bruitée s'il y a eu collision ou non en cas d'envoi de message. Le défi de ce problème est que les observations sont bruitées et donc que les agents ne peuvent construire qu'une croyance sur les résultats de leurs actions.*

Graphiquement, un DEC-POMDP se représente comme montré par la figure 2.9 pour le cas à deux agents. Chacun des agents connaît seulement une observation de l'état à chaque étape de décision et influence l'état suivant. Les récompenses ont été omises pour des raisons de clarté. Les états de croyance de chaque agent ont été représentés pour bien marquer le fait qu'aucun des deux agents n'a accès à de l'information sur les observations ou les actions de l'autre agent.

Résoudre un DEC-POMDP revient à trouver une politique locale par agent, c'est à dire une politique jointe pour l'équipe, telle que son exécution distribuée, mais synchrone maximise l'espérance des récompenses accumulées par le système. On peut donc de la même manière que les MMDPs définir des politiques locales et jointes, mais basées maintenant sur les observations.

Définition 8. *(*Politique individuelle*) [Seuken et Zilberstein, 2005] Une politique individuelle pour l'agent i, q_i, est une association d'un historique local des observations*

$$\bar{o} = o_{i1} \dots o_{it}, o_{ij} \in \Omega_i$$

à une action $a \in \mathcal{A}_i$ de l'agent i.

Définition 9. *(*Politique Jointe*) [Seuken et Zilberstein, 2005] Une politique jointe, $\boldsymbol{q} = \langle q_1, \dots, q_n \rangle$, est un tuple de politiques locales, une pour chaque agent.*

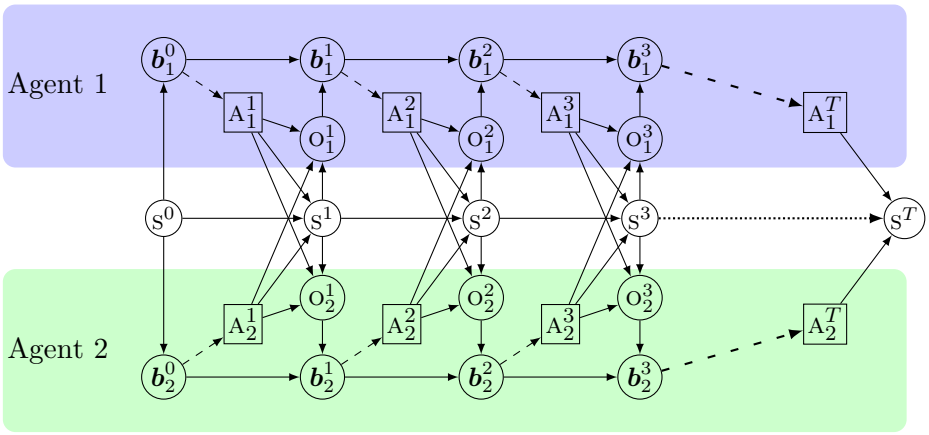


FIGURE 2.9 – Problème de décision de Markov décentralisé partiellement observable (DEC-POMDP).

Selon Szer [2006], établir une fonction de valeur dans le cadre d'un DEC-POMDP nécessite la redéfinition du concept d'état de croyance. Tout d'abord, l'*état de croyance multiagent* devient subjectif, et chaque agent possède une autre croyance sur le système due aux observations locales. Ensuite, l'état de croyance multiagent ne consiste plus seulement en une distribution de probabilité sur l'état sous-jacent du système, mais doit en outre prendre en compte le comportement futur des autres agents. Pour déterminer la valeur d'une politique dans un cadre multiagent, il faut donc prendre en compte le comportement futur des autres agents. Si Q_i dénote l'ensemble des politiques probables de l'agent i, et $\boldsymbol{Q}_{-i} = \langle Q_1, \ldots, Q_{i-1}, Q_{i+1}, \ldots, Q_n \rangle$ les ensembles des politiques probables de tous les autres agents, alors un état de croyance multiagent $\boldsymbol{b}_i$ pour un agent i est une distribution de probabilité sur $\mathcal{S}$ et sur $\boldsymbol{Q}_{-i}$. $\boldsymbol{b}_i \in \mathcal{B}_i = \Delta(\mathcal{S} \times \boldsymbol{Q}_{-i})$.

La définition d'un espace de croyance multiagent permet d'étendre le concept de fonction de valeur aux DEC-POMDPs. On note V_i la fonction de valeur locale de l'agent i. Si $\boldsymbol{q}_{-i}$ est la politique jointe pour tous les autres agents sauf i, et si $\langle \boldsymbol{q}_{-i}, q_i \rangle$ dénote une politique joint complète, alors la valeur de la politique q_i en l'état de croyance $\boldsymbol{b}_i$ peut être déterminée de la manière suivante :

$$V_{i,q_i}(\boldsymbol{b}_i) = \sum_{s \in \mathcal{S}} \sum_{\boldsymbol{q}_{-i} \in \boldsymbol{Q}_{-i}} \boldsymbol{b}(s,) V(s, \langle \boldsymbol{q}_{-i}, q_i \rangle)$$

Bernstein *et al.* [2002] ont montré que calculer une politique qui apporte une récompense cumulée à partir d'un état s_0 supérieure à K dans un DEC-POMDP est NEXP-complet en réduisant le problème à un problème de TILING.

Notons quand même un cas particulier de DEC-POMDPs que sont les DEC-MDPs. Ce modèle fait l'hypothèse d'un environnement localement totalement observable. Voyons

les simplifications induites par cette hypothèse.

2.2.2.1 DEC-POMDP localement observable (DEC-MDP)

L'hypothèse exacte réalisée dans un DEC-MDP est que l'environnement est parfaitement observable, mais seulement localement. L'union de toutes les observations des agents à un instant donné est égale à l'état réel du système à cet instant :

$$\begin{aligned} &\forall \boldsymbol{a} \in \boldsymbol{\mathcal{A}}, \forall s, s' \in \mathcal{S} \text{ et } \forall \boldsymbol{o} \in \boldsymbol{\Omega} \text{ tel que } \mathcal{O}(s, \boldsymbol{a}, \boldsymbol{o}, s') > 0 \\ &\exists L : \boldsymbol{\Omega} \mapsto \mathcal{S} \text{ tel que } L(\boldsymbol{o}) = s' \end{aligned} \tag{2.8}$$

Hélas, on peut également montrer (et c'est même à partir de ce modèle que Bernstein *et al.* [2002] ont fait la réduction) que calculer une politique qui apporte une récompense cumulée à partir d'un état s_0 supérieure à K dans un DEC-MDP est NEXP-complet. Nous reviendrons également plus en détail sur ce modèle au chapitre 4 puisque nous étudierons plus en profondeur les effets des observations sur la complexité du problème.

D'autres modèles multiagents de Markov existent également. Par exemple, les POMDPs interactifs (I-POMDPs) [Gmytrasiewicz et Doshi, 2005] représentent chaque agent par un POMDP (identique pour tous) ayant un état de croyance explicite sur les autres agents. Incidemment, chaque agent qui maintient un état de croyance complet sur les autres maintient également un état de croyance sur les croyances des autres agents et donc un état de croyance sur les croyances des autres sur ses propres croyances, ceci menant à des croyances infiniment imbriquées et non résolvables sans autre hypothèse.

Un autre exemple est le problème de décision d'équipe multiagent (MTDP) proposé par Pynadath et Tambe [2002]. Ce modèle a été démontré comme complètement équivalent aux DEC-POMDPs par Seuken et Zilberstein [2005] sous certaines hypothèses. Par exemple, MTDP a considéré le cas où les agents peuvent avoir une mémoire limitée ou bien peuvent communiquer. Ainsi, un DEC-POMDP est un MTDP sans communication et avec mémoire infinie. Pour une analyse plus complète et détaillée de ces deux derniers modèles, le lecteur intéressé est invité à lire l'article de Seuken et Zilberstein [2008].

Voyons rapidement les différents modèles avec communication.

2.2.2.2 Modèles avec communication

Les modèles multiagents incluant la communication se basent principalement sur les modèles précédent en ajoutant simplement des actions spécifiques de communication à chaque étape qui consistent à choisir le message à envoyer parmi un ensemble de messages. Plus formellement :

Définition 10. *(*DEC-POMDP-COM*) Un* DEC-POMDP-COM *est défini par un tuple* $\langle \alpha, \mathcal{S}, \{\mathcal{A}_i\}_{i\in\alpha}, \mathcal{T}, \{\Omega_i\}_{i\in\alpha}, \mathcal{O}, \Sigma, C_\Sigma \mathcal{R}, T\rangle$*, où :*

- $\langle \alpha, \mathcal{S}, \{\mathcal{A}_i\}_{i\in\alpha}, \mathcal{T}, \{\Omega_i\}_{i\in\alpha}, \mathcal{O}, T\rangle$ *définit un* DEC-POMDP *sans la fonction de récompense ;*
- Σ *est l'alphabet de communication.* $\sigma_i \in \Sigma$ *est un message atomique envoyé par l'agent* i *et* $\sigma = \langle \sigma_1, ..., \sigma_n \rangle$ *est un message joint, i.e. le tuple des messages échangés à une étape de temps. Un message particulier* ε_σ *fait également partie de* Σ *et représente le message* null *de coût zero qui est envoyé lorsqu'un agent ne désire pas communiquer.*
- $C_\Sigma : \Sigma \mapsto \mathbb{R}$ *est la fonction de coût des messages et* $C_\Sigma(\varepsilon_\sigma) = 0$*.*
- $\mathcal{R} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \times \boldsymbol{\Sigma} \mapsto \mathbb{R}$ *définit la fonction de récompense correspond à la récompense produite par le système lorsque l'action jointe* $\boldsymbol{a}$ *est exécutée dans l'état* s *et que le message* σ *est envoyé.*

Il est important de mentionner que pour ce modèle, les ensembles d'actions $\{\mathcal{A}_i\}$ *ne contiennent aucune action de communication entre les agents.*

La différence de ce modèle avec le modèle MTDP-COM [Pynadath et Tambe, 2002] réside dans la gestion des états de croyance. Ce dernier effectue en réalité deux mise à jour de l'état de croyance. Une première est faite lors de la réception de l'observation locale de l'agent, mais avant la communication pour décider quelle doit être le message à transmettre aux autres agents. La seconde est faite lors de la réception du message des autres agents pour incorporer l'information obtenue à sa croyance avant le choix de la prochaine action.

Malheureusement, les problèmes décentralisés munis de la communication restent tout aussi difficiles que les problèmes sans communication si celle-ci n'est pas gratuite ou si le contenu des messages n'est pas explicitement représenté. La table 2.1 résume la complexité des modèles multiagents présentés ci-avant pour plusieurs formes de communication.

Voyons maintenant comment résoudre un DEC-POMDP.

	Complètement Observable	Localement Observable	Partiellement Observable	Non Observable
Pas de Comm.	P-complet	NEXP-complet	NEXP-complet	NP-complet
Comm. avec coût	P-complet	NEXP-complet	NEXP-complet	NP-complet
Comm. gratuite	P-complet	P-complet	PSPACE-complet	NP-complet

TABLE 2.1 – Complexité en temps des DEC-POMDPs-COM

2.2.2.3 Résolution optimale de DEC POMDPs

La NEXP-complétude suggère que quelque soit l'algorithme optimal utilisé pour résoudre un problème formulé comme un DEC-POMDP, un temps doublement exponentiel en pire cas sera nécessaire. Cependant en dehors des travaux effectués sur les algorithmes optimaux, il existe également des algorithmes heuristiques qui, bien qu'ils n'offrent aucune garantie théorique, trouvent tout de même de ''bonnes'' politiques. Nous allons dans un premier temps s'intéresser aux algorithmes optimaux existant pour bien comprendre la difficulté de résoudre un DEC-POMDP. Nous verrons par la suite quelles hypothèses et quelles approximations ont été effectuées pour en arriver aux algorithmes approchés.

Une politique optimale à horizon fini [1] peut-être représentée par un arbre de décision q, où les nœuds sont marqués par des actions et les arcs par des observations. Une solution d'un DEC-POMDP à horizon t peut alors être vue comme un vecteur $\boldsymbol{q}^t$ d'arbres de décision de profondeur t (appelés aussi arbres de politique dans la littérature), $\boldsymbol{q}^t = \langle q_1^t, ..., q_n^t \rangle$, un pour chaque agent, où $q_i^t \in Q_i^t$. La figure 2.10 donne un exemple d'arbres de décisions joints pour un problème avec deux actions et deux observations à horizon 1 et 2 et montre comment le nombre de politiques possible pour chacun des agents croît de manière exponentielle.

Une construction incrémentale de la politique est utilisée ici. L'ensemble Q_i^{t+1} des politiques de l'agent i à l'horizon $t+1$ peut être construit à partir de l'ensemble des politiques à l'horizon t en ajoutant une étape observation-décision à chaque feuille de l'arbre. Ainsi, pour chaque feuille de $q_i^t \in Q_i^t$, $|\Omega|$ nouvelles feuilles filles sont ajoutées, et pour chacune d'entre elles, une décision est choisie.

Le nombre total d'arbres de politique pour un agent est le nombre de combinaison d'assignation d'actions possible étant donné tous les historiques possibles. Pour l'horizon 1, une seule décision doit être prise donc $|\mathcal{A}|$ politiques peuvent être suivies. Pour l'horizon 2, pour chaque observation possiblement obtenue après la première action, il

1. Les horizons infinis étant indécidables même pour un seul agent Madani *et al.* [1999].

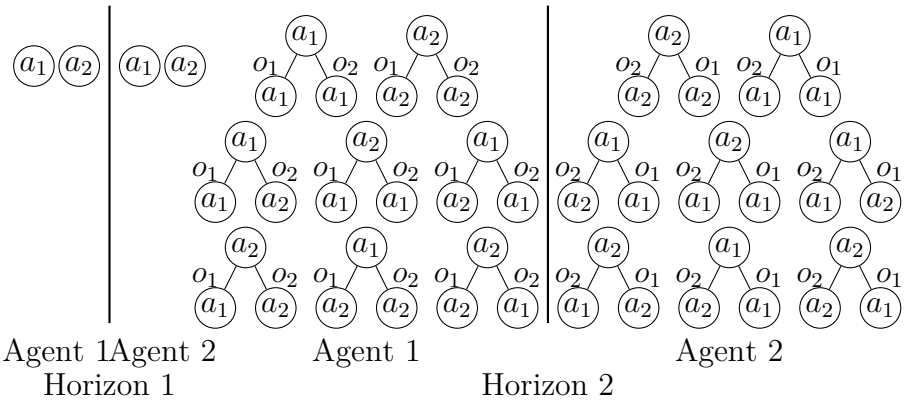


FIGURE 2.10 – Politiques possibles pour l'horizon 1 et 2 pour le problème MABC.

va falloir prendre un décision. Dans ce cas, $|\Omega|$ branches sont donc ajoutées à chacune des politiques initiales, et toutes les assignations d'actions au second niveau sont alors possibles. Ceci résulte en $|\mathcal{A}|^{(1+|\Omega|)}$ politiques possibles à l'horizon 2. En continuant de manière similaire, on trouve qu'il faudra encore ajouter un troisième niveau étant donné chaque séquence possible de deux observations pour se retrouver avec $|\mathcal{A}|^{(1+|\Omega|+|\Omega|^2)}$ politiques possibles. Il suit de ce raisonnement, qu'en général, pour un horizon t, le nombre de politiques possibles est de $|\mathcal{A}|^{(1+|\Omega|+\cdots+|\Omega|^{t-1})} = |\mathcal{A}|^{\frac{|\Omega|^t-1}{|\Omega|-1}} \in O(|\mathcal{A}|^{|\Omega|^t})$. Et de fait, l'algorithme est doublement exponentiel en l'horizon t. Seuken et Zilberstein [2008] font tout de même remarquer que $|\mathcal{A}|^{|\Omega|^{t-1}} < |\mathcal{A}|^{\frac{|\Omega|^t-1}{|\Omega|-1}} < |\mathcal{A}|^{|\Omega|^t}$.

Malheureusement, le problème est également exponentiel en nombre d'agents, puisque le nombre de politiques jointes est le produit des ensembles de politiques de chacun des agents. Ce nombre est donc $|\mathcal{A}|^{\left(\frac{|\Omega|^t-1}{|\Omega|-1}\right)^n}$. La table 2.2 montre le nombre de politiques qu'il faudrait explorer, et ce, pour seulement 2 agents lorsqu'il y a deux actions et deux observations.

Horizon	# de politiques pour un agent	# de politiques jointes (2 agents)
1	2	4
2	8	64
3	128	16 384
4	32 768	1 073 741 824
5	2 147 483 648	$4.6{\cdot}10^{18}$

TABLE 2.2 – Nombre de politiques jointes pour le problème MABC par force brute.

Deux algorithmes radicalement différents ont été développés pour rechercher la

meilleure politique jointe dans cet espace incommensurable : Un basé sur la programmation dynamique, qui élimine à chaque étape les politiques dominées, et un autre basé sur la recherche arborescente heuristique ($A^\star$). Détaillons maintenant ces deux algorithmes.

Programmation dynamique (dp)

Présenté par Hansen *et al.* [2004], la Programmation Dynamique (DP) pour les DEC-POMDPs généralise deux techniques du domaine : La programmation dynamique pour les POMDPs et l'élimination de politiques dominées.

L'idée principale de la programmation dynamique est de rechercher au travers de l'ensemble des politiques de manière incrémentale, en éliminant le plus tôt possible les politiques dominées, évitant ainsi de multiplier exagérément le nombre de politiques possibles. Le problème ici, soulevé un peu plus tôt dans ce chapitre, est que la notion d'état de croyance est également étendue aux systèmes multiagents. Ainsi, à l'instar des POMDPs, l'état de croyance de l'agent i à l'étape t n'est plus seulement une distribution sur les états du système, mais également sur l'ensemble des politiques possibles des autres agents $\mathcal{B}_i^t$. Incidemment, puisqu'à chaque étape de temps le nombre de politiques des autres agents croît de manière doublement exponentielle, l'espace des états de croyance croît de manière similaire.

De la même manière que les POMDPs, la fonction de valeur V est représentée sous la forme d'un ensemble d'$\boldsymbol{\alpha}$-vecteurs v_j de dimension $|\mathcal{S}|$. Elle est dénotée $V = \{v_1, ..., v_k\}$, où :

$$V_i^t(\boldsymbol{b}_i) = \max_{1 \leqslant j \leqslant k} \sum_{s \in \mathcal{S}} \sum_{\boldsymbol{q}_{-i} \in \boldsymbol{Q}_{-i}^t} \boldsymbol{b}_i(s, \boldsymbol{q}_{-i}^t) v_j(s, \langle \boldsymbol{q}_{-i}^t, q_i^t \rangle)$$

Où chaque v_j est un $\boldsymbol{\alpha}$-vecteur correspondant à un arbre de politique. La programmation dynamique exploite la possibilité que certains vecteurs puissent être dominés par d'autres :

Définition 11. *(Arbre de politique faiblement dominé) Un arbre de politique $q_j \in Q_j$ auquel correspond l'α-vecteur $v_j \in V_i^t$ est faiblement dominé si :*

$$\forall \boldsymbol{b}_i \in \mathcal{B}_i^t, \exists v_k \in V_i^t \setminus v_j \text{ tel que } \boldsymbol{b}_i v_k \geqslant \boldsymbol{b}_i v_j$$

Cependant, du fait que l'élimination de la politique d'un agent peut influencer la meilleure politique d'un autre agent, l'élimination de politique doit être fait itérativement jusqu'à ce qu'aucun agent ne puisse plus éliminer aucune politique. Ainsi, la programmation dynamique pour DEC-POMDPs entrelace la génération exhaustive des politiques

à l'horizon $t+1$ sachant l'horizon t, et l'élimination des politiques dominées dans le nouvel ensemble des politiques possibles pour chacun des agents. Dès que l'algorithme atteint un horizon voulu T, une politique est alors choisie, celle qui garantie la meilleure valeur espérée étant donné l'état de croyance initial.

Bien sûr, dans le pire cas, cet algorithme reste doublement exponentiel selon l'horizon voulu. Cependant, l'amélioration par rapport à un algorithme de type force brute réside dans l'élimination des politiques dominées à chaque étape, et ceci dépend du problème en particulier. Par exemple pour le problème MABC, les résultats reportés dans la table 2.3 de l'élimination des politiques dominées sont significatifs. Notons que la programmation dynamique ne produit que 1% des politiques produites par l'algorithme de force brute à l'itération 3 et 4, et que l'algorithme de force brute n'est même pas capable de se rendre à l'itération 4.

Horizon	# de politiques par force brute	# de politiques par DP
1	$(2, 2)$	$(2, 2)$
2	$(8, 8)$	$(6, 6)$
3	$(128, 128)$	$(20, 20)$
4	$(32768, 32768)$	$(300, 300)$

TABLE 2.3 – Nombre de politiques pour chaque agent pour le problème MABC par Programmation Dynamique.

Malheureusement, la programmation dynamique explose la mémoire disponible dès l'itéra-tion 4 à cause de la croissance doublement exponentielle du nombre de politiques. Même avec l'élimination des politiques dominées, le passage de l'horizon 4 à l'horizon 5 par énumération exhaustive aurait besoin de produire $2 \cdot 300^4$, soit plus de 16 milliards d'α-vecteurs, avant même de tenter d'éliminer les dominés. Ceci explique clairement pourquoi l'algorithme explose les limites spatiales avant d'exploser les limites temporelles.

Résolution avec heuristique (maa*)

Une autre approche appelée *multiagent* A*, proposée par Szer *et al.* [2005], est radicalement différente de la programmation dynamique vue précédemment. Elle consiste à rechercher par profondeur d'abord dans l'arbre des politiques jointes décentralisables en se concentrant, à l'aide d'heuristiques admissibles, sur les sous-politiques les plus prometteuses à la manière d'un algorithme de type A*.

Si on dénote par $V(s_0, \boldsymbol{q}^T)$ la valeur espérée d'exécuter la politique jointe $\boldsymbol{q}^T$ dans l'état initial s_0, alors l'objectif est de trouver la politique $\boldsymbol{q}^{\star T} = \arg\max_{\boldsymbol{q}^T} V(s_0, \boldsymbol{q}^T)$. De la même manière que l'algorithme force brute, l'algorithme MAA* recherche dans l'espace des vecteurs de politiques (ou politiques jointes), où les nœuds de l'arbre de recherche au niveau t correspondent aux solutions partielles du problème, des politiques à horizon t. Néanmoins, ce ne sont pas tous les nœuds de l'arbre qui sont complètement explorés ; à la place, une heuristique est utilisée pour évaluer les feuilles de l'arbre de recherche. Et c'est le nœud ayant la plus haute estimation qui est exploré à l'étape suivante. La figure 2.11 montre une section d'une telle recherche pour le problème MABC.

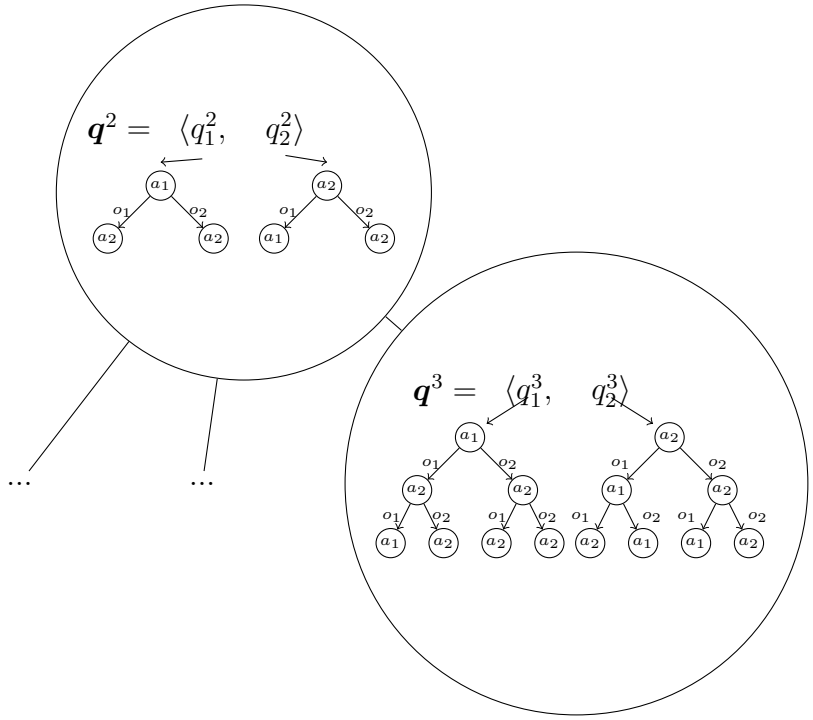


FIGURE 2.11 – Section de la recherche par MAA*, montrant une politique jointe à horizon 2 avec une des expansions possibles à l'horizon 3 pour le problème MABC.

À la manière d'A*, la fonction de valeur est décomposée en deux parties : une évaluation exacte de la solution partielle (le vecteur des politiques jusqu'au niveau courant), et une estimée par l'heuristique de la partie restante à compléter de la politique courante appelée *complétion* :

Définition 12. *(Complétion d'une politique jointe) Une complétion Φ^{T-t} d'une politique jointe $\boldsymbol{q}^t$ de profondeur t est un ensemble de politiques de profondeur $T-t$ qui peuvent être ajoutées à toutes les feuilles de la politique $\boldsymbol{q}^t$ de telle sorte que $\{\boldsymbol{q}^t, \Phi^{T-t}\}$ est une politique jointe de profondeur T.*

Ainsi, on peut décomposer la valeur V de toute politique de profondeur T en deux parties :

$$V(s_0, \{\boldsymbol{q}^t, \Phi^{T-t}\}) = V(s_0, \boldsymbol{q}^t) + V(\Phi^{T-t}|s_0, \boldsymbol{q}^t)$$

Évidemment, la valeur de la complétion dépend de l'état de croyance atteint après avoir exécuté la politique $\boldsymbol{q}^t$ dans l'état s_0. Plutôt que de calculer sa valeur exactement, l'algorithme l'estime à l'aide d'une heuristique donnée H. L'estimation de la valeur de la politique $\boldsymbol{q}^t$ dans l'état s_0 est alors donnée par :

$$F(s_0, \boldsymbol{q}^t) = V(s_0, \boldsymbol{q}^t) + H^{T-t}(s_0, \boldsymbol{q}^t)$$

Ici, de la même manière que A*, pour que la recherche soit *complète* et *optimale*, la fonction H doit être *admissible*, c'est à dire qu'elle doit surestimer la valeur de la complétion optimale de la politique $\boldsymbol{q}^t$:

$$\forall \Phi^{T-t} : H^{T-t}(s_0, \boldsymbol{q}^t) \geqslant V^*(\Phi^{T-t}|s_0, \boldsymbol{q}^t)$$

Le choix de l'heuristique est important aussi bien sur sa capacité à estimer un borne supérieure proche de la valeur optimale (pour pouvoir éliminer le plus possible des branches lors de la recherche), que sur la rapidité à pouvoir l'estimer. Pour décrire les différentes heuristiques proposées par Szer *et al.* [2005], les notations suivantes seront utilisées :

- $\Pr(s|s_0, \boldsymbol{q}^t)$ est la probabilité d'être dans l'état s après avoir exécuté la politique $\boldsymbol{q}^t$ dans l'état s_0.
- $h^t(s)$ est une estimation optimiste de la récompense espérée dans l'état s de la politique optimale : $h^t(s) \geqslant V^{\star t}(s)$.

La classe suivante d'heuristiques est alors définie :

$$H^{T-t}(s_0, \boldsymbol{q}^t) = \sum_{s \in \mathcal{S}} \Pr(s|s_0, \boldsymbol{q}^t) h^{T-t}(s)$$

Cette classe d'heuristiques simule la situation où l'état réel du système est révélé à l'agent à l'étape t après exécution de la politique $\boldsymbol{q}^t$. Il est possible de montrer que toute heuristique H définie de cette manière est admissible si h est admissible Szer *et al.* [2005]. De plus, le calcul de l'heuristique h pour un état s_0 et une politique $\boldsymbol{q}^t$ doit seulement être exécuté dans chacun des états, ce qui permet une amélioration significative des temps de calcul par rapport au temps doublement exponentiel initial. Trois heuristiques sont présentées dans Szer *et al.* [2005] :

mdp La plus simple consiste à considérer que les agents peuvent observer l'état à chaque étape à partir de l'étape t, un MMDP à horizon fini est ainsi défini pour les $T-t$ étapes restantes : $h^{T-t}(s) = V^{\text{MMDP}}_{T-t}(s)$. Du fait que la résolution du MMDP ne prend qu'un temps polynomial, cette heuristique est très rapide, mais également très loin de la valeur optimale.

pomdp En considérant non plus le MMDP, mais le MPOMDP, en faisant l'hypothèse qu'à chaque étape chaque agent aura les observations des autres agents, la borne obtenue est nettement plus serrée. Malheureusement, calculer cette heuristique est plus complexe puisqu'un POMDP est PSPACE-complet.

maa* récursif La borne la plus serrée consiste à supposer qu'on ne révèle l'état qu'une fois, et ce, à l'étape t, $h^t(s) = V^{\star t}(s)$, puis qu'on applique à nouveau l'algorithme MAA* avec cette nouvelle connaissance. En appliquant MAA* de manière récursive, $h^{T-t}(s) = \text{MAA}^{\star T-t}(s)$, la valeur est calculé en temps seulement exponentiel puisqu'utiliser l'algorithme MAA$^{\star T}$ consiste à utilise $|\mathcal{S}|$ fois l'algorithme MAA$^{\star T-1}$.

Évidemment, il y a ici un équilibre à trouver entre l'écart de la borne à la valeur optimale et le temps nécessaire à calculer cette borne. Seulement, à cause de la croissance doublement exponentielle de l'arbre de recherche, une borne plus serrée même plus coûteuse reste la solution la plus efficace en pratique. À ce jour, MAA* est le seul algorithme pouvant résoudre optimalement un problème simple à 2 agents, 2 états, 3 actions et 2 observations jusqu'à l'horizon 4. MAA* sort des limites de temps lors du calcul de la politique à l'horizon 5. Une critique plus approfondie de cet algorithme peut-être trouvée dans [Seuken et Zilberstein, 2008, sect. 3.1.3]. Pour améliorer la mise à l'échelle de ces algorithmes, Seuken et Zilberstein [2007b,a]; Carlin et Zilberstein [2008] ont proposé des algorithmes approximatifs que nous allons détailler maintenant.

2.2.2.4 Résolution approchée

Dans la partie précédente, nous avons vu que la complexité des DEC-POMDPs n'autorisait pas une résolution optimale au-delà de l'horizon 4, et ce, même pour des problèmes de très petite taille (2 états, 3 actions, 2 observations). L'idée de développer des algorithmes approximatifs est donc une avenue intéressante en pratique. Cependant, il a été montré par Rabinovich *et al.* [2003] que trouver une politique basée sur l'historique et optimale à ε près est NEXP-difficile pour les DEC-POMDPs et les DEC-MDPs. Nous parlerons donc ici de trouver une politique approximative, mais ne sous-tendant absolument aucune garantie de performance, c'est-à-dire donnant de ‘‘bons’’ résultats en pratique.

Parmi les algorithmes qui donnent ce genre de politiques, nous mous attarderons plus particulièrement sur deux algorithmes combinant astucieusement recherche arborescente (de haut en bas) et programmation dynamique (de bas en haut) et notamment le deuxième qui, avec quelques améliorations, a montré des résultats pour le moins intéressants sur des horizons bien plus grands et des problèmes moins triviaux que ceux adressés précédemment dans la littérature. Pour une revue complète de la littérature sur les algorithmes approximatifs, le lecteur intéressé est invité à lire [Seuken et Zilberstein,

2008, sect. 4.1].

Recherche basée sur un équilibre joint des politiques (jesp)
JESP (pour *Joint Equilibrium-Based Search for Policies*), présenté par Nair *et al.* [2003], est un algorithme qui recherche une solution localement optimale par ajustements successifs des politiques de chacun des agents.

Le principe de cet algorithme repose sur l'amélioration itérative des politiques des agents. La politique d'un agent est optimisée tandis que celles des autres agents sont fixées. Chacun des agents améliore ainsi sa politique jusqu'à ce que l'ensemble des agents converge vers l'équilibre de Nash. Une idée majeure ici consiste à optimiser les politiques seulement sur l'ensemble des états de croyance accessibles à partir d'un état de croyance initial donné $\boldsymbol{b}^0$. Puisque le nombre d'états de croyance augmente ''seulement'' de manière exponentielle, un gain substantiel est apporté ici. Pour passer à un horizon supérieur, JESP se base sur la génération exhaustive comme présentée à la section 2.2.2.3.

Dans sa version la plus efficace, DP-JESP, l'algorithme obtient des résultats singulièrement meilleurs que les approches optimales puisqu'il obtient de bons résultats pour le problème du tigre multiagent jusqu'à l'horizon 7 (cf. les résultats présentés table 2.4 à la section suivante). Cependant, cet algorithme supporte mal la mise à l'échelle en particulier du à la génération exhaustive des politiques et le nombre exponentiel de politiques à générer au contraire de l'algorithme que nous allons voir maintenant.

Programmation dynamique à mémoire limitée (mbdp)
Comme nous l'avons vu dans la section 2.2.2.3 et dans la sous-section précédente, le problème majeur de la programmation dynamique est l'explosion de l'utilisation de la mémoire due au nombre exponentiel de politiques possibles. Cependant, une analyse plus fine du processus d'élimination des politiques dominées révèle que la plupart des politiques non dominées sont inutiles étant donné un certain état de croyance initial. En effet, puisqu'après un certain nombre d'étapes, seulement un nombre restreint d'états de croyance sont accessibles, la plupart des politiques non dominées le sont sur ce sous-ensemble d'états de croyance.

L'idée principale de cet algorithme consiste donc à n'effectuer le processus de génération exhaustive des politiques jointes en utilisant que les meilleures politiques dans les états de croyance accessibles. Un portefeuille d'heuristiques est utilisé pour sélectionner un ensemble restreint d'états de croyance *accessibles* tandis que la programmation dyna-

mique n'est effectuée que pour les meilleures politiques dans ces états de croyance fixés. L'algorithme se décompose donc principalement en deux phases : une phase *forward* et une phase *backward* :

***Forward* : selection d'états de croyance :** Même si l'état de croyance joint n'est pas accessible durant l'exécution de la politique décentralisée, rien n'empêche de l'utiliser pour l'évaluation des politiques calculées à l'aide d'un algorithme de programmation dynamique par exemple. Évidemment, la séquence des états de croyance obtenue lors de l'exécution de la politique optimale n'est pas connue lors de la recherche de cette politique. Il existe cependant de nombreuses heuristiques possibles pour la sélection d'états de croyance : depuis une recherche complètement aléatoire, jusqu'à l'utilisation de la politique du MMDP sous-jacent. Il est même possible, une fois l'algorithme exécuté une première fois, de l'appeler *récursivement*, en utilisant la politique retournée la première fois pour sélectionner de nouveaux états de croyance. On sélectionne ensuite pour chacun des états de croyance sélectionnés la meilleure politique.

***Backward* : dp :** Néanmoins, pour éviter l'explosion exponentielle lors de la génération exhaustive des politiques, seulement on nombre restreint d'états de croyance sont sélectionnés ($maxTree$) de telle sorte que la mémoire utilisée à chaque itération reste constante et égale à $|\mathcal{A}|maxTree^{|\Omega|}$. Ainsi, après sélection par heuristique de $maxTree$ états de croyance, les meilleures politiques sont sélectionnées pour chacun d'entre eux puis une itération de programmation dynamique est effectuée par la génération exhaustive des $|\mathcal{A}|maxTree^{|\Omega|}$ combinaisons possibles de politiques. Finalement, après la T-ième génération, la meilleure politique pour l'état de croyance initial est choisie et retournée.

Améliorations par l'énumération partielle des observations (imbdp) : Bien que l'algorithme MBDP soit de complexité linéaire en temps et en espace par rapport à l'horizon, il reste exponentiel selon le nombre d'observations. Cela reste une limitation sévère lorsque l'on cherche à résoudre des problèmes autres que des problèmes jouets. Par exemple, même pour un problème à 2 actions, 5 observations, le nombre de paires de politiques avec un $maxTree$ de 5 serait de 39 062 500 !

Par conséquent, Seuken et Zilberstein [2007a] ont observé que ce ne sont pas toutes les observations qui sont importantes dans tous les états de croyance et à toutes les étapes. Il est possible par exemple d'avoir une observation à un instant donné et de ne plus jamais l'avoir par la suite.

Dans cet ordre d'idée la, ils ont donc proposé d'améliorer MBDP en ne conservant que les *maxObs* observations les plus probables lors de la génération exhaustive des politiques. Ainsi, après avoir identifié les meilleures politiques pour un ensemble donné d'états de croyance, la génération des politiques à l'horizon s'effectue de la manière suivante : Tout d'abord, les observations jointes sont triées selon leur probabilité pour l'état de croyance $\boldsymbol{b}^{t-1}$ le plus probable et l'action $\boldsymbol{a}$ dictée par l'heuristique :

$$\Pr(\boldsymbol{o}) = \sum_{s \in \mathcal{S}} \boldsymbol{b}^{t-1}(s)\mathcal{O}(\boldsymbol{o}|\boldsymbol{a}, s)$$

Puis, un rang est attribué à chacune des observations pour chacun des agents et les *maxObs* premières observations sont mémorisées. Une génération *partielle* est alors faite basée sur ces *maxObs* observations. Cela résulte aboutit, comme représenté dans la figure 2.12, à un ensemble de politiques incomplètes qui sont alors complétées par simple recherche locale.

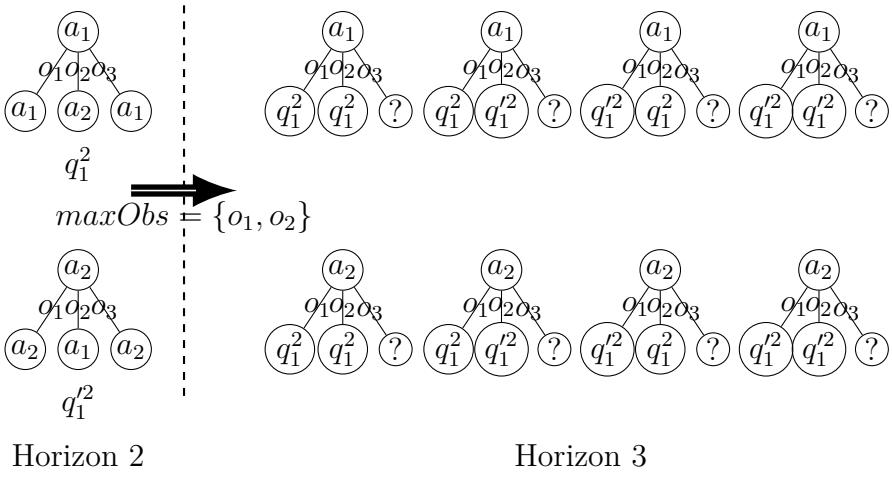


FIGURE 2.12 – Génération partielle pour l'horizon t sachant $t-1$ pour un problème à 2 actions et 3 observations avec $maxTree = 2$ et $maxObs = 2$ avant remplissage par recherche locale.

Amélioration utilisant la compression des observations (mbdp-oc) : Une dernière amélioration a été portée à l'algorithme MBDP en se basant sur la génération partielle des politiques, mais sans sacrifier arbitrairement les observations les moins probables. Cette amélio-ration, basée sur la compression des observations selon leur valeur permet de limiter tout autant le nombre d'observations à *maxObs* pendant la génération des politiques, mais le fait de manière à garantir la valeur perdue dans cette compression.

Ainsi, la où l'amélioration de MBDP (IMBDP) coupait simplement les observations les moins probables en leur assignant une politique trouvée par recherche locale, MBDP-OC regroupe les observations de telles sortes que l'application de la même politique garantisse la perte de valeur minimale et ce jusqu'à atteindre un nombre de groupe égal à *maxObs*. La figure 2.13 montre la différence avec IMBDP pour le même problème à la même itération.

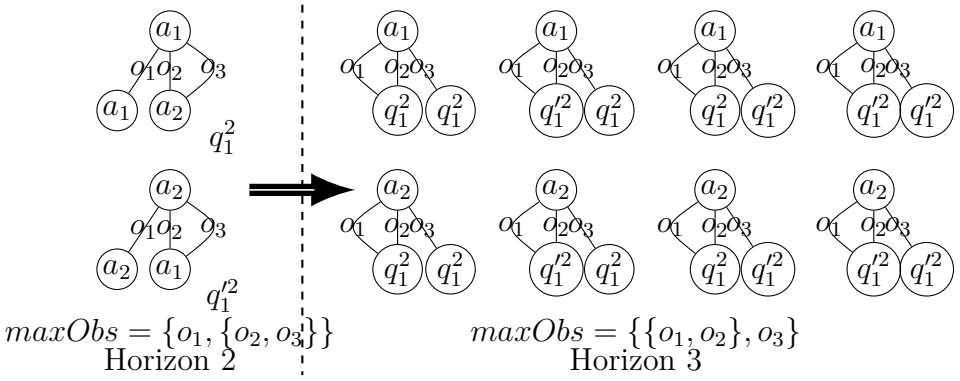


FIGURE 2.13 – Génération partielle avec compression des observations pour l'horizon t sachant $t-1$ pour un problème à 2 actions et 3 observations avec $maxTree = 2$ et $maxObs = 2$.

2.2.2.5 Résultats

La table 2.4 résume les différents résultats des algorithmes présentés dans cette section pour trois problèmes du domaine. Le problème du Tigre possède 2 états, 3 actions et 2 observations, le problème de partage du canal de communication (MABC) a pour sa part 4 états, 2 actions et 2 observations et le problème de déménagement (CBP) consiste en un problème plus complexe de 123 états, 4 actions et 5 observations.

Ces résultats montrent la progression des algorithmes depuis le milieu des années 2000. On remarquera que la résolution optimale restera très probablement bloquée à ce niveau de résolution à moins d'une évolution majeure des technologies (avec un ordinateur quantique par exemple) étant donné la complexité doublement exponentielle. Néanmoins, des avancées restent encore possibles dans le domaines des approximations ou des sous-classes de DEC-POMDPs.

Horizon	DP	MAA*	JESP	MBDP	DP	Random	MBDP	IMBDP	MBDP-OC
	Tiger				MABC			CBP	
2	−4	−4	−4	−4	2	1	2	4.7	(NA)
3	5.19	5.19	5.19	5.19	2.99	1.49	2.99	5.3	(NA)
4	-	4.80	3.99	4.80	3.89	1.99	3.89	52.4	(NA)
5	-	-	2.93	5.38	-	2.47	4.79	56.77	72.3
6	-	-	2.39	9.91	-	2.96	5.69	59.89	(NA)
9	-	-	-	12.57	-	4.41	8.39	68.93	(NA)
10	-	-	-	13.49	-	4.9	9.29	71.33	103.9
20	-	-	-	(NA)	-	(NA)	(NA)	96	149.8
50	-	-	-	(NA)	-	(NA)	(NA)	80.8	278.7
100	-	-	-	93.24	-	48.39	90.29	72.8	503.8

TABLE 2.4 – Résultats de valeur espérée pour les différents problèmes et algorithmes. Les "-" représentent le fait que l'algorithme a dépassé le temps ou la mémoire allouée. Les "(NA)" représentent les données non disponibles dans la littérature.

2.3 Discussion

Nous avons présenté dans ce chapitre un aperçu général des modèles de Markov monoagents et multiagents complètement ou partiellement observables. Nous avons ensuite décrit des algorithmes optimaux pour les modèles généraux et des algorithmes non optimaux et leurs différentes évolutions jusqu'au plus récent. Nous n'avons pas présenté d'algorithmes pour les sous-classes particulières de DEC-POMDPs, mais le lecteur intéressé peut se référer à Goldman et Zilberstein [2004] pour ces différents algorithmes et leurs caractéristiques.

Pour finir, il convient de préciser que les DEC-POMDPs permettent de modéliser un grande partie des problèmes réels, mais souffrent notamment de plusieurs limitations :

Malédictions dimensionnelles : Les DEC-POMDPs, comme les POMDPs ou les MDPs, représentent l'état de manière extensive et raisonne souvent sur chacun des états du monde. Ce point, souvent critiqué, peut mener, dans le cas où l'état dépend d'un grand nombre de variables, à des espaces d'états exponentiellement grands et donc à une difficulté inhérente à la représentation. Différents modèles factorisés sont apparus pour pallier à ce problème, comme Boutilier *et al.* [2000]; Guestrin *et al.* [2001a]; Koller et Milch [2003]; Pralet *et al.* [2006b]; Oliehoek *et al.* [2008], pour ne citer qu'eux. Des problèmes similaires surviennent également lorsqu'on se retrouve avec un grand nombre d'actions ou un grand nombre d'observations.

Représentation temporelle : Dès lors que l'on s'intéresse à des problèmes reliés au temps, la représentation par des modèles de Markov devient extrêmement complexe. En effet, une des hypothèses stipule que les agents agissent de manière synchrone et que le temps est discrétisable, ce qui est souvent, mais pas nécessairement le cas. Relâcher cette hypothèse conduit à utiliser des modèles dits semi-markoviens qui incluent le temps et la durée dans le modèle [Goldman et Zilberstein, 2008].

Communication : Les modèles généraux incluant la communication demeurent aussi complexes que ceux ne l'incluant pas. Ce résultat contre-intuitif s'explique notamment par le fait que le modèle proposé par Goldman *et al.* [2004] ne fait aucune supposition sur le contenu informatif des messages. Il reste donc de nombreux travaux de modélisation des messages et de la communication à explorer avant de pouvoir effectivement utiliser ces modèles sur des applications réelles.

Interactions : On retrouve dans les travaux de Guestrin *et al.* [2001a,b], de nombreux exemples d'interactions plus simples que l'interaction complète des agents les uns envers les autres. Les modèles de Markov multiagents peinent encore à représenter ces interactions locales et à les exploiter comme le fait Oliehoek *et al.* [2008] pour affiner la recherche d'une politique jointe efficace.

En bref, les DEC-POMDPs offrent un framework extrêmement puissant permettant de modéliser beaucoup de problèmes actuels. Cependant, leur complexité inhérente les rend, en pratique, difficiles à utiliser. De nombreux travaux restent à mener pour déterminer les caractéristiques qui rendraient un DEC-POMDP à la fois simple et expressif.

2.4 Conclusion

Nous avons présenté dans ce chapitre les différents modèles sur lesquels se basent nos recherches et les notations que nous utiliserons dans les chapitres suivants.

Nous avons présenté dans un premier temps le modèle de décision de Markov pour un agent dans un environnement complètement observable puis celui où l'observabilité de l'état peut être altérée. Les principales méthodes de résolution dans ces modèles ont également été présentées dans le but de faciliter la compréhension des algorithmes des chapitres suivants.

Nous avons ensuite étendu les modèles aux cas où plusieurs agents interagissent dans un environnement complètement observable puis à celui dans lequel les agents ne perçoivent plus complètement l'état et doivent se coordonner à partir d'observations décentralisées. Nous avons finalement discuté les capacités et les limitations de ces

modèles.

Parmi les limitations présentées dans la discussion précédente, deux points ont plus précisé-ment attiré notre attention. Premièrement, concernant la malédiction de la dimensionalité, nous nous sommes intéressés dans le chapitre suivant au problème de l'augmentation du nombre d'actions possibles et à comment trouver des solutions lorsque ces actions sont contraintes et peuvent mener à des situations non désirables. Deuxièmement, nous nous sommes intéressé à un problème à la communication entre agents. Plus particulièrement, dès les agents communiquent et ont accès à une représentation bruitée de l'état complet du système, nous avons étudié dans le chapitre 4 la difficulté réelle de ce problème.

Voyons donc maintenant comment utiliser les contraintes issues de situations non désirables pour tenter d'améliorer la performance d'algorithmes dans un modèle décisionnel de Markov.

Chapitre 3

Contraintes et processus de Markov

«L'histoire de l'humanité est une statistique de la contrainte.»

Testament Phonographique -- Léo Ferré

Ce chapitre introduit tout d'abord le problème d'allocation de ressources à des tâches dont la réalisation peut être incertaine puis fait un survol de la littérature à ce sujet. Les bases de la satisfaction de contraintes sont ensuite introduites avant de détailler le modèle proposé qui combine processus décisionnels de Markov et satisfaction de contraintes en vue de résoudre le problème d'allocation de ressources. Des résultats théoriques pour ce modèle sont présentés ainsi que des résultats expérimentaux pour un algorithme spécialement adapté.

3.1 Introduction

Nous avons vu dans le chapitre précédent qu'un des problèmes majeurs des processus décisionnels de Markov réside dans l'explosion combinatoire du temps de calcul lors de l'augmentation exponentielle de la taille des données d'entrées tels l'espace des états ou encore l'espace des actions. Cette augmentation exponentielle de la taille des données d'entrées est d'autant plus fréquente que les problèmes à résoudre sont complexes et structurés.

Prenons par exemple un problème d'allocations séquentielles de n ressources à m tâches dont chaque réalisation peut être incertaine. S'il existe potentiellement C_m^n

allocations possibles des ressources aux tâches, à chaque étape de décision, il faut alors vérifier chacune de ces allocations possibles et estimer leur valeur espérée à long terme. Ce travail de calcul machine peut éventuellement être optimisé si l'information sur la structure de l'espace d'état est utilisée lors de la résolution du problème.

Dans ce contexte, nous proposons d'utiliser certaines formes de structuration de l'espace d'état basées sur les contraintes entre les états pour améliorer l'efficacité des algorithmes issus des MDPs. Cette structuration permet en effet d'induire une factorisation de la fonction de transition et de réduire le facteur de branchement dans la recherche de politiques d'allocations de ressources optimales. Le problème formel d'allocation dynamique et stochastique de ressources à des tâches est donc tout d'abord présenté ainsi que les solutions auparavant proposées de la littérature à ce sujet. Les bases formelles des problèmes de satisfaction de contraintes statiques et dynamiques sont ensuite exposées avant de présenter l'adaptation de ceux-ci au contexte markovien. L'application de ces méthodes à base de contraintes sur un problème non trivial de défense en milieu marin démontre l'efficacité de l'approche même si l'analyse théorique en pire cas pourrait laisser croire le contraire.

3.2 Problème statique d'allocation de ressources

Un des problèmes fondamentaux de la recherche opérationnelle est le problème d'assignation d'armes à des cibles (ou *Weapon-Target Assignment ou* WTA). Ce problème, étudié plus largement depuis la fin des années cinquante suite à sa formulation par Manne [1958], a été montré comme NP-complet par Lloyd et Witsenhausen [1986]. Il consiste à assigner de manière optimale des armes de sorte à minimiser l'espérance totale de survie des cibles à la fin des engagements.

Plus formellement, Lloyd et Witsenhausen [1986] ont défini le problème WTA de la manière suivante : Soit un ensemble $\mathcal{W}$ de w armes et un ensemble $\mathcal{M}$ de m cibles. Soit V_i la valeur de la cible i, $(1 \leqslant i \leqslant m)$ et α_{ij} la décision d'affecter l'arme j, $(1 \leqslant j \leqslant w)$ à la cible i. Si un tir de l'arme j sur la cible i est décidé alors il existe une probabilité p_{ij} que la cible soit détruite, indépendamment des autres tirs. Une solution au problème WTA statique est donc la matrice $\mathcal{A}$ des $m \times w$ variables de décision α_{ij} définies comme suit :

$$\alpha_{ij} = \begin{cases} 1 & \text{si l'arme } j \text{ est affecté à la cible } i \\ 0 & \text{sinon} \end{cases}$$

avec comme contrainte qu'une arme ne peut être assignée qu'a une seule cible :

$$\sum_{i=1}^{m} \alpha_{ij} = 1, \forall j \in \{1, \ldots, w\}$$

Ainsi, la probabilité que la cible i survive après la résolution de l'engagement est donnée par :

$$\prod_{j=1}^{w} (1 - \alpha_{ij} p_{ij})$$

Donc, au terme d'un engagement, la valeur espérée R peut être évaluée comme la somme des valeurs des cibles V_i détruites à la fin d'un engagement moins le coût C_{ij} induit par cet engagement et peut être traduit par :

$$R = \underbrace{\sum_{i=1}^{m} V_i \left[1 - \prod_{j=1}^{w} (1 - \alpha_{ij} p_{ij}) \right]}_{(a)} - \underbrace{\sum_{i=1}^{m} \sum_{j=1}^{w} C_{ij} \alpha_{ij}}_{(b)} \tag{3.1}$$

L'objectif étant de **maximiser** R.

Nous remarquerons ici une sorte de ''déséquilibre'' dans la fonction de valeur au niveau du coût (partie (b)) par rapport à la valeur espérée (partie (a)). Cela est partiellement dû au fait que le coût, dont la définition n'a pas été donnée, est intimement lié aux probabilités p_{ij}. Une définition plus complète sera donnée dans les sections suivantes car le coût est supposé nul dans le reste de cette section.

A noter également que dans le contexte de la formulation 3.1, il est également supposé que chaque arme ne possède qu'une seule munition ; toutefois, plusieurs armes de même type peuvent exister. Ceci permet, sans perte de généralité, de s'affranchir d'une non-linéarité supplémentaire qui apparaîtrait sous la forme d'une puissance dans le produit des probabilités :

$$R = \sum_{i=1}^{m} V_i \left[1 - \prod_{j=1}^{w} (1 - p_{ij})^{\alpha_{ij}} \right] - \sum_{i=1}^{m} \sum_{j=1}^{w} C_{ij} \alpha_{ij}$$

Voyons maintenant quelques cas particuliers pour lesquels un algorithme optimal a déjà été proposé.

3.2.0.6 Armes identiques

Dans le cas particulier ou toutes les armes sont supposées identiques, alors la probabilité de détruire une cible est la même quelque soit l'arme et $p_{ij} = p_i$. Cette

hypothèse peut être valide dans le cas où le défenseur ne possède qu'un seul type d'arme et qu'elles sont toutes localisées au même endroit si bien que les temps et probabilités d'interception sont identiques. Le problème peut alors être simplifié et réécrit de la manière suivante :

$$R = \sum_{i=1}^{m} V_i (1 - p_i)^{x_j}$$

avec

$$x_j = \sum_{j=1}^{w} \alpha_i$$

L'objectif étant cette fois-ci de **minimiser** R.

Un algorithme en temps en $\mathcal{O}(m + w \log w)$ a été proposé par den Broeder *et al.* [1959] pour résoudre ce problème de manière optimale généralement référencé sous le nom de *Maximum Marginal Return* (MMR) dans la littérature [Hosein, 1989]. Cet algorithme consiste à assigner les armes une par une à la cible de plus grande valeur. Dans le cas particulier des armes identiques, il est possible alors de montrer que cette stratégie gloutonne est optimale. MMR est donné par l'algorithme 8.

Algorithm 8 *Maximum Marginal Return* [den Broeder *et al.*, 1959]

Inputs : $\mathcal{M}$ l'ensemble des cibles et leurs valeurs
$\mathcal{W}$ l'ensemble des armes toutes identiques
pour tout $i \in \mathcal{M}$ **faire**
  Soit $x_i = 0$
  Soit $S_i = q_i^{x_i}$ la probabilité que la cible i soit détruite par x_i missiles
fin pour
$j \leftarrow 1$

tant que $j \leqslant w$ **faire**
  Trouver la cible i_{max} pour laquelle une munition a le plus d'impact sur la valeur espérée R :

$$i_{max} = \arg\max_i \{V_i S_i (1 - q_i)\}$$

  Assigner la munition j à la cible i_{max} :

$$x_{i_{max}} = x_{i_{max}} + 1 \quad \text{et} \quad S_{i_{max}} = S_{i_{max}} q_{i_{max}}$$

  $j \leftarrow j + 1$
fin tant que

A noter que si toutes les menaces sont aussi identiques, alors q_i est remplacé par q et les V_i deviennent égaux à 1. Lloyd et Witsenhausen [1986] ont montré que dans ce

cas encore plus particulier il suffit de répartir le plus uniformément possible les armes sur les cibles pour avoir une solution optimale.

L'algorithme MMR est extrêmement simple autant que rapide. Il s'initialise en temps $\mathcal{O}(|\mathcal{W}|)$ et après chaque itération, la mise à jour se fait en temps $\mathcal{O}(\log|\mathcal{W}|)$. Puisque le nombre d'itérations est de $|\mathcal{M}|$, la complexité en temps de l'algorithme MMR est en $\mathcal{O}(|\mathcal{W}| + |\mathcal{M}|\log|\mathcal{W}|)$. On peut se référer à Katter [1986] pour une implémentation de MMR et quelques résultats numériques.

Hosein [1989] a également développé un algorithme de recherche locale basé sur MMR qui trouve une solution optimale à partir de n'importe quelle allocation de départ. Ainsi, on remarquera que la simple hypothèse des armes identiques rend la fonction à optimiser convexe et donc triviale à explorer.

3.2.0.7 Autres cas particuliers

Deux autres cas particuliers ont été résolus de manière optimale en temps polynomial. Les cas ou le défenseur possède moins d'armes que l'assaillant, ou encore le cas ou l'on ne peut assigner au plus qu'une arme à chaque cible.

Ainsi, dans le cas où le nombre de munitions est inférieur ou égal au nombre de cibles, la solution optimale consiste à allouer les armes disponibles maximisant le produit valeur/efficacité le plus uniformément possible [Lloyd et Witsenhausen, 1986]. De la même manière, dans le cas où l'on ne peut assigner qu'au plus une arme à chaque cible, le problème est alors un cas tout aussi particulier et strictement équivalent au cas précédent où le nombre de munitions serait égal au nombre de cibles ($|\mathcal{W}| = |\mathcal{M}|$). La solution optimale est du coup identique et consiste à assigner les armes le plus uniformément possible [Lloyd et Witsenhausen, 1986].

Évidemment, puisque ces cas particuliers sont résolus de manière optimale, il sera supposé dans la suite de ce chapitre que, dans le cas général, les armes ont une efficacité différente, que le nombre d'armes est supérieur à celui des cibles et qu'il est possible de tirer avec plusieurs armes sur la même cible :

$$\forall j,\, k \in \{1,\ldots,m\},\, j \neq k \wedge p_{ij} \neq p_{ik} \quad \text{et que} \quad m > n.$$

3.2.1 Cas général statique

Les différentes approches proposées pour résoudre le cas général couvrent un large éventail des méthodes existantes utilisées en Intelligence Artificielle : programmation dynamique, programmation linéaire, processus décisionnels de Markov (partiellement observables ou non), etc. Les buts de ce bref état de l'art sont de montrer un large panel des techniques déjà utilisées et d'en extraire les avantages et les faiblesses. Ainsi, on verra que plusieurs formulations du problème existent s'attachant à résoudre des aspects plus spécifiques, mais en posant des hypothèses parfois contraignantes.

Depuis l'expression du cadre général par Manne [1958], Maltin [1970] a produit une revue de littérature couvrant la période 1958--1970 et dans laquelle de nombreuses références sont classées par le modèle qui est pris en considération. Une deuxième revue de la littérature de ces années là a également été proposée par Eckler et Burr [1972]. Maltin [1970] pour sa part a présenté le problème sous sa forme offensive, Eckler et Burr [1972] l'ont, quant à eux, présenté sous la forme défensive tout en donnant plusieurs modèles mathématiques ainsi qu'une analyse de ces modèles. Le lecteur est invité à se référer aux Tables I et II de [Maltin, 1970] pour savoir quelles méthodes ont été employées et sur quels modèles sur la période précédant les années 70.

3.2.2 Approches récentes employées

Parmi les approches plus récentes, de nombreuses techniques ont été employées. Parmi ces approches, il convient de retenir plus particulièrement les travaux de Ahuja *et al.* [2003]. Ces auteurs ont investigué les performances des algorithmes de type *Branch & Bound* et de recherche locale, mais également plusieurs autres approches pour calculer des bornes inférieures et les injecter dans le *Branch & Bound*. Une première borne est calculée en résolvant le problème de programmation linéaire en nombres entiers relaxé aux réels, une deuxième borne est calculée par programmation linéaire mixte, une troisième par la formulation du problème sous forme de minimisation de coût de réseaux de flux (*min cost flow*) et une dernière utilisant l'algorithme MMR en supposant qu'à chaque cible est assignée la meilleure arme possible. Toutes ces bornes sont ensuite utilisées dans un *Branch & Bound* classique et ont été comparées par Ahuja *et al.* [2003]. Cependant, même si cette approche est très efficace lorsque le nombre de cibles n'excède pas de beaucoup le nombre d'armes, elle devient inefficace lorsque le nombre d'armes approche du double du nombre de cibles. Les auteurs de l'approche ont alors proposé une méthode employant une recherche locale à très grand voisinage (VLSN pour *very large-scale neighborhood*) et une autre basée sur une recherche heuristique utilisant

la formulation *min cost flow*. Ces deux dernières approches présentent des résultats optimaux ou proches de l'optimal avec des temps de calcul très prometteurs sur des instances de grande taille. Les auteurs suggèrent également d'utiliser ces approches comme sous-routine pour la formulation du problème dynamique développé au chapitre suivant.

D'autres travaux ont bien sur été réalisés employant des techniques telles que les algorithmes génétiques [Lee *et al.*, 2003], les réseaux de neurones [Wacholder, 1989] ou encore certaines approches considérant des environnements multiplateformes [Rosenberger *et al.*, 2005]. Cependant, elles restent moins efficaces sur le papier que celle proposée par Ahuja *et al.* [2003].

De plus, toutes ces approches se limitent strictement à la formulation statique du problème. En effet, la version dynamique du problème d'assignation d'armes à des cibles a été formulée pour la première fois par Hosein *et al.* [1988]. Elle consiste à résoudre la version statique du problème sur plusieurs étapes de temps en considérant le résultat de l'étape précédente d'assignation pour l'étape courante. Évidemment, certains des algorithmes présentés ci-avant pourront être utilisés comme sous-routines pour la formulation dynamique comme suggéré par Ahuja *et al.* [2003], mais nécessiteront une adaptation pour pouvoir prendre en compte les informations supplémentaires inhérentes à ce problème plus général.

3.3 Problème dynamique d'allocation de ressources

Dans la version statique du problème d'allocation d'armes, l'allocation est effectuée à un instant donné. Maintenant, supposons que les allocations peuvent être décidées à n'importe quel instant t parmi T intervalles discrets de temps. Selon Murphy [2000], il existe deux formulations fondamentales du problème dynamique d'allocation d'armes à des cibles. Le premier, et le plus étudié, est le cas ou toutes les cibles sont connues a priori (leur nombre et leurs positions). Dans ce cas-ci, puisque le résultat de l'engagement de chaque étape d'assignation est stochastique, la survie d'une cible est dépendante du nombre d'armes assignées dessus et va influencer les allocations des étapes suivantes. Ce type de problème est connu sous le nom de *shoot-look-shoot* puisqu'une observation (*look*) de la première allocation est nécessaire avant de faire une nouvelle allocation.

Le deuxième cas, un peu moins étudié dans ce contexte, mais largement étudié dans le problème de routage de transport (*Vehicle Routing* e.g. [Hentenryck *et al.*, 2005]), est le cas ou les cibles ne sont pas observables (dans le sens ou leur nombre et leurs

positions ne sont pas connues), mais ou le résultat de l'engagement est essentiellement déterministe. En d'autres mots, à chaque étape, un sous-ensemble de cibles est connu avec certitude alors que le reste des cibles est soit inconnu soit partiellement connu. La difficulté résidant dans la capacité de prédiction de l'apparition aléatoire de nouvelles cibles (problème de *Stochastic Demand*) au cours de l'engagement.

Bien entendu, l'idée directrice de ce chapitre est de considérer les deux cas pris ensemble. Ce chapitre va donc commencer par présenter le problème du *shoot-look-shoot* puis celui de *stochastic demand* avant de présenter une formulation générale du problème dynamique.

3.3.1 Shoot-Look-Shoot

Dans les problèmes de type *shoot-look-shoot*, l'assignation des armes aux cibles est effectuée sur un certain nombre d'intervalles discrets de temps. De plus, la supposition est faite qu'après chaque intervalle, une observation parfaite du résultat de l'engagement en résulte. Hosein et Athans [1990] ont formulé le problème de la manière suivante :

L'état de la cible i (μ_i) est une fonction du temps et de l'assignation à l'étape précédente. À chaque étape t, l'état est simplement :

$$\mu_i^t = \begin{cases} 1 & \text{si } i \text{ a survécu à l'étape } t-1 \\ 0 & \text{sinon} \end{cases}$$

Ainsi, puisque la survie de la cible i dépend de l'assignation α_{ij}^{t-1} et de p_{ij}^{t-1}, l'état de la cible évolue de manière stochastique de la manière suivante :

$$P(\mu_i^t = k) = k \underbrace{\prod_{j=1}^{w}(1-p_{ij}^{t-1})^{\alpha_{ij}^{t-1}}}_{(a)} + (1-k) \underbrace{\left(1 - \prod_{j=1}^{w}(1-p_{ij}^{t-1})^{\alpha_{ij}^{t-1}}\right)}_{(b)} \tag{3.2}$$

En d'autres termes, si $k = 1$, la probabilité que l'état μ_i de la cible i à l'étape t est donnée par la partie (a) de l'équation 3.2 qui indique la probabilité que la cible aie survécu à l'engagement. Dans le cas inverse ($k = 0$) la partie (b) de l'équation 3.2 est appliquée.

De la même manière, l'état de l'arme j est défini par :

$$\omega_j^t = \begin{cases} 1 & \text{si } j \text{ n'a pas été utilisée à l'étape } t-1 \\ 0 & \text{sinon} \end{cases}$$

et évolue suivant :

$$\omega_j^t = 1 - \sum_{i=1}^{m} \alpha_{ij}^{t-1}$$

qui décris que l'arme est disponible à l'étape t si elle n'a pas été utilisée à l'étape $t-1$ ainsi qu'à toutes les étapes précédentes.

Hosein et Athans [1990] ont ensuite défini le coût $F_1(\boldsymbol{\mu}, \boldsymbol{\omega})$ de l'étape 1, le coût à l'étape 2 étant $F_2(\boldsymbol{\mu}, \boldsymbol{\omega})$, et ainsi de suite pour chaque étape de temps t. Le problème est enfin formulé comme le programme linéaire en nombre entier suivant :

$$\begin{aligned}
\min F_1 &= \min_{\alpha_{ij}} \sum_{\mathbf{k} \in \{0,1\}^m} P[\boldsymbol{\mu} = \mathbf{k}] F_2^*(\mathbf{k}, \boldsymbol{\omega}) \\
\text{sous contraintes} &: \alpha_{ij} \in \{0, 1\} \\
i &= 1, 2, \ldots, m \\
j &= 1, 2, \ldots, w \\
\omega_j &= 1 - \sum_{i=1}^{m} \alpha_{ij}.
\end{aligned}$$

Où la solution optimale de l'étape 2, notée $F_2^*(\mathbf{k}, \boldsymbol{\omega})$, est obtenue par la résolution de l'étape 3 définie par $F_3^*(\mathbf{k}, \boldsymbol{\omega})$, et ainsi de suite pour chaque étape de temps t. Le coût optimal de l'étape finale $T+1$ étant la somme des valeurs des cibles ayant survécu à la fin :

$$F_{T+1}^*(\boldsymbol{\mu}, \boldsymbol{\omega}) = \sum_{i=1}^{m} V_i \mu_i$$

Cette formulation est clairement adaptable à la programmation dynamique. Malheureusement, même pour $T = 2$, le nombre de calculs requis par la programmation dynamique est incroyablement grand. Il existe en effet 2^w sous-ensembles d'armes qui peuvent être choisis à l'étape 1. Et si w_1 armes sont utilisées à l'étape 1, le nombre d'allocations possibles à vérifier est m^{w_1}. À cela s'ajoute en plus l'étape 2, où si $\overline{m}$ cibles ont survécu à l'engagement précédent et qu'il reste w_2 armes disponibles, alors il faudra vérifier les $\overline{m}^{w_2}$ assignations possibles.

Néanmoins, Hosein [1989] a trouvé quelques éléments de solution pour un cas plus simple où les probabilités p_{ij} ne dépendent plus des armes et des cibles, mais seulement de l'étape à laquelle est faite l'allocation. Avec cette hypothèse, la décision à chaque étape est alors juste de la forme w^t, le nombre d'armes à assigner à l'étape t. L'auteur a ainsi montré que dans ce cas précis il existe une stratégie optimale :

Proposition 1. *[Hosein, 1989] Pour $t = 1, 2, \ldots, T$, les w^t armes utilisées à l'étape t doivent être réparties équitablement sur l'ensemble des cibles.*

Évidemment, il reste à déterminer les valeurs optimales de w^t pour tout t. Hosein [1989] présente 3 cas particuliers :

Cas 1. Si $w \leqslant m$, i.e. que le nombre d'armes est inférieur au nombre de menaces, alors il faut assigner toutes les armes dans l'étape où elles ont la plus grande efficacité.

Cas 2. Si $w \geqslant m$ et $p^t > p^{t+1}$ avec $t = 1, 2, \ldots, T-1$, i.e. lorsque le nombre d'armes est plus grand que le nombre de menaces, mais que leur efficacité diminue avec le temps, alors la solution optimale est d'allouer toutes les armes le plus vite possible.

Cas 3. Si $T > 1 + \frac{w-m}{2}$ pour $w > m$ et $p^t = p$, $\forall t$, i.e. lorsque l'horizon de planification est suffisamment grand et que les armes sont identiques, alors la première étape optimale est d'allouer exactement autant d'armes qu'il y'a de cibles pour ne pas gaspiller de munitions.

Athans et Hosein [1990] ont également développé une limite sur F_1 lorsque le nombre de cible tends vers l'infini.

3.3.2 Apparition stochastique de cibles

Considérons maintenant un problème d'allocation où, à un certain instant t, seulement une partie de l'ensemble des cibles est connu. Si on note le nombre de cibles connues à l'instant t, m^t alors on a $m^t \leqslant |\mathcal{M}|$, $\forall t$. À mesure que le temps avance, certaines cibles sont découvertes, ainsi m^t ne décroît pas au cours du temps. Mais puisque $|\mathcal{M}| = m^T$ est inconnu, l'allocation des armes ne peut être effectuée que sur les m^t cibles connues ou réservée pour les cibles encore inconnues. Ce type de problème dynamique d'assignation d'armes à des cibles à apparition stochastique à été introduit en premier par Murphey [1999], mais se rapproche également des problèmes de demandes stochastiques comme on peut le trouver dans la littérature sous la forme de routage de véhicules par exemple [Hentenryck *et al.*, 2005]. La formulation pour le WTA a toutefois été donnée par Murphey [1999] : Considérant que les cibles sont ordonnées suivant leur valeur, alors il est possible qu'au temps t, seulement les cibles ayant une valeur basse aient été découvertes. Dans ce cas, il est préférable d'attendre avant d'effectuer une allocation. Si, après avoir attendu un temps τ, toutes les cibles sont connues (i.e. $m^\tau = m^T$), il en résulte un problème statique WTA. Supposons néanmoins qu'un coût C^t est associé à l'action d'attendre. Ce coût peut être justifié par le fait que plus le temps passe et plus la cible s'approche de nous dans un cas d'auto-défense par exemple.

Appelons D_i la limite de temps à partir de laquelle on ne peut plus engager une cible i. La résolution du problème d'apparition stochastique de cibles consiste donc à trouver l'ensemble des vecteurs d'assignation $\boldsymbol{\alpha}_i^t$ qui maximisent les valeurs des cibles

détruites sur l'ensemble des instants $t \in [0, D_i]$, $D_i \leqslant T$ en considérant une fonction de coût d'attente qui augmente de manière monotone avec le temps, et qui comme dans le cas statique du WTA est pondéré par les valeurs V_i des cibles. Dans sa formulation, Murphey [1999] considère que les valeurs V_i des menaces restent constantes au cours du temps.

Enfin, Murphey [1999] formule le problème de la manière suivante :

$$\begin{aligned}
\text{minimiser} \quad & \sum_{t=1}^{T} C^t \left[\sum_{i=1}^{m^t} V_i \left(\prod_{j=1}^{w} (1 - p_{ij})^{\alpha_{ij}^t} \right) \right] \\
\text{sous contraintes} \quad & : \\
\sum_{t=1}^{T} \sum_{i=1}^{m^t} \alpha_{ij}^t = 1 \quad & i = 1, 2, \ldots, m^T \\
V_i \in \mathbb{R}_+ \quad & j = 1, 2, \ldots, w \\
\alpha^t \in \{0,1\}^{m^t} \quad & t = 1, 2, \ldots, T.
\end{aligned}$$

3.3.3 Généralisation du problème

Nous avons vu dans les sections précédentes que le problème dynamique relatif au WTA pouvait se décliner sous plusieurs formulations différentes. Le premier travail consiste donc à unifier les deux modèles présentés ci-avant pour présenter un modèle dynamique où les observations sont permises et incertaines et où la totalité des cibles n'est pas connue initialement.

Formellement, le problème se définit à la manière de la version statique du chapitre précédent sur plusieurs étapes temporelles : Soit un ensemble $\mathcal{W}$ de w armes et un ensemble $\mathcal{M}$ de m cibles. Soit V_i^t la valeur de la cible i, $(1 \leqslant i \leqslant m)$ à l'étape $t, t \in \{1, \ldots, T\}$ et α_{ij}^t la décision d'affecter l'arme j, $(1 \leqslant j \leqslant w)$ à la cible i à l'étape t. Si un tir de l'arme j sur la cible i est décidé alors il existe une probabilité p_{ij}^t que la cible soit détruite, indépendamment des autres tirs à la même étape. Une solution au problème WTA dynamique est donc l'ensemble $\mathcal{A}^t$ des $m \times w \times T$ variables de décision α_{ij}^t définies de la même manière que dans la version statique. Dans ce cadre, nous pouvons évaluer la valeur espérée d'une solution au cours du temps, soit :

$$R = \sum_t R^t = \sum_{t=1}^{T} \sum_{i=1}^{m} \left[V_i^t \left(1 - \prod_{j=1}^{w} (1 - \alpha_{ij}^t p_{ij}^t) \right) - \sum_{j=1}^{w} C_{ij}^t \alpha_{ij}^t \right] \tag{3.3}$$

Le but étant de **maximiser** une telle valeur espérée R.

Il convient cependant de préciser que le coût C_{ij} tel qu'inclut dans l'équation 3.3 peut inclure de nombreuses données comme la date limite avant laquelle une cible doit être engagée (équation 3.4), les pertes d'efficacité des armes au cours du temps, les interactions positives/négatives entre les armes si l'assignation est effectuée au même instant (équation 3.5), etc. Par exemple,

$$C_{ij}^{t} = C_{i,t_0}^{deadline} + C_{ij}^{interactions} + \cdots$$

avec

$$C_{i,D_i}^{deadline} = \begin{cases} V_i^t & \text{si } t \geqslant D_i \text{ avec } D_i \text{ la limite} \\ 0 & \text{sinon} \end{cases} \tag{3.4}$$

$$C_{ij}^{interactions} = V_i^t \sum_{k=1}^{w} \alpha_{ik}^t p_{ijk} \tag{3.5}$$

Où p_{ijk} est l'efficacité gagnée (resp. perdue) par l'interaction positive (resp. négative) entre l'arme j et l'arme k.

Pour les mêmes raisons que la version statique, la supposition est faite que chaque arme ne possède qu'une seule munition, mais qu'il est possible de posséder plusieurs armes de même type. Cela permet de s'affranchir à nouveau d'une non-linéarité qui apparaîtrait sous la forme d'une puissance dans le produit des probabilités soit $\prod_{j=1}^{w}(1-p_{ij}^t)^{\alpha_{ij}^t}$ en lieu et place de $\prod_{j=1}^{w}(1-\alpha_{ij}^t p_{ij}^t)$. Dès lors, R se réduirait à :

$$R = \sum_t R^t = \sum_{t=1}^{T}\sum_{i=1}^{m}\left[V_i^t\left(1-\prod_{j=1}^{w}(1-p_{ij}^t)^{\alpha_{ij}^t}\right) - \sum_{j=1}^{w} C_{ij}^t \alpha_{ij}^t\right]$$

Malheureusement, cela induit une perte en modélisation : alors que dans la modélisation statique nous n'avions qu'un seul engagement à faire, dans la version dynamique il est possible qu'une arme ne puisse engager deux cibles à la fois et dès lors le plan est ralongé d'autant. Ce problème est pris en compte par notre modèles présenté à la section 3.5.

On a vu dans la littérature sur le problème WTA que la version statique avait été largement étudiée et que de nombreuses solutions efficaces quoique sous-optimales avaient été trouvées. Néanmoins, la littérature sur le problème dynamique est significativement moins fournie en partie dû au fait de sa complexité [Hosein *et al.*, 1988]. Toutefois, certaines approches approximatives existent pour le cas général, notamment celle de Yost et Washburn [2000] basée sur la composition de la programmation linéaire et des Processus de Markov Partiellement Observables (POMDPs) pour maintenir l'état de croyance sur la survie des cibles. Castañon et ses collègues ont également proposé plusieurs approches pour le cas dynamique en s'intéressant principalement à la maintenance des contraintes de ressources [Castanon et Wohletz, 2002; Castanon et Wu, 2004].

Plus récemment, Glazebrook et Washburn [2004] ont proposé une revue de la littérature du *shoot-look-shoot* dans laquelle ils identifient plusieurs cas selon les contraintes sur les armes, l'information disponible pour la validation de la destruction d'une menace et de l'horizon de planification disponible. Par exemple, le cas contraint [1] avec information complète [2] et à horizon infini [3] est un cas de programmation dynamique typique qui devient insurmontable dès que les nombres de cibles et d'armes dépassent la demi-douzaine. Le cas contraint[1] avec information complète[2] mais à horizon fini est également présenté sous l'hypothèse que les cibles et les armes sont toutes identiques. Ces auteurs présentent également les travaux de Manor et Kress [1997] portant sur le cas ou l'information perçue est incomplète ou incertaine. Dans ce cas précis, Manor et Kress [1997] ont montré qu'une stratégie gloutonne (*greedy*) suivant l'efficacité des armes sur certaines cibles était optimale. Glazebrook et Washburn [2004] ont pour leur part, fait état du cas non contraint où la limite de ressources est remplacée par un coût à chaque utilisation d'armes avec un horizon infini. L'objectif étant de minimiser ce coût tout en maximisant la probabilité de destruction des cibles. Ces auteurs utilisent ici une représentation markovienne partiellement observable (POMDP) de l'environnement combiné à de la programmation dynamique stochastique. Dans le cas de l'horizon fini et où le nombre de types de cible est réduit, ils suggèrent d'utiliser l'approche fournie par Yost et Washburn [2000] qui permet de résoudre le problème donné en séparant chaque type de cible.

Dans la suite de ce chapitre, le problème dynamique d'allocation d'armes à des cibles est considéré, en supposant un cas contraint et une information complète à horizon fini, mais où d'autres types de contraintes peuvent s'ajouter sur les armes que le nombre limité de munitions. Ces contraintes peuvent par exemple impliquer d'utiliser un radar pour le guidage de missile ou des limites de portée pour les armes. Pour pouvoir prendre en compte ce type de contraintes non linéaires, un formalisme de satisfaction de contraintes plus général que la programmation linéaire est présenté à la section suivante.

3.4 Problèmes de satisfaction de contraintes (CSP)

Un CSP (Constraint Satisfaction Problem) est défini par un ensemble de variables, chacune pouvant prendre une valeur dans un domaine associé à la variable. Des contraintes portant sur des sous-ensembles de variables restreignent les choix possibles. Chaque contrainte est définie par une relation précisant quelles combinaisons de valeurs sont

1. Où il ne faut pas dépasser un nombre maximum d'armes.
2. Ou la validation de la destruction de la cible est immédiate.
3. Où les tirs peuvent être faits un par un.

autorisées. Les variables sur lesquelles porte une contrainte sont dites liées (par cette contrainte). Plus formellement,

Définition 13. *(CSP) Un Problème de Satisfaction de Contraintes est défini par le triplet $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, où :*

- $\mathcal{X} = \{\mathrm{X}_1, \mathrm{X}_2, \ldots, \mathrm{X}_i, \ldots, \mathrm{X}_n\}$ *est l'ensemble des n variables du problème ;*
- $\mathcal{D} = \{\mathcal{D}_{\mathrm{X}_i} | \mathrm{X}_i \in \mathcal{X}\}$ *est l'ensemble des n domaines finis associés aux variables (i.e., leurs valeurs possibles) ;*
- $\mathcal{C} = \{c_1, \ldots, c_j, \ldots, c_m\}$ *est l'ensemble des m contraintes, spécifiant les combinaisons de valeurs mutuellement compatibles, avec $c_j = \langle \chi_{c_j}, \rho_{c_j} \rangle$:*
  - $\chi_{c_j} = \{\mathrm{X}_{j_1}, \ldots, \mathrm{X}_{j_k}\} \subseteq \mathcal{X}$ *est l'ensemble des k variables liées par c_j ; on appelle χ_{c_j} le* cadre *de c_j et k est appelé arité de c_j ;*
  - $\rho_{c_j} \subseteq \mathcal{D}_{\mathrm{X}_{j_1}} \times \cdots \times \mathcal{D}_{\mathrm{X}_{j_k}}$ *est la* relation *associée à la contrainte c_j ; elle représente les n-uplets autorisés par cette contrainte.*

Nous noterons $\mathbb{S}(P) \subset \bigtimes_{i=1}^{n} \mathcal{D}_{\mathrm{X}_i}$ l'ensemble des assignations des variables qui satisfont l'ensemble des contraintes $\mathcal{C}$ de P. Résoudre un CSP équivaut à assigner une valeur à chaque variable en respectant toutes les contraintes du problème et donc à trouver un élément $\sigma \in \mathbb{S}(P)$. En trouver un est un problème NP-complet [Haralick *et al.*, 1978].

À noter ici que cette représentation à base de variables rappelle le principe de la factorisation et pourrait donc probablement permettre d'identifier les indépendances entre les variables (et éventuellement les agents dans un cadre multiagent) la où il n'y a pas de contraintes.

3.4.1 CSP dynamique (DCSP)

Lorsque les CSPs se suivent et se ressemblent, Verfaillie et Jussien [2005] définissent un CSP dynamique (DCSP) Ψ comme une suite finie de CSP ''statiques'' $P_0, \ldots, P_i, P_{i+1}, \ldots, P_\eta$, chacun résultant du précédent par l'ajout ou le retrait d'un ensemble de contraintes (ces opérations sont respectivement appelées restriction et relaxation) et/ou d'un ensemble variable. Par conséquent si on note $P_i = \langle \mathcal{X}_i, \mathcal{D}_i, \mathcal{C}_i \rangle$, on a :

$$\begin{aligned} P_{i+1} &= \langle \mathcal{X}_{i+1}, \mathcal{D}_{i+1}, \mathcal{C}_{i+1} \rangle \\ &= \langle \mathcal{X}_i \cup \{\mathrm{X}_{n+1}, \ldots, \mathrm{X}_{n+p}\}, \mathcal{D}_i \cup \{\mathcal{D}_{\mathrm{X}_{n+1}}, \ldots, \mathcal{D}_{\mathrm{X}_{n+p}}\}, \mathcal{C}_i \cup \{c_{m+1}, \ldots, c_{m+q}\} \rangle \end{aligned} \tag{3.6}$$

Cependant, il n'existe ici aucun modèle du futur permettant de garantir la robustesse de la solution au cours du temps même si ce modèle permet de s'intéresser à la dynamique des CSP.

Résoudre un DCSP Ψ consiste à trouver une solution $\sigma_i \in \mathbb{S}(P_i)$ pour chaque i, $1 \leqslant i \leqslant \eta$. La plupart des recherches effectuées sur ce modèle ont consisté à trouver une solution σ_{i+1} basée sur σ_i qui minimise le nombre de changements. Par exemple, la réutilisation de solution ou de raisonnement [Verfaillie et Jussien, 2005] sont deux approches qui bénéficient des solutions calculées aux étapes précédentes pour calculer celle aux étapes futures.

Malheureusement, les CSPs et DCSPs supposent un monde déterministe ou chacune des variables est contrôlable et prend une valeur parmi l'ensemble de son domaine. Or, comme nous l'avons vu précédemment, la notion de variable incontrôlable et/ou partiellement observable peut être nécessaire dans la représentation de problèmes.

3.4.2 CSP stochastique (SCSP)

Puisque les CSPs et DCSPs sont déterministes, mais qu'il est souvent important de pouvoir modéliser des notions d'incertitudes sur certaines variables, Walsh [2002] a défini les CSPs stochastiques (SCSP) qui intègrent des variables aléatoires dans l'ensemble des variables. Walsh définit pour cela une distribution de probabilités sur le domaine de chacune de ces variables. Plus formellement :

Définition 14. *(SCSP) Un Problème de Satisfaction de Contraintes Stochastique est défini par le tuple* $S = \langle \mathcal{X}, \mathcal{S}, \mathcal{T}, \mathcal{D}, \mathcal{C}, \theta \rangle$*, où :*

- $\mathcal{X} = \{\mathrm{X}_1, \mathrm{X}_2, \ldots, \mathrm{X}_i, \ldots, \mathrm{X}_n\}$ *est l'ensemble des* n *variables du problème ;*
- $\mathcal{S} \subset \mathcal{X}$ *est l'ensemble des variables stochastiques du problème ;*
- $\mathcal{D} = \{\mathcal{D}_{\mathrm{X}_i} | \mathrm{X}_i \in \mathcal{X}\}$ *est l'ensemble des* n *domaines finis associés aux variables (i.e., leurs valeurs possibles) ;*
- $\mathcal{T} : \mathcal{S} \mapsto \Delta_{\mathrm{X} \in \mathcal{S}}(\mathcal{D}_{\mathrm{X}})$ *est un fonction qui associe à chaque variable une distribution de probabilité sur leur domaine ;*
- $\mathcal{C} = \{c_1, \ldots, c_j, \ldots, c_m\}$ *est l'ensemble des* m *contraintes, spécifiant les combinaisons de valeurs mutuellement compatibles.*

Résoudre un SCSP consiste à trouver une solution $\sigma \in \mathbb{S}(S)$ telle que la probabilité que le SCSP soit satisfait sachant l'assignation σ et les distributions $\mathcal{T}$, soit supérieure à θ. Il est à noter qu'un SCSP peut être répété plusieurs fois, et que dès lors une notion de résolution en séquence est nécessaire. Dans ce contexte, il a été montré par Walsh [2002] que trouver une solution à un SCSP à plusieurs étapes est un problème PSPACE-complet.

Plus récemment, Pralet *et al.* [2006b] ont proposé un cadre algébrique appelé PFU (pour Plausibilité, Faisabilité, Utilité) qui généralise tous les modèles ci-dessus ainsi

que les réseaux bayésiens et les diagrammes d'influence. Ils ont également proposé un algorithme à base d'élimination de variable [Dechter, 2003, chap. 13.3.3] pour résoudre n'importe quel type de problème modélisable à l'aide de ce cadre algébrique. Dans la section ci-après, nous verrons comment nous l'avons adapté pour mieux coller aux besoins réels de notre domaine d'application en termes de modélisation.

3.5 Problèmes de satisfaction de contraintes markoviens

C'est donc en ayant en tête l'idée de représenter un MDP factorisé contraint que nous avons proposé ce que nous avons appelé un CSP markovien :

Définition 15. (MARKOVIAN CSP) *un* CSP *markovien (*MaCSP*s) est défini par le tuple* $\Phi = \langle \mathcal{S}, \mathcal{A}_i, \Psi, \mathcal{T}, \mathcal{R} \rangle$ *où :*

- $\mathcal{S} = \{s_i\}$ *où* $s_i = \langle \mathrm{X}_1, ..., \mathrm{X}_m \rangle$ *est un état factorisé avec* $\mathrm{X}_k \in \mathcal{X}_{s_i}$ *;*
- $\mathcal{A} = \{a_i\}$ *où* $a_i = \langle \mathrm{A}_1, ..., \mathrm{A}_n \rangle$ *est une action factorisée avec* $\mathrm{A}_k \in \mathcal{X}_{s_i}$*. On dénote par* $\mathbb{S}_i = \{\sigma_{s_i} = \{\mathrm{A}_1, ..., \mathrm{A}_\varsigma\} : \mathrm{A}_k, \mathrm{A} \in \mathcal{D}_{s_i}, \sigma_{s_i} \in \mathbb{S}(P_{s_i})\}$ *l'ensemble des assignations des variables de décisions consistantes avec* $\mathcal{C}_{s_i}$ *;*
- $\Psi = \{P_{s_i}\}$ *est un* DCSP *avec* $P_{s_i} = \langle \mathcal{X}_{s_i}, \mathcal{D}_{s_i}, \mathcal{C}_{s_i} \rangle$ *basé sur les variables de l'état* s_i *et sur les variables de décisions* σ_{s_i} *;*
- $\mathcal{T}^{\sigma}_{s_i s_{i+1}} = Pr(s_{i+1}|s_i, \sigma)$ *est la probabilité que l'assignation* σ *provoque la transition de l'état* s_i *vers l'état* s_{i+1} *;*
- $\mathcal{R}^{\sigma}_{s}$ *est une fonction de récompense pour chaque assignation* σ *satisfaisante dans* s*.*

En d'autres mots, $\mathcal{S}$ modélise un espace d'état factorisé $\langle \mathrm{X}_1, ..., \mathrm{X}_m \rangle$ où des contraintes $\mathcal{C}_{s_i}$ peuvent exister entre les variables X et les variables A de $\mathcal{A}$ dans chaque état s_i. Les variables sont partitionnées en deux types : les variables de décision A de $\mathcal{A}$ (par exemple l'assignation d'une tâche à une ressource) et les variables d'état X de $\mathcal{S}$ (par exemple l'état d'une tâche). $\mathbb{S}_i$ est le sous-ensemble de toutes les combinaisons d'assignations possibles des variables de décision qui est consistent avec $\mathcal{C}_{s_i}$. La fonction de transition $\mathcal{T}^{\sigma}_{s_i s_{i+1}}$ représente l'évolution de toutes les variables d'état du système selon l'assignation des variables de décision. Incidemment, si certaines contraintes de $\mathcal{C}_{s_i}$ dépendent de variables d'états, alors, ces contraintes peuvent évoluer au cours du temps (changer, apparaître, ou disparaître). La fonction de récompense $\mathcal{R}^{\sigma}_{s}$ qui attribue une valeur à chaque assignation de variable de décision, peut être vue comme un ordre de préférence sur les tâches ou comme la récompense d'avoir terminé certaines tâches. Par conséquent,

un DCSP est un cas particulier de MaCSP où le modèle de transition $\mathcal{T}^{\sigma}_{s_i s_{i+1}}$ est déterministe et possiblement inconnu et où il n'y a que des variables de décision.

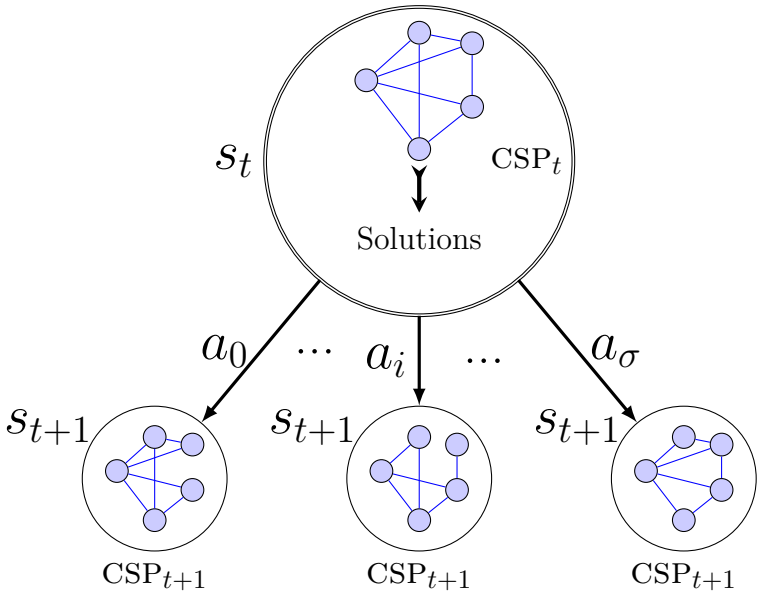


FIGURE 3.1 – CSP dynamique Markovien.

La figure 3.1 illustre comment se fait le calcul des transitions d'un état s_t à un état s_{t+1}. L'ensemble $\{a_0, ..., a_i, a_\sigma\}$ représente les solutions du CSP$_t$ qui sont les actions consistantes dans l'état s_t. Ces actions n'étant pas déterministes, elles peuvent conduire à des états s_{t+1} différents. Voyons maintenant un exemple illustratif de MaCSPs.

3.5.1 Exemple illustratif

Considérons un problème d'allocation de ressources non fiables renouvelables, 3 robots jardiniers par exemple $(R_i, i = 1, 2, 3)$, et un ensemble de tâches à réaliser (par exemple planter un cerisier (T_{pc}), tondre la pelouse (T_{tp}), éliminer les parasites (T_{ep}) et planter des tomates (T_{pt})). Ces robots étant chacun spécifique, leur capacité pour réaliser chaque tâche n'est pas la même. On représentera cette capacité sous la forme d'une probabilité de réaliser ou non cette tâche.

Si on ajoute à cela le fait que deux robots risquent d'avoir des problèmes s'ils s'attellent à la même tâche au même moment, il est possible de représenter le problème sous forme de MDP classique et sous forme de MaCSPs.

Formellement le MDP pourrait être représenté comme suit :$\langle \mathcal{S}, \{\mathcal{A}\}, \mathcal{P}, \mathcal{R}, s_0 \rangle$

	T_{pc}	T_{tp}	T_{ep}	T_{pt}
R_1	0.9	0.2	0.5	0.7
R_2	0.1	0.9	0.7	0.1
R_3	0.3	0.5	0.5	0.6

TABLE 3.1 – Probabilités p_{ij} que le Robot R_j réalise la tâche T_i

- $\mathcal{S} = 2^{\{T_i\}}$ Le produit cartésien de l'état des tâches (réalisées ou non) ;
- $\mathcal{A} = \{\mathcal{A}_i\}$ Le produit cartésien des actions de chaque robot ;
- $\mathcal{P}^a_{ss'} = \mathcal{P}(s, a, s') = \prod_{i,j}(1 - p_{ij}\alpha_{ij})$ suivant la table 3.1 ;
- $\mathcal{R}^a_s = \mathcal{R}(s, a) = \sum$ Valeurs des tâches réalisées dans l'état s ;
- s_0 État initial (toutes les tâches sont non réalisées).

Quand aux MACSPs, ils pourraient être représentés par :$\langle \mathcal{S}, X, D, C, \{\mathcal{A}\}, \mathcal{P}, \mathcal{R}, s_0 \rangle$

- $\mathcal{S} = 2^{\{T_i\}}$ Le produit cartésien de l'état des tâches (réalisées ou non) ;
- $X = \{R_j\}$ L'ensemble des ressources à assigner ;
- $D = \{D_i\} = \{T_i\}$ L'ensemble des tâches à réaliser sont les actions possibles de chaque robot ;
- $C = \{\forall R_i, R_j \in X, R_i \neq R_j\}$ Deux robots ne doivent pas être assignés à la même tâche ;
- $\mathcal{A} = \{X^D/_C\}$ Le produit cartésien des actions de chaque robot *consistantes avec* C ;
- $\mathcal{P}^a_{ss'} = \mathcal{P}(s, a, s') = \prod_{i,j}(1 - p_{ij}\alpha_{ij})$ suivant la table 3.1 ;
- $\mathcal{R}^a_s = \mathcal{R}(s, a) = \sum$ Valeurs des tâches réalisées dans l'état s ;
- s_0 État initial (toutes les tâches sont non réalisées).

Une première estimation permet d'évaluer le nombre d'états à $2^4 = 16$ et le nombre d'actions à $|T|^{|X|} = 4^3 = 64$ dans le MDP classique. Comme on peut le constater, le nombre d'actions est exponentiel par rapport au nombre de tâches. Ainsi, en ajoutant le filtrage des contraintes sur l'espace des actions, on passe de 64 actions possibles à seulement tous les arrangements possibles des ressources aux tâches : $A^{|X|}_{|T|} = 4! = 24$.

Plus formellement, le gain dans le cas d'un problème de ressources où les assignations doivent être toutes différentes :

Proposition 2. *Dans un problème d'allocation de ressources à des tâches, le gain en facteur de branchement de l'utilisation de la contrainte* allDiff *qui impose que toutes les ressources soient appliquées à des tâches différentes est égal à :*

$$Gain = \frac{|T|^{|X|}}{C^{|X|}_{|T|}}$$

Quand les nombres de tâches et de ressources tendent vers l'infini ce gain est alors de :

$$Gain_{\infty} = \lim_{|T|\to\infty,|X|\to|T|} \frac{|T|^{|X|}}{C_{|T|}^{|X|}} = \lim_{|T|\to\infty} \frac{|T|^{|T|}}{|T|!} = \infty$$

À noter tout de même que ceci est un cas particulier de contraintes, il est également possible d'avoir tout autre type de contraintes tant le pouvoir expressif des CSPs est grand. Ce résultat est extrêmement dépendant du nombre de contraintes du problème. Plus le problème sera contraint et plus l'espace d'actions sera réduit ainsi que le facteur de branchement qui va avec. Voyons maintenant un algorithme qui pourrait traiter notre MaCSP.

3.5.2 Algorithme en ligne de résolution de MaCSPs

Du fait que le MaCSP mixe deux approches pour lesquelles de nombreux algorithmes ont été proposés, nous avons commencé directement par élaborer un algorithme (Algorithme 9) en ligne exploitant les différentes avancées déjà effectuées dans les domaines des MDPs et des CSPs. Plus précisément, nous avons choisi d'utiliser les techniques temps réel de programmation dynamique (RTDP) pour l'aspect markovien du modèle proposé. Ces approches sont, à ce jour, considérées comme les approches optimales les plus efficaces. Nous avons trouvé dans la littérature plusieurs versions dérivées de l'algorithme RTDP original [Bonet et Geffner, 2003; McMahan *et al.*, 2005]. Nous avons choisi l'un des plus récents, Focused RTDP (FRTDP), développé par Smith et Simmons [2006] et qui est présentement, selon les auteurs, le plus efficace. Cet algorithme se base sur deux bornes, une borne supérieure $\mathcal{H}_U$ et une borne inférieure $\mathcal{H}_L$. L'utilisation de la borne inférieure offre une garantie en termes de fonction de valeur alors que la borne supérieure est utilisée pour guider la recherche.

En ce qui concerne l'aspect contraint du modèle, l'approche adoptée consiste à utiliser l'élimination de variables (ligne 27 de l'algorithme 9) [Dechter, 2003, chap. 13.3.3] qui permet de calculer toutes les solutions d'un CSP assez efficacement et donc de réduire le facteur de branchement du MDP en ne sélectionnant que les actions exécutables dans l'état courant. L'élimination de variable consiste essentiellement à propager les contraintes d'une variable que l'on cherche à éliminer sur les autres variables concernées par ces contraintes pour pouvoir l'enlever du CSP. Dès lors, une fois que toutes les variables sauf une ont été éliminées, une propagation inverse donne l'ensemble des solutions.

Algorithm 9 Focused RTDP pour MaCSPs

1: ▷ Initialisation de tous les états s
2: $(s.L, s.U) \leftarrow (\mathcal{H}_L, \mathcal{H}_U)$
3: $s.prio \leftarrow \Delta(s)$
4: **fonction** FRTDP(s_0, ε,$\mathcal{H}_L$,$\mathcal{H}_U$)
5: $D \leftarrow D_0$
6: **tant que** $s_0.U - s_0.L > \varepsilon$ **faire**
7: $(q_p, n_p, q_c, n_c) \leftarrow (0, 0, 0, 0)$
8: TRIALREC(s_0,W=1,d=0)
9: **si** $(q_c/n_c) \geqslant (q_p/n_p)$
10: **alors** $D \leftarrow k_D D$
11: **fin tant que**
12: **fin fonction**
13: **fonction** TRIALREC(s, W, d)
14: $(a^*, s^*, \delta) \leftarrow$ backup(s)
15: TRACKUPDQUAL($\delta W, d$)
16: **si** $\Delta(s) \leqslant 0$ **or** $d \geqslant D$
17: **alors retourner**
18: TRIALREC(s^*,$\gamma T^{a^*}_{s,s^*} W$,$d+1$)
19: backup(s)
20: **fin fonction**
21: **fonction** TRACKUPDQUAL(q, d)
22: **si** $d > D/k_D$ **alors**
23: $(q_c, n_c) \leftarrow (q_c + q, n_c + 1)$
24: **sinon** $(q_p, n_p) \leftarrow (q_p + q, n_p + 1)$
25: **fin fonction**
26: **fonction** BACKUP(s_i)
27: $\mathcal{A}_i \leftarrow$ VARELIMINATION(s_i)
28: $s_i.L \leftarrow \max_{a \in \mathcal{A}_i} \mathrm{QL}(s_i, a)$
29: $u \leftarrow \max_{a \in \mathcal{A}_i} \mathrm{QU}(s_i, a)$
30: $a^* \leftarrow \sup_{a \in \mathcal{A}_i} \mathrm{QU}(s_i, a)$
31: $\delta \leftarrow |s_i.U - u|$
32: $s_i.U \leftarrow u$
33: $p \leftarrow \max_{s_{i+1} \in \mathcal{S}} \gamma T^{a^*}_{s_i,s_{i+1}} s_{i+1}.prio$
34: $s^* \leftarrow \sup_{s_{i+1} \in \mathcal{S}} \gamma T^{a^*}_{s,s_{i+1}} s_{i+1}.prio$
35: $s_i.prio \leftarrow \min(\Delta(s_i), p)$
36: **retourner** (a^*, s^*, δ)
37: **fin fonction**
38: **fonction** $\Delta(s)$
39: **retourner** $|s.U - s.L| - \varepsilon/2$
40: **fin fonction**
41: **fonction** $\mathrm{QL}(s, a)$:
42: **retourner** $R(s, a) + \gamma \sum_{s' \in \mathcal{S}} \gamma T^a_{s,s'} s'.L$
43: **fin fonction**
44: **fonction** $\mathrm{QU}(s, a)$
45: **retourner** $R(s, a) + \gamma \sum_{s' \in \mathcal{S}} \gamma T^a_{s,s'} s'.U$
46: **fin fonction**

Description de l'algorithme 9 : Comme tous les algorithmes de type RTDP, l'exécution de FRTDP consiste à réaliser des trajectoires (*trials* en anglais, ligne 18) qui commencent toutes dans l'état initial s_0 donné et explore ensuite les états accessibles à partir de s_0 en utilisant la borne supérieure $\mathcal{H}_U$ pour choisir l'action à effectuer et un système de priorité pour sélectionner l'état suivant parmi les états accessibles avec l'action choisie. Une fois qu'un état final est atteint, l'algorithme effectue des *backups* sur la trajectoire suivie. Ces *backups* consistent essentiellement en des mises à jour de Bellman [Bellman, 1957] qui sont appliquées sur chaque borne (lignes 28, 29), sur la priorité de l'état suivant (ligne 33) et sur la qualité de la trajectoire : plus grande est la mise à jour sur la borne supérieure, meilleure est la qualité (ligne 31). La fonction BACKUP retourne l'action optimale courante basée sur $\mathcal{H}_U$, le prochain état à explorer basé sur sa priorité et la qualité du *backup*.

Comme il a été précédemment évoqué, FRTDP maintient aussi une borne inférieure $\mathcal{H}_L$ et utilise un critère de priorité (ligne 33) pour choisir l'état suivant une fois l'action sélectionnée. La borne inférieure $\mathcal{H}_L$ est utilisée à la terminaison de l'algorithme puisque la politique retournée se base sur la meilleure action selon $\mathcal{H}_L$. Cette borne contribue également au calcul de la priorité des états à explorer (ligne 35). La détection de la fin d'une trajectoire est également légèrement différente dans FRTDP : elle a été améliorée par l'ajout d'une profondeur de recherche D adaptative dans le but d'éviter de faire de trop longues trajectoires peu informatives trop tôt. Cette profondeur de recherche est augmentée (ligne 23) à chaque fois que la trajectoire précédente n'a pas été assez contributive à l'utilité. Cette utilité est représentée par δW où δ mesure la valeur de la mise à jour sur la borne supérieure de l'état s et W la probabilité que, selon la politique courante, on va passer par l'état s à partir de s_0. Le lecteur est invité à se référer au pseudo-code de l'algorithme 9 et à l'article de Smith et Simmons [2006] pour plus de détails sur l'algorithme FTRDP.

Détaillons maintenant la complexité en pire cas du modèle proposé avant de faire une analyse empirique de l'algorithme proposé pour ce modèle.

3.6 Résultats théoriques

Le problème majeur de l'utilisation de MaCSPs réside dans la complexité en pire cas. En effet, dans le pire des scénarios, il est possible de construire des exemples où toutes les solutions du CSP d'un état donné soient vraiment plus complexes à trouver que le calcul de la politique du MDP. En effet, la complexité en pire cas de trouver toutes les solutions d'un CSP est #P-complète [Dechter, 2003]. Résoudre un MaCSP consiste donc à résoudre complètement un nombre polynômial de CSPs si le nombre d'états est polynômialement borné, puis à trouver la politique optimale pour le MDP correspondant :

Proposition 3. *Trouver une politique pour un* MaCSP*, qui obtiendrait une récompense espérée d'au moins C, est* #P*.*

Démonstration. En supposant que les données d'entrées sont de taille polynômiale, i.e. que l'on a un nombre polynômial d'états, d'actions et que les fonctions de transition et de récompense peuvent être encodées à l'aide d'un nombre polynômial de bits, on peut en déduire que le nombre de variables dans le système est logarithmique en la taille de ces données d'entrées : $|\mathcal{X}| = \mathcal{O}(\log |\mathcal{S}|)$. De fait, avant de trouver la politique dans le MDP qui rapporte une récompense espérée d'au moins C, qui est un problème P-complet [Papadimitriou et Tsisiklis, 1987], il faut trouver pour chacun des états

l'ensemble des actions possibles dans cet état. Ce dernier problème consiste à énumérer toutes les solutions possibles du CSP de l'état correspondant qui aurait $\mathcal{O}(\log|\mathcal{A}|)$ variables. Sachant que l'énumération de toutes les solutions d'un CSP ayant un nombre polynômial de variables est #P-complet [Dechter, 2003], alors une borne supérieure pour la résolution d'un MaCSP est également #P puisqu'il s'agit de résoudre complètement un nombre polynômial de CSPs avant de calculer la récompense espérée. □

#P [Dechter, 2003] est la classe de complexité qui englobe le problème de trouver le nombre de solutions à un problème NP-complet ce qui par conséquent nécessite d'énumérer lesdites solutions. Comparativement à la P-complétude des MDPs, cette classe de complexité parait totalement irréalisable. Cependant, il faut garder à l'esprit que les données d'entrées des deux problèmes considérés diffèrent complètement. Pour les CSPs, c'est le nombre de variables et la taille de leurs domaines qui sont considérés comme données d'entrées. Alors que pour le MDP, il s'agit des tailles de l'espace d'état et de l'espace des actions. En supposant que la taille de l'espace d'état croît seulement polynômialement, et en considérant que le nombre de variables croît logarithmiquement en fonction de la taille de l'espace d'état et de l'espace des actions, la complexité en pire cas reste ''acceptable''.

En d'autres termes, on pourrait avoir à résoudre par exemple quelques milliers de CSPs similaires comprenant seulement une dizaine de variables et donc quasiment instantanés à résoudre. Nous allons voir dans la section suivante qu'en pratique, il est préférable d'utiliser les contraintes théoriquement plus complexes à résoudre que de modéliser ces contraintes directement dans le MDP.

De plus, l'étude de l'utilité des contraintes dans la représentation que l'agent se fait du monde est également applicable aux systèmes multiagents où le gain sera d'autant plus grand que la complexité de ces systèmes est de loin supérieure à la classe #P.

3.7 Résultats expérimentaux

L'exemple sur lequel nous avons réalisé nos expérimentations est le problème d'allocation dynamique d'armes à des menaces (DWTA en anglais pour Dynamic Weapon Target Allocation) présenté au début de ce chapitre. Considérons une plate-forme militaire possédant un arsenal de w armes attaquée par m menaces.

Sur cette plate-forme nous avons plusieurs types d'armes : deux lanceurs de missiles (ML), une mitrailleuse lourde (Gun) et un CIWS (pour Close-In Weapon System). Ces

armes ont leur utilisation contrainte comme expliqué dans la table 3.2. A cela s'ajoute des angles morts donnés par la table 3.3, des portées limitées et des probabilités de réussite données par la table 3.4 et un nombre fini de munitions. La plate-forme possède également deux STIRs (Spot & Track Illumination Radars) qui permettent de ''voir'' et suivre l'état des menaces.

C_1 :	L'arme doit ''voir'' la cible (cf. tables 3.3 et 3.4)
C_2 :	Un ML doit être guidé par un STIR du tir jusqu'à l'interception,
	Un Gun doit utiliser un STIR pour tirer,
C_3 :	Deux STIRs ne peuvent cibler la même menace.

TABLE 3.2 – Contraintes : La première contrainte spécifie que la menace allouée à une arme doit être à portée et ne pas être dans un angle mort (cf. table 3.3). La seconde spécifie qu'une arme doit nécessairement être accompagnée d'un STIR tout au long de son allocation pour pouvoir être guidé. La dernière spécifie que deux STIRs ne peuvent être alloués à la même menace à cause d'interférences qu'ils pourraient provoquer les uns envers les autres.

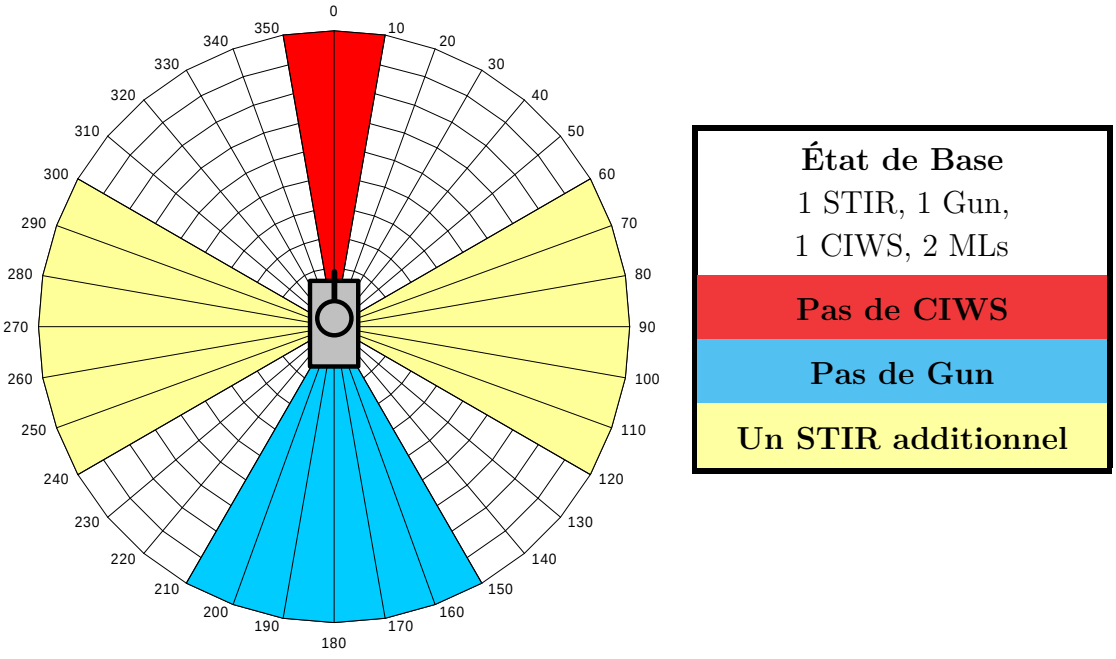


TABLE 3.3 – Angles morts : Le Gun et le CIWS étant placé respectivement à l'avant et à l'arrière du navire, il ne peuvent pas tirer au travers de celui-ci. Les deux stirs sont eux aussi orienté l'un vers l'avant, l'autre vers l'arrière, permettant ainsi que les deux soient disponibles sur chacun des flancs du bâtiment.

Un problème statique d'allocation d'armes représenté sous forme de CSP tel que présenté figure 3.2 peut-être formalisé de la manière suivante :

Arme	Portée	Probabilité de succès
ML	From 2.2 to 20km	95%
Gun	From 1.5 to 5km	50%
CIWS	From 0.2 to 2km	30%

TABLE 3.4 – Résultats des actions : Cette table spécifie la portée des différentes armes et les probabilités que chacune d'elle réussisse à détruire la menace qui lui aurait été allouée.

- $X = \mathcal{W}$ ensemble des w ressources du problème ;
- $D = \mathcal{M}$ ensemble des $m + 1$ tâches associées aux ressources (i.e., leurs allocations possibles à un instant donné) plus la possibilité de ne rien faire ;
- $C = \{c_1, ..., c_k, ...c_n\}$ ensemble des n contraintes, spécifiant les combinaisons de tâches mutuellement compatibles.

L'objectif étant de trouver une allocation possible des tâches aux ressources suivant les contraintes entre les ressources.

Il est cependant possible d'avoir des contraintes temporelles entre les tâches plutôt que des contraintes entre les ressources. Dans ce cas précis, on préférera employer la formulation ci-dessous, duale de celle exprimée précédemment :

- $X = \mathcal{M}$ ensemble des m tâches du problème ;
- $D = \mathcal{W}$ ensemble des w ressources associés aux tâches (i.e., leurs allocations possibles à un instant donné) ;
- $C = \{c_1, ..., c_k, ...c_n\}$ ensemble des n contraintes, spécifiant les combinaisons de ressources mutuellement compatibles.

L'objectif étant de trouver une allocation possible des ressources aux tâches suivant les contraintes entre les tâches.

3.7.1 Utilisation des contraintes seules

Dans notre problème, nous avons choisi d'implémenter l'instance du problème précédemment évoqué en spécifiant le MaCSP de la manière suivante : Les variables de décision sont les armes dont les domaines sont les menaces (on doit assigner à chaque arme une menace). Les contraintes et les probabilités de transition sont spécifiées telles qu'indiquées dans les tables 3.2, 3.3 et 3.4. Les menaces apparaissent à des distances distribuées selon une gaussienne autour de 20 km de la plate-forme et selon un azimut distribué uniformément autour de la plate-forme. On suppose que l'état des menaces (distance, azimut) évolue de manière déterministe. Ainsi, on connaît précisément quelle

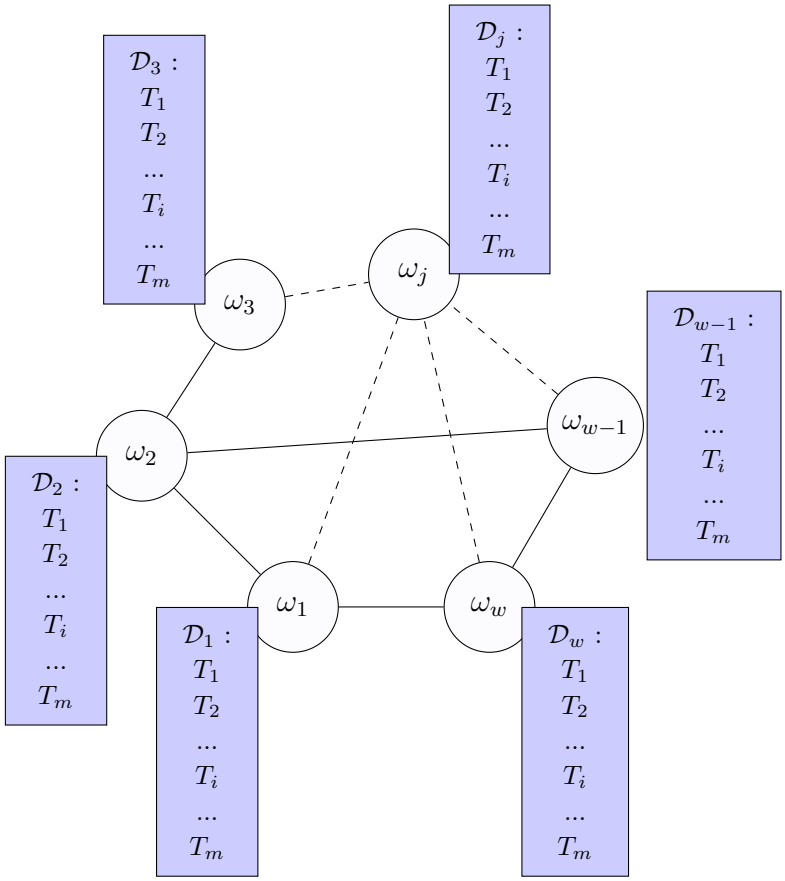


FIGURE 3.2 – Modélisation CSP du *Weapon-Target Assignment* où les armes sont les variables et les tâches représentent le domaine. Voir le texte pour plus de détails.

va être l'évolution des contraintes au cours du temps. La Figure 3.3 montre le DCSP à l'état initial et comment il peut évoluer durant un épisode. Il existe des contraintes unaires qui spécifient quelles sont les menaces ''visibles'' par les armes selon leur distance et leur azimut, et des contraintes binaires qui associent des radars à des armes pour pouvoir tirer ou spécifient certaines impossibilités. Dans cet exemple, les contraintes évoluent à chaque changement d'état, ou résolution d'engagement, à savoir si une arme a atteint une cible ou non. Les contraintes sont alors modifiées dynamiquement pour refléter l'utilisation actuelle des radars et des armes.

Nous avons comparé notre modèle avec un MDP factorisé sans contrainte, mais en donnant des probabilités nulles de succès lorsqu'une assignation était interdite (par exemple un missile tiré sur une menace trop éloignée a une chance nulle de l'atteindre). Les résultats présentés en Figure 3.4 sont obtenus en utilisant le FRTDP classique pour le MDP et notre adaptation pour le MaCSP. Chaque point est une moyenne sur 300 simulations. Nous avons utilisé pour ces tests les ''pires bornes possibles'' : une borne

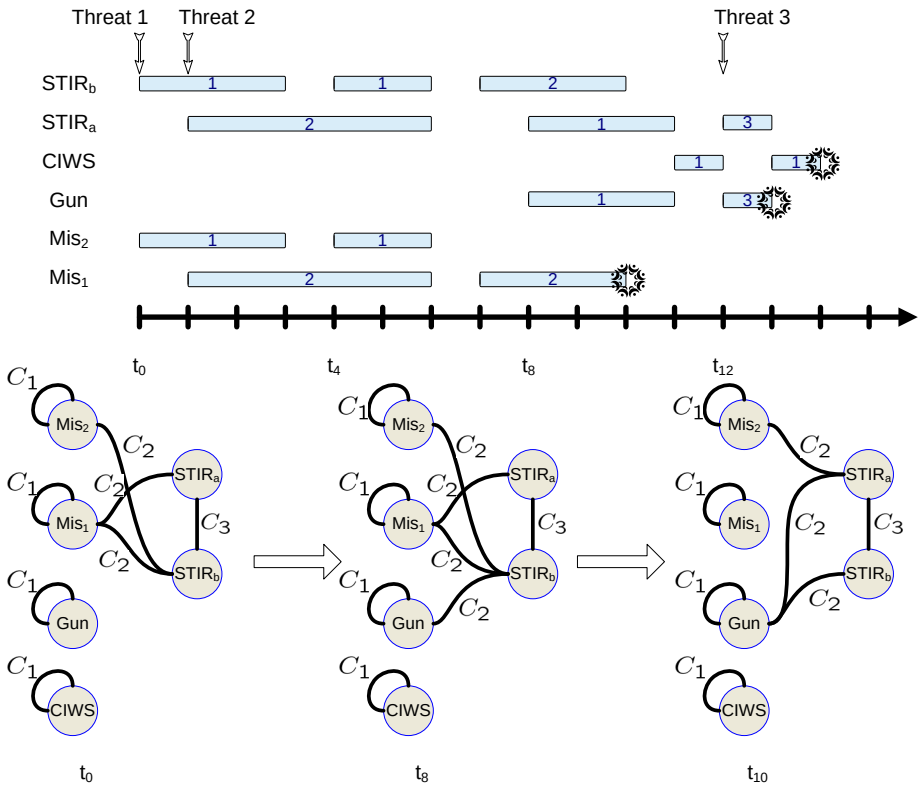
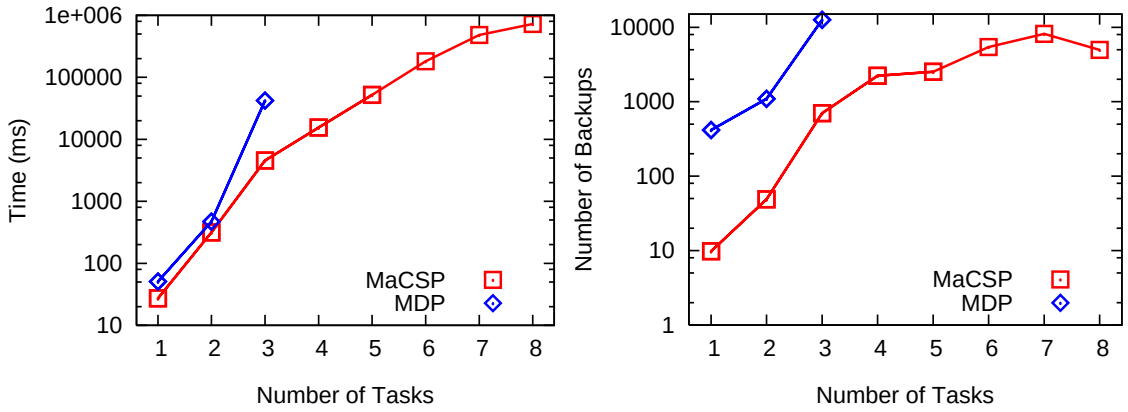


FIGURE 3.3 – Exemple de plan (en haut) et d'évolution du DCSP associé (en bas) pendant un épisode.



(a) Time (ms) vs # Tasks

(b) # Backups vs # Tasks

FIGURE 3.4 – Comparaison entre MDP et MACSPS.

inférieure nulle et une borne supérieure comme si les assignations d'armes faisables étaient toujours un succès. Ces bornes ont été utilisées pour vérifier la réelle efficacité

de l'algorithme sans aucun bénéfice tiré des bornes utilisées. Les résultats de la figure 3.4(a) représentent le temps de calcul pour la planification optimale tandis que la figure 3.4(b) représente le nombre de mises à jour de Bellman (appel à la fonction BACKUP de l'algorithme 9) en fonction du nombre de tâches. On pouvait s'attendre à ce genre de résultats pour les MDPs puisque la taille de l'espace d'état croît exponentiellement suivant le nombre de tâches et de ressources. On voit cependant qu'on gagne un ordre de magnitude en restreignant l'espace de recherche par les contraintes.

3.7.2 Utilisation des contraintes dans une borne

Nous avons également adapté les bornes initialement proposées par Plamondon *et al.* [2007] pour améliorer les performances de l'algorithme. La borne supérieure n'a pas été modifiée et utilise la même factorisation en tâche pour améliorer le temps d'évaluation de la borne. En revanche, la borne inférieure se base sur la résolution par contraintes et sur un algorithme réactif réalisant la tâche la plus urgente pour obtenir une bien meilleure estimation de la fonction de valeur minimum à un coût de calcul très légèrement supérieur. Ainsi, les gains réalisés en temps de calcul sont plus conséquent dus à un meilleur élagage de l'arbre de recherche. Les résultats sont rapportés dans la figure 3.5

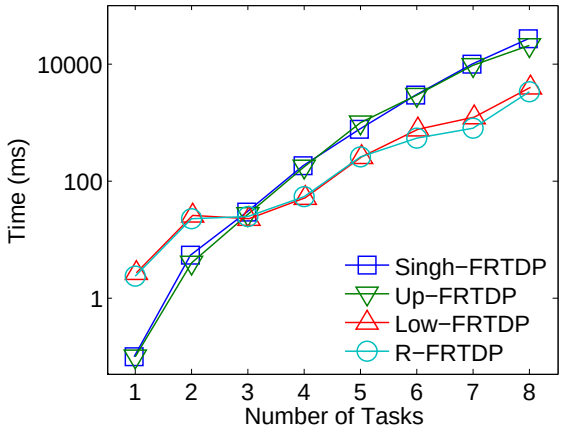


FIGURE 3.5 – Comparaison de l'utilisation de borne à base de CSP versus les bornes de la littérature.

Pour ces résultats, nous avons comparé l'utilisation des contraintes dans la borne inférieure versus les bornes disponibles dans la littérature. Singh-FRTDP représente l'algorithme de base FRTDP utilisant les bornes proposées par Singh et Cohn [1998]. Up-FRTDP est l'algorithme FRTDP utilisant la borne supérieure de Plamondon *et al.* [2007] et la borne inférieure de Singh et Cohn [1998]. Low-FRTDP utilise la borne supérieure de

Singh et Cohn [1998] et la borne inférieure à base de contraintes. R-FRTDP utilise la borne supérieure de Plamondon *et al.* [2007] et la borne inférieure à base de contraintes.

Les résultats indiquent clairement que les contraintes ne deviennent intéressantes à utiliser que lorsque le nombre de tâches devient important (ici supérieur à 3) mais que le gain s'avère finalement substantiel lorsque le nombre de tâches continue à croître (la figure utilise une échelle logarithmique sur les ordonnées).

3.8 Conclusion

Ce chapitre a, tout d'abord, fait état du problème d'allocation d'armes à des cibles tel que présenté dans la littérature. La version statique de ce problème a premièrement été détaillée avant de présenter la version dynamique et les différentes versions sous incertitudes. Un aperçu des modèles de satisfaction de contraintes a également été introduit comme introduction au modèle de Markov proposé dans ce chapitre. Un analyse théorique a été réalisée montrant que la complexité *en pire cas* n'est pas favorable au développement de tels modèles. Cependant, les expérimentations ont montré que, vu que le nombre de variables d'état est très souvent très inférieur (selon une loi logarithmique) au nombre d'états rencontrés, l'approche proposée permet des réductions substantielles du temps de calcul dans le problème d'allocation de ressources.

Parmi les travaux futurs envisagés à la suite de ce projet, plusieurs avenues de recherche ont attiré notre attention. Premièrement, l'hypothèse est faite que les perceptions de l'agent sont parfaites et donc que l'état du système est entièrement observé à chaque instant de décision. De fait, dans la réalité, les radars de navires sont rarement parfaits et un certaine incertitude est présente sur la position de menaces, sur les temps de vol des missiles et sur les observations de réussites (ou non) des contre-mesures. Il était donc naturel de regarder du côté des processus de Markov partiellement observables pour continuer nos travaux. D'autre part, nous avons considéré dans ce chapitre qu'un seul bâtiment avait à faire face à une attaque ennemie. En pratique, aucun navire ne circule seul et c'est souvent une flottille qui est sous l'effet d'une attaque ennemie. Il a donc été envisagé d'étudier les comportements d'un ensemble de navires sous les conditions précédemment évoquées.

Malheureusement, comme nous le verrons au chapitre suivant, la complexité des modèles de Markov multiagents partiellement observables est telle qu'aucun modèle actuel n'est en mesure de proposer une solution efficace pour le problème considéré. Nous allons donc vous détailler dans les chapitres suivants comment nous pensons qu'il

est nécessaire de modéliser ce type de problème pour le rendre resolvable, et ce, même lorsque plusieurs agents doivent agir dans un environnement hostile et incertain.

Chapitre 4

Contraintes sur l'observabilité des processus de Markov

«L'important, c'est de savoir ce qu'il faut observer.»

Histoires Extraordinaires -- Edgar Allan Poe

Ce chapitre introduit tout d'abord la problématique de la complexité associée à l'observabilité partielle dans les processus décisionnels de Markov. Il débute par un aperçu de la complexité des différents modèles existants et introduit ensuite un nouveau type d'observabilité contrainte par le concepteur. Ce nouveau type d'observabilité, qualifié de bijectivement observable *permet, dans certain cas identifiés, de réaliser des gains substantiels en complexité en pire cas. Ce chapitre détaille donc les apports pour le modèle à un seul agent partiellement observable (*POMDP*) avant d'étendre au modèle à plusieurs agents (*DEC-POMDP*). Des exemples d'applications sont ensuite modélisés et présentés.*

4.1 Introduction

La disponibilité et la complétude de l'information ont toujours été des facteurs prépondérant dans le domaine de la prise de décision. Dans les processus décisionnels de Markov, où toute l'information nécessaire est rassemblée dans l'état du système, l'observabilité de cet état influence énormément la complexité de la prise de décision

dans cet état. Toutefois, il est très rare de trouver des cas ou l'information disponible est parfaite et complète à la fois.

Introduite par Drake [1962], l'observabilité partielle modélise cet aspect incomplet ou bruité de l'observation de l'état par l'agent intelligent. Une *fonction d'observation* est alors utilisée qui associe à chaque état une distribution sur un ensemble restreint de symboles. Une autre manière de voir cette observabilité partielle est de considérer que l'état est transmis au travers d'un canal de communication bruité (e.g. les capteurs). Cette représentation très générale permet de modéliser tous les niveaux d'observabilité, depuis l'observabilité complète, où à chaque état est associé un symbole particulier qui peut être l'état lui-même, jusqu'à l'observabilité nulle, où à chaque état est associé le même symbole. Cette représentation à été utilisée dans tous les modèles de Markov depuis, et en particulier dans les POMDPs pour le cas à un seul agent, et dans les DEC-POMDPs dans le cas multiagent.

Cependant, avec l'observabilité partielle vient nécessairement une augmentation de la complexité algorithmique en pire cas. Comme démontré dans la figure 4.1, à mesure que l'observabilité diminue, i.e. que les agents ont de moins en moins d'information pour raisonner, la complexité augmente généralement.

Ainsi, en présence d'information complète et parfaite -- lorsque le ou les agents ont entièrement accès à l'état réel du système -- le problème à résoudre est seulement P-complet. Seulement, dès que l'observation devient imparfaite, et même si elle reste complète -- lorsque tous les agents ont la même information bruitée de l'état du système -- résoudre le problème devient alors PSPACE-complet, et ce, même si la fonction de transition est déterministe (c.f. les POMDPs déterministes ou DetPOMDPs au chapitre 6 de la thèse de Littman [1996]). La décentralisation de l'information -- lorsque les agents n'ont plus nécessairement accès à la même information bruitée sur l'état du monde -- accroît encore cette complexité du problème jusqu'à la NEXP-complétude. Les cas particuliers du DEC-MDP-COM avec communications gratuites et du UMDP (Unobservable MDP) sont facilement explicables. Dans le cas du DEC-MDP-COM, les agents ont une vue non bruitée locale du système. Ainsi, de par la communication, ils peuvent s'échanger toutes leurs vues locales pour obtenir une vue globale équivalente au problème où chacun aurait déjà cette observabilité complète en partant. Dans le cas du MDP non observable, le cas est simple puisque ne sachant pas où ils se trouvent, les agents ne raisonnent plus sur les états possibles du monde, mais seulement sur les objectifs à atteindre et sur leur probabilité de réalisation.

Il convient toutefois de remarquer, comme le montre la figure 4.1, qu'il existe un fossé extrême lorsque l'on passe du cas complètement observable (ou non observable),

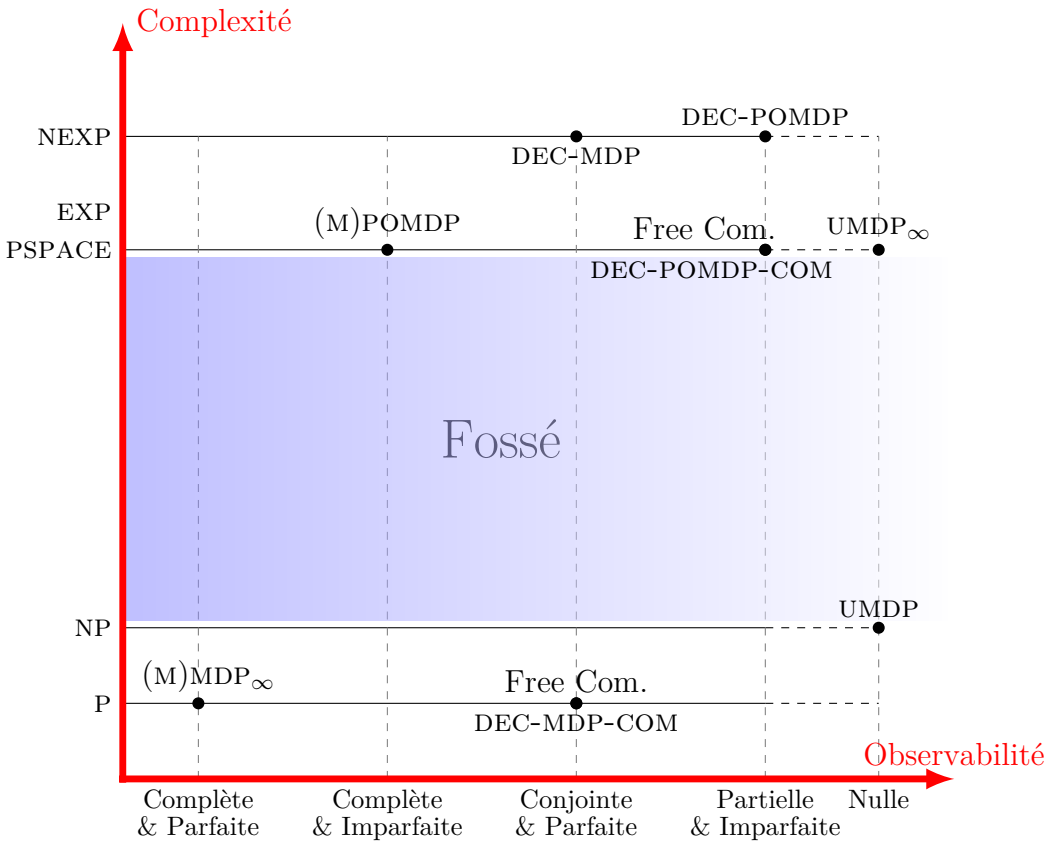


FIGURE 4.1 – Mesure de la complexité fonction de l'observabilité

au cas partiellement observable. Ceci s'explique en partie par la très grande capacité de représentation des modèles partiellement observables, leur conférant une expressivité à la mesure de leur complexité. Dans les sections suivantes, les travaux réalisés cherchent donc à réduire cette complexité tout en équilibrant au mieux l'expressivité. Des modèles particuliers d'observabilité ont été développés à cette fin permettant ainsi de réduire cette complexité tout en gardant une expressivité suffisante.

Pour réduire une telle complexité, nous avons tout d'abord convenu de restreindre notre étude aux modèles monoagents à transition déterministe. La version multiagent sera ensuite présentée avant de discuter de l'extension possible au cas stochastique de la fonction de transition. Rappelons donc tout d'abord les processus de Markov partiellement observables déterministes avant de proposer de nouveaux modèles d'observation.

4.2 POMDPs quasi-déterministes

Les POMDPs déterministes ont été définis par Littman [1996], comme suit :

Définition 16. *Un Processus Décisionnel de Markov Partiellement Observable Déterministe (*DET-POMDP*) est un tuple* $\langle \mathcal{S}, \mathcal{A}, \Omega, \mathcal{T}, \mathcal{O}, \mathcal{R}, \gamma, \boldsymbol{b}^0 \rangle$*, où :*

- $\mathcal{S}$ *est un ensemble fini d'*états $s \in \mathcal{S}$ *;*
- $\mathcal{A}$ *est un ensemble fini d'*actions $a \in \mathcal{A}$ *;*
- $\mathcal{T}(s, a, s') : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \{0, 1\}$ *est la* fonction déterministe de transition *du système de l'état s vers l'état s' après l'exécution de l'action a ;*
- $\mathcal{R}(s, a) : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ *est la* récompense *produite par le système lorsque l'action a est exécutée dans l'état s,*
- Ω *est un ensemble fini d'observations* $o \in \Omega$*,*
- $\mathcal{O}(s, a, o, s') : \mathcal{S} \times \mathcal{A} \times \Omega \times \mathcal{S} \mapsto \{0, 1\}$ *est la* fonction déterministe *qui indique si l'observation o survient ou non lors de la transition du système de l'état s vers l'état s' sous l'effet de l'action a.*
- γ *est le facteur d'escompte ;*
- $\boldsymbol{b}^0$ *est la connaissance* a priori *sur l'état, i.e. l'état de croyance initial, supposé non déterministe.*

Remarquons que l'état de croyance initial $\boldsymbol{b}^0$ -- qui décrit les différentes possibilités pour l'état de départ -- est crucial. En effet, si l'état de départ est connu, et la transition d'un état vers un autre déterministe, alors la séquence complète des états suivants est également prévisible, et le problème de DET-POMDP se ramène alors au problème déjà très étudié dans la littérature de l'IA de la planification déterministe [Nau *et al.*, 2004].

Par comparaison au POMDP déterministe, l'extension que nous proposons modifie la définition de la fonction d'observation de telle sorte qu'elle soit maintenant stochastique tout en assurant qu'il existe certaines observations qui sont vraiment plus probables que d'autres dans chacun des états. Formellement :

Définition 17. *Un processus décisionnel de Markov partiellement observable quasi-déterministe (*QDET-POMDP*) est un tuple* $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{O}, \mathcal{R}, \gamma, \boldsymbol{b}^0 \rangle$*, où :*

- $\mathcal{S}$ *est un ensemble fini d'*états $s \in \mathcal{S}$ *;*
- $\mathcal{A}$ *est un ensemble fini d'*actions $a \in \mathcal{A}$ *;*
- $\mathcal{T}(s, a, s') : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \{0, 1\}$ *est la* fonction déterministe de transition *du système de l'état s vers l'état s' après l'exécution de l'action a ;*
- $\mathcal{R}(s, a) : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ *est la* récompense *produite par le système lorsque l'action a est exécutée dans l'état s,*
- $\mathcal{O}(z, a, s') : \Omega \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ *est une* fonction d'observation *qui indique la probabilité d'obtenir l'observation z lorsque le monde arrive dans l'état s' après*

avoir exécuté l'action a ;
De plus, $\forall\, s' \in \mathcal{S}$, $a \in \mathcal{A}$, $\exists\, z \in \Omega$, s.t. $\mathcal{O}(z, a, s') \geqslant \theta > \frac{1}{2}$, i.e. le monde est minimalement observable et la probabilité d'obtenir une des observations est bornée inférieurement par un demi ;

- *γ est le facteur d'escompte ;*
- *$\boldsymbol{b}^0$ est la connaissance* a priori *sur l'état, i.e. l'état de croyance initial, supposé non déterministe.*

En d'autres termes, ce modèle considère les systèmes possédant une fonction de transition déterministe, mais une fonction d'observation stochastique où il existe une observation dans chaque état telle que sa probabilité d'être observée peut-être bornée inférieurement par un réel θ. Il convient de noter ici que θ n'est qu'une borne inférieure sur la probabilité d'observer chaque état et que par conséquent celle-ci peut éventuellement être plus grande. Remarquons également que l'horizon de planification n'est pas fixé *a priori*. Ceci est dû au fait -- comme le présente la sous-section 4.2.2 -- que l'on peut démontrer sous certaines conditions sur l'observation que l'état de croyance converge avec grande probabilité vers une distribution déterministe après un nombre fini d'étapes. Il reste donc à définir dans ce type de modèle quasi-déterministe des fonctions d'observations particulières qui permettront de réduire la complexité en pire cas.

4.2.1 Variantes sur l'observabilité

Comme énoncé à la sous-section précédente, le modèle d'observation influence fortement la caractérisation de la complexité en pire cas. Pour rappel, voici différentes variantes du modèle d'observation :

Modèles inobservables pour lesquels $|\Omega| = 1$ ou $\forall s, s' \in \mathcal{S}, \forall a \in \mathcal{A}, \forall o \in \mathcal{O}, P(o|s, a, s') \sim \mathcal{U}$ et donc aucune information ne peut être obtenue sur l'état. Cette classe est une sous-classe du problème de planification avec observation nulle à l'exécution (*conformant planning*[1] [Goldman et Boddy, 1996]).

Modèles complètement observables pour lesquels $\Omega = \mathcal{S}$ et $\mathcal{O}(z, a, s') = 1$ ssi $z = s'$. Cette classe correspond exactement à la classe des processus décisionnels de Markov (MDPs) où seulement l'état initial est inconnu (e.g. QMDP [Kaelbling *et al.*, 1998]).

Modèles partiellement observables pour lesquels $|\Omega| > 1$. Cette classe est exactement le complémentaire de l'ensemble des problèmes inobservables. Parmi ces problèmes, on peut distinguer :

1. Les différents types de planification peuvent être trouvés dans l'annexe B

Modèles bijectivement observables pour lesquels $\Omega = \mathcal{S}$. Cette classe regroupe tous les problèmes où la fonction d'observation est linéaire, mais bruitée. L'état réel du système peut être perçu avec éventuellement du bruit. Cette classe regroupe typiquement les problèmes de contrôle robotique où l'état est perçu au travers de capteurs bruités.

Modèles bijectivement observables factorisés pour lesquels $|\Omega| = \cup_{\mathrm{x}\in\mathcal{X}}|\mathcal{D}_{\mathrm{x}}|$. Où $\mathcal{X}$ est l'ensemble des variables d'états et $\mathcal{D}_{\mathrm{x}}$ est le domaine de la variable x. L'espace d'état est alors donné par $\mathcal{S} = \bigtimes_{\mathrm{x}\in\mathcal{X}} \mathcal{D}_{\mathrm{x}}$. Cette classe est très similaire de la précédente à la différence que le bruit est maintenant distribué sur les domaines des variables et donc supposément non corrélé au bruit ni au valeurs des autres variables. De plus, les observations sont restreintes selon les ''dimensions'' de l'espace d'état. En effet, l'hypothèse est faite ici que seulement de l'information d'un seul capteur arrive à la fois et qu'ainsi de l'information selon une seule ''dimension'' est perçue à chaque étape. Cette classe peut se ramener à la classe précédente si on considère une seule variable d'état dont le domaine est $\mathcal{S}$.

Modèle général qui inclut tous les cas précédents sans aucune hypothèse sur le nombre d'observations ou la fonction d'observation.

Dans la suite du document, ni le cas général, ni les cas inobservable et totalement observable ne seront décrits ni étudiés puisqu'ils ont déjà fait l'objet de nombreuse publications [Nau *et al.*, 2004; Kaelbling *et al.*, 1998]. Nous proposons cependant d'étudier plus en particulier le cas bijectivement observable ainsi que le cas avec observations factorisées. En effet, une majorité des problèmes réels se rapprochent du cas avec observation factorisée ou au moins bijectivement observable. La prochaine section explique comment ces problèmes sont en réalité plus simple que la formulation générale en bornant la mémoire nécessaire qu'un agent doit avoir pour planifier ε-optimalement.

4.2.2 Résultats théoriques

Comme il a été présenté dans le chapitre 2, une façon de représenter de manière compacte la séquence d'observation obtenue est l'état de croyance [Sondik, 1971]. Un tel état (noté $\boldsymbol{b}^t$) est une distribution de probabilité sur l'espace d'états qui représente la probabilité que l'agent soit dans chacun des états à l'instant t. Formellement, $\boldsymbol{b}^t(s) = \Pr(s|z^t, a^t, \boldsymbol{b}^{t-1})$ est la probabilité d'être dans l'état s à l'étape t sachant que l'observation z^t venait d'être obtenue après avoir effectué l'action a dans l'état de croyance $\boldsymbol{b}^{t-1}$. Cette mise à jour de l'état de croyance s'effectuait de la manière suivante :

$$\boldsymbol{b}^t(s) = \frac{\mathcal{O}(z^t, a^t, s)\sum_{s'\in\mathcal{S}}\mathcal{T}(s', a^t, s)\boldsymbol{b}^{t-1}(s')}{\sum_{s''\in\mathcal{S}}\mathcal{O}(z^t, a^t, s'')\sum_{s'\in\mathcal{S}}\mathcal{T}(s', a^t, s'')\boldsymbol{b}^{t-1}(s')} \tag{4.1}$$

En utilisant une représentation matricielle des différentes fonctions, cette équation (4.1) peut être réécrite de la manière suivante :

$$\boldsymbol{b}^k(s) = \frac{D_k T_{a^k} \cdots D_1 T_{a^1} \boldsymbol{b}^0}{\mathbb{1}^\top D_k T_{a^k} \cdots D_1 T_{a^1} \boldsymbol{b}^0} \tag{4.2}$$

Où $\boldsymbol{b}^0$ est l'état de croyance initial, T_{a^t} sont les matrices de transitions selon les action a^t choisies, D_i sont des matrices diagonales dont chaque terme représente la probabilité d'observer z_i sachant chaque état, et $\mathbb{1}$ est un vecteur de 1 de dimension $|\mathcal{S}|$.

Intuitivement, la convergence de l'état de croyance vers une distribution déterministe dépend du nombre n d'observations réussies parmi les k étapes dans un contexte non inobservable. Néanmoins, le critère de partiellement observable n'est pas suffisant pour garantir la convergence. C'est pourquoi nous avons suggéré les deux sous-classes d'observabilité que nous nous proposons d'étudier maintenant.

4.2.2.1 Modèles bijectivement observables

Un modèle bijectivement observable garantit qu'il n'existe qu'une seule observation la plus probable (MLO pour *Most Likely Observation*) dans chaque état et que chaque MLO d'un état n'est la MLO d'aucun autre état. Plus formellement :

Définition 18. *Un* QDET-POMDP *bijectivement observable est un* QDET-POMDP *où l'hypothèse suivante est faite :*

$$\begin{aligned}
&\exists o_1 \in \Omega,\, \forall a \in \mathcal{A},\, \forall s \in \mathcal{S}^{o_1}, \\
&\textit{avec } \mathcal{S}^{o_1} = \{s \in \mathcal{S} | \exists o_1 \in \Omega, P(o_1|s,a) > P(o|s,a), \forall o \neq o_1\}, \\
&\textit{alors } |\Omega| = |\mathcal{S}| \textit{ et } |\mathcal{S}^{o_1}| = 1
\end{aligned}$$

En d'autres mots, $\mathcal{S}^{o_1}$ est l'ensemble des états ou o_1 est la MLO.

Considérant cette définition, il est maintenant possible d'énoncer le théorème suivant :

Théorème 2. *Sous l'hypothèse d'observation bijective, l'état de croyance $\boldsymbol{b}^k$ est tel que $\boldsymbol{b}^k(s) \geqslant 1 - \varepsilon$ ssi*

$$n \geqslant \frac{1}{2 \ln \frac{\nu\theta}{(1-\theta)}} \ln \left[\frac{1-\varepsilon}{\varepsilon} \left(1 + \nu^{1-\frac{k}{2}}\right) \right] + \frac{k}{2} \tag{4.3}$$

Où $\nu = \max_{s,a} \sum_{z \in \Omega} I(\theta > \mathcal{O}(z,a,s) > 0) < |\Omega|$ est le nombre d'observations autres que la MLO *qui peuvent être perçues dans un état.*

Démonstration. En pire cas, la probabilité d'observer l'observation la plus probable (soit l'état sous-jacent du système) est toujours minimale et égale à θ à chaque étape. De plus, si à chaque fois que l'observation ''échoue'' (i.e. que l'observation obtenue n'est pas la plus probable), celle-ci sous-tend toujours le deuxième état le plus probable, cela résulte en une augmentation de la probabilité de ce dernier dans l'état de croyance. Étant donné l'équation (4.2) et sachant que les transitions sont déterministes, induisant des matrices de permutations pour les matrices de transition, on peut montrer que :

$$\frac{\theta^n \frac{(1-\theta)^p}{\nu^p}}{\theta^n \frac{(1-\theta)^p}{\nu^p} + \theta^p \frac{(1-\theta)^n}{\nu^n} + (\nu - 1)\frac{(1-\theta)^k}{\nu^k}} \geqslant 1 - \varepsilon \tag{4.4}$$

Où n est le nombre d'observations ''réussies'' (i.e. où l'observation obtenue était la plus probable dans l'état caché sous-jacent) et $p = k - n$ est le nombre d'observations échouées. Le numérateur est donc le résultat de l'obtention de n ''bonnes'' observations et de p ''mauvaises'' pendant l'exécution. Le dénominateur somme sur l'ensemble des états pour la même séquence d'observations. Le premier terme correspond à l'état le plus probable, le second terme au second état le plus probable (soutenu par les p ''mauvaises'' observations) et le troisième terme correspond aux autres états selon le nombre ν d'états où on aurait pu percevoir les ''mauvaises'' observations. L'hypothèse est faite dans cette preuve que la probabilité d'obtenir une ''mauvaise'' observation est uniforme sur l'ensemble des observations qui ne sont pas la MLO. Cette hypothèse n'induit pas une perte de généralité [2] et est justifiée par le *principe d'entropie maximale* qui veut que, selon notre connaissance *a priori*, la distribution d'entropie maximale -- la loi uniforme dans notre cas -- soit la distribution la plus appropriée. La résolution de l'équation (4.4)

2. Et poser $\nu = 1$ ramène au pire cas où l'observation est soit ''bonne'' soit ''mauvaise''.

mène à l'équation (4.3) de la manière suivante :

$$
\begin{aligned}
& \frac{\theta^n \frac{(1-\theta)^p}{\nu^p}}{\theta^n \frac{(1-\theta)^p}{\nu^p} + \theta^p \frac{(1-\theta)^n}{\nu^n} + (\nu-1)\frac{(1-\theta)^k}{\nu^k}} \geqslant 1-\varepsilon \\
\Leftrightarrow \quad & \frac{\nu^n\theta^n(1-\theta)^p}{\nu^n\theta^n(1-\theta)^p + \nu^p\theta^p(1-\theta)^n + (\nu-1)(1-\theta)^k} \geqslant 1-\varepsilon \\
\Leftrightarrow \quad & \frac{\nu^p\theta^p(1-\theta)^n}{\nu^n\theta^n(1-\theta)^p} + \frac{(\nu-1)(1-\theta)^k}{\nu^n\theta^n(1-\theta)^p} \leqslant \frac{1}{1-\varepsilon} - 1 \\
\Leftrightarrow \quad & \nu^{k-2n}\theta^{k-2n}(1-\theta)^{2n-k} + (\nu-1)\frac{\theta^{-n}\nu^{-n}}{(1-\theta)^{-n}} \leqslant \frac{\varepsilon}{1-\varepsilon} \\
\Leftrightarrow \quad & \frac{\nu^{k-2n}\theta^{k-2n}}{(1-\theta)^{k-2n}}\left[1 + (\nu-1)\frac{\nu^{-p}\theta^{-p}}{(1-\theta)^{-p}}\right] \leqslant \frac{\varepsilon}{1-\varepsilon} \\
\Leftrightarrow \quad & (k-2n)\ln\frac{\nu\theta}{(1-\theta)} + \ln\left[1 + (\nu-1)\frac{\nu^{-p}\theta^{-p}}{(1-\theta)^{-p}}\right] \leqslant \ln\frac{\varepsilon}{1-\varepsilon} \\
\Leftrightarrow \quad & (k-2n)\ln\frac{\nu\theta}{(1-\theta)} \leqslant \ln\frac{\varepsilon}{1-\varepsilon} - \ln\left[1 + (\nu-1)\frac{(1-\theta)^p}{\nu^p\theta^p}\right] \\
\Leftrightarrow \quad & (2n-k)\ln\frac{\nu\theta}{(1-\theta)} \geqslant \ln\frac{1-\varepsilon}{\varepsilon} + \ln\left[1 + (\nu-1)\frac{(1-\theta)^p}{\nu^p\theta^p}\right] \\
\Leftrightarrow \quad & (2n-k) \geqslant \frac{\ln\frac{1-\varepsilon}{\varepsilon}}{\ln\frac{\nu\theta}{(1-\theta)}} + \frac{1}{\ln\frac{\nu\theta}{(1-\theta)}}\ln\left[1 + (\nu-1)\frac{(1-\theta)^p}{\nu^p\theta^p}\right] \\
\Leftrightarrow \quad & n \geqslant \frac{\ln\frac{1-\varepsilon}{\varepsilon}}{2\ln\frac{\nu\theta}{(1-\theta)}} + \frac{1}{2\ln\frac{\nu\theta}{(1-\theta)}}\ln\left[1 + (\nu-1)\frac{(1-\theta)^p}{\nu^p\theta^p}\right] + \frac{k}{2} \qquad (4.5)
\end{aligned}
$$

mais puisque $1 \leqslant \nu \leqslant |\mathcal{S}| - 1$, $n > \frac{k}{2}$ et $\frac{1-\theta}{\theta} < 1$,

$$
\begin{aligned}
\ln\left[1 + \frac{|\mathcal{S}|-2}{\nu^{k-n}}\frac{(1-\theta)^{k-n}}{\theta^{k-n}}\right] & \leqslant \ln\left[1 + \frac{|\mathcal{S}|-2}{\nu^{\frac{k}{2}}}\left(\frac{1-\theta}{\theta}\right)^{\frac{k}{2}}\right] \\
& \leqslant \ln\left[1 + \nu^{1-\frac{k}{2}}\right]
\end{aligned}
$$

Ainsi, s'assurer de l'inégalité (4.3) assure également l'inégalité (4.5). □

En d'autres termes, ν représente également la manière dont l'erreur se répartit sur l'espace d'état. Le théorème 2 déclare que si l'observation est suffisante (avec une large probabilité θ d'observer l'état réel sous-jacent du système) et si l'erreur se répartit sur un grand nombre d'états (lorsque ν est grand), alors il suffit que seulement la moitié des observations plus une correspondent à l'état réel du système pour que l'état de croyance converge vers une distribution déterministe, i.e. vers un seul état.

4.2.2.2 Modèles factorisés

D'une manière plus générale et surtout plus structurée que les modèles bijectivement observables, les modèles dit factorisés assurent que chaque valeur de chaque variable est observée un nombre suffisant de fois. Ainsi, ils permettent de déterminer également l'état sous-jacent du système en un nombre fini d'étapes :

Définition 19. *Un* qDet-pomdp *à observations factorisées est un* qDet-pomdp *où les hypothèses suivantes sont faites :*

- *l'espace d'état est* factorisé *en μ variables d'états : $\mathcal{S} = \bigtimes_{\mathrm{x}\in\mathcal{X}} \mathcal{D}_{\mathrm{x}}$ et les observations possibles sont $\Omega = \bigcup_{\mathrm{x}\in\mathcal{X}} \mathcal{D}_{\mathrm{x}}$.*
- *La somme des probabilités d'obtenir une des valeurs réelles des variables d'état est bornée inférieurement par $\theta > \frac{1}{2}$.*

Cette définition implique que, dans le pire cas et pour chaque variable d'état, il existe une probabilité θ/μ d'observer la vraie valeur de cette variable et une probabilité $(1-\theta)/(|\Omega|-\mu)$ d'observer une autre valeur. Remarquons également que cette définition est une généralisation de la définition 18 qui correspond au cas $\mu = 1$. Cette constatation mène au théorème suivant :

Théorème 3. *Sous l'hypothèse d'observations factorisées bijectives, l'état de croyance $\boldsymbol{b}^k$ est tel que $\boldsymbol{b}^k(s) \geqslant 1-\varepsilon$ ssi*

$$n \geqslant \frac{1}{2\ln\frac{(|\Omega|-\mu)\theta}{\mu(1-\theta)}} \ln\left[\frac{1-\varepsilon}{\varepsilon}\left(1+|\mathcal{S}|-\mu\right)\right] + \frac{k}{2} \tag{4.6}$$

Démonstration. La preuve suit les mêmes arguments que la preuve du théorème 2. □

Une fois que le nombre n d'observations les plus probables est borné, trouver la probabilité d'obtenir au moins n ''bonnes'' observations est simplement une application de la loi de la binômiale d'avoir au moins n succès sur k essais :

Corollaire 1. *Dans tout* qDet-pomdp *satisfaisant les hypothèses d'observabilité bijective, la probabilité que l'état de croyance $\boldsymbol{b}^k(s)$ soit ε-déterministe après k étapes est :*

$$\exists s, \Pr(\boldsymbol{b}^k(s) \geqslant 1-\varepsilon) \geqslant \sum_{i=n}^{k} \binom{k}{i} \theta^i (1-\theta)^{k-i} \tag{4.7}$$

Où n est donné par l'équation (4.3) dans le cas simple et par l'équation (4.6) dans le cas factorisé.

Cette équation indique que pour être certain (à δ près) d'avoir un état de croyance déterministe (à ε près), il se peut que l'horizon à explorer soit très grand si θ est trop proche de $\frac{1}{2}$.

Voyons maintenant l'impact sur la complexité de ce modèle.

4.2.2.3 Analyse de la complexité du modèle proposé

Un implication majeure des théorèmes 2 et 3 est la réduction de la complexité par rapport aux problèmes généraux représentables par un POMDP, mais où un QDET-POMDP pourrait être utilisé. En fait, Papadimitriou et Tsisiklis [1987] ont montré que les POMDPs à horizon fini sont PSPACE-complets. Ceci repose principalement sur le fait que chaque agent a à choisir une action qui, étant donné n'importe quelle observation, va mener au choix d'une autre action et ainsi de suite, jusqu'à atteindre un horizon T. Cependant, ramener cet horizon T à une constante diminue artificiellement la complexité dans la hiérarchie polynômiale [Stockmeyer, 1976]. La hiérarchie polynômiale est une classe générale de problèmes inclue dans PSPACE et qui est basée sur les oracles. Stockmeyer [1976] a tout d'abord défini $\Sigma_2^{\mathrm{P}} = \mathrm{NP}^{\mathrm{NP}}$ comme étant la classe des problèmes de décision qui peuvent être résolus en temps polynômial par une machine de Turing non déterministe tout en utilisant un oracle NP. Le problème ''canonique'' pour cette classe de complexité (si SAT est celle de la classe NP) est le problème 2-QBF. Ce problème consiste à déterminer si la formule quantifiée booléenne suivante est vraie : $\exists\vec{a}\forall\vec{b}\,\phi(\vec{a},\vec{b})$. Monter d'un cran dans cette hiérarchie polynômiale (par exemple Σ_3^{P}) consiste à ajouter un autre quantificateur pour un autre ensemble de variables de la formule booléenne -- i.e. vérifier si $\exists\vec{a}\forall\vec{b}\exists\vec{c}\,\phi(\vec{a},\vec{b},\vec{c})$ est vraie -- et ainsi de suite. Dans le cas précis d'un POMDP à horizon *constant*, il est donc possible d'énoncer la proposition suivante :

Proposition 4. *Trouver une politique pour un* POMDP *à horizon k constant, qui obtiendrait une récompense espérée d'au moins C, est dans la classe de complexité* $\Sigma_{2k-1}^{\mathrm{P}}$.

Démonstration. Pour montrer que le problème est dans $\Sigma_{2k-1}^{\mathrm{P}}$, il est possible d'utiliser l'algorithme suivant qui utilise un oracle $\Sigma_{2k-2}^{\mathrm{P}}$: Demander à l'oracle une politique pour $k-1$ étapes, vérifier que cette politique obtient une récompense espérée d'au moins C en temps polynomial en vérifiant les $|\Omega|^k$ historiques possibles, puisque k est constant. □

Puisque les QDET-POMDPs sont une sous-classe de POMDPs et puisque choisir $1-\delta$, i.e. la probabilité d'être dans un état de croyance ε-déterministe, induit un horizon constant sous les hypothèses d'observabilité bijective :

Corollaire 2. *Trouver une politique pour un* QDET-POMDP *à horizon infini, respectant l'hypothèse d'observabilité bijective, et qui permette d'obtenir une récompense escomptée espérée d'au moins C avec probabilité $1-\delta$, est $\Sigma^{\mathrm{P}}_{2k-1}$.*

Démonstration. Pour montrer que cette classe de problème est dans $\Sigma^{\mathrm{P}}_{2k-1}$, l'algorithme suivant donne une politique ε-optimale en temps polynômial sous l'hypothèse de l'accès à un oracle $\Sigma^{\mathrm{P}}_{2k-2}$: Demander à l'oracle de fournir un politique pour $k-1$ étapes et vérifier que cette politique obtient une récompense espérée d'au moins C en calculant la valeur de chacun des états de croyance dans chaque feuille de l'arbre des observations possibles (qui est de taille polynômiale puisque k est constant), puis ajouter la valeur escomptée espérée du MDP sous-jacent puisque l'état de croyance est ε-déterministe dans $100(1-\delta)\%$ des feuilles de l'arbre. □

En pratique, trouver une politique probablement approximativement correctement ε-optimale pour un QDET-POMDP qui respecte une des hypothèses sur l'observabilité implique de calculer une politique optimale pour les k premières étapes puis ensuite d'utiliser la politique optimale du MDP sous-jacent pour l'infinité du temps restant. Ainsi, il suffit de borner la probabilité voulue d'être dans un état de croyance ε-déterministe pour borner supérieurement l'horizon de planification et garantir que la politique du MDP sous-jacent va bien se comporter par la suite.

4.2.3 Résultats expérimentaux : application au dirigeable

Un autre application pratique des théorèmes 2 et 3 réside dans l'estimation de l'information disponible à l'agent à chaque instant. En effet, en pratique la dynamique des systèmes complexes est plus souvent stochastique et donc le résultat précédemment proposé peut avoir une autre interprétation correspondant à la quantité d'information que fournit la fonction d'observation au cours du temps.

Pour illustrer nos idées, prenons un exemple de contrôle de ballon dirigeable dans un espace à une dimension (la hauteur) [Dallaire *et al.*, 2009b]. L'objectif de l'agent contrôleur serait d'apprendre à maintenir l'altitude du ballon à une certaine hauteur sans connaissance à priori de la dynamique sous-jacente du ballon, mais en ayant connaissance toutefois de l'incertitude associée à son altimètre. Le théorème 2 permet alors de prévoir pour différentes valeurs de précision de l'altimètre (donnée par θ et ν), différentes valeurs de l'horizon nécessaire à partir duquel plus aucune information ne sera extractible du processus de filtrage de l'état. Des exemples numériques sont donnés dans la table 4.1 pour le cas général bijectif et dans la table 4.2 pour le cas factorisé.

θ	ν	k	$n \geqslant$
0.6	3	75	40
0.6	10	59	31
0.6	100	50	26
0.7	3	22	13
0.7	10	19	11
0.7	100	14	8
0.8	3	9	6
0.8	10	6	4
0.8	100	6	4

TABLE 4.1 – Modèle bijectivement observable. $\Pr(\boldsymbol{b}^K(s) \geqslant 1-\varepsilon) \geqslant 1-\delta$. $\varepsilon = 10^{-3}$ et $\delta = 10^{-1}$.

θ	μ	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{S}\rvert$	k	$n \geqslant$
0.6	2	10	100	84	44
0.6	3	5	125	98	52
0.6	10	6	10^6	112	60
0.7	2	10	100	30	17
0.7	3	5	125	33	19
0.7	10	6	10^6	39	23
0.8	2	10	100	13	8
0.8	3	5	125	16	10
0.8	10	6	10^6	20	13

TABLE 4.2 – Modèle factorisé. $\Pr(\boldsymbol{b}^K(s) \geqslant 1-\varepsilon) \geqslant 1-\delta$. $\varepsilon = 10^{-3}$ et $\delta = 10^{-1}$.

Comme l'on pouvait s'y attendre, les horizons de planification nécessaires pour la convergence sont plus grands dans le cas factorisé que dans le cas bijectif classique pour des tailles d'espace d'états et d'observations similaires. En effet, l'agent reçoit à chaque étape de temps seulement de l'information sur une partie de l'état. De fait, les observations ne permettent que de discriminer une partie de l'espace d'état à chaque étape plutôt que l'état lui-même comme dans le cas classique. Cependant, vu que le nombre d'observations est exponentiellement moins grand que dans le cas classique, certains algorithmes pourraient avoir moins de difficulté à résoudre ce type de problèmes. Une évaluation empirique pour les deux modèles est donnée dans le cas multiagent à la section 4.3.2.

Sachant la probabilité minimale de l'observation la plus probable pour tous les états et la variance du modèle d'observation, il est possible de prédire à partir de quel moment

l'agent sera localisé avec une précision désirée. Ainsi, il est possible de trouver une politique non stationnaire pour cet horizon fixé à l'aide de l'algorithme POMDP de son choix, puis de se référer à la politique du MDP sous-jacent ensuite. Il est également possible dans le cadre de l'apprentissage, et selon la connaissance actuelle de la fonction de transition et d'observation, de déterminer la longueur optimale des trajectoires à effectuer dans l'environnement. Cet aspect n'a pas été exploité pour le moment, mais serait envisageable dans des travaux futurs pour améliorer l'efficacité d'algorithmes d'apprentissage à base d'épisodes.

Il est également intéressant de remarquer que dès que la probabilité d'observer l'état réel sous-jacent est au-dessus de 80% dans le cas bijectif classique, il est alors suffisant de calculer la politique optimale seulement pour les 4 premières étapes pour avoir un état de croyance quasiment déterministe (à ε près) avec une probabilité supérieure à 90%.

Il peut cependant arriver que l'observabilité soit très faible dans certaines situations causant une convergence lente de l'état de croyance. La figure 4.2 indique la probabilité d'être dans un état de croyance déterministe en fonction du nombre d'étapes effectuées dans l'environnement et de la probabilité minimale θ. À mesure que le nombre d'étapes avance, de l'information est accumulée à propos de l'état sous-jacent pour être presque capable de le déterminer de manière certaine. Comme l'on pouvait s'y attendre, plus θ est faible plus la convergence est lente.

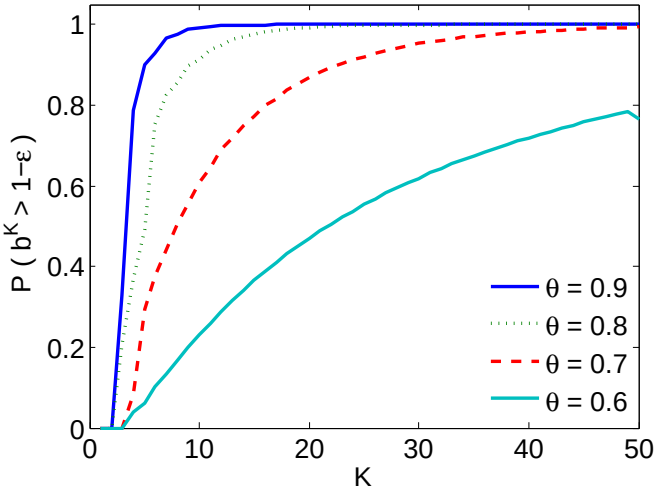


FIGURE 4.2 – Probabilité d'être dans un état de croyance déterministe selon le nombre d'étapes pour différentes probabilités d'observation. $\nu = 100$ et $\varepsilon = 10^{-3}$.

4.2.4 Et si les transitions sont stochastiques ?

Si l'on regarde en particulier l'information contenue dans l'état de croyance en mesurant l'*entropie* (c.f. sous-section suivante) de celui-ci, on peut voir qu'elle converge *en moyenne* vers zero. Néanmoins, lorsque les transitions ne sont pas déterministes, i.e. que de l'incertitude réapparaît à chaque action effectuée, une *entropie résiduelle moyenne* persiste comme démontré par les figures 4.3(a) à 4.3(f). Plus précisément, ces figures montrent l'évolution moyenne de l'entropie sur 10000 séquences d'états de croyance lorsque les transitions sont à peine stochastiques et pour différentes valeurs de la probabilité minimale d'observation θ.

4.2.4.1 Entropie d'un état de croyance

Une mesure couramment utilisée en théorie de l'information de la quantité d'information et de la qualité de l'information contenue dans un distribution de probabilité telle que l'estimation d'un état de croyance [Fox *et al.*, 1998] est l'entropie de Shannon [1948]. L'entropie mesure la quantité d'incertitude contenue dans une distribution de probabilité discrète ou continue. Dans le cas particulier d'un état de croyance sur un ensemble fini d'états, l'entropie H est calculée en utilisant la formule suivante :

$$H(\boldsymbol{b}^t) = -\sum_{s \in \mathcal{S}} \boldsymbol{b}^t(s) \log \boldsymbol{b}^t(s) \tag{4.8}$$

L'entropie est maximale est égale à $\log |\mathcal{S}|$ lorsque l'état de croyance est uniforme -- i.e il est possible d'être dans tous les états de manière équiprobable -- et tend vers zero à mesure que l'état de croyance devient déterministe [3].

Dans la littérature de la théorie de l'information, l'entropie associée à l'estimation de l'état courant a rarement été étudiée, excepté par Rezaeian [2006] qui a appelé cette quantité *entropie d'estimation*. En fait, cette entropie d'estimation, que nous noterons $H(\boldsymbol{b}^t)$, est égale à l'entropie de la distribution sur les états étant donnée une séquence d'observations reçues. Elle se calcule donc de la manière suivante :

$$H(\boldsymbol{b}^t) = H(s^t | o^t, o^{t-1}, \ldots, o^1) = H(s^t | o^t, \boldsymbol{b}^{t-1}) \tag{4.9}$$

Ainsi, en utilisant les règles connues de théorie de l'information, il est possible d'énoncer la proposition suivante :

3. Par convention, $0 \log 0 \equiv 0$.

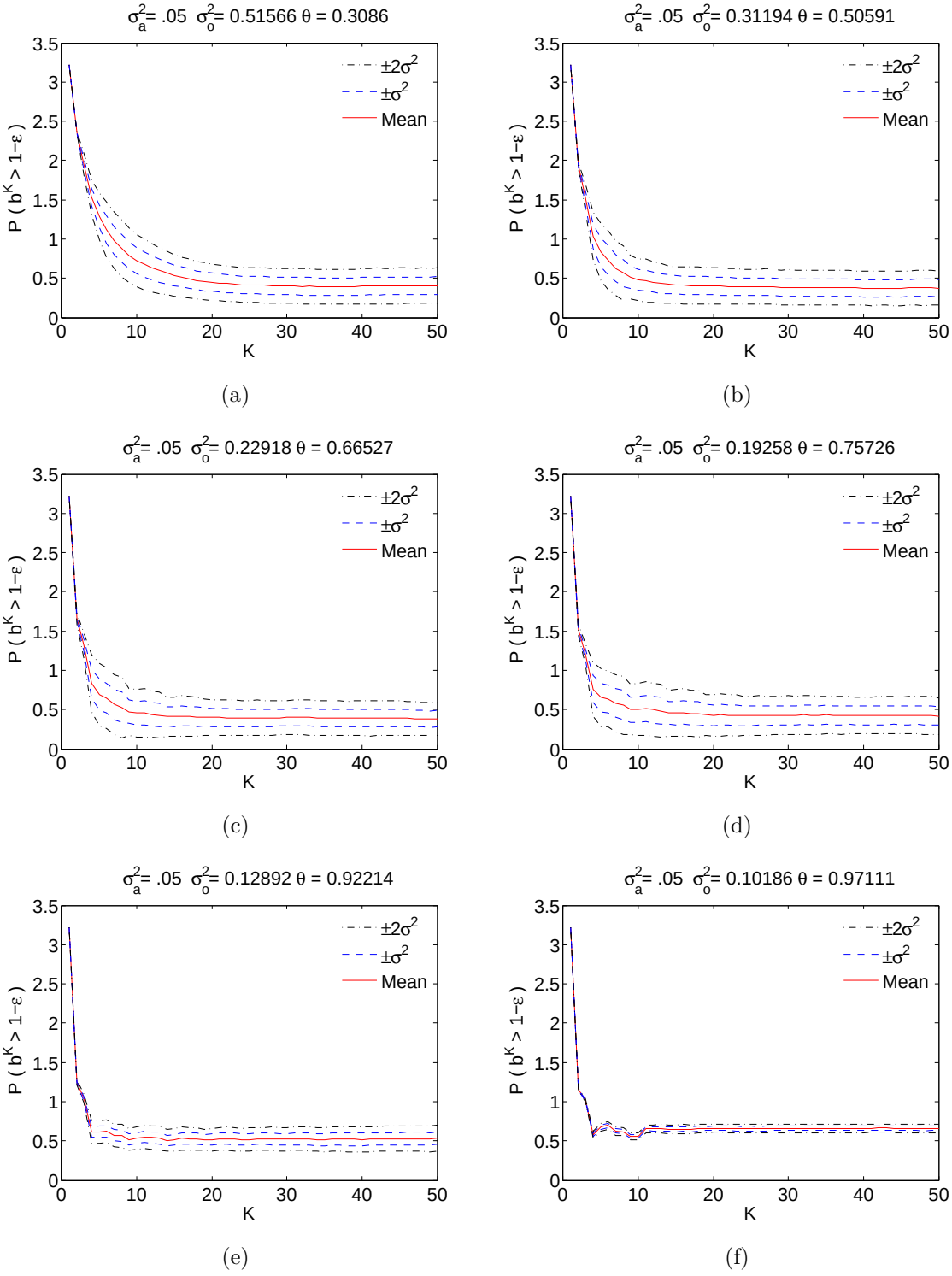


FIGURE 4.3 – Évolution de l'entropie de l'état de croyance au cours du temps pour différentes valeurs de θ et en presence d'incertitude sur les transitions.

Proposition 5. *L'entropie $H(\boldsymbol{b}^t)$ d'un état de croyance à l'étape t depuis un état de croyance $\boldsymbol{b}^{t-1}$ étant donnée n'importe quelle politique est donnée par l'équation suivante :*

$$H(\boldsymbol{b}^t) = H(s^t) - I(s^t; \boldsymbol{b}^{t-1}) - I(o^t; s^t | \boldsymbol{b}^{t-1}) \tag{4.10}$$

Démonstration.

$$\begin{aligned} H(\boldsymbol{b}^t) &= H(o^t|s^t, \boldsymbol{b}^{t-1}) - H(o^t|\boldsymbol{b}^{t-1}) + H(s^t|\boldsymbol{b}^{t-1}) \\ &= -I(o^t; s^t|\boldsymbol{b}^{t-1}) + H(s^t|\boldsymbol{b}^{t-1}) \\ \text{since } I(X;Y) &= H(X) - H(X|Y) \end{aligned}$$

Où $H(s^t)$ est le *taux instantané d'entropie* [Cover et Thomas, 1991] de la chaîne de Markov sous-jacente à la politique qui se construit par la combinaison de la politique et de la fonction de transition. $I(s^t; \boldsymbol{b}^{t-1})$ est l'*information mutuelle* entre l'état s^t et l'état de croyance à l'étape $t-1$ et $I(o^t; s^t|\boldsymbol{b}^{t-1})$ l'*information mutuelle conditionnelle* entre l'état s^t et l'observation o^t étant donné l'état de croyance à l'étape précédente. □

L'information mutuelle mesure l'indépendance mutuelle de deux variables aléatoires ou la quantité d'information que deux variables aléatoires partagent. Elle est donnée par :

$$\begin{aligned} I(X;Y) &= \sum_{y\in Y}\sum_{x\in X} p(x,y) \log\left(\frac{p(x,y)}{p_1(x)\,p_2(y)}\right), \\ &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \\ &= H(X) + H(Y) - H(X,Y) \\ &= H(X,Y) - H(X|Y) - H(Y|X). \end{aligned}$$

L'information mutuelle conditionnelle mesure la même quantité d'information, mais étant donnée une troisième variable aléatoire :

$$I(X;Y|Z) = \mathbb{E}_Z\Big(I(X;Y)|Z\Big) = \sum_{z\in Z}\sum_{y\in Y}\sum_{x\in X} p_Z(z) p_{X,Y|Z}(x,y|z) \log \frac{p_{X,Y|Z}(x,y|z)}{p_{X|Z}(x|z) p_{Y|Z}(y|z)},$$

Qui peut être simplifiée en :

$$I(X;Y|Z) = \sum_{z\in Z}\sum_{y\in Y}\sum_{x\in X} p_{X,Y,Z}(x,y,z) \log \frac{p_Z(z) p_{X,Y,Z}(x,y,z)}{p_{X,Z}(x,z) p_{Y,Z}(y,z)}.$$

De fait, dans le contexte d'information bijective exprimé par la définition 18 où les observations apportent la même quantité d'information quelque soit l'action effectuée,

il serait possible de croire que l'équation (4.10) correspond à l'entropie de l'état de croyance sachant que l'observation o^t a été reçue quelque soit l'action effectuée. En réalité, cette action effectuée a un impact sur la fonction de transition, et ainsi la politique influence fortement l'entropie de l'état de croyance. L'entropie $H(\boldsymbol{b}^t)$ peut donc être interprétée comme l'incertitude ajoutée par la fonction de transition à laquelle on retranche l'information apportée par l'observation ($I(o^t; s^t|\boldsymbol{b}^{t-1})$) et l'information déjà contenue dans l'état de croyance à l'étape précédente ($I(s^t; \boldsymbol{b}^{t-1})$).

Si l'on tente de sommer cette entropie récursive sur K étapes, on obtient :

Corollaire 3. *L'entropie $H(\boldsymbol{b}^K)$ d'un état de croyance après K étapes depuis un état de croyance $\boldsymbol{b}^0$ et étant donnée n'importe quelle politique est donnée par :*

$$H(\boldsymbol{b}^K) = H(s^K) - \sum_{i=1}^{K} \Big(I(s^i; \boldsymbol{b}^{i-1}) + I(o^i; s^i|\boldsymbol{b}^{i-1}) \Big) \tag{4.11}$$

Sous la condition que $H(\boldsymbol{b}^0) = H(s^0)$.

Démonstration. Ce corollaire découle directement de la proposition 5. □

Dans cette formulation, $H(s^t)$ est toujours le taux d'entropie [Cover et Thomas, 1991] de la chaîne de Markov sous-jacente. Ce taux d'entropie converge exponentiellement rapidement sous de petites hypothèses [Hochwald et Jelenkovic, 1999] vers

$$H(s) = \sum_{s \in \mathcal{S}} \sum_{s' \in \mathcal{S}} \mu_s \mathcal{T}(s, \pi(s), s') \log \left[\mathcal{T}(s, \pi(s), s') \right] \tag{4.12}$$

Où μ_s est la distribution stationnaire de la chaîne de Markov construite par la combinaison de la politique et de la fonction de transition.

Toutefois, aucune recherche à notre connaissance ne fait état de l'évolution de ce terme au cours du temps. La majorité des recherches à ce sujet portent essentiellement sur l'étude en *régime d'équilibre*, i.e. lorsque la chaîne de Markov est censée avoir déjà convergé vers sa distribution stationnaire et ne dépend alors plus de la distribution initiale $\boldsymbol{b}^0$. Certains travaux [Subelman, 1976; Lindqvist, 1981; Brosh et Gerchak, 1978] expliquent qu'il existe des conditions sur la chaîne permettant la convergence vers la distribution stationnaire en un nombre d'étapes bornées, mais une adaptation reste à faire quant aux processus décisionnels puisque ces travaux restent sur les versions sans contrôle des chaînes de Markov. Très récemment des travaux de physique quantique semblent également faire état de résultats en régime non stationnaire du taux d'entropie, mais sont restés hermétiques à notre compréhension.

Voyons plutôt expérimentalement quelles sont les conditions sur la qualité des transitions et des observations telles que l'agent soit capable d'extraire l'information exacte à propos de l'état sous-jacent de l'environnement.

4.2.4.2 Expérimentations

Pour tester ces valeurs de convergence de l'entropie, nous avons pris un problème simple de déplacement d'un robot sur une grille torique (une grille où les cellules sur les bords sont adjointes au cellules du bord opposé). Ce robot peut choisir de se déplacer dans les quatre directions cardinales ou de ne pas se déplacer, mais reçoit quand même à chaque étape une observation. Cet agent connaît évidemment les résultats espérés de chacune de ses actions. Il sait également que, dû à certaines conditions environnementales, il peut glisser lors d'un de ses déplacements et se rendre dans une case adjacente à celle souhaitée initialement avec une certaine probabilité. Nous supposons ici que l'action de ne pas se déplacer entraîne également un glissement probable pour des raisons d'uniformité dans la génération de l'entropie au cours du temps.

Nous avons choisi de représenter ces glissements par des distributions gaussiennes discrétisées puisque c'est souvent ainsi que les bruits sont représentés dans la littérature robotique. Ces glissements surgissent donc selon une distribution de probabilité conjointe définie par une gaussienne discrétisée isotropique à deux dimensions paramétrée par une variance σ_τ^2. La figure 4.4 illustre donc la fonction de transition pour passer de l'état du centre de la grille à l'état en dessous par une distribution de probabilité représentant les chances de se retrouver dans les cases adjacentes.

Comme nous l'avons dit ci-dessus, dès que l'agent effectue une action, celui perçoit alors immédiatement une observation de l'environnement (de son système de positionnement global -- GPS -- par exemple). Cette observation est aussi caractérisée par une distribution de probabilité gaussienne discrétisée où la moyenne est exactement l'état dans lequel est arrivé l'agent suite à son action et où la variance est de σ_o^2. Cette distribution de probabilité est donc similaire à celle de la fonction de transition et plus particulièrement lorsque les variances sont égales.

Puisque nous utilisons des distributions gaussiennes discrétisées sur une grille torique dans nos expérimentations, les résultats seront présentés par rapport à la variance des ces distributions. L'erreur relative de l'observation étant donnée la variance de la gaussienne est présentée dans la figure 4.5. Il convient de remarquer que cette erreur devient inférieure au millième lorsque la variance tombe en dessous de 0.3. L'erreur sur les transitions évoluant de manière identique, nous ne la présentons pas ici.

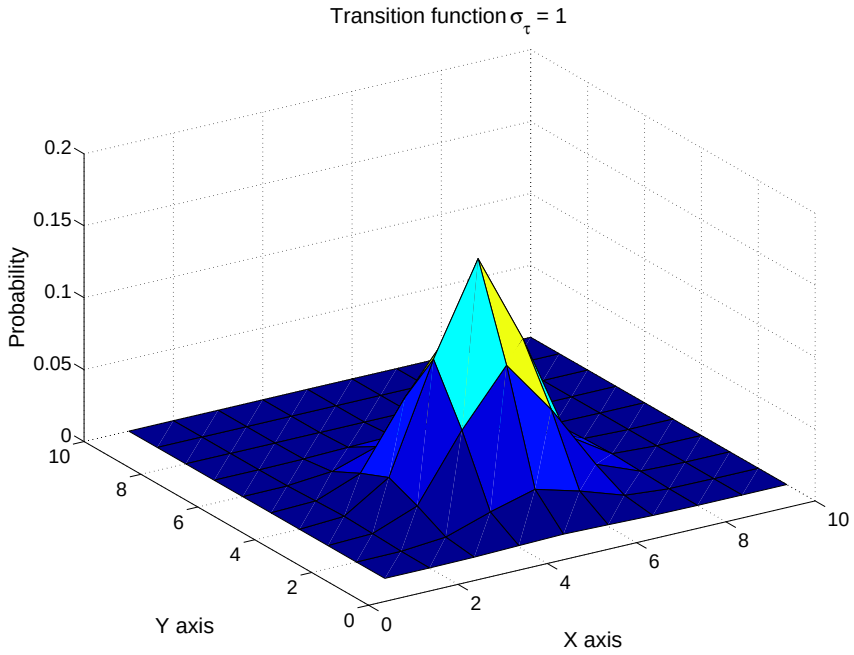


FIGURE 4.4 – Densité de probabilité de la fonction de transition depuis l'état $(5, 5)$ au travers de l'action DOWN. $\sigma_\tau = 1$.

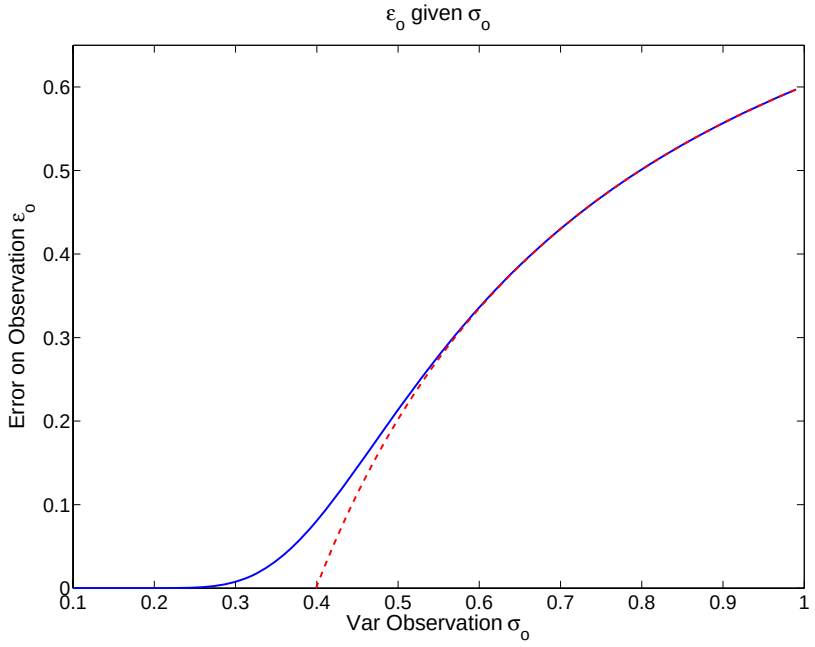


FIGURE 4.5 – Erreur ε_o sur l'observation étant donnée la variance de la gaussienne utilisée pour représenter le bruit. L'erreur est représentée par la ligne pleine alors que la ligne hachée représente $1 - \frac{1}{\sigma_o\sqrt{2\pi}}$.

La première expérimentation que nous avons réalisée porte sur une politique aléatoire et nous calculons l'entropie d'estimation après 3000 étapes pour différentes variances allant de 0 à 3. Nous ne présentons toutefois que les résultats variant de 0.1 à 1, puisqu'en dessous de 0.1, la fonction `normpdf` de Matlab® souffre de problèmes de précision, et au-delà de 0.8, le mode de la distribution devient plus petit que 1/2 et les résultats sont très similaires à ceux pour des variances comprises entre 0.8 et 1.

La figure 4.7 montre le temps minimal de convergence sur 100 trajectoires pour que l'entropie de l'état de croyance converge vers $\varepsilon = 10^{-3}$ sous l'influence d'une politique aléatoire. Cette courbe mets en valeur la fait que, dès que la politique fait en sorte que la chaîne de Markov sous-jacente est ergodique, l'entropie peut prendre un temps exponentiel avant de converge vers ε, si seulement elle converge. En fait, les figures 4.8(a) à 4.8(d) montrent l'entropie à différentes étapes par rapport à la variance sur la transition et sur l'observation. Malheureusement, ces figures montrent également qu'il existe des valeurs de variance sur la transition qui induisent des valeurs de convergence de l'entropie largement supérieures à ε, comme il était possible de s'y attendre.

En fait, la figure 4.6 montre la convergence de l'entropie d'estimation après avoir suivi une politique aléatoire pendant trois milles étapes. Ces résultats montrent qu'après un certain seuil sur l'erreur de transition (sur la figure entre 0.2 et 0.3), l'entropie de l'état de croyance ne peut être réduite plus. Ce seuil survient en fait lorsque la transition n'est plus déterministe puisqu'au-delà de 0.3 l'erreur sur la transition devient supérieure au millième comme nous l'avons vu dans la figure 4.5.

Un autre résultat intéressant survient également lorsqu'une politique déterministe est utilisée plutôt qu'une politique aléatoire (par exemple lorsque le robot cherche à se rendre à une position donnée et à y rester). Les résultats expérimentaux de la figure 4.9 montrent en effet que dans ce cas la convergence est plus rapide pour des valeurs identiques de variances. Par exemple, la figure 4.7 montre qu'il faut au minimum 60 étapes pour converger lorsque la variance sur l'observation est de 0.5 et que l'on suit une politique aléatoire, alors que ce temps est rarement atteint -- même avec une variance de 1 -- lorsqu'une politique déterministe est utilisée. Ce résultat s'explique très simplement par le fait qu'une politique stochastique induit nécessairement un taux d'entropie supplémentaire sur la chaîne de Markov sous-jacente. Il convient alors de n'utiliser que des politiques déterministes lorsque l'on cherche à récolter de l'information sur l'état sous-jacent du système.

Finalement, ces résultats préliminaires ont conduit aux résultats théoriques exprimés un peu plus haut dans ce chapitre. Il serait extrêmement intéressant d'étudier plus avant la théorie de l'information dans certaines chaînes de Markov spécifiques (et non

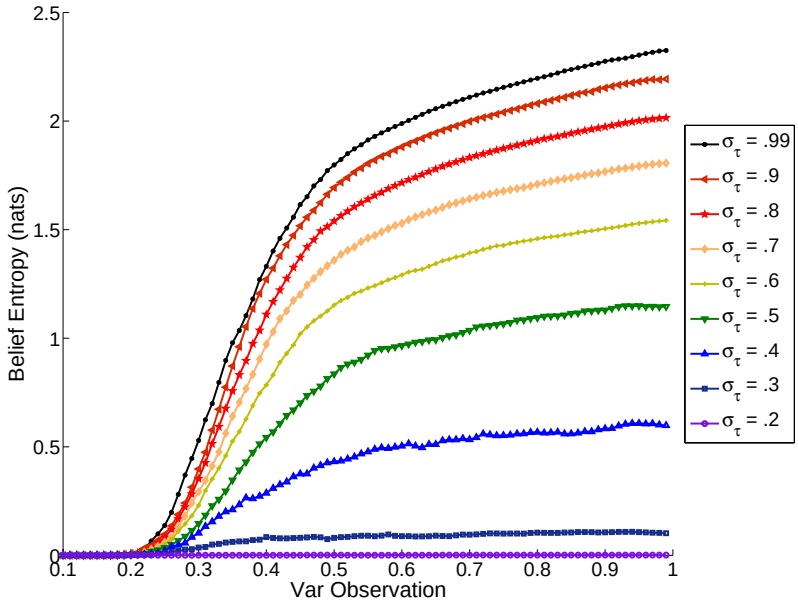


FIGURE 4.6 – Entropie de l'état de croyance $\boldsymbol{b}^{3000}$ par rapport à différentes valeurs de σ_τ et σ_o.

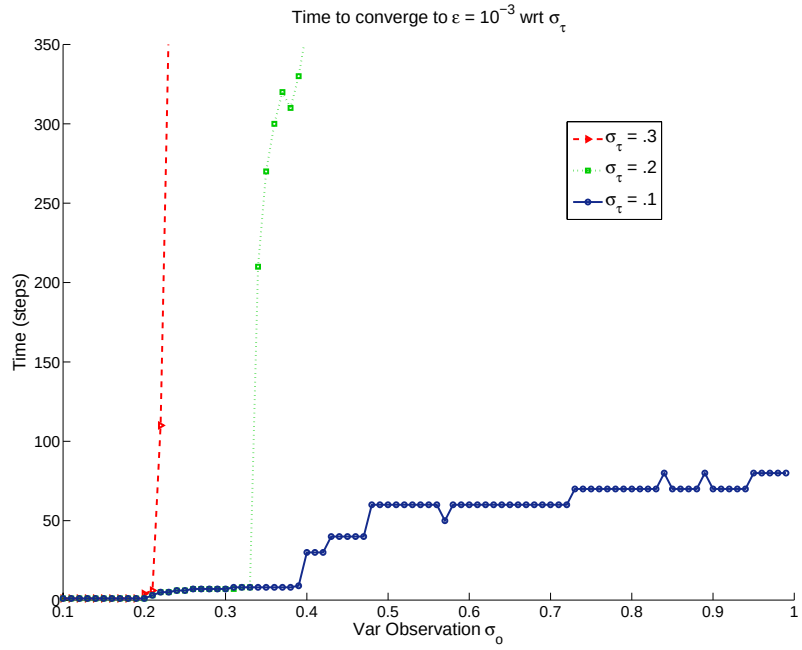
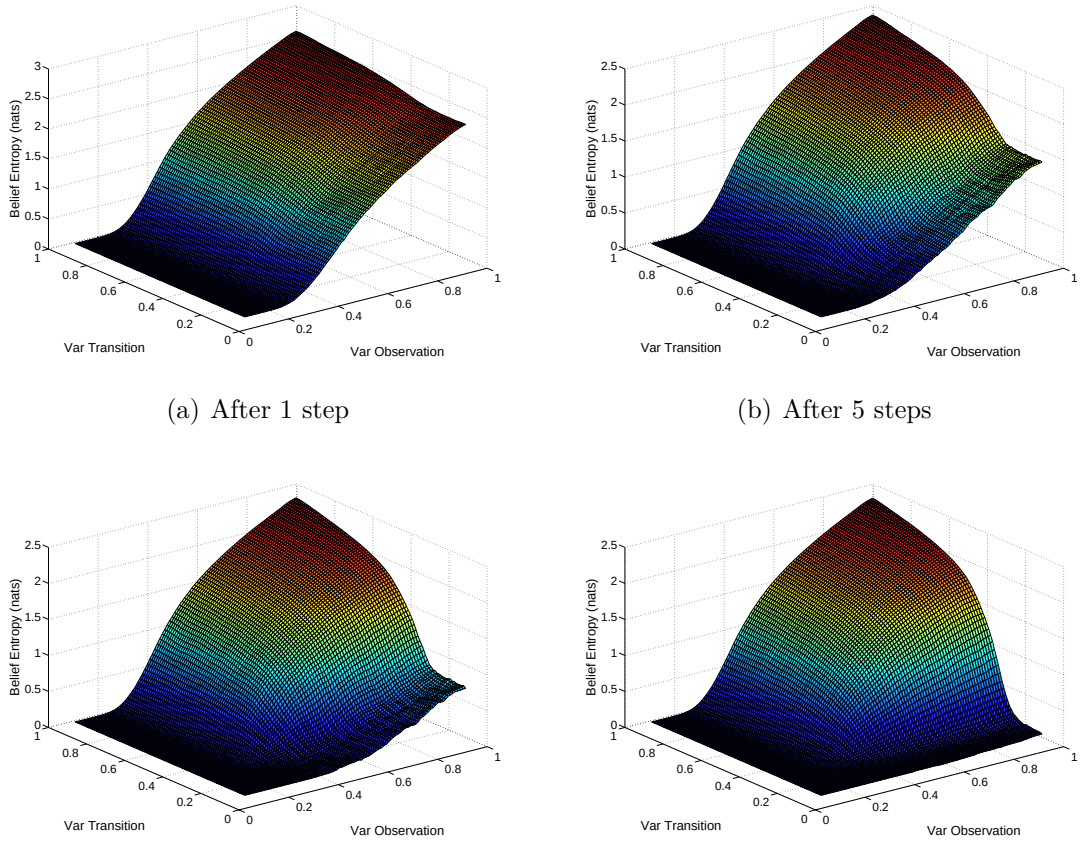


FIGURE 4.7 – Temps minimal de convergence de l'entropie d'estimation avec politique aléatoire pour différentes variances sur la transition σ_τ et différentes variances sur l'observation σ_o.



(a) After 1 step

(b) After 5 steps

(c) After 8 steps

(d) After 50 steps

FIGURE 4.8 – Étude de la variation de l'entropie pour différents valeurs de bruits sur la transition et l'observation.

déterministes) où le taux d'entropie instantané peut-être borné supérieurement, induisant ainsi une convergence assurée de l'entropie d'estimation vers 0.

Voyons maintenant l'extension aux problèmes multiagents de ces modèles à observation quasi-déterministe et à transition déterministe.

4.3 QDET-POMDPs multiagents

L'extension du cas monoagent au cas multiagent se fait de manière directe en considérant un ensemble d'agents α où chaque agent i possède maintenant son ensemble propre d'actions $\mathcal{A}_i$ et d'observations Ω_i et où l'ensemble des actions jointes $\boldsymbol{\mathcal{A}}$ est le produit cartésien des ensembles des actions de tous les agents. Les fonctions de

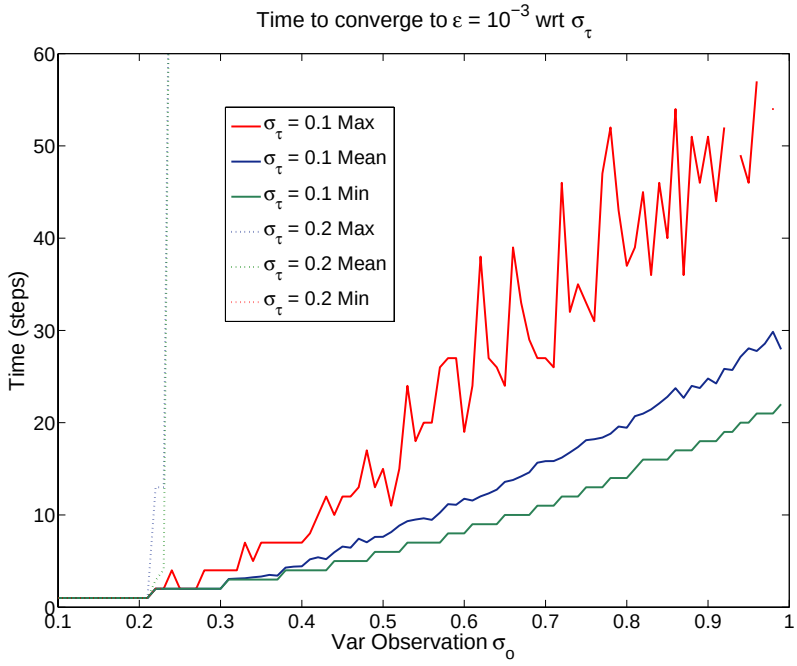


FIGURE 4.9 – Temps de convergence de l'entropie d'estimation pour des variances sur la transition de $\sigma_\tau = 0.1$ et $\sigma_\tau = 0.2$ en fonction de la variance sur l'observation σ_o lorsque l'agent suit une politique déterministe. La moyenne, le minimum et le maximum sont calculés sur 100 simulations.

transitions sont ainsi définies sur l'ensemble des actions jointes et les conditions requises d'observation minimale sont maintenant définies pour toutes les actions jointes $\boldsymbol{a}$ dans $\boldsymbol{\mathcal{A}}$:

Définition 20. *Un Processus Décisionnel de Markov Partiellement Observable Décentralisé Quasi-Déterministe (*QDET-DEC-POMDP*) est un tuple* $\langle \mathcal{S}, \{\mathcal{A}_i\}_{i\in\alpha}, \Omega, \mathcal{T}, \mathcal{O}, \mathcal{R}, \gamma, \boldsymbol{b}^0 \rangle$*, où :*

- $\mathcal{S}$ *est un ensemble fini d'*états $s \in \mathcal{S}$ *;*
- $\mathcal{A}_i$ *est un ensemble fini d'*actions $a \in \mathcal{A}_i$ *pour chaque agent* $i \in \alpha$ *et* $\boldsymbol{a} \in \boldsymbol{\mathcal{A}}$ *denote une action jointe de tous les agents ;*
- $\mathcal{T}(s, \boldsymbol{a}, s') : \mathcal{S} \times \boldsymbol{\mathcal{A}} \times \mathcal{S} \mapsto \{0, 1\}$ *est la* fonction déterministe de transition *du système de l'état* s *vers l'état* s' *après l'exécution de l'action jointe* $\boldsymbol{a}$ *;*
- $\mathcal{R}(s, \boldsymbol{a}) : \mathcal{S} \times \boldsymbol{\mathcal{A}} \mapsto \mathbb{R}$ *est la* récompense *produite par le système lorsque l'action jointe* $\boldsymbol{a}$ *est exécutée dans l'état* s*,*
- Ω_i *est un ensemble fini d'observations* $o \in \Omega_i$ *pour chaque agent* i *et* $\boldsymbol{o} \in \boldsymbol{\Omega}$ *denote une observation jointe de tous les agents,*
- $\mathcal{O}(\boldsymbol{z}, \boldsymbol{a}, s') : \boldsymbol{\Omega} \times \boldsymbol{\mathcal{A}} \times \mathcal{S} \mapsto [0, 1]$ *est une* fonction d'observation *qui indique la probabilité d'obtenir l'observation jointe* $\boldsymbol{z}$ *lorsque le monde arrive dans l'état* s' *après avoir exécuté l'action jointe* $\boldsymbol{a}$ *;*

De plus, $\forall\, s' \in \mathcal{S}$, $\boldsymbol{a} \in \boldsymbol{\mathcal{A}}$, $\exists\, \boldsymbol{z} \in \boldsymbol{\Omega}$, *s.t.* $\mathcal{O}(\boldsymbol{z}, \boldsymbol{a}, s') \geqslant \theta > \frac{1}{2}$, *i.e. le monde est minimalement observable et la probabilité d'obtenir une des observations jointe est bornée inférieurement par un demi ;*
- γ *est le facteur d'escompte ;*
- $\boldsymbol{b}^0$ *est la connaissance* a priori *sur l'état, i.e. l'état de croyance initial, supposé non déterministe.*

On peut également étendre la condition d'observation bijective à plusieurs agents :

Définition 21. *Un* qDet-dec-pomdp *bijectivement observable est un* qDet-dec-pomdp *où l'hypothèse suivante est faite :*

$$\begin{aligned}
&\forall i \in \alpha, \exists o_1 \in \Omega_i, \forall \boldsymbol{a} \in \boldsymbol{\mathcal{A}}, \forall s \in \mathcal{S}^{o_1},\\
&\textit{avec } \mathcal{S}^{o_1} = \{s \in \mathcal{S} | \exists o_1 \in \Omega_i, P(o_1|s, \boldsymbol{a}) > P(o|s, \boldsymbol{a}), \forall o \neq o_1\},\\
&\textit{alors } |\Omega_i| = |\mathcal{S}| \textit{ et } |\mathcal{S}^{o_1}| = 1
\end{aligned}$$

En d'autres termes, chacun des agents possède une observation bijective de l'état *complet* du système. Les agents perçoivent ainsi le même état même si ce n'est pas nécessairement la même observation qui est reçue à chaque étape par tous les agents. Cette hypothèse est relativement plus forte que dans le cas monoagent pour deux raisons principales :
- Les agents peuvent n'avoir qu'une vue locale de leur environnement et donc ne pas percevoir ce que perçoivent d'autres agents vis a vis de l'état du système ;
- Les agents peuvent avoir un état *interne* qui leur est propre et que personne d'autre qu'eux ne peut observer, mais qui fait nécessairement partie de l'état global pour une prise de décision optimale.

En regard de ces deux points, il n'est pas interdit de regarder vers la communication pour résoudre ces deux problèmes. La communication permet aux agents de fusionner l'information qu'ils ont sur le monde et de partager leur état interne. Bien qu'elle ne soit pas parfaite, on peut néanmoins assumer qu'elle l'est quasiment au regard des techniques actuelles de correction d'erreur. On peut également faire l'hypothèse que dans la majorité des problèmes, la communication est beaucoup plus rapide que la fréquence des instants de décision. Ces points seront plus amplement discutés dans la section 4.4.

4.3.1 Résultats théoriques

Les résultats théoriques valides dans le cas monoagent, peuvent s'étendre aisément au cas multiagent et permettre, dans ce contexte, des gains en complexité en pire cas encore

plus intéressants. En effet, la où un gain relatif en espace se faisait par la limitation de l'historique à maintenir pour être certain de l'état sous-jacent du système, il est maintenant possible non seulement de borner son propre historique à mémoriser, mais également l'historique de tous les autres agents du système.

Pour rappel, les DEC-POMDPs sont réputés pour être particulièrement difficiles à résoudre optimalement dans le cas à horizon fini (NEXP-complet [Bernstein *et al.*, 2000]) et même à approximer [Rabinovich *et al.*, 2003]. En restreignant la modélisation aux problèmes quasi-déterministes avec observabilité bijective, il est possible de réduire de beaucoup cette complexité en pire cas et on peut donc énoncer le corollaire suivant :

Corollaire 4. *Trouver une politique pour un* QDET-DEC-POMDP *à horizon infini, sous l'hypothèse d'observabilité bijective, qui produit une récompense escomptée espérée d'au moins C avec probabilité $1-\delta$, est* PSPACE.

Démonstration. Pour montrer que ce problème est PSPACE, l'algorithme suivant donne la politique ε-optimale en espace polynômial : Construire l'arbre de tous les historiques possibles à horizon k (possible en espace $\mathcal{O}((|\boldsymbol{\mathcal{A}}||\boldsymbol{\Omega}|)^k)$), et calculer ensuite pour chaque feuille de l'arbre construit la valeur espérée escomptée de l'état de croyance quasi-déterministe atteint en utilisant la valeur de la politique optimale du MMDP sous-jacent. Rétropropager finalement la valeur jusqu'à la racine pour vérifier que la récompense C est effectivement obtenue. □

En d'autre termes, trouver une politique optimale pour un QDET-DEC-POMDP à horizon infini est aussi complexe en pire cas que de résoudre un POMDP à horizon fini.

Voyons maintenant un exemple permettant d'illustrer l'utilisation de ce modèle avant de discuter plus avant les hypothèses et les applications d'un tel modèle.

4.3.2 Résultats expérimentaux : application à la lutte contre l'incendie

Comme le suggère la preuve des corollaires 2 et 4, un algorithme pour résoudre ε-optimalement un problème QDET-DEC-POMDP à horizon infini sous l'hypothèse d'observabilité bijective consiste à calculer une politique escomptée classique pour le problème à horizon fini à k étapes et à utiliser ensuite le MMDP sous-jacent pour terminer l'exécution de la tâche. Un tel algorithme serait la Programmation Dynamique [Hansen *et al.*, 2004] vue au chapitre 2 par exemple. En pratique cependant, résoudre un DEC-POMDP

à horizon k -- même quasi-déterministe -- devient extrêmement complexe dès lors que $k > 2$ et une approximation devient nécessaire. Les résultats présentés ci-après sont donc réalisés à partir d'un algorithme d'approximation sur un problème de chaîne de seaux d'eau (cf figure 4.10).

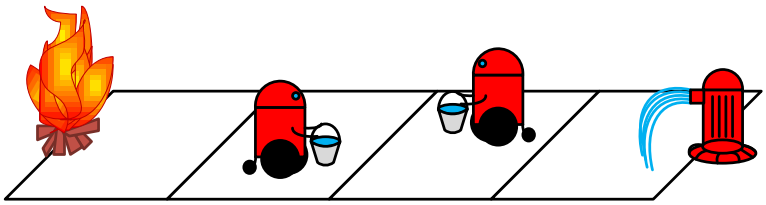


FIGURE 4.10 – Le problème de la chaîne de seaux.

La version générale du problème de chaîne de seaux d'eau est comme suit : deux agents sont situés sur une grille linéaire et portent chacun un seau. Ils peuvent aller à *droite* ou à *gauche*, ou encore *jeter de l'eau*, chaque action infligeant une petite pénalité de retard (-0.1 par agent). Dès qu'un agent se trouve sur la case la plus à droite, le seau se remplit automatiquement avec une probabilité φ. Chaque action de déplacement a également une probabilité φ de réussir, l'action consistant à jeter de l'eau réussissant systématiquement. Les agents peuvent s'échanger leurs seaux au travers de la même action de jet d'eau. À chaque fois qu'un seau est vidé dans la case la plus à gauche par le biais d'un jet d'eau, une récompense est obtenue (1 par agent). Les agents sont initialement positionnés aléatoirement sans eau. Les agents sont supposés avoir une observation bruitée de leur position, i.e. ils ont une probabilité θ d'observer leur position réelle et une probabilité $(1 - \theta)/2$ d'observer une des cases adjacentes. Ils reçoivent également une communication bruitée de la position de l'autre agent et de l'état de son seau. Puisque le problème peut-être un problème à horizon infini, un facteur d'escompte $\gamma = 0.95$ a été utilisé. Dans les expérimentations présentées ci-après, une valeur de $\varphi = 1$ a été aussi utilisée pour l'hypothèse de transition déterministe. Lorsque $\varphi = 1$ le nombre d'états accessibles est de 49 (7 pour chaque agent) et donc 49 observations peuvent être obtenues. Dans le cas bijectif factorisé, le nombre d'observations est seulement de 10 puisque chaque agent a 4 positions plus l'état du seau de l'autre agent.

Pour ce problème, nous avons utilisé une version adaptée de l'algorithme IMBDP [Seuken et Zilberstein, 2007a] présentée à la section 2.2.2.4 de telle sorte qu'il utilise la valeur optimale du MMDP sous-jacent et inclut également le facteur d'escompte. Cet algorithme est dénoté par IMBDP^i dans les figures.

Plusieurs simulations ont été réalisées utilisant différents paramètres. La figure 4.11 montre la valeur espérée escomptée de IMBDP^i sur le problème bijectif classique pour plusieurs valeurs de θ allant de 0.6 à 0.95. Les meilleurs paramètres trouvés par validation croisée de IMBDP^i sont $maxTree = 5$ et $maxObs = 1$. Augmenter ces paramètres

n'améliore pas significativement la valeur espérée escomptée, mais détériore significativement les performances spatiales et temporelles de l'algorithme. Comme l'on pouvait le prévoir, à mesure que θ approche de un, l'horizon de planification nécessaire diminue jusqu'à deux et la valeur de la politique à horizon infini s'approche de la valeur de la politique optimale du MMDP sous-jacent.

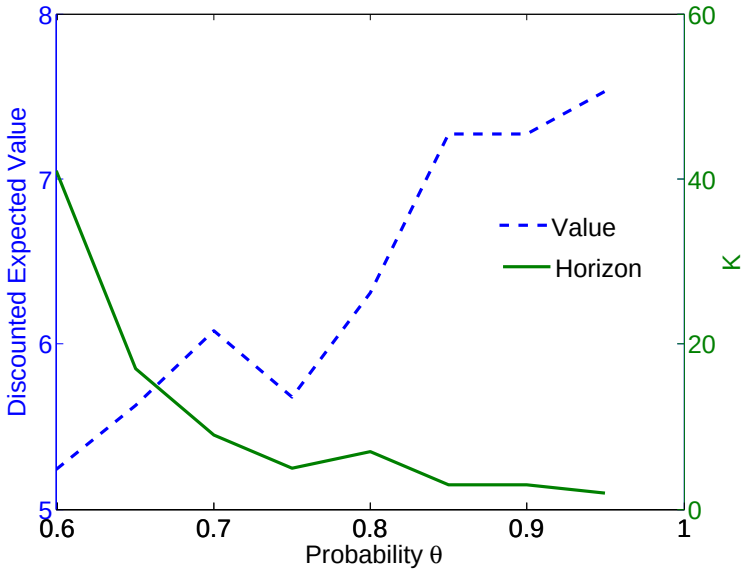


FIGURE 4.11 – Valeur espérée escomptée et horizon espéré pour différentes valeurs de θ.

La figure 4.12 montre les valeurs espérées escomptées pour $\theta = 0.8$ sur plusieurs valeurs de l'horizon allant de 3 à 101 montrant que l'algorithme tend vers la valeur à horizon infinie trouvée précédemment. L'algorithme IMBDP standard a été utilisé avec les mêmes paramètres que ci-dessus ($maxTree = 5$ et $maxObs = 1$).

La figure 4.13 montre le temps de calcul nécessaire à l'approximation de la politique à horizon infini en utilisant les mêmes paramètres pour l'algorithme IMBDP. Comme espéré, le temps diminue rapidement à mesure que θ croît puisque l'horizon nécessaire de planification diminue également.

Les figure 4.14 à 4.16 comparent respectivement la valeur espérée escomptée obtenue, l'horizon nécessaire de planification et le temps de calcul de l'algorithme IMBDP^i entre le cas d'observabilité bijective classique et le cas factorisé. Comme l'on pouvait s'y attendre, le cas factorisé nécessite un horizon de planification supérieur puisque la borne est plus lâche du fait que chaque agent reçoit moins d'information à chaque étape. Cependant, le temps de calcul est lui bien inférieur (d'un ordre de grandeur) pour algorithme IMBDP^i puisque le nombre d'observations à considérer est beaucoup plus petit (de 49 à 10). Du côté de la valeur, les deux modèles étudiés se comportent de manière similaire avec un léger avantage pour le cas classique encore dû au fait que l'information sur l'état est

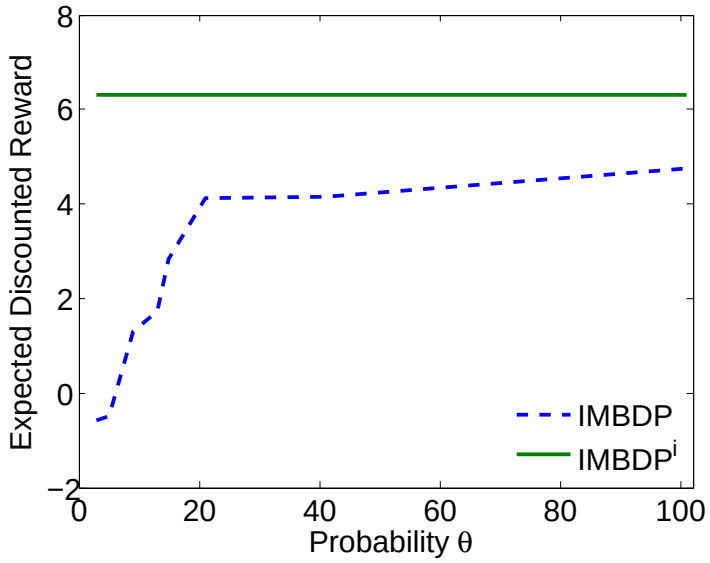


FIGURE 4.12 – Valeur espérée escomptée à horizon fini pour différentes valeurs d'horizons.

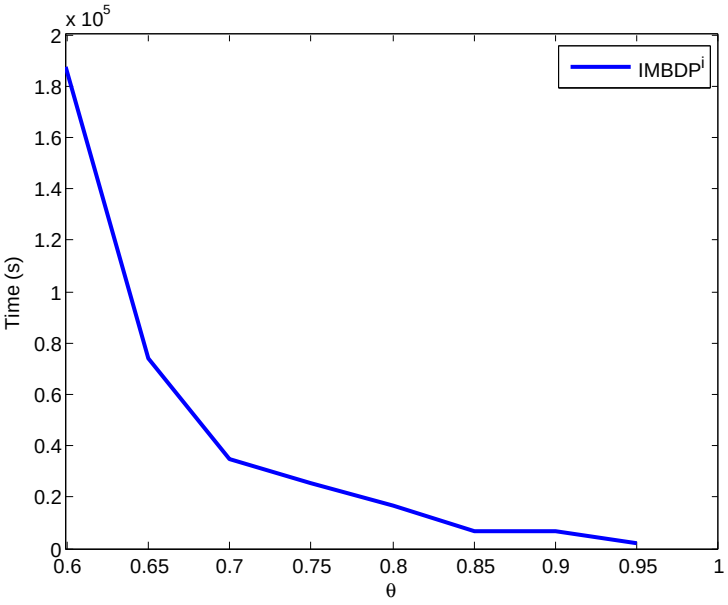


FIGURE 4.13 – Temps de calcul pour différentes valeurs de θ.

disponible plus rapidement dans ce cas-ci.

Discutons maintenant de l'approche générale, des hypothèses utilisées, des résultats obtenus ainsi que des différents travaux futurs avant de conclure.

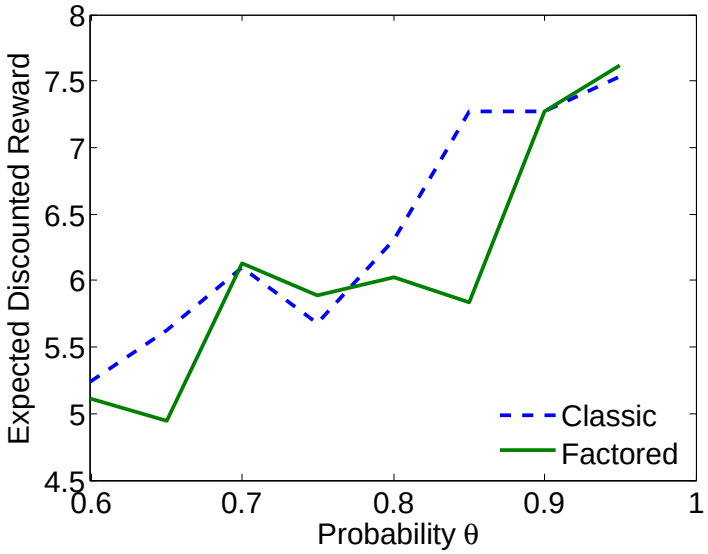


FIGURE 4.14 – Comparaison en valeur des modèles classique et factorisé.

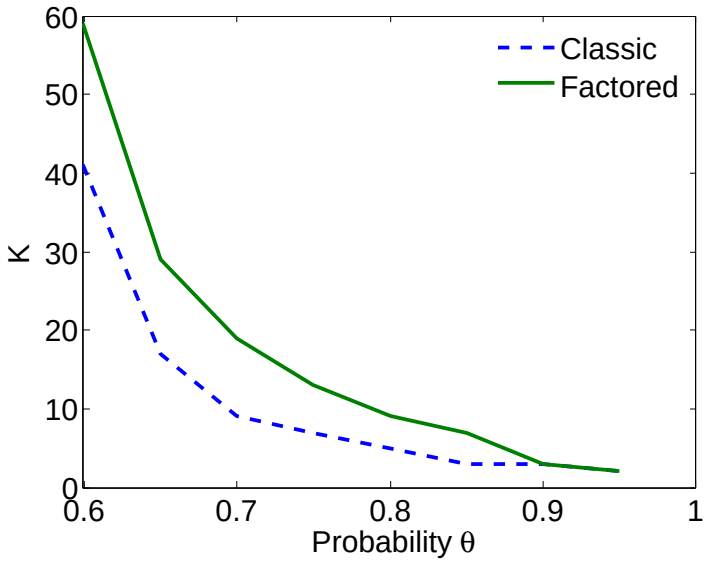


FIGURE 4.15 – Comparaison en horizon des modèles classique et factorisé.

4.4 Discussion et Conclusion

La première chose qu'il convient de remarquer concerne les hypothèses faites au travers de ce chapitre. Il peut en effet paraître curieux d'étudier les systèmes de Markov à transition déterministe lorsque l'on sait qu'historiquement ces modèles étaient introduits avant tout pour étudier les comportements asymptotiques de systèmes dynamiques stochastiques. Cependant, au vu des récentes avancées algorithmiques pour ces modèles

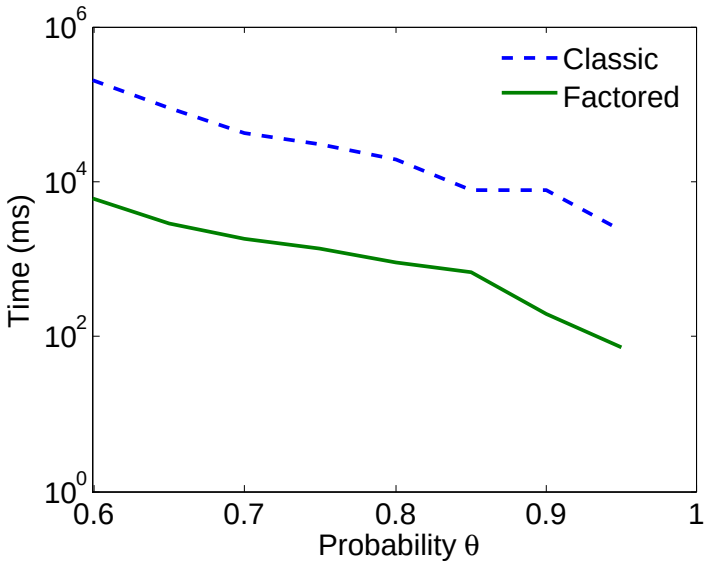


FIGURE 4.16 – Comparaison en temps de calcul des modèles classique et factorisé.

il peut être très intéressant de s'approprier ces algorithmes dans des cas déterministes spécifiques où l'observabilité peut faire défaut. Il convient donc d'équilibrer plusieurs choses :

- Le choix des algorithmes stochastiques pour des problèmes partiellement observables purement déterministes n'est pas le plus approprié selon Bonnet [2009]. L'auteur suggère d'employer plutôt des algorithmes pour les arbres de recherche ET/OU [Marinescu et Dechter, 2009]. En revanche, dans le cas où l'observation est stochastique mais la transition déterministe, les résultats de ce chapitre montrent qu'il n'est nécessaire de considérer que les K dernières observations pour ensuite pouvoir utiliser la politique du MDP sous-jacent comme politique ε-optimale.
- Le choix du modèle d'observation, surtout dans le cas multiagent, influence l'équilibre des performances temps/valeurs espérées de l'algorithme considéré. Il convient donc de savoir si, dans le cas où l'observation peut être factorisée, il est préférable de ne considérer que certaines parties de l'observation à chaque étape, ou s'il faut considérer l'observation complète. Dans le premier cas, l'algorithme bénéficie d'un avantage certain dans la complexité en temps de calcul, mais produit des solutions à valeur moindre, alors que dans le second cas, les temps de calcul peuvent s'avérer prohibitifs pour obtenir un gain en valeur possiblement avantageux.

La seconde chose à remarquer concerne l'hypothèse de l'observabilité bijective. Il est possible en effet de trouver cette hypothèse très restrictive dans certains contextes. En robotique toutefois, la factorisation du domaine et des observations est inhérente aux capteurs attachés au robot et cette formulation devient alors naturelle lorsque l'on observe des positions GPS (pour *Global Positioning System*) bruitées ou des caractéristiques

spécifiques de l'environnement à l'aide de capteurs plus ou moins fiables. Cette étude permet ainsi de sélectionner dans certains cas les capteurs adaptés au robot souhaité maximisant ainsi l'équilibre entre la précision voulue des capteurs selon la capacité de calcul du robot, et le prix de ces capteurs, sous l'hypothèse que des capteurs plus fiables coûtent plus cher.

Dans le cas multiagent, cette hypothèse de bijectivité n'est pas aussi restrictive qu'on pourrait le croire. Même si chaque agent perçoit effectivement l'état *complet* du monde avec une certaine probabilité, chaque agent ne reçoit pas nécessairement la même observation que les autres agents et leurs croyances sur l'état réel sous-jacent du système peuvent-être différentes. D'un côté, cette hypothèse semble donc plus difficilement applicable dans le cadre décentralisé (DEC-POMDP) que dans le cadre centralisé (POMDP) puisque les valeurs de variables internes aux agents ne sont alors plus obligatoirement observables. D'un autre côté, ces variables internes peuvent éventuellement être transmises à autrui sans nécessairement communiquer leur observation complète, et l'incertitude sur cette communication peut être alors incluse dans l'incertitude associée à l'observabilité de ces variables.

Dans la littérature, cette hypothèse d'observabilité bijective s'approche beaucoup des travaux effectués par Goldman et Zilberstein [2004] sur les DEC-MDPs où les agents, lorsqu'ils communiquent leurs observations, ont accès à l'état réel sous-jacent du système. La différence avec ces travaux provient du fait que chacun des agents ne perçoit pas sa propre partie de l'état avec certitude, fournissant ainsi à l'ensemble du système l'observabilité complète au travers de la communication. L'observabilité bijective assure uniquement que chaque agent *peut éventuellement* observer l'état réel du système avec probabilité θ. En d'autres termes, un DEC-MDP avec communication complète est un DEC-POMDP avec observabilité bijective où $\theta = 1$, qui est aussi un MMDP.

En termes de travaux futurs, deux voies restent à être explorées. Une première concerne l'extension de ce modèle aux problèmes avec transition quasi-déterministe et le contrôle dans ce domaine. Des travaux non publiés présentés dans la section 4.2.4 utilisant des notions de théorie de l'information montrent qu'il est impossible de montrer la convergence de l'état de croyance vers une entropie nulle comme l'on laissé entendre les figures 4.3(a) à 4.3(f). Il est toutefois envisageable, comme le montre les travaux de Hsu *et al.* [2007] sur l'approximabilité des POMDPs, de trouver des classes de fonctions de transitions particulières forçant la convergence des états de croyance vers un ensemble fini d'états de croyance sur lesquels des algorithmes d'itération de valeur seraient applicables par exemple. La seconde avenue consiste à considérer un problème réel permettant de mettre en œuvre ce modèle et des travaux ont été commencés en ce sens avec Jean-Samuel Marier sur un système de patrouille d'UAVs (pour *Unmanned Aerial*

Vehicles) [Marier *et al.*, 2009, 2010] ou plusieurs agents doivent surveiller un ensemble de positions stratégiques régulièrement et s'adapter en cas de changement des positions ou en cas de dysfonctionnement des UAVs.

Dans le chapitre suivant nous allons donc nous concentrer sur les aspects distribués de la planification en ligne et particulièrement sur l'état de croyance lorsque les agents ne communiquent pas nécessairement.

Chapitre 5

Observabilité et communication dans les processus de Markov décentralisés

«Coopérer, c'est résoudre à deux des problèmes qu'on n'aurait jamais eus tout seul.»

adapté de Sacha Guitry

*Ce chapitre introduit tout d'abord la résolution en ligne de processus décisionnels de Markov décentralisés (*DEC-POMDP*s) puis présente un algorithme approximatif et à utilisation de mémoire constante dans le cas où aucune communication n'est autorisée. L'algorithme est ensuite évalué empiriquement sur différents problèmes de la littérature et ses résultats sont comparés à des adaptations en ligne des meilleurs algorithmes disponibles. Une discussion de ces résultats est également effectuée ainsi que les différentes extensions possibles autour de la communication.*

5.1 Introduction à la résolution en ligne de DEC-POMDPs

Comme l'a souligné le chapitre précédent, l'observabilité est un facteur clé de la complexité d'un problème de planification multiagent. Dès lors que l'observabilité n'est

plus bijective en regard de l'état, le problème devient alors extrêmement complexe (NEXP-complet) et l'on doit alors faire appel à des algorithmes d'approximation dépourvus de garanties, l'approximabilité de ce type de problème ayant été montrée au moins tout aussi complexe (NEXP-difficile) [Rabinovich *et al.*, 2003].

Parmi ces algorithmes d'approximation, plusieurs ont été présentés au chapitre 2 de cette thèse, et ils s'avèrent être tous une résolution *hors ligne* du problème de planification markovienne multiagent décentralisé. Le *hors ligne* signifie précisément que toute la planification est réalisée *avant* l'exécution et qu'aucune planification n'est effectuée une fois que l'exécution a commencé. Par opposition, *en ligne* signifie que l'agent peut planifier entre deux exécutions d'actions dans l'environnement et peut donc utiliser l'information récoltée pendant l'exécution pour améliorer sa planification. Les avantages de la planification en ligne sont doubles : il n'est pas nécessaire de planifier pour l'ensemble des états de croyance possible (soit le simplexe au complet), et l'intégration de la connaissance acquise au cours du temps se fait naturellement à chaque re-planification à partir de l'état de croyance courant. Très répandus dans le contexte monoagent POMDP (avec RTBSS [Paquet *et al.*, 2006], AEMS [Ross et Chaib-draa, 2007; Stephane Ross et Paquet, 2008]), le problème de la planification en ligne en multiagent et dans le cas partiellement observable provient de la connaissance décentralisée de l'état et du fait que les agents ne partagent justement pas la même croyance sur l'état courant selon leurs observations respectives.

Ce chapitre présente donc un algorithme décentralisé de planification en ligne pour les DEC-POMDPs. Dans un premier temps, l'algorithme dont s'inspire notre propre algorithme sera présenté. Il s'agit de l'algorithme de *Rollout* de Bertsekas *et al.* [1997] qui a été pensé et réalisé pour le cas monoagent complètement observable (MDP). L'extension au cas multiagent partiellement observable sera ensuite présentée et évaluée empiriquement avant de conclure. L'algorithme de *Rollout* est maintenant détaillé.

5.2 Algorithme à base de "rollouts"

Le principe de base de l'algorithme de Rollout, tel que présenté par Bertsekas *et al.* [1997], consiste à faire de l'amélioration incrémentale en ligne de politique. Cette méthode nécessite une politique π qu'on améliore ensuite successivement à l'aide de trajectoire Monte-Carlo par exemple. Pour cela, à chaque étape de temps, le Rollout considère l'état courant et calcule l'ensemble des états suivants possibles à partir des actions disponibles. Puis, il approxime la valeur espérée de la politique π dans chaque état suivant généré à l'aide de simulations Monte-Carlo, et cette évaluation est ensuite utilisée

pour sélectionner la meilleure action disponible qui peut éventuellement être différente de celle suggérée initialement par la politique π. L'action sélectionnée se trouve ainsi être celle ayant la plus grande valeur espérée (toujours selon l'estimation Monte-Carlo) dans l'état courant. Il est en fait possible de montrer, dans le cas monoagent et sous certaines conditions légères, que la politique effectivement exécutée produit une valeur espérée au moins aussi élevée que la politique π initiale, à moins que la politique π soit optimale, auquel cas, le Rollout se comporte alors aussi optimalement.

Dans ce contexte, Chang *et al.* [2004] ont proposé une extension de cet algorithme au cas monoagent partiellement observable (POMDP) en utilisant les états de croyance plutôt que les états directement. Ainsi, dans le cas des POMDPs, le Rollout calcule, à partir de l'état de croyance courant de l'agent, l'ensemble des états de croyance possibles selon les actions disponibles et les observations résultantes possibles. Une estimation de la valeur de ces états de croyance est également évaluée à partir de simulation Monte-Carlo non plus d'une seule politique initiale π mais d'un ensemble de politiques de base Π. La meilleure action -- correspondant à la meilleure valeur espérée -- est ensuite retournée. L'algorithme 10 donne le pseudo-code du cas partiellement observable.

Algorithm 10 Rollout pour POMDP

requiert: Un ensemble d'heuristiques $\Pi = \langle \pi_1, \ldots, \pi_m \rangle$, un état de croyance $\boldsymbol{b}^0$ et un horizon T.

retourne: Une action que l'agent doit exécuter dans l'état de croyance $\boldsymbol{b}^0$.

1 **fonction** ROLLOUT(Π,$\boldsymbol{b}^0$,T)
2 **pour tout** $a \in \mathcal{A}$, $o \in \mathcal{O}$ **faire**
3 $\{\boldsymbol{b}^1\}_{a,o} \leftarrow \textsc{Update}(\boldsymbol{b}^0, a, o)$ ▷ Mise à jour de l'état de croyance (Eq. 2.7)
4 **pour tout** $\boldsymbol{b}^1 \in \{\boldsymbol{b}^1\}_{a,o}$ **faire**
5 $\Upsilon \leftarrow \textsc{smce}(\Pi, \boldsymbol{b}^1, T)$ ▷ Valeur estimée des simulations Monte-Carlo
6 **fin pour**
7 **fin pour**
8 $a^\star \leftarrow \arg\max_{a \in \mathcal{A}} \sum_{o \in \mathcal{O}} \Pr(o|\boldsymbol{b}^0, a) \max_{\pi \in \Pi} \Upsilon[\pi]$
9 **retourner** $a^\star$
10 **fin fonction**

L'estimation de la valeur des politiques par simulation Monte-Carlo (à la ligne 5 de l'algorithme 10) notée $\textsc{smce}(\Pi, \boldsymbol{b}^1, T)$ sera plus amplement expliquée à la section 5.3.1. Voyons maintenant l'extension au cas multiagent.

5.3 Rollout décentralisé (dRollout)

L'approche proposée dans ce chapitre est similaire à celle de Bertsekas *et al.* [1997] et de Chang *et al.* [2004] dans l'utilisation et l'amélioration d'heuristiques. Une estimée de la valeur espérée de chaque politique donnée est calculée pour chaque action que peut réaliser un agent en se basant sur des méthodes Monte-Carlo séquentielles. L'approche utilise ainsi une quantité fixe de temps et de mémoire à chaque itération d'amélioration. Plus formellement, étant donné un ensemble d'heuristiques multiagent Π selon un état de croyance multiagent, la politique associée au Rollout sélectionne l'action selon l'équation suivante :

$$\boldsymbol{a} = \arg\max_{\boldsymbol{a}\in\boldsymbol{\mathcal{A}}} \max_{\pi\in\Pi} \mathrm{E}\left[\sum_{t=0}^{T} \mathcal{R}(\boldsymbol{b}^t, \pi(\boldsymbol{b}^t))\right]$$

Où $\mathcal{R}(\boldsymbol{b}^t, \boldsymbol{a})$ est la récompense espérée selon l'état de croyance $\boldsymbol{b}^t$:

$$\mathcal{R}(\boldsymbol{b}^t, \boldsymbol{a}) = \mathop{\mathrm{E}}_{s\in\mathcal{S}}[\mathcal{R}(s, \boldsymbol{a})] = \sum_{s\in\mathcal{S}} \boldsymbol{b}^t(s)\mathcal{R}(s, \boldsymbol{a})$$

Un des problèmes de l'algorithme de Rollout provient de sa nécessité d'avoir des *bornes inférieures* sur la fonction de valeur optimale pour garantir que le choix de l'action sera bien meilleur. De fait, toute politique sous-optimale permettra de diriger la recherche de l'algorithme Rollout de manière correcte. C'est pour cela, qu'en sus des bornes disponibles dans la littérature, rappelées à la section 2.2.2.3 du chapitre 2, nous avons choisi aussi d'utiliser l'algorithme MBDP et ses dérivés de la section 2.2.2.4 du chapitre 2 et une version en ligne de la programmation dynamique (section 2.2.2.3 du chapitre 2) -- qui utilise un horizon ''mobile'' de 2 pour sélectionner la meilleure action -- comme borne inférieure sur la valeur optimale. Voyons maintenant comment évaluer ces heuristiques à l'Aide de simulation Mont-Carlo.

5.3.1 Simulations Monte-Carlo

Les simulations Monte-Carlo (cf. annexe C), bien qu'efficaces, sont surtout reconnues pour consommer beaucoup de temps de calcul. Cependant, de récents travaux ont considérablement réduit les temps de simulation. Ces techniques, appelées *filtres à particules*, permettent la simulation en parallèle de plusieurs essais Monte-Carlo séquentiels tout en utilisant une taille fixe de la mémoire et offrant des garanties de performance sur l'estimation de la distribution de probabilité postérieure.

Thrun [1999] a appliqué la technique des filtres à particules avec succès aux POMDPs. Il estimait l'état de croyance à chaque étape de temps à l'aide de filtres à particules

puis sélectionnait ensuite la meilleure action à exécuter en utilisant une procédure mixant programmation dynamique et l'algorithme des *k-plus-proches-voisins* pour la généralisation. Les filtres à particules étaient utilisés comme variante à base d'échantillons des filtres de Bayes pour récursivement estimer la densité postérieure, ou l'état de croyance $\boldsymbol{b}$, sur un état s, du système dynamique [Fox *et al.*, 2001] :

$$\boldsymbol{b}^{t+1}(s') \propto \mathcal{O}(\boldsymbol{o}|\boldsymbol{a}, s') \int \mathcal{T}(s'|s, \boldsymbol{a})\boldsymbol{b}^t(s)\,\mathrm{d}s$$

Où, comme défini précédemment, $\boldsymbol{o}$ est l'observation jointe reçue et $\boldsymbol{a}$ l'action jointe effectuée.

Le filtre à particules représente l'état de croyance par un ensemble S de N particules pondérées $\langle s^{(i)}, w^{(i)}\rangle$. Chaque $s_{(i)}$ est un échantillon représentant un état, et les $w^{(i)}$ sont des réels non négatifs, appelés *poids d'importance*, qui somment à un. Dans sa forme basique, le filtre à particule réalise un filtrage bayésien récursif selon une procédure d'échantillonnage souvent référée par l'*échantillonnage séquentiel par importance avec rééchantillonnage* (SISR) :

1. *Échantillonnage* : Utilise l'état courant s et l'action jointe [1] $\boldsymbol{a}$ pour échantillonner l'état suivant s' selon la fonction de transition $\mathcal{T}(s'|s, \boldsymbol{a})$, qui décrit la dynamique du système selon l'interaction de l'agent ;
2. *Pondération des échantillons* : Pondère l'échantillon s' par la vraisemblance de l'observation $w' = \mathcal{O}(\boldsymbol{o}|s, \boldsymbol{a}, s')$. Cette vraisemblance est extraite de la fonction d'observation de l'agent (e.g. du modèle de ses capteurs et de l'environnement) .
3. *Rééchantillonnage* : Tire avec remplacement un nouvel échantillon -- un état -- s de l'ensemble S (représentant l'état de croyance courant $\boldsymbol{b}(s)$) selon la distribution discrète définie par la pondération $w^{(i)}$ appliquée à l'étape précédente.

Dans certains cas, toutefois, les filtres à particules peuvent prendre du temps à faire cette mise à jour de l'état de croyance, quand le nombre de particules est beaucoup trop grand par exemple. Dans ces cas ci, des améliorations existent dans la littérature. Parmi ceux-ci, un des plus intéressants est le filtre adaptatif et temps-réel de Kwok *et al.* [2003]. Il permet de déterminer dynamiquement le nombre de particules à utiliser à chaque instant pour garantir une borne sur l'erreur faite sur l'approximation de l'état de croyance selon la divergence de Kullback-Leibler (KL divergence).

Si l'on suppose que la distribution discrète estimée est répartie dans k "boîtes" (e.g. l'état de croyance est réparti sur k états), alors, pour des bornes sur l'erreur fixes ε et δ, l'équation (5.1) donne le nombre d'échantillons N minimal en fonction de k [Fox, 2001] tel que, avec probabilité $1 - \delta$, la KL divergence entre l'estimée du maximum de

1. Il est également possible d'utiliser une distribution sur les actions plutôt que l'action seule.

vraisemblance basée sur les échantillons et la vraie distribution *a posteriori* n'excède pas un certain seuil ε :

$$N = \frac{k-1}{2\varepsilon}\left[1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}}z_{1-\delta}\right]^3 \tag{5.1}$$

Où $z_{1-\delta}$ est le $1-\delta$-quantile supérieur d'une distribution normale $\mathcal{N}(0,1)$. Le nombre requis de particules est donc proportionnel à l'inverse de l'erreur voulue et linéaire en fonction du nombre k de ''boîtes'' sur lesquelles est la distribution. L'échantillonnage KLD estime k par le nombre d'états qui sont représentés par au moins une particule.

Pour intégrer l'échantillonnage KLD dans un algorithme de filtre à particule standard, une grille fixe est initialement utilisée pour approximer la distribution sur les états (e.g. les états eux-mêmes). Puis, pendant la phase de prédiction du filtrage (l'étape 1 dans la procédure SISR), l'algorithme détermine si un des états suivants générés tombe dans une case vide de la grille ou non (la grille est remise à zéro après chaque mise à jour). Si la case de la grille est vide, alors le nombre de ''boîtes'' k est incrémenté et la case est marquée comme non vide. Après chaque échantillonnage, le nombre requis d'échantillons est mis à jour en utilisant l'équation (5.1). L'échantillonnage s'arrête dès lors qu'aucune nouvelle ''boîte'' ne se remplit et que N converge.

Ainsi, l'échantillonnage KLD utilise initialement un grand nombre de particules lorsque rien n'est connu sur l'état courant et donc que la distribution est uniforme. Ce même nombre de particules diminue ensuite rapidement à mesure que de l'information sur l'état arrive. Pour plus de détails sur les filtres temps-réels adaptatifs, le lecteur intéressé est invité à se référer aux travaux de Fox [2001] sur le sujet. Voyons maintenant comment nous avons utilisé ces filtres dans le contexte d'un Rollout multiagent.

5.3.2 Rollout décentralisé

Les entrées de l'algorithme décentralisé de Rollout sont les suivantes : un état de croyance joint initial $\boldsymbol{b}^0$ qui est l'information commune partagée par tous les agents, et un ensemble d'heuristiques (ou de politiques sous-optimales) $\Pi = \langle \pi_1, \ldots, \pi_m \rangle$. On suppose que cet ensemble d'heuristiques est ordonné initialement par la valeur espérée de chaque heuristique dans l'état de croyance initial : $EU(\pi_1, \boldsymbol{b}^0) \geqslant ... \geqslant EU(\pi_m, \boldsymbol{b}^0)$.

L'algorithme 11 décrit le pseudo-code de l'algorithme de Rollout décentralisé. L'algorithme est construit de telle sorte que chaque agent calcule tout d'abord l'état de croyance joint en faisant l'hypothèse que les autres agents suivent l'heuristique la

Algorithm 11 Rollout pour DEC-POMDPs

requiert: Un ensemble d'heuristiques $\Pi = \langle \pi_1, \ldots, \pi_m \rangle$, l'état de croyance de l'agent i : $\boldsymbol{b}_i^0$, un horizon T.

retourne: Une action que l'agent i doit exécuter dans l'état de croyance $\boldsymbol{b}_i^0$.

1 **fonction** dROLLOUT(Π,$\boldsymbol{b}_i^0$,T)
2 $\boldsymbol{a}_{-i} \leftarrow \pi_1(\boldsymbol{b}_i^0)$
3 **pour tout** $a \in \mathcal{A}$, $\boldsymbol{o} \in \boldsymbol{\Omega}$ **faire**
4 $\boldsymbol{a} \leftarrow \langle a, \boldsymbol{a}_{-i} \rangle$
5 $\{\boldsymbol{b}_i^1\}_{\boldsymbol{a}_{-i},\boldsymbol{o}} \leftarrow$ UPDATE$(\boldsymbol{b}^0, \boldsymbol{a}, \boldsymbol{o})$
6 **pour tout** $\boldsymbol{b}_i^1 \in \{\boldsymbol{b}_i^1\}_{\boldsymbol{a}_{-i},\boldsymbol{o}}$ **faire**
7 $\Upsilon \leftarrow$ SMCE$(\Pi, \boldsymbol{b}_i^1, T)$
8 **fin pour**
9 **fin pour**
10 $a_i^\star \leftarrow \arg\max_{a \in \mathcal{A}} \sum_{\boldsymbol{o} \in \boldsymbol{\Omega}} \Pr(\boldsymbol{o} | \boldsymbol{b}_i^0, \langle a, \boldsymbol{a}_{-i} \rangle) \max_{\pi \in \Pi} \Upsilon[\pi]$
11 **retourner** $a_i^\star$
12 **fin fonction**

plus prolifique, i.e. la politique π_1 (ligne 4). L'ensemble des états de croyance possible $\{\boldsymbol{b}_i^1\}_{\boldsymbol{a}_{-i},\boldsymbol{o}}$ est ainsi calculé sur la base que tous les autres agents vont suivre l'heuristique π_1 et vont recevoir une observation jointe dépendante de l'état de croyance courant $\boldsymbol{b}^0$. L'ensemble des heuristiques $\Pi = \langle \pi_1, \ldots, \pi_m \rangle$ est alors évalué sur chacun de ces états de croyance $\{\boldsymbol{b}_i^1\}_{\boldsymbol{a}_{-i},\boldsymbol{o}}$ à partir des méthodes Monte-Carlo séquentielles présentées à la sous-section précédente et dont le pseudo-code est donné par l'algorithme 12.

L'algorithme 12 retourne une évaluation de chacune des actions possible dans l'état de croyance courant selon chacune des heuristiques π_k. Ainsi, en choisissant l'action ayant la plus haute valeur espérée en $\boldsymbol{b}^0$ selon l'estimation faite dans tous les $\boldsymbol{b}^1$ possibles, l'algorithme garantit une amélioration de la meilleure des heuristiques pour l'état de croyance considéré dépendamment de la qualité de l'estimation réalisée.

Il convient ici de remarquer que l'algorithme 12 se doit d'être implanté prudemment pour effectivement garantir cette amélioration. Il faut pour cela s'assurer que les estimations faites des politiques sous-optimales restent des bornes inférieures de l'exécution réelle de toutes les mixtures possibles de ces heuristiques. Ainsi, chaque agent doit estimer chacune de ces politiques comme s'il n'allait connaître dans le futur que sa propre séquence d'actions et d'observations. Le filtre à particule est donc mis à jour sur cette hypothèse et en supposant que les autres agents choisiront éventuellement n'importe laquelle des heuristiques de l'ensemble Π (ligne 4). La récompense accumulée moyenne est ainsi garantie d'être estimée inférieurement au mieux par rapport à ce qui

Algorithm 12 Estimation Monte-Carlo séquentielle de chaque $\pi \in \Pi$

requiert: Un ensemble d'heuristiques $\Pi = \langle \pi_1, \ldots, \pi_m \rangle$, un état de croyance $\boldsymbol{b}_i^1$, un horizon T.

retourne: Une valeur espérée pour chaque heuristique fournie pour l'état de croyance considéré $\Upsilon = \langle EU(\pi_1, \boldsymbol{b}_i^1), \ldots, EU(\pi_m, \boldsymbol{b}_i^1) \rangle$.

1 **fonction** SMCE(Π,$\boldsymbol{b}_i^1$,T)
2 **pour tout** $\pi_k \in \Pi$ **faire**
3 **pour** $t = 1$ **jusqu'à** $T - 1$ **faire**
4 $\phi_{-i}^{\pi_k} \sim \langle \pi_k, \mathcal{U}(\{\pi_j(\boldsymbol{b}_i^1)\}) \rangle$ ▷ Définit l'heuristique jointe stochastique
5 $\boldsymbol{b}_i^{t+1} \leftarrow \text{STEP}(\boldsymbol{b}_i^t, \phi_{-i}^{\pi_k})$ ▷ Calcule l'état de croyance suivant par SISR
6 **fin pour**
7 $V_{\pi_k}(\boldsymbol{b}_i^T) \leftarrow \mathcal{R}(\boldsymbol{b}_i^T, \phi_{-i}^{\pi_k}(\boldsymbol{b}_i^T))$ ▷ Évalue $\pi_k(\boldsymbol{b}_i^1)$
8 **pour** $t = T - 1$ **jusqu'à** 1 **faire**
9 $V_{\pi_k}(\boldsymbol{b}_i^t) \leftarrow \mathcal{R}(\boldsymbol{b}_i^t, \phi_{-i}^{\pi_k}(\boldsymbol{b}_i^t)) + V_{\pi_k}(\boldsymbol{b}_i^{t+1})$
10 **fin pour**
11 $\Upsilon \leftarrow \Upsilon \cup V_{\pi_k}(\boldsymbol{b}_i^1)$
12 **fin pour**
13 **retourner** Υ
14 **fin fonction**

15 **fonction** STEP(Un état de croyance $\boldsymbol{b}_i^t$, Une politique stochastique $\phi_{-i}^{\pi_k}$)
16 SAMPLEPF($\phi_{-i}^{\pi_k}(\boldsymbol{b}_i^t)$) ▷ Étape 1.
17 $\omega(\boldsymbol{o}) \leftarrow$ GETOBSERVATIONLIKELYHOOD() ▷ Depuis le modèle
18 IMPORTANCESAMPLEPF($\omega(\boldsymbol{o})$) ▷ Étape 2.
19 $\boldsymbol{b}_i^{t+1} \leftarrow$ RESAMPLEPF($\boldsymbol{b}^t$) ▷ Étape 3.
20 **retourner** $\boldsymbol{b}^{t+1}$
21 **fin fonction**

sera obtenu en réalité. L'étape 1 d'échantillonnage du filtre à particule (ligne 16) est donc réalisée en utilisant la politique jointe stochastique $\phi_{-i}^{\pi_k}$ qui suppose que l'agent i suit l'heuristique $\pi_k \in \Pi$ et que les autres agents suivent n'importe laquelle des politiques de Π selon une distribution uniforme :

$$\phi_{-i}^{\pi_k}(\boldsymbol{b}) = \begin{cases} \pi_k(\boldsymbol{b}) & \text{pour l'agent } i \\ \mathcal{U}(\{\pi_j(\boldsymbol{b}) : \pi_j \in \Pi\}) & \text{pour les autres agents.} \end{cases} \tag{5.2}$$

Où $\mathcal{U}(\{\pi_j(\boldsymbol{b}) : \pi_j \in \Pi\})$ dénote d'une croyance *a priori* uniforme sur la politique suivi dans l'état de croyance $\boldsymbol{b}$.

Notons également que la qualité de l'approximation produite par l'algorithme décentralisé de Rollout dépend du nombre de particules utilisées dans l'estimation de la valeur

des politiques futures, comme retranscrit par l'algorithme 12. Ainsi, il a été montré dans la section 5.3.1 que si le rééchantillonnage est effectué correctement, alors l'erreur dans l'estimation de l'état de croyance devient inférieure à ε avec probabilité $1-\delta$ selon l'équation (5.1). Il doit donc être possible de quantifier l'erreur faite sur l'estimation lorsque ce nombre de particules et suffisant. Nous laissons toutefois la preuve de cette conjecture pour de travaux futurs pour le moment.

Algorithm 13 Simulation du Rollout pour DEC-POMDPs

requiert: Un ensemble d'heuristiques Π, un état de croyance joint $\boldsymbol{b}^0$, un horizon mobile T, un horizon de test $\mathbb{T}$.
retourne: r la récompense totale accumulée.

1 **fonction** SIMULATEROLLOUT(Π,$\boldsymbol{b}^0$,T)
2 $r \leftarrow 0$; $s \leftarrow$ SAMPLEFROM($\boldsymbol{b}^0$) ; $\boldsymbol{a} \leftarrow \langle\rangle$
3 **pour tout** $i \in \alpha$ **faire**
4 $\boldsymbol{b}_i \leftarrow \boldsymbol{b}^0$
5 **fin pour**
6 **pour** $t = 0$ **jusqu'à** $\mathbb{T}$ **faire**
7 **pour tout** $i \in \alpha$ **faire**
8 $a_i \leftarrow$ DROLLOUT($\Pi, \boldsymbol{b}_i, T$) ▷ Action de l'agent i
9 **pour tout** $j \in \alpha, i \neq j$ **faire**
10 $a_j \leftarrow$ DROLLOUT($\Pi, \boldsymbol{b}_i, T$)
11 $\boldsymbol{a}_{-i} \xleftarrow{\cup} a_j$ ▷ Croyances de l'agent i sur les actions des autres agents.
12 **fin pour**
13 $\boldsymbol{b}_i \leftarrow$ SINGLEUPDATE($\boldsymbol{b}_i, \boldsymbol{a}, o_i$) ▷ Mise à jour de l'état de croyance de l'agent i (cf. section 5.3.3).
14 $\boldsymbol{a} \xleftarrow{\cup} a_i$
15 **fin pour**
16 $r \leftarrow r + \mathcal{R}(s, \boldsymbol{a})$
17 $s \leftarrow$ SAMPLEFROM($\mathcal{T}(s'|\boldsymbol{a}, s)$) ▷ Simulation de l'évolution de l'état.
18 **fin pour**
19 **retourner** r
20 **fin fonction**

L'algorithme 13 fournit le pseudo-code utilisé pour la simulation du Rollout décentralisé. Cette simulation repose essentiellement sur le fait que chaque agent n'a pas accès aux observations ni aux actions des autres agents pendant l'exécution. Pour déterminer les actions des autres agents, chaque agent n'a donc pas le choix d'effectuer lui aussi des simulations sur ce que font les autres, mais en supposant que l'état de croyance initial est son propre état de croyance (ligne 11). De la même manière, il met à jour son état de croyance en faisant des hypothèses sur les observations des autres agents. Voyons maintenant différentes manières de faire cette mise à jour.

5.3.3 Maintenance de l'état de croyance joint au cours du temps

Un des problèmes majeurs de l'algorithme 13 se trouve à la ligne 13. On retrouve également ce problème dans l'algorithme 12 à la ligne 5. Il concerne la mise à jour de l'état de croyance multiagent du point de vue d'un seul agent. En fait, dès la seconde étape de décision, aucun agent ne possède les mêmes croyances sur l'état joint du système et la croyance sur les politiques des autres agents devient alors très difficile à maintenir. Ceci est essentiellement dû au fait que chaque agent ne perçoit que sa propre observation de l'état joint du monde, et, à moins de se retrouver dans les conditions d'observations bijectives comme dans le chapitre précédent, les états de croyance de chacun des agents divergent rapidement à mesure que les observations s'accumulent. Ce problème, implicitement considéré dans les algorithmes hors-lignes par l'état de croyance sur les politiques des autres agents et une coordination temporelle parfaite, doit aussi être considéré pour les algorithmes en-ligne. Cela permettrait d'assurer la maintenance d'un état de croyance cohérent sur l'état de l'environnement et sur les politiques des autres agents.

En conséquence de cette divergence au niveau des croyances, aucun agent ne peut désormais plus supposer que les autres agents suivent exactement le même raisonnement que lui-même puisque l'état de croyance courant est différent pour chacun. La coordination devient alors extrêmement difficile et incertaine puisque l'action jointe espérée peut se trouver à être extrêmement différente de l'action jointe réellement effectuée. Une hypothèse valide serait que, puisque les agents avaient un état de croyance commun au départ et s'étaient mis d'accord sur une heuristique à utiliser dans ce cas ci, ils continuent à penser que les agents vont continuer de se comporter de cette manière dans le futur. D'un autre côté, les agents peuvent également penser que les autres agents ont reçu une information similaire à celle reçue par l'agent sur l'état du monde et peuvent ainsi utiliser l'action jointe qu'ils ont calculée pour mettre à jour leur état de croyance.

D'autres approches impliquant l'observation des actions des autres agents ou encore la communication sont envisageables et nous en discuterons dans la section 5.3.4. Pour le reste de ce chapitre les agents feront l'hypothèse que les autres agents peuvent prendre une politique uniformément dans l'ensemble des heuristiques disponibles, mais selon leur propre état de croyance. L'option de la communication ou de l'observation des autres agents restant une option valide pour de travaux futurs. Nous avons donc retenu deux méthodes de mise à jour de leur état de croyance joint que nous avons évalué dans la section 5.4.

L'une des façons (suggérée par Seuken et Zilberstein [2005]) de calculer la mise à jour de l'état de croyance multiagent à chaque étape est de maintenir une distribution de probabilité sur les états de croyance des autres agents selon chaque probabilité des observations jointes, et de calculer pour chacun de ces états de croyance quelle serait la politique exécutée par les autres agents. Malheureusement, à mesure que le temps passe, le nombre d'états de croyance possible croît doublement exponentiellement et une telle distribution devient rapidement impossible à représenter. Nous proposons donc deux approximations simples de cette mise à jour, selon comment sont considérées les observations des autres agents, tout en supposant que leurs actions sont choisies selon l'équation (5.2).

Une première manière simple d'approximer cette mise à jour en n'utilisant que l'observation locale à l'agent et de ne considérer, comme l'on fait Seuken et Zilberstein [2007a] dans leur amélioration d'MBDP, que l'observation la plus probable pour les autres agents. Ce cas correspond également à ne considérer que l'état de croyance le plus probable, parmi tous les états de croyance futurs :

$$\begin{aligned} \boldsymbol{o}_i^\star &\leftarrow \langle o_i, \boldsymbol{o}_{-i}^\star \rangle \\ \text{où } \boldsymbol{o}_{-i}^\star &= \arg\max_{\boldsymbol{o}_{-i} \in \boldsymbol{\Omega}_{-i}} \mathcal{O}(\langle o_i, \boldsymbol{o}_{-i} \rangle | \boldsymbol{a}, s') \boldsymbol{b}_i^t(s') \\ \boldsymbol{b}_i^{t+1}(s') &= \frac{\mathcal{O}(\boldsymbol{o}_i^\star | \boldsymbol{a}, s') \sum_{s \in \mathcal{S}} \mathcal{T}(s' | s, \boldsymbol{a}) \boldsymbol{b}_i^t(s)}{\sum_{s,s' \in \mathcal{S}} \mathcal{O}(\boldsymbol{o}_i^\star | \boldsymbol{a}, s') \mathcal{T}(s' | s, \boldsymbol{a}) \boldsymbol{b}_i^t(s)} \end{aligned} \tag{5.3}$$

Où $\boldsymbol{o}_i^\star$ représente l'observation jointe la plus probable que l'agent i présuppose étant donné son propre état de croyance sur l'état s'. Elle est composée de l'observation o_i qu'il a réellement reçu et de l'observation la plus probable des autres agents $\boldsymbol{o}_{-i}^\star$ selon son état de croyance.

Une deuxième manière moins naïve consiste à marginaliser toutes les observations jointes possibles dans l'état de croyance local à l'agent :

$$\boldsymbol{b}_i^{t+1}(s') = \sum_{\boldsymbol{o}_{-i} \in \boldsymbol{\Omega}_{-i}} \frac{\mathcal{O}(\boldsymbol{o} | \boldsymbol{a}, s') \sum_{s \in \mathcal{S}} \mathcal{T}(s' | s, \boldsymbol{a}) \boldsymbol{b}_i^t(s)}{\sum_{s,s' \in \mathcal{S}} \mathcal{O}(\boldsymbol{o} | \boldsymbol{a}, s') \mathcal{T}(s' | s, \boldsymbol{a}) \boldsymbol{b}_i^t(s)}, \quad \boldsymbol{o} = \langle o_i, \boldsymbol{o}_{-i} \rangle \tag{5.4}$$

Dans ce cas-ci, chaque agent incorpore dans son état de croyance la possibilité que tous les autres agents peuvent recevoir n'importe laquelle des observations jointes, selon son propre état de croyance et selon l'observation o_i qu'il a lui-même reçu. Bien que cette méthode produit des états de croyance joint possédant une entropie plus élevée, i.e. beaucoup d'incertitude sur l'état sous-jacent multiagent, nous verrons que dans la pratique cette méthode produit également de meilleurs résultats.

Avant de présenter les résultats empiriques, voyons quelles seraient les possibilités

de synchronisation des agents pour améliorer leurs états de croyance tout en utilisant un minimum de communication.

5.3.4 Vers la synchronisation via la communication

L'approche présentée jusqu'ici ne présuppose aucune communication entre les agents. Il serait éventuellement possible de l'étendre en vue de synchroniser régulièrement l'état de croyance conjoint régulièrement. Cela changerait néanmoins beaucoup la dynamique du raisonnement effectué par chacun des agents lors de la planification puisque plutôt que de raisonner sur toutes les observations possibles des autres agents, celles-ci seraient désormais disponibles régulièrement au coût -- pas nécessairement raisonnable -- d'une communication.

Dans ce sens, plusieurs approches ont été proposées dans la littérature. Elles s'articulent toutes autour de deux axes principaux qui consistent tout d'abord à calculer une politique jointe pour le POMDP multiagent comme si tous les agents avaient accès à toutes les observations de tous les agents à chaque instant. Il resterait alors à chaque agent à calculer en parallèle un état de croyance conjoint et un état de croyance mono-agent. Ceci lui permettrait de mesurer la distance existant entre sa propre croyance sur son environnement et les croyances que peuvent avoir les autres agents sur sa propre croyance, lui permettant ainsi de déterminer quand il devient nécessaire de communiquer. Pour ce faire, trois approches existent à notre connaissance.

Une première de ces approches à été proposée par Roth *et al.* [2005a,b] et consiste tout d'abord à trouver une politique pour le POMDP sous-jacent au DEC-POMDP. Cette politique est nommée QPOMDP par les auteurs. À partir de cette politique, les agents exécutent leurs actions et ne communiquent leur historique d'observations que lorsque c'est nécessaire. Cette nécessité est déterminée dès lors que l'action jointe effectuée par l'ensemble des agents serait différente sachant l'historique d'un des agents. Ainsi, chacun mémorise l'état de croyance conjoint depuis la dernière synchronisation ainsi qu'un autre état de croyance incorporant sa propre connaissance reçue depuis. Dès qu'il détecte une différence possible de politique, il communique les dernières observations qu'il a reçues pour que tout le monde puisse être à jour. Ce principe de communication a l'avantage d'être asynchrone dans le sens où dès qu'un agent communique, les autres agents ne sont pas forcés de communiquer s'il ne le juge pas opportun.

Une autre de ces approches proposée par Wu *et al.* [2009] consiste également à trouver une politique QPOMDP en se basant sur des politiques stochastiques permettant ainsi l'utilisation de la programmation linéaire. Pour déterminer quand communiquer,

Wu *et al.* [2009] utilisent cependant la notion d'inconsistance d'historique. Cette notion permet d'une part de fusionner des historiques dont les politiques futures sont identiques (mais en réalité il regarde juste la prochaine action à effectuer), et d'autre part, lorsque des historiques ont été fusionnés à tort, de corriger le tir en communiquant l'ensemble de l'historique des observations. L'avantage par rapport à la méthode précédente vient du fait que très peu d'historiques sont stockés (en réalité exactement $|\mathcal{A}|$ si on considère juste les actions suivantes comme politiques futures). Toutefois, une telle approche présente l'inconvénient que dès qu'un agent communique, alors tous les agents doivent le faire également pour synchroniser leur état de croyance conjoint. Le deuxième inconvénient réside dans le fait que dès que la fonction d'observation est particulièrement ardue (i.e comprendre non bijective au sens du chapitre 4), l'inconsistance d'historique arrive fréquemment, obligeant les agents à communiquer fréquemment.

La dernière approche tout récemment proposée par young Kwak *et al.* [2010] utilise également la politique QPOMDP calculée de la manière que l'on préfère (i.e. comprendre que l'algorithme a juste besoin d'une politique jointe fixe). Cette approche détermine alors à l'aide d'une heuristique issue des BDI (*Beliefs, Desire, Intentions* [Georgeff *et al.*, 1998]) des *points de communication* en évaluant à quel moment un agent estime que son propre historique d'observation peut-être mal interprété par les autres agents et conduire à une politique non désirable. Du fait du nombre exponentiel d'historiques possible, ils éliminent également les historiques les moins probables en ne conservant que les k plus probables et en se reposant sur le fait que dès que la synchronisation des états de croyance arrivera bien assez tôt pour que l'erreur accumulée ne soit pas trop grande.

Il serait donc intéressant de considérer la combinaison d'une politique jointe et d'une politique décentralisée dans certains des cas présentés ci-avant pour en étudier les améliorations possibles empiriquement. Néanmoins, même si nous pouvons voir ces travaux comme des avenues intéressantes pour de futures recherches, l'utilisation de la communication modifie trop la dynamique du raisonnement des agents (puisqu'ils raisonnent sur une politique jointe plutôt que décentralisée) pour que nous envisagions de l'intégrer pour l'instant. Dans le reste de ce chapitre, nous supposerons donc que les agents ne disposent pas de communication pour synchroniser leur état de croyance.

5.4 Évaluation empirique

Selon Seuken et Zilberstein [2007b], la plupart des chercheurs du domaine des DEC-POMDPs comparent les performances de leurs algorithmes sur le problème du tigre

multiagent de Nair *et al.* [2003]. Cependant, depuis leur amélioration de MBDP [Seuken et Zilberstein, 2007a], les mêmes auteurs ont proposé un nouvel environnement de test permettant de mieux mettre à l'épreuve les algorithmes du domaine. Nous avons donc réalisé nos expérimentations sur ces deux domaines : le problème du tigre multiagent et le problème des déménageurs.

5.4.1 Domaine du tigre multiagent

Dans le problème du tigre multiagent [Nair *et al.*, 2003], deux agents se trouvent face à deux portes et doivent choisir laquelle ouvrir. Une des deux portes mène à un tigre sanguinaire, l'autre à un trésor fabuleux (voir figure 5.1). La difficulté réside dans le fait que les agents, d'une part ne savent pas où est initialement le tigre, et d'autre part n'ont qu'une observation partielle de l'environnement.

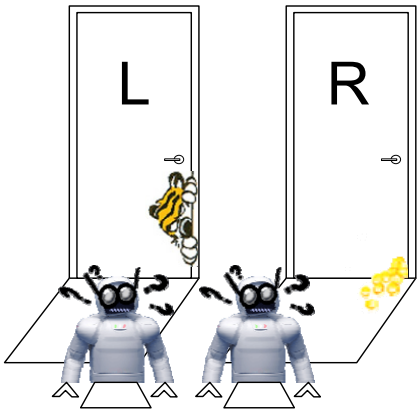


FIGURE 5.1 – Le problème du tigre multiagent.

Il existe donc deux états du monde : $\mathcal{S} = \{s_L, s_R\}$, soit le tigre est à droite, soit à gauche. Ils ont la possibilité à chaque étape soit d'écouter, pour tenter de découvrir où se trouve le tigre, soit d'ouvrir une des deux portes ($\mathcal{A}_i = \{oL, oR, Listen\}, \forall i$). Lorsqu'ils tentent d'écouter où se trouver le tigre, ils n'ont qu'une certaine probabilité [2] d'entendre le tigre derrière la bonne porte, et le complémentaire [3] à 1 d'entendre le tigre du mauvais côté. À partir du moment où un des agents ouvre une porte, le jeu est réinitialisé. En ce qui concerne les récompenses, elles se décomposent comme suit :

- si les deux agents ouvrent la même porte, ils reçoivent une récompense de +20 si c'est la bonne, -50 sinon ;
- si les deux agents ouvrent chacun une porte différente ils reçoivent -100 ;
- si les deux agents écoutent, une pénalité de -2 est infligée ;

2. en général $\mathcal{O}(o_L|s_L) = 0.85$ et pareil à droite
3. $\mathcal{O}(o_L|s_R) = 0.15$ par conséquent

– si l'un des deux écoute alors que l'autre ouvre une porte, si celui qui ouvre se trompe, les deux agents reçoivent -101, sinon +9.

Il s'agit d'un problème d'observabilité partielle simple possédant 2 états, 3 actions et 2 observations. La figure 5.2 montre le graphe d'état sous-jacent ainsi que les probabilités de transitions et d'observations selon l'action choisie par chaque agent. Pour pouvoir planifier de manière optimale dans ce problème, il est nécessaire pour chacun des agents de maintenir un état de croyance sur la position de tigre et sur la politique de l'autre agent en fonction des observations. Généralement, les métriques utilisées dans ce problème consistent à calculer la récompense obtenue après un horizon T.

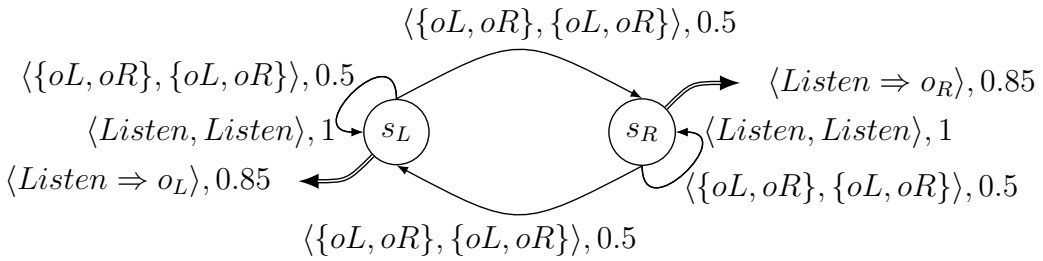


FIGURE 5.2 – Graphe d'état sous-jacent du problème du tigre.

Concernant la difficulté de ce problème, elle réside principalement dans la présence d'actions *épistémiques*. Une action est dite épistémique dès lors qu'elle ne modifie pas l'état du monde, mais agit uniquement sur l'état de croyance de l'agent effectuant l'action. Ces actions sont souvent décrites comme ''observer'', ''écouter'', ''vérifier'', etc. La difficulté du Tiger est essentiellement due à la présence de ces actions, car les heuristiques habituellement utilisées pour les DEC-POMDPs s'avèrent inefficaces (notamment la Q_{MDP}).

5.4.2 Domaine des déménageurs

Ce problème a été introduit dans la section 1.3.2.3 du chapitre 1 mais a été proposé par Seuken et Zilberstein [2007b] et décrit le comportement de deux robots déménageurs qui doivent ranger des boites dans un environnement de type grille. Il y'a 3 boites dans l'environnement, deux petites et une grande. La grande nécessite que les deux agents se coordonnent pour pouvoir la pousser. Le but est d'amener une boite (et une seule) dans la zone but (en haut de la grille, voir figure 5.3). L'optimal étant bien sur d'apporter la grosse boite dans la zone but.

Les robots ont comme possibilité pour se déplacer d'avancer (FWRD), de tourner à droite (RGHT), à gauche (LEFT), ou de rester sur place (STAY). Il y'a toujours une

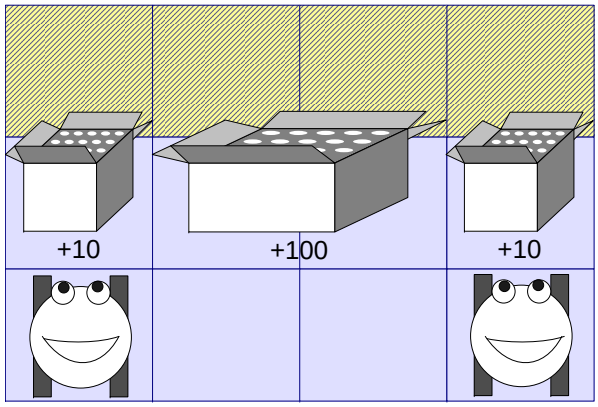


FIGURE 5.3 – Le problème du déménageur en multiagent sur une grille 3×4.

probabilité 0.9 que l'action réussisse. Dans le cas inverse, l'action STAY est exécutée. A chaque fois qu'un robot exécute une action, il reçoit une observation parmi les 5 suivantes : case vide, mur, autre agent, petite boite ou grande boite. La fonction de récompense stipule que dès qu'un agent se cogne contre une boite ou un mur, alors il reçoit +6, dès qu'un agent pousse une petite boite dans l'en-but, chaque agent reçoit +10 et la partie est réinitialisée. Si les deux agents poussent ensemble la grosse boite dans l'en-but, chaque agent reçoit alors +100 et la partie est aussi réinitialisée. À chaque étape de temps, un coût de -1 est infligé à chaque agent.

Ce problème possède un beaucoup plus grand nombre d'états que les problèmes précédents (123 à 2 agents et dans une grille de 3×4). Cependant, ce problème reste limité à 4 actions et 5 observations si on considère que les actions de l'autre agent sont totalement observables. À l'image de l'autre problème, la principale mesure dans celui-ci reste la valeur espérée amassée par les agents au cours du temps.

Voyons maintenant l'application de notre algorithme à ces deux problèmes et notre comparaison avec des algorithmes similaires de la littérature.

5.4.3 Résultats expérimentaux

Pour comparaison avec notre algorithme de Rollout décentralisé, nous avons implémenté deux approches de la littérature présentées à la section 2.2.2.4 du chapitre 2, IMBDP [Seuken et Zilberstein, 2007a] (pour *Improved Memory Bounded Dynamic Programming*) et MBDP-OC [Carlin et Zilberstein, 2008] (pour *Memory Bounded Dynamic Programming with Observations Compressed*), avec une légère adaptation vers une version en-ligne puisque ces deux algorithmes sont initialement hors-ligne. Pour cela

T	online IMBDP				online MBDP-OC				Rollout			
	ATPS (ms)		AAR		ATPS (ms)		AAR		ATPS (ms)		AAR	
	MLO	Marg.	MLO	Marg.	MLO	Marg.	MLO	Marg.	MLO	Marg.	MLO	Marg.
5	39.32	38.87	-65.45	-14.96	40.024	39.48	-41.18	-11.9	138.292	146.8	-50.55	-12.4
10	129.96	129.4	-152.42	-99.08	129.08	129.46	-143.07	-117.66	836.18	856.04	-96.24	-46.95
15	224.17	227.14	-218.9	-214.23	220.07	216.37	-248.69	-193.15	1717.94	1713.25	-193.67	-76.36

TABLE 5.1 – IMBDP en-ligne,MBDP-OC en-ligne et dRollout dans le problème du Tigre. $maxTree = 6$ et $maxObs = 2$. ATPS est le temps moyen par étape (*average time per step*) et AAR et la récompense accumulée moyenne (*average accumulated reward*).

nous avons simplement utilisé l'algorithme 13 où les lignes 8 et 11 ont été remplacées par l'algorithme correspondant.

Ces algorithmes dérivés de MBDP ont toutefois besoin de paramètres prépondérants dans leur efficacité. Un premier paramètre issu de MBDP concerne le niveau d'approximation des politiques des autres agents ($maxTree$), un autre concerne le nombre d'observations $maxObs$ prises en compte lors de l'opération de *backup* (cf. section 2.2.2.4 pour plus de détails). Puisque le problème du tigre contient seulement deux observations, nous avons choisi de ne pas approximer pour ne pas trop dégrader les performances et avons choisi de laisser $maxObs = 2$. Sur le problème des déménageurs nous avons fait varier $maxObs$ de 2 à 4. Pour le paramètre $maxTree$, nous l'avons fait varier de 3 à 7 sur le problème du tigre et de 2 à 5 sur le problème des déménageurs. Aller au-delà de ces intervalles les résultats des algorithmes ne s'amélioraient pas significativement en regard du temps de calcul. L'horizon de planification est également un paramètre important puisqu'il influence particulièrement la récompense accumulée moyenne (AAR pour *accumulated average reward*). Nous avons donc choisi de laisser les algorithmes planifier pour l'ensemble de l'horizon nécessaire. Les résultats reportés ci-après sont ceux ayant les paramètres les plus adaptés, i.e. apportant la meilleure récompense accumulée moyenne.

Pour les heuristiques Π utilisées dans l'algorithme du Rollout décentralisé, nous avons choisi la QMDP et MBDP pour le problème du Tigre mais seulement le QMDP dans le problème des déménageurs pour des raisons de limites de temps dans le calcul des valeurs de chaque action jointe, mais surtout parce que le QMDP est une heuristique extrêmement efficace dans tous les domaines de type grille.

Pour évaluer l'algorithme de Rollout décentralisé, nous avons donc tout d'abord comparé sa récompense accumulée moyenne (AAR) et le temps de calcul nécessaire pour qu'il l'obtienne versus la récompense accumulée moyenne et le temps de calcul des deux autres algorithmes en-ligne.

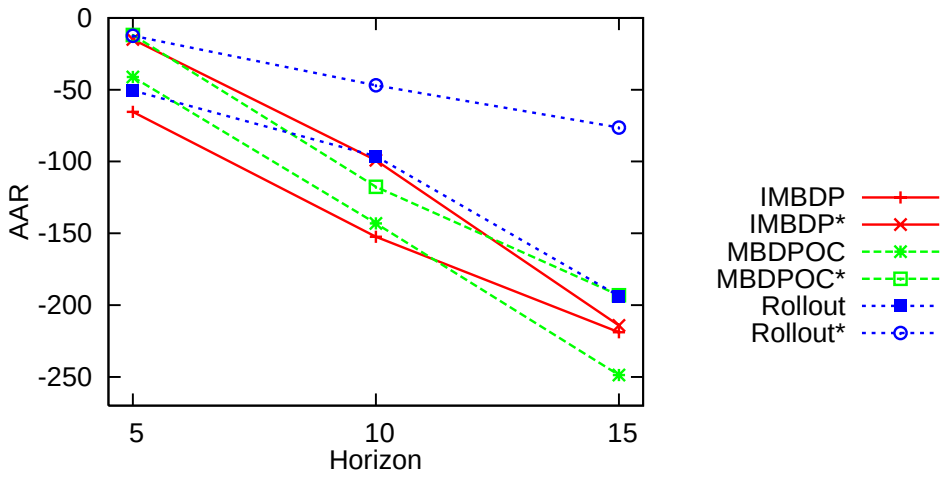


FIGURE 5.4 – online IMBDP, online MBDP-OC and Rollout on the Tiger Problem : Average Accumulated Reward (AAR) / horizon

T	online IMBDP				online MBDP-OC				Rollout			
	ATPS (ms)		AAR		ATPS (ms)		AAR		ATPS (ms)		AAR	
	MLO	Marg.	MLO	Marg.	MLO	Marg.	MLO	Marg.	MLO	Marg.	MLO	Marg.
5	5.33	4.54	-3.6	-6.8	13434.68	13426.68	-3.5	5.3	304.67	301.93	13.4	11
10	4.7	4.62	-48.8	-28.8	30040.33	30360.31	6.6	18.8	305.125	303.05	35.2	32.8
15	4.75	4.59	-59.1	-53.4	58883.06	60925.91	18	20	306.18	305.07	51.6	51.6

TABLE 5.2 – IMBDP en-ligne,MBDP-OC en-ligne et dRollout dans le problème des déménageurs. $maxTree = 2$ et $maxObs = 2$. ATPS est le temps moyen par étape (*average time per step*) et AAR et la récompense accumulée moyenne (*average accumulated reward*).

Deux versions de chaque algorithme sont évaluées dans les tables 5.1 et 5.2 et dans les figures 5.4, 5.5, 5.6 et 5.7. Les versions ‘‘étoilées’’ (IMBDP*, MBDP-OC*, Rollout*) correspondent à la mise à jour de l'état de croyance joint par marginalisation des observations des autres agents (cf. équation (5.4)), alors que les versions sans étoile utilisent l'heuristique de l'observation la plus probable (cf équation (5.3)).

Les résultats rapportés sur le problème multiagent du Tigre sont des moyennes sur 100 épisodes de chacun des algorithmes. La figure 5.4 montre la récompense accumulée moyenne (AAR) en fonction de l'horizon tandis que la figure 5.5 montre le temps de calcul moyen respectif pour trouver la meilleure action à chaque étape. Les valeurs correspondantes sont données par la table 5.1.

Pour les résultats rapportés sur le problème des déménageurs, 20 épisodes de chacun des algorithmes ont été utilisés pour la moyenne. Nous mettons également les résultats de

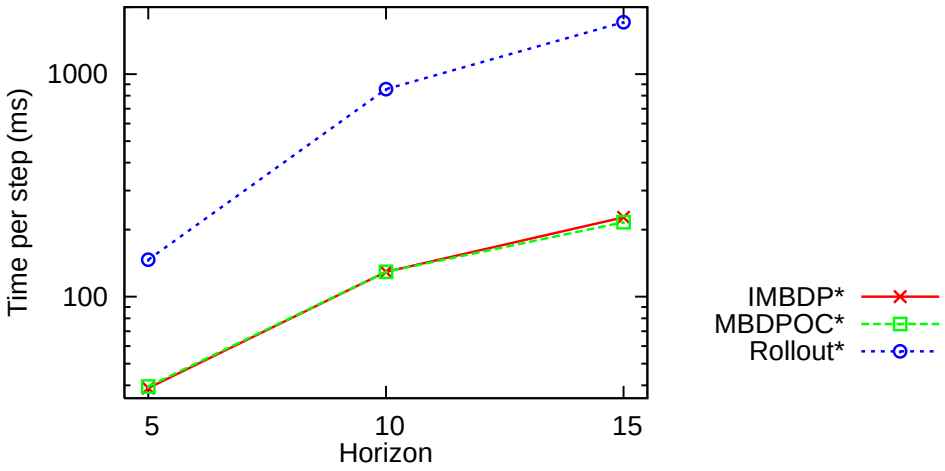


FIGURE 5.5 – online IMBDP, online MBDP-OC and Rollout on the Tiger Problem : Average Time (ms) per step / horizon

l'heuristique QMDP sous l'étiquette MDP⋆. La figure 5.6 montre la récompense accumulée moyenne en fonction de l'horion alors que la figure 5.7 montre le temps moyen nécessaire par étape pour calculer la meilleure action disponible. Les valeurs sont données dans la table 5.2. Nous avons volontairement omis les résultats des algorithmes ''non-étoilés'', i.e. ceux utilisant l'observation la plus probable pour la mise à jour de l'état de croyance joint, car les temps de calcul étaient également très similaires sur ce problème. Ceci nous mène à la discussion de ces résultats et à la conclusion de ce chapitre.

5.5 Discussion et conclusion

Depuis l'apparition d'algorithmes approximatifs efficaces pour les problèmes de la littérature DEC-POMDPs, le domaine complet des algorithmes en ligne pour les DEC-POMDPs s'est à nouveau retrouvé sous les feux de la rampe. De fait, la possibilité de calculer des heuristiques efficientes de manière efficace nous a naturellement porté à proposer un algorithme de type Rollout pour les domaines de Markov décentralisés, où les recherches dans de très larges espaces et les modèles dynamiques complexes sont courants.

Nous avons vu cependant qu'avec ce nouveau domaine, vient de nouveaux problèmes, et notamment le problème de la maintenance d'état de croyance joint commun pour l'ensemble des agents du système. Ce problème réduit également à néant l'hypothèse couramment utilisée dans les DEC-POMDPs que les agents partagent la même information initiale pour planifier, permettent ainsi une coordination implicite dans le calcul de la

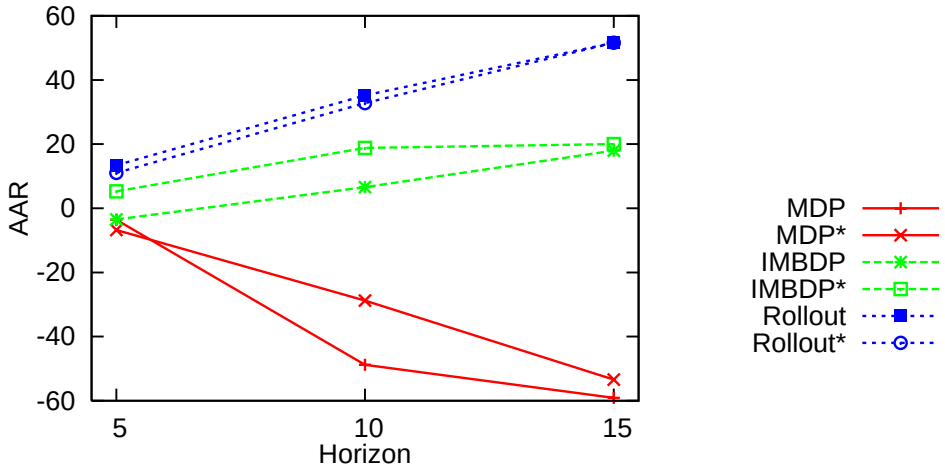


FIGURE 5.6 – online IMBDP, online MBDP-OC and Rollout on the Coordinate Box Pushing Problem : Average Accumulated Reward (AAR) / horizon

politique sur l'horizon fixé. À mesure que les agents agissent dans l'environnement, leurs états de croyance évoluent selon leurs observations reçues, souvent très différemment, induisant des problèmes de coordination jamais rencontrés auparavant.

Nous avons proposé dans ce chapitre deux manières simples de résoudre ce problème sans utilisation de la communication. Une première basée sur l'utilisation de la vraisemblance maximale des observations et une seconde basée sur la marginalisation des observations des autres agents dans la mise à jour de l'état de croyance. Nous avons alors vu que le temps de calcul étant identique, il est préférable d'utiliser la marginalisation des observations des autres agents puisqu'elle est plus efficace dans certains cas et offre des résultats similaires dans d'autres.

En ce qui concerne le Rollout décentralisé, les résultats montrent que cet algorithme reste adapté aux DEC-POMDPs dès lors que des heuristiques adaptées sont choisies. En fait, le Rollout utilise $|\mathcal{A}| \times |\Omega|^n$ fois l'ensemble de toutes les heuristiques pour sélectionner la meilleure action à utiliser à la prochaine étape. Un équilibre doit donc être trouvé entre le coût de l'estimation des actions dans les états de croyance suivants et les récompenses effectivement récupérées. C'est la principale raison qui nous a poussés à abandonner l'utilisation de IMBDP comme heuristique sur le problème des déménageurs. Cette heuristique coûte beaucoup trop cher pour le gain espéré en termes de récompenses par rapport à l'heuristique QMDP basée sur le MDP sous-jacent.

Les performances de l'approche Rollout ouvrent des perspectives intéressantes pour la recherche dans le domaine des DEC-POMDPs. En fait, de par sa capacité à garantir de meilleurs résultats que la meilleure heuristique utilisée dans toutes les situations,

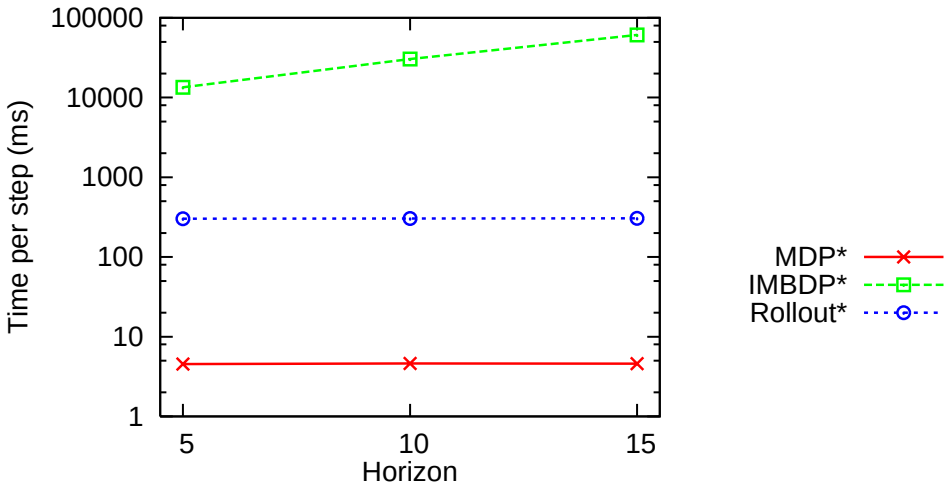


FIGURE 5.7 – online IMBDP, online MBDP-OC and Rollout on the Coordinate Box Pushing Problem : Average Time (ms) per step / horizon

le Rollout décentralisé apparaît comme une avenue de recherche intéressante et peu coûteuse dans certains cas pour améliorer n'importe quelle politique sous-optimale pour les DEC-POMDPs. Toutefois, dès que l'heuristique devient suffisamment bonne pour donner des valeurs proches de celles de la politique optimale, la contribution du Rollout devient plus modeste puisque le coût de l'évaluation de multiples heuristiques n'équilibre pas l'amélioration induite.

De fait, le Rollout est donc extrêmement bien adapté à tous les domaines de type grille avec but fixé et sans actions épistémiques. Le rapport/coût qualité de l'heuristique QMDP pour ce type de problème est tel qu'aucune heuristique n'est plus profitable. Dans le cas de problèmes munis d'actions épistémiques, il est également possible d'utiliser l'heuristique à base d'entropie de Melo *et al.* [2005] (cf. annexe 4.2.4) qui inclut le prix de l'information dans les récompenses au travers de la réduction d'entropie des états de croyance. Il est donc possible que d'autres domaines bénéficient également de ce type d'heuristiques efficaces permettant l'emploi de Rollout plutôt que d'autres algorithmes plus coûteux.

Selon notre connaissance du domaine, l'algorithme présenté dans ce chapitre est le second algorithme en-ligne pour les DEC-POMDPs. Le premier a été présenté par Emery-Montemerlo *et al.* [2004] dans le cadre des jeux stochastiques partiellement observables. Selon Seuken et Zilberstein [2007b], leur version en-ligne de MBDP réalisait déjà de meilleures performances que l'algorithme de Emery-Montemerlo *et al.* [2004] et c'est pourquoi nous ne présentons pas de comparatif dans ce chapitre. D'autre part, depuis la parution de notre article [Besse et Chaib-draa, 2008], Wu *et al.* [2009] ont également proposé un algorithme multiagent en-ligne, mais initialement basé sur la communication.

Ils présentent toutefois des résultats extrêmement prometteurs même lorsqu'aucune communication n'est disponible. Nous n'avons pas pu nous comparer à leur algorithme puisque leur modèle d'observation et leur modèle de récompense dans le problème des déménageurs ne correspondent pas exactement au modèle que nous avons utilisé.

Comme nous l'avons évoqué dans la section 5.3.4, il serait donc intéressant de voir du côté de la communication pour les travaux futurs. Nous avons pour cela réalisé des travaux conjointement avec Jean-Samuel Marier [Marier *et al.*, 2009, 2010] sur un problème de patrouille multiagent incluant la communication. Il resterait toutefois à limiter cette communication pour pouvoir comparer les algorithmes développés aux algorithmes en-ligne multiagents existants. Nous discuterons dans le chapitre suivant plus en avant les conclusions et les perspectives de recherche de cette thèse.

Chapitre 6

Conclusions et Perspectives

Ce chapitre rappelle tout d'abord le contexte de cette thèse et les contributions apportées aux différents domaines abordés. Les perspectives de recherches offertes par les contributions sont ensuite décrites et certaines sont détaillées au travers de quelques travaux effectués en collaboration avec d'autres collègues. Ce chapitre conclusif se termine par des remarques finales quant à l'application à des problèmes réels des modèles et algorithmes présentés dans cette thèse.

La prise de décisions dans les modèles de Markov multiagents est particulièrement complexe et difficile. Elle l'est également dans les modèles monoagents lorsque le nombre d'états et/ou d'actions est particulièrement grand. C'est pourquoi nous nous sommes intéressés à différentes approches permettant de réduire cette difficulté combinatoire dans les modèles de Markov monoagents et multiagents. Dans ce contexte de réduction de la complexité combinatoire, nous avons proposé plusieurs méthodes telles que l'utilisation des contraintes dans les processus décisionnels de Markov, la restriction de la fonction d'observation dans les processus partiellement observables ou encore la résolution approximée en ligne de processus de Markov décentralisés. Dans cette conclusion nous allons donc rappeler les contributions de cette thèse quant à la réduction de la complexité combinatoire des processus de Markov avant de présenter quelques travaux en cours ainsi que des travaux futurs issus de ces recherches.

6.1 Résumé des contributions

Conceptuellement, cette thèse s'est attaquée au problème de la réduction de la complexité combinatoire (c.f. annexe A) dans les processus décisionnels de Markov

monoagents et multiagents depuis les problèmes les plus simples -- monoagents et complètement observables -- jusqu'aux problèmes les plus complexes -- multiagents et partiellement observables. Dans une première partie (au chapitre 3) nous avons réduit la complexité inhérente aux MDPs lorsqu'un grand nombre d'actions sont possibles sous certaines contraintes difficiles à représenter dans le modèle MDP initial [Besse et Chaib-draa, 2007; Besse *et al.*, 2007]. Dans une seconde partie (au chapitre 4) nous nous sommes attaqués au problème de la représentation de la fonction d'observation trop générale et avons donc proposé une sous-classe d'observabilité permettant de réduire considérablement la complexité en pire cas dans les cas monoagent et multiagent [Besse et Chaib-draa, 2009, 2010a,b]. Enfin, nous avons proposé au chapitre 5 un algorithme en ligne de planification pour les modèles multiagents partiellement observables lorsqu'aucune communication n'est possible [Besse et Chaib-draa, 2008]. Voyons donc rapidement un résumé de ces contributions et les différentes complémentarités entre ces parties.

Réduction de l'espace de recherche par des contraintes sur les actions : Cette contribution suggère l'utilisation des représentations à base de contraintes pour la résolution de problèmes d'allocation de ressources sous différents types de contraintes. L'idée consiste à utiliser un algorithme permettant de dissocier la résolution des contraintes des allocations aux ressources et la résolution du problème séquentiel de planification dans un modèle unifié. Un algorithme en ligne adapté à cette dissociation a également été proposé offrant ainsi des gains en performance significatifs sur un problème de défense navale.

Réduction de l'incertitude par des contraintes sur les observations : C'est en commençant par l'extension du modèle déterministe partiellement observable que nous avons proposé une nouvelle classe spécifique d'observabilité assurant la convergence en probabilité de l'état de croyance vers un état spécifique. Cette contribution s'aligne également dans la direction de la réduction combinatoire des problèmes de Markov, mais dans le cas partiellement observable. Elle permet notamment de comprendre plus précisément quel est l'effet de l'observabilité dans la complexité des modèles partiellement observables tout en ouvrant la porte à d'autres chercheurs en vue d'induire une heuristique admissible.

Résolution en ligne de problèmes décentralisés : Un algorithme de résolution en ligne de DEC-POMDPs ne faisant aucune hypothèse sur la communication ou l'interaction des agents *a priori* a été proposé dans ce cadre. Nous avons vu qu'un tel algorithme est difficilement applicable sans communication et que de mauvaises décisions locales peuvent éventuellement conduire à des résultats globaux catastrophiques. Il convient toutefois de modérer ces propos dans la mesure ou peu de problèmes réels sont aussi couplés que le sont les problèmes ''jouets'' sur lesquels sont effectuées la majorité des expérimentations. Des travaux en ce sens

seront présentés dans la section suivante.

En résumé, cette thèse vise à contribuer à la réduction combinatoire des modèles de Markov et à l'étude de l'observabilité dans de tels modèles. Une certaine complémentarité ressort des contributions de cette thèse puisque chacune de ces contributions s'attaque à un problème distinct des modèles de Markov. On peut ainsi imaginer regrouper ces contributions dans un modèle multiagent unique et unifié associant les contraintes sur les actions, l'observabilité bijective et autorisant la communication implicite à des fins de synchronisation. Voyons donc maintenant le cadre des perspectives de recherche découlant des contributions de cette thèse.

6.2 Perspectives de recherche

Cette thèse peut être étendue dans diverses directions. Dans ce contexte, cette section décrit brièvement les perspectives de recherches associées à chaque approche développée dans ce manuscrit avant de présenter une application du modèle rassemblant toutes les contributions ci-avant.

6.2.1 Multiagent MaCSP

Brièvement discutée dans la conclusion du chapitre 3, l'extension au cas multiagent des problèmes de Markov de satisfaction de contraintes permettrait éventuellement une plus grande expressivité du modèle tout en assurant une réduction combinatoire encore plus grande que le modèle monoagent proposé. En effet, on peut supposer qu'un tel modèle permettrait de modéliser certains types d'interactions spécifiques existant entre les agents tels que les interactions négatives par exemple. Il est possible que des agents se trouvent dans l'impossibilité de réaliser une action de par la présence d'un autre agent ou de par l'action d'un autre agent à une étape précédente. Ce type d'interactions se représente très bien à l'aide de contraintes et permettrait ainsi également de réduire l'espace des actions jointes disponibles lors de la recherche d'une solution. Les contraintes sont donc un bon moyen de représenter certaines interactions entre les agents en vue de réduire la complexité d'algorithmes déjà existants.

6.2.2 Étude des modèles à observabilité bijective et transition stochastique

Nous nous sommes limités, dans le chapitre 4, à l'étude des modèles à transition déterministes même si des travaux préliminaires concernant les fonctions de transitions stochastiques sont rapportés dans l'annexe 4.2.4. Poursuivre ces travaux en étudiant à la fois la structure de la fonction de transition et le bruit généré par celle-ci est également une perspective de recherche intéressante. Il serait ainsi possible de relier les travaux proposés à ceux de Hsu *et al.* [2007] puisque ces auteurs ont étudié certaines fonctions de transitions permettant l'approximation de POMDPs. Les auteurs ont en effet réussi à montrer qu'il existe des classes de problèmes dont le nombre d'états de croyance accessible depuis un état de croyance donné est fini. Ainsi, il suffit de faire une itération de valeur sur cet ensemble fini d'états de croyance pour trouver une politique optimale pour le POMDP à horizon infini.

Il est également possible d'étendre nos travaux sur la bijectivité des observations aux POMDPs continus où la fonction d'observation est une fonction bijective exacte sur laquelle un bruit gaussien est ajouté. Il serait ainsi possible de définir une bijectivité de la transition aussi appelée réversibilité de la matrice de transition dans les modèles usuels de Markov en mathématiques [Aldous et Fill, 2002]. Ces aspects seront abordés dans la section 6.4 de ce chapitre qui décrit des travaux entrepris en collaboration avec Patrick Dallaire sur les POMDPs a espaces continus [Dallaire *et al.*, 2009b].

6.2.3 Utilisation de la communication pour la synchronisation des états de croyance

Une autre avenue de recherche intéressante apportée par l'observabilité bijective concerne la communication dans les systèmes multiagents. Dans l'exemple multiagent de chaîne de seaux d'eau traité dans le chapitre 4, pour que les agents aient accès à l'état complet du système, nous avons fait l'hypothèse que les agents pouvaient communiquer, comme bon leur semblait, la partie de l'état non visible par un autre agent. Nous n'avons cependant fait aucune hypothèse quant à la qualité de cette communication qui peut ainsi être bruitée. Ce type de communication partielle, en dehors des schémas classiques de communication dans les travaux actuels dans le domaine des DEC-POMDPs, devrait être comparé aux algorithmes faisant l'hypothèse que la communication sert à synchroniser l'état de croyance des agents en communiquant l'historique des actions et observations, tels que présentés au chapitre 5. On pourrait alors imaginer une forme de

communication efficace et sémantiquement non redondante dans des espaces factorisés et structurés et ainsi réduire les coûts associés à la communication.

6.3 Un modèle unifié appliqué à la patrouille d'un espace représenté sous forme de graphe

C'est dans le cadre de la maîtrise de Jean-Samuel Marier que nous avons essayé de mettre en pratique les approches proposées dans cette thèse [Marier *et al.*, 2009, 2010]. Un modèle multiagent pour le problème de patrouille d'agents autonomes a donc été proposé sur la base des principes suivants :

- Les agents sont contraints dans leurs actions par un graphe de localisations à surveiller aussi souvent que possible ;
- Les agents observent avec certitude leur position et communiquent celle-ci aux autre agents ;
- Les agents peuvent communiquer leur état local et leur politique locale inaccessible aux autres agents avec une fréquence suffisamment élevée telle que les délais ou la fiabilité du réseau de communication ne puissent pas être mise en cause ;
- Les agents n'ont pas accès à l'information qu'ils surveillent, ils n'ont qu'une représentation probabiliste (un état de croyance) de l'état de fraîcheur de l'information qu'ils cherchent à surveiller. Ainsi, dans le cadre de la surveillance de forêts pour la détection d'incendie par exemple, les agents pourraient être munis de capteurs infrarouges détectant avec une certaine probabilité un feu à une position donnée, mais il n'auront aucune confirmation de la part de ce capteur si quelque chose a été découvert ou pas. Cela permet un niveau d'abstraction supplémentaire et ainsi de découpler la tâche de surveillance avec l'objet à surveiller. On peut ainsi supposer avoir le modèle des capteurs spécifiques à utiliser sans savoir exactement quels sont ces capteurs ou à quoi ils servent.

Il a donc été proposé un modèle [Marier *et al.*, 2009] pour ce problème basé sur un mélange de processus de Markov multiagents partiellement observables et de processus semimarkoviens généralisés [Younes, 2004] incluant la prise en compte du temps continu et la gestion d'événements concurrents. Nous le détaillerons rapidement dans cette section. Le lecteur intéressé par sa résolution est invité à se référer à la maîtrise de Marier [2010].

6.3.1 Formalisation du problème de patrouille

Comme nous l'avons énoncé plus haut, le problème de patrouille est contraint et structuré par un graphe. Ainsi, les agents ne peuvent se rendre d'une position à une autre que si une arête du graphe existe entre ces deux nœuds. Appelons V l'ensemble des nœuds et E l'ensemble des arêtes. Soit L une matrice de taille $|V| \times |V|$, dans laquelle L_{ij} est un nombre réel *non-négatif* qui représente le temps requis pour aller d'un nœud i à un nœud j si l'arête $[i, j]$ appartient à E ; L_{ij} est infini sinon. Dans un cas plus général, L_{ij} est une distribution de probabilité sur le temps. Chaque nœud a un poids d'importance w_i. On dénotera par $\mathbf{w}$ le vecteur de tous les poids. Lorsque l'on suppose que toutes les positions sont accessibles depuis n'importe laquelle position, alors le graphe est complet.

La mesure de performance habituellement utilisée dans la littérature sur le problème de patrouille est l'*oisiveté*. L'oisiveté d'un sommet i, notée τ_i représente le temps écoulé depuis la dernière visite d'un agent à ce sommet. L'oisiveté est 0 lorsqu'un agent est sur le sommet et $\tau_i^{t+\Delta t} = \tau_i^t + \Delta t$ si le sommet n'a pas eu de visite dans l'intervalle de temps $(t, t + \Delta t)$. Parce que l'oisiveté est une quantité non bornée, nous avons décidé d'utiliser ce que nous avons appelé la *fraîcheur* d'un sommet. La fraîcheur d'un sommet i est donné par $k_i^t = b^{\tau_i^t}$, avec $0 < b < 1$. Ainsi, k_i^t est toujours compris entre 0 et 1 et peut être vue comme la valeur espérée d'un variable aléatoire de Bernoulli qui vaut 1 lorsque le sommet i est correctement observé par l'argent en patrouille et 0 sinon. De fait, k_i^t est la probabilité que cette variable soit 1 au temps t. L'idée d'observer ''correctement'' reste générale comme expliquée au quatrième point de l'introduction de cette section. Ainsi, au temps t, k_i^t est la probabilité que l'agent ait réussi à observer ce qu'il avait à observer à la position i. La mesure de performance k_i^t est ensuite calculée en faisant la somme pondérée par la valeur des sommets $\mathbf{w}$ du vecteur $\mathbf{k}^t$.

Cette probabilité évolue de la manière suivante : $k_i^{t+\Delta t} = k_i^t b^{\Delta t}$ si aucune visite n'a eu lieu dans l'intervalle de temps $(t, t + \Delta t)$. Si un agent avec des observations bruitées visite le sommet i au temps t, l'oisiveté devient alors 0 avec probabilité $(1 - a)$, où a satisfait $b < (1 - a) \leqslant 1$. Si n agents visitent le sommet simultanément au temps $t + \Delta t$ et qu'il n'y eu aucune visite depuis le temps t, alors on peut écrire :

$$k_i^{t+\Delta t} = k_i^t a^n b^{\Delta t} + 1 - a^n. \tag{6.1}$$

En résumé, une instance du problème de patrouille est un tuple $\langle L, \mathbf{w}, a, b \rangle$ constitué de (i) la matrice L des longueurs des arêtes, (ii) du vecteur $\mathbf{w}$ des poids des sommets, et (iii) des paramètres a et b représentant respectivement la probabilité que l'oisiveté ne soit pas remise à 0 lorsqu'un agent passe par un sommet et le taux de décroissance de k_i

au cours du temps, ou encore la rapidité de l'obsolescence de l'information au sommet i. Il convient ici de remarque que b peut être différent d'un sommet à l'autre et que a peut être différent pour chaque pair d'agent/sommet. Nous n'utiliserons toutefois qu'un seul paramètre a et b pour des raisons de clarté.

6.3.2 Représentation par un MMDP en temps discret

En utilisant les capacités de communication évoquées précédemment, il est possible de représenter ce problème sous la forme d'un MMDP (c.f. chapitre 2) où l'état des agents est complètement observable, mais hybride. Tous les agents ont effectivement la même information complète quant à la position des autres agents et l'état (incertain et continu) des sommets du graphe, pour pouvoir prendre une décision coordonnée. Du fait que les actions peuvent avoir des durées différentes et ainsi désynchroniser les agents, il est nécessaire d'étendre ce modèle au temps continu avec la gestion événementielle d'actions concurrentes. Cette gestion se fait très bien dans les processus décisionnels semi-markoviens généralisés (GSMDP [Younes, 2004]) que nous ne détaillerons pas ici.

L'état se décompose alors en plusieurs variables, certaines décrivant les positions de chacun des agents, d'autres modélisant la fraîcheur de chacun des sommets. Ainsi, si N agents sont dans l'environnement, le nombre d'états possible est de

$$\mathcal{S} = V^N \times [0,1]^{|V|}. \tag{6.2}$$

Étant donné un état $s = (\mathbf{v}, \mathbf{k}) \in \mathcal{S}$, v_i est la position de l'agent i et k_i est l'oisiveté du sommet i. Nous utilisons $s^t = (\mathbf{v}^t, \mathbf{k}^t)$ pour l'état et ses composantes à l'instant t.

A certains instants, appelés étapes de décision, les agents doivent choisir une action. Tous les agents doivent être en train d'effectuer une action à chaque instant et ne peuvent changer d'action que lorsqu'ils sont à une étape de décision. Les actions sont contraintes par la structure du graphe et sa position : si un agent est dans un sommet v, il ne peut choisir son action que parmi $\mathcal{A}_v = \{u : [v,u] \in E\}$. Si un agent choisit l'action u depuis le sommet v au temps t^i, la prochaine étape de décision surviendra au temps $t^{i+1} = t^i + L_{vu}$, et $v^t = v$ tant que $t \in [t^i, t^{i+1})$ et $v^t = u$ dès que $t = t^{i+1}$. En d'autres mots, l'agent est considéré comme étant à son sommet de départ pendant tout le trajet, et ce, jusqu'à ce qu'il arrive. Si L_{vu} est une distribution de probabilité sur le temps d'arrivée, alors t^{i+1} en est une aussi étant donné t^i.

Du fait que les agents n'agissent plus nécessairement de manière synchronisée, les actions deviennent alors concurrentes et peuvent s'entremêler arbitrairement puisque chaque composante k_i de $\mathbf{k}$ évolue indépendamment des autres. On peut alors réécrire

l'équation (6.1) de telle sorte à prendre en compte cette concurrence des actions en temps discret. Soit $\{t^j\}_j$ une séquence d'étapes de décisions et n_i^j le nombre d'agents arrivant au sommet i au temps t^j. Si l'on dénote $\Delta t^j = t^{j+1} - t^j$, on peut réécrire l'équation (6.1) :

$$k_i^{t^{j+1}} = k_i^{t^j} a^{n_i^{j+1}} b^{\Delta t^j} + 1 - a^{n_i^{j+1}}.$$

Les récompenses R sont alors calculées en fonction du vecteur $\mathbf{k}$. Plus spécifiquement, le gain en récompense est donné par la somme pondérée des k_i par les w_i et la fonction de valeur du GSMDP est alors calculée de la manière suivante :

$$V^\pi(s) \underset{(a)}{=} \mathrm{E} \int_0^\infty \gamma^t \mathbf{w}^\top \mathbf{k}^t \, dt. \underset{(b)}{=} \mathrm{E} \sum_{j=0}^{\infty} \gamma^{t^j} \mathbf{w}^\top \mathbf{k}^{t^j} \frac{(b\gamma)^{\Delta t^j} - 1}{\ln(b\gamma)} \tag{6.3}$$

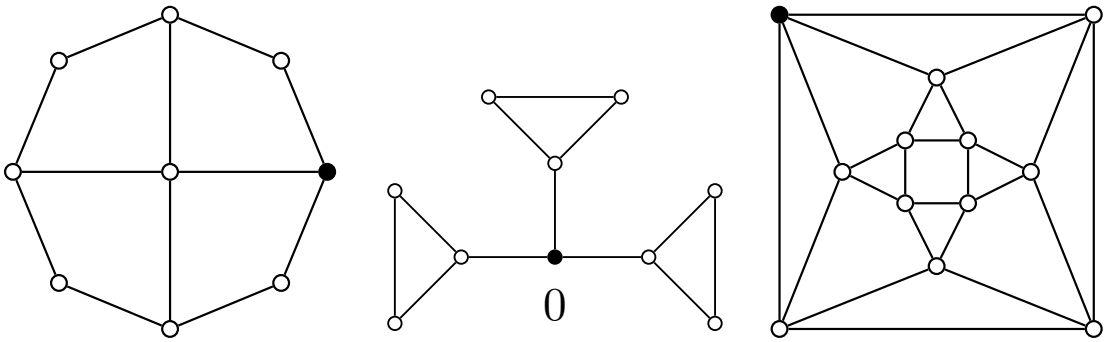
Où $\gamma \in (0, 1]$ est le facteur d'escompte.

Il convient de préciser que l'égalité (a) est donnée par la définition d'une fonction de valeur à temps continu, et (b) est obtenue par l'intégration par morceaux entre les différentes étapes de décisions. En outre, l'espérance est prise ici sur la durée stochastique des actions. La séquence d'états rencontrés dépend des actions choisies par les agents représentées par leur politique π. Le problème étant de trouver la politique maximisant cette valeur. Plusieurs algorithmes ont été proposés par Marier [2010]. Des exemples de graphes sont donnés par les figures 6.1 et 6.2.

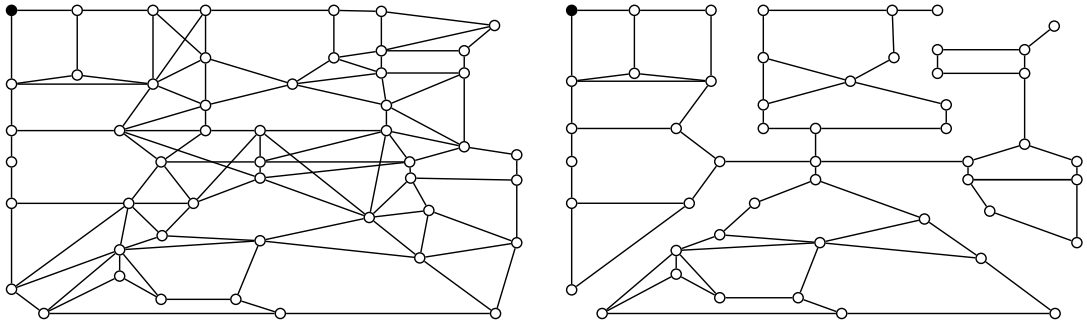
6.3.3 Discussion

Nous avons présenté dans cette section un modèle découlant directement des travaux formulés dans cette thèse alliant contraintes sur les actions au travers d'un graphe, et communication à haute fréquence de sous-parties de l'état observables uniquement par certains agents du système. Beaucoup de ces hypothèses restent discutables comme la communication gratuite, fiable, sécuritaire et à haute fréquence. Nous avons effectivement choisi ces hypothèses sciemment dans un contexte civil de surveillance de sites sensibles. Il pourrait évidemment en être autrement dans un domaine militaire par exemple et certaines hypothèses devraient alors être reconsidérées. Néanmoins, dans le contexte choisi ces hypothèses sont tout a fait justifiées et permettent une simplification efficace du problème dont il serait inconsidéré de se priver.

Quelques travaux futurs propres à ces recherches peuvent également être trouvés dans le mémoire de Marier [2010].

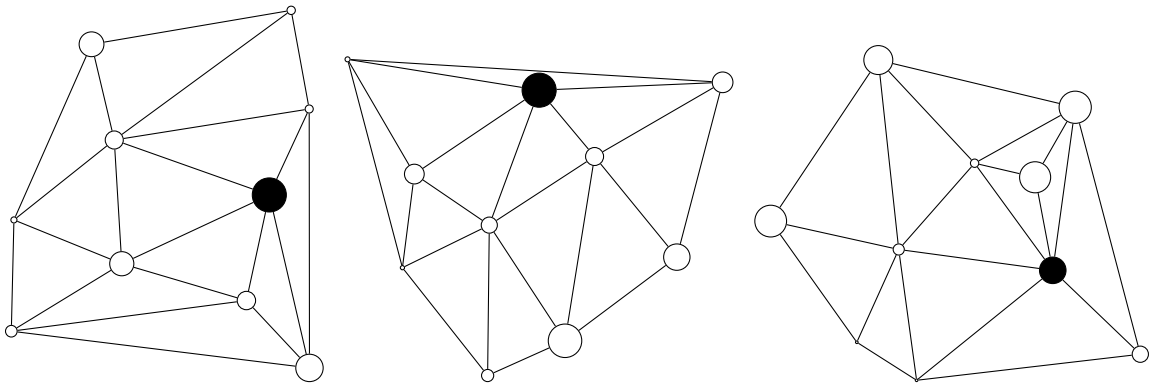


(a) Wheel (b) Cloverleaf (c) Cuboctahedron



(d) Map-A (e) Map-B

FIGURE 6.1 – Instances de patrouille. Le nœud de départ est rempli, les arêtes ont une longueur unitaire et les nœuds un poids unitaire à moins que cela ne soit précisé.



(a) (b) (c)

FIGURE 6.2 – Instances aléatoires de patrouille. Le nœud de départ est rempli, les arêtes ont une longueur unitaire et les nœuds un poids proportionnel à la taille de l'affichage.

6.4 Un modèle de POMDP à espaces continus

Une autre avenue de recherche de l'observabilité bijective que nous avons commencé à explorer concerne l'apprentissage d'une sous-classe de POMDPs continus à l'aide de processus gaussiens (c.f. annexe D). Ce travail réalisé dans le cadre de la maîtrise de Patrick Dallaire [Dallaire *et al.*, 2009b,a, 201X] concerne l'apprentissage d'une fonction d'observation bijective et d'une fonction de transition également représentable à l'aide d'une fonction déterministe adjointe d'un bruit gaussien.

6.4.1 Formalisation du modèle

Pour ces recherches, nous avons considéré le problème où le POMDP possède un espace d'états continu (tel qu'un plan en robotique par exemple), un espace d'action continu (tel que le contrôle d'une vitesse en robotique), et dont les fonctions de transition, d'observation et de récompenses se trouvent elles aussi à être continues. Placés dans un contexte d'apprentissage, nous cherchons à déterminer la fonction d'observation et la fonction d'observation tout en assurant un contrôle optimal vis-à-vis de la fonction de récompense supposée connue. Pour ce faire, nous avons supposé que le modèle est composé de fonctions déterministes adjointes d'un bruit de la manière suivante :

$$\begin{aligned} \mathbf{s}_t &= T'(\mathbf{s}_{t-1}, \mathbf{a}_{t-1}) + \epsilon_T \\ \mathbf{z}_t &= O'(\mathbf{s}_t, \mathbf{a}_{t-1}) + \epsilon_O \\ r_t &= R'(\mathbf{s}_t, \mathbf{a}_{t-1}) + \epsilon_R \end{aligned} \tag{6.4}$$

Où ϵ_T, ϵ_O and ϵ_R sont des bruits gaussiens de moyenne nulle et les fonctions T' et O' sont respectivement les fonctions déterministes et inconnues de transition et d'observation.

Bien que discutable, cette hypothèse de fonction déterministe adjointe d'un bruit blanc est largement utilisée en robotique et dans de nombreux autres domaines tels que la théorie de l'information, la sociologie ou encore la physique.

Plus de détails sur ces recherches sont disponibles dans les résultats de Dallaire *et al.* [2009b].

6.4.2 Apprentissage à partir de données bruitées

Si l'on ajoute l'hypothèse que l'on connaît maintenant le bruit de la fonction d'observation (i.e. le bruit associé au capteur), les résultats du chapitre 4 permettraient éventuellement de démontrer qu'avant un nombre fini de transitions, il ne sera pas possible d'apprendre quoi que ce soit sur la fonction de transition puisque l'incertitude associée à l'état de croyance empêche de corréler efficacement une récompense ou une observation à un état donné.

Les premiers travaux réalisés dans ce sens ont concerné l'apprentissage par régression d'une fonction de transition à partir de données d'entrées bruitées (un état de croyance à l'tape t) dont le niveau d'incertitude est connu, et vers des données de sorties bruitées (un autre état de croyance à l'étape $t+1$) dont le niveau d'incertitude est connu également. Plus de détails sont disponibles dans les résultats de Dallaire *et al.* [2009a] et de Dallaire *et al.* [201X].

6.5 Remarques finales

La dernière décennie a vu l'avènement des modèles de Markov pour le multiagent là ou il n'y avait que croyances, engagements et intentions. La possibilité de représenter sous formes de probabilités les connaissances du monde a permis l'avancée de beaucoup de domaines et notamment en robotique. Néanmoins, de plus en plus de modèles restreints et spécifiques font chaque année leur apparition dans les conférences du domaine de la prise de décision séquentielle multiagent. En particulier récemment, ces nouveaux modèles s'attaquent à la structure de tels problèmes où les interactions entre les agents sont supposées connues et peu nombreuses [Witwicki et Durfee, 2010].

Il ressort de la communauté multiagent encore jeune un grand désir d'essayer de modéliser tous les problèmes possibles à l'aide de modèles de Markov, qu'ils soient observables ou pas, stochastiques ou pas, possible à résoudre ou pas. Il conviendrait de faire une taxonomie de tous les modèles existants sur plusieurs dimensions :

- Comment un modèle représente-t'il les transitions entre les états ?
- Comment un modèle représente-t'il les observations des agents ?
- Comment un modèle représente-t'il les récompenses des agents ?
- Comment un modèle représente-t'il les interactions entre les agents ?
- Comment un modèle représente-t'il la communication entre les agents ?
- Comment un modèle représente-t'il les contraintes sur l'état, les actions, les

interactions, etc ?
- Quels types d'algorithmes sont disponibles pour ce modèle ?

Une étude d'une telle taxonomie aiderait grandement le novice dans ce domaine à trouver la voie qui lui convient le mieux et les applications possibles des ces modèles.

D'autre part, il convient également de remarquer que les DEC-POMDPs ne sont clairement pas adaptés aux problèmes actuels de par leurs hypothèses restrictives telles que le synchronisme des agents où la disponibilité de la fonction de transition complète du monde environnant. De gros efforts doivent être investis dans la représentation de modèles structurés, interactifs, faiblement couplés, asynchrones, etc. Dans l'état actuel des modèles de Markov multiagents, peu de résultats ne sauraient être mis en application sans un énorme travail préalable de modélisation et de nombreuses adaptations à l'exécution.

Dans cette thèse nous nous sommes attachés à explorer ces caractéristiques que ce modèle idéal multiagent devrait avoir pour être utilisable, mais énormément de travail reste encore à faire dans ce domaine.

Annexe A

Complexité algorithmique des modèles de Markov multiagents

Dans cette annexe une autre approche de la complexité dans les processus de Markov multiagents à horizon fini est présentée selon les degrés d'observabilité de chacun des agents. Les modèles les plus simples sont tout d'abord présentés, suivis ensuite par les modèles les plus complexes. La complexité algorithmique des modèles monoagents n'est pas précisée dans cette annexe puisqu'elle reste identique à celle des modèles multiagents dans la mesure où le nombre d'agents est fixé.

Les hypothèses classiques des modèles de Markov stipulent qu'un groupe d'agents évolue *simultanément* dans un environnement stochastique *commun à tous*. À chaque étape de temps discrète, chacun des agents choisit une action à partir de sa politique locale, et l'ensemble des actions choisies constitue l'action jointe effectuée dans l'environnement. Cette action jointe provoque la transition de l'environnement depuis l'état s vers un état s' déterminé par la fonction de transition de l'environnement. Dans le cas partiellement observable, chacun des agents reçoit alors une observation de ce nouvel état, ou le nouvel état lui-même si l'environnement est complètement observable. Plus formellement, on peut rappeler qu'un processus de décision de Markov complètement observable est défini de la manière suivante :

Définition 22. *(*MMDP*) Un* MMDP *est défini par un tuple* $\langle \alpha, \mathcal{S}, \{\mathcal{A}_i\}_{i\in\alpha}, \mathcal{T}, \mathcal{R}, T\rangle$*, où :*

- α *est un ensemble fini d'*agents $i \in \alpha, 1 \leqslant i \leqslant n$ *;*
- $\mathcal{S}$ *est un ensemble fini d'*états $s \in \mathcal{S}$ *;*
- $\mathcal{A}_i$ *est un ensemble fini d'*actions *pour l'agent* i $a_i \in \mathcal{A}_i$ *;*
- $\mathcal{T}(s, a_1, \ldots, a_n, s') : \mathcal{S} \times \boldsymbol{\mathcal{A}} \times \mathcal{S} \mapsto [0, 1]$ *est la* probabilité de transition *du système de l'état* s *vers l'état* s' *après l'exécution de l'action jointe* $\boldsymbol{a} = \langle a_1, \ldots, a_n, \rangle$ *;*

– $\mathcal{R}(s, a_1, \ldots, a_n) : \mathcal{S} \times \boldsymbol{\mathcal{A}} \mapsto \mathbb{R}$ *est la* récompense *produite par le système lorsque l'action jointe* $\boldsymbol{a}$ *est exécutée dans l'état* s *;*
– T *est l'horizon de planification.*

Graphiquement, ce modèle se représente comme montré par la figure A.1 pour le cas à deux agents. Chacun des agents connaît l'état à chaque étape de décision et influence l'état suivant. Les récompenses ont été omises pour des raisons de clarté.

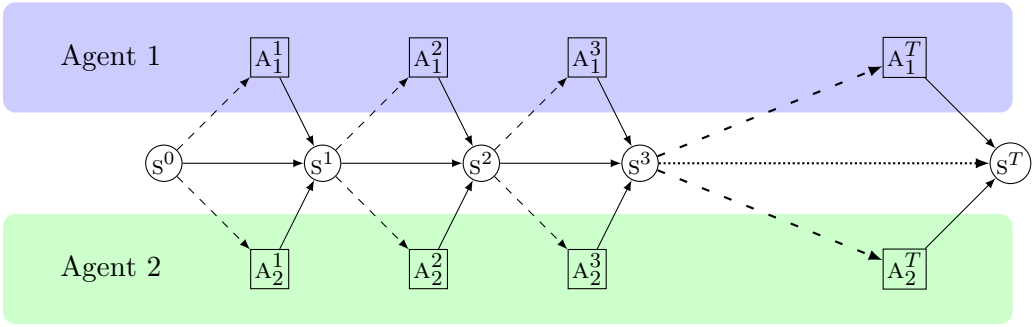


FIGURE A.1 – Problème de décision de Markov à deux agents et à horizon fini (MMDP).

Dès lors que les agents n'accèdent plus directement à l'état mais seulement à une observation de celui-ci, le problème devient alors partiellement observable. Si tous les agents ont tout de même accès aux observations de tous les autres agents, le modèle utilisé est un POMDP multiagent ou MPOMDP :

Définition 23. *Un* MDP *partiellement observable multiagent (*MPOMDP*) est défini par un tuple* $\langle \alpha, \mathcal{S}, \{\mathcal{A}_i\}_{i\in\alpha}, \mathcal{T}, \{\Omega_i\}_{i\in\alpha}, \mathcal{O}, \mathcal{R}, T\rangle$*, où :*
– $\langle \alpha, \mathcal{S}, \{\mathcal{A}_i\}_{i\in\alpha}, \mathcal{T}, \mathcal{R}, T\rangle$ est un MMDP ;
– Ω_i est l'ensemble fini des *observations* de l'agent i et $\boldsymbol{\Omega} = \bigtimes_{i\in\alpha} \Omega_i$ est l'ensemble des observations jointe, où $\boldsymbol{o} = \langle o_1, \ldots, o_n\rangle \in \boldsymbol{\Omega}$, $o_i \in \Omega_i$, est une observation jointe ;
– $\mathcal{O}(\boldsymbol{o}|\boldsymbol{a}, s') : \mathcal{S} \times \boldsymbol{\mathcal{A}} \times \boldsymbol{\Omega} \mapsto [0, 1]$ est la *fonction d'observation* représentant la probabilité de recevoir l'observation jointe $\boldsymbol{o}$ lors de la transition vers s' sous l'effet de l'action jointe $\boldsymbol{a}$.

La figure A.2 représente graphiquement un MPOMDP à deux agents à horizon fini. Chacun des agents connaît l'observation reçue par tous les autres agents à chaque étape de décision. Comparativement à la figure A.1, les agents ne reçoivent qu'une information partielle à propos de l'état. Tous les agents peuvent toutefois maintenir la même distribution pour l'état de croyance puisqu'ils ont tous accès à toutes les observations. Finalement, dans le cas où ils n'ont précisément pas accès aux observations des autres agents, le modèle devant être utilisé est un DEC-POMDP :

Définition 24. *Un* POMDP *décentralisé (*DEC-POMDP*) est un tuple* $\langle \alpha, \mathcal{S}, \{\mathcal{A}_i\}_{i\in\alpha}, \mathcal{T}, \{\Omega_i\}_{i\in\alpha}, \mathcal{O}, \mathcal{R}, T\rangle$*, où :*

- $\langle \alpha, \mathcal{S}, \{\mathcal{A}_i\}_{i\in\alpha}, \mathcal{T}, \{\Omega_i\}_{i\in\alpha}, \mathcal{O}, \mathcal{R}, T\rangle$ est un MPOMDP ;
- Chaque agent i a uniquement accès à son propre historique h_i^t des observations o_i^1 à o_i^{t-1} pour prendre sa décision a_i^t au temps t.

Dans le reste de cette annexe, nous supposerons que $\forall i, \mathcal{A}_i = \mathcal{A}$ et $\Omega_i = \Omega$ pour la simplicité des explications.

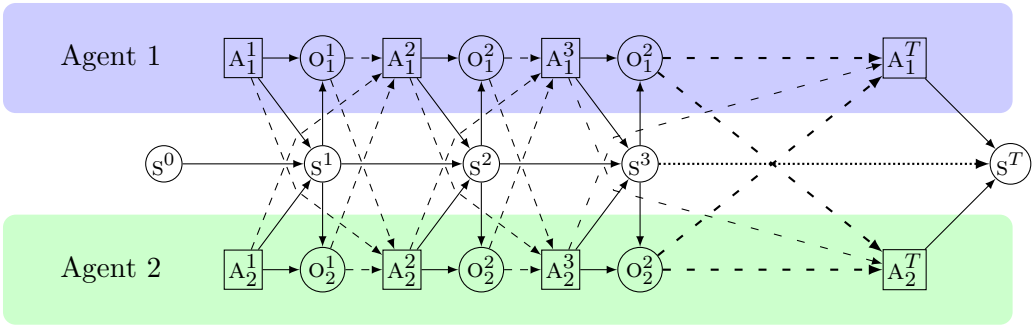


FIGURE A.2 – Problème de décision de Markov partiellement observable à deux agents et à horizon fini (MPOMDP).

La figure A.3 représente graphiquement un DEC-POMDP à deux agents à horizon fini. Chacun des agents ne connaît dans ce modèle que son propre historique des observations et des actions représenté par les cadres. Comparativement à la figure A.2, les agents ne reçoivent plus aucune information quant à l'observation de l'autre agent ou à son action.

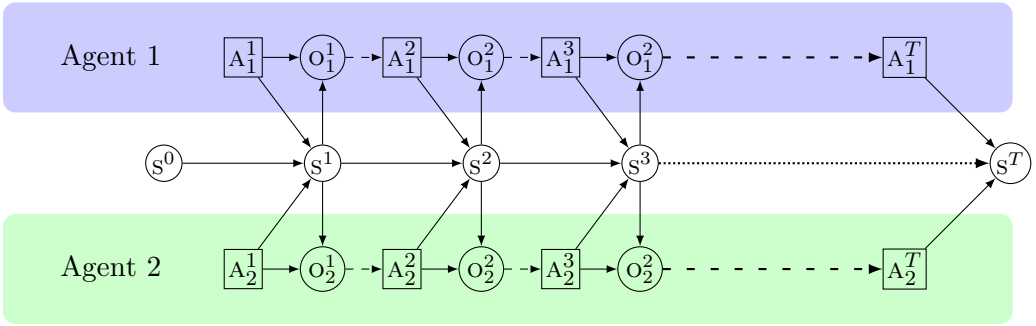


FIGURE A.3 – Problème de décision de Markov partiellement observable décentralisé à deux agents et à horizon fini (DEC-POMDP).

Comme nous l'avons expliqué au chapitre 2, résoudre un modèle de Markov multiagent consiste à trouver une *politique* pour chaque agent qui maximise l'*utilité espérée EU* (pour *expected utility*) de tous les agents. Une politique est un ensemble de règles de décision, une pour chaque étape de temps, qui associe l'historique des observations des agents à l'action jointe à effectuer dans l'environnement. Dans le cas complètement observable (MMDP), puisque l'état est une statistique suffisante pour le choix de la meilleure action à effectuer selon la fonction de récompense (Propriété de Markov), une politique associe simplement une action à tous les états accessibles à une étape de temps. Pour résumer :

Définition 25. *Dans le cas complètement observable,*

- *Une* règle de décision individuelle δ_i^t *pour l'agent* i*, est une association* $\mathcal{S} \mapsto \mathcal{A}$ *pour chaque étape* t *;*
- *Une* règle de décision jointe $\delta^t = \{\delta_i^t\}_{i \in \alpha}$ *est une association* $\mathcal{S} \mapsto \boldsymbol{\mathcal{A}}$ *pour chaque étape* t *;*
- *Une* politique individuelle Δ_i *est une séquence de* T *règles de décisions individuelles, une pour chaque étape ;*
- *Une* politique jointe $\Delta = (\delta^1, ..., \delta^T) = \langle \Delta_1, ..., \Delta_n \rangle$ *est une séquence de* T *règles de décisions jointes, une pour chaque étape, ou bien un ensemble de politiques individuelles, une pour chaque agent.*

Cependant, dans le cas partiellement observable, les agents ne perçoivent l'état réel du système qu'au travers d'observations souvent bruitées ou statistiquement insuffisantes. Chacun d'eux doit alors mémoriser la séquence complète des observations perçues (appelé l'historique) pour pouvoir agir de manière optimale. Une *politique jointe* Δ à horizon T est alors définie comme une séquence de T règles de décisions δ^t où chaque δ^t associe un historique de longueur t à une action jointe de $\boldsymbol{\mathcal{A}}$:

Définition 26. *Dans le cas partiellement observable,*

- *Une* règle de décision individuelle δ_i^t *pour l'agent* i*, est une association* $\mathcal{H}_i^t \mapsto \mathcal{A}$ *pour chaque étape* t *;*
- *Une* règle de décision jointe $\delta^t = \{\delta_i^t\}_{i \in \alpha}$ *est une association* $\bigtimes_{i \in \alpha} \mathcal{H}_i^t \mapsto \boldsymbol{\mathcal{A}}$ *pour chaque étape* t *;*
- *Une* politique individuelle Δ_i *et une* politique jointe Δ *sont identiques à celles de la définition 25.*

Où $\mathcal{H}_i^t$ *est l'ensemble des historiques* $h_i^t = (o_i^1, ..., o_i^{t-1})$ *à horizon* t*.*

Si l'on dénote par w^T le *monde* à horizon T qui contient l'historique joint à horizon T de toutes les observations et la séquence -- possiblement non observable -- des états de l'environnement sur les T étapes de temps, on peut alors calculer la probabilité de

cette séquence étant donné une politique Δ :

$$\Pr_{\Delta}(w^T) = \Pr(s^0) \prod_{t=1}^{T-1} \mathcal{O}(\boldsymbol{o}^t|s^t, \boldsymbol{a}^t) \prod_{t=1}^{T} \mathcal{T}(s^t|s^{t-1}, \boldsymbol{a}^t) \tag{A.1}$$

Où $\boldsymbol{a}^t = \delta^t(h^t)$. Il convient de remarquer que dans les MMDPs $\mathcal{O}(\boldsymbol{o}^t|s^t, \boldsymbol{a}^t) = 1$ si et seulement si $\forall i, o_i^t = s^t$.

La contribution à l'utilité espérée d'un monde à horizon T est alors donnée par :

$$eu_{\Delta}(w^T) = \Pr_{\Delta}(w^T) \sum_{t=1}^{T} \mathcal{R}(s^t, \boldsymbol{a}^t) \tag{A.2}$$

Finalement, trouver une *politique optimale* correspond à trouver une *politique jointe* Δ qui maximise l'utilité espérée de tous les mondes possibles sur l'horizon T :

$$EU(\Delta) = \sum_{\{s^t\}_{t=0}^{T}} \sum_{\{\boldsymbol{o}^t\}_{t=1}^{T-1}} eu_{\Delta}(w^t) \tag{A.3}$$

$$\text{etdonc}\Delta^* = \arg\max_{\{\delta^t\}_{t=1}^{T}} EU(\Delta) \tag{A.4}$$

Décrivons maintenant un exemple permettant d'aider à la compréhension des concepts précédents et de la complexité des algorithmes détaillés dans la suite de cette annexe.

Problème du tigre multiagent [**Nair *et al.*, 2003**] **:** Décrit au chapitre 5 de cette thèse, ce problème représente deux agents faisant face à deux portes. Derrière l'une d'entre elles se trouve un tigre féroce (associé d'une grosse pénalité). Derrière l'autre se trouve un trésor fabuleux (la récompense). Chacun des agents peut ouvrir l'une ou l'autre des portes (par une action o_{R} ou o_{L} pour *open right* et *open left*) ou se contenter d'écouter (L*is* pour *listen*). Lorsqu'un agent écoute, une petite pénalité est infligée et l'agent qui écoute reçoit une observation bruitée de là où se situe le tigre (R ou L pour *right* et *left*). Dès qu'un des agents tente d'ouvrir une porte, le monde est réinitialisé, replaçant le tigre derrière l'une des portes aléatoirement. Les deux agents ont T étapes de temps pour maximiser leur utilité. Le problème contient donc deux états -- R*ight* et L*eft*--, trois actions -- $o_{\text{R}}, o_{\text{L}}$ et L*is* -- et deux observations par agent -- R et L --. Il peut être factorisé en utilisant une variable d'état S avec $\mathcal{S} = \{\text{R}, \text{L}\}$, deux variables de décision A_1 et A_2 avec $\mathcal{A} = \{o_{\text{R}}, o_{\text{L}}, \text{L}is\}$ et deux variables d'observation O_1 et O_2 avec $\Omega = \{\text{R}, \text{L}\}$ par étape de temps.

La figure A.3 donne une représentation graphique de l'exemple pour T étapes de temps.

Une politique jointe peut aussi être représentée par un *arbre de décision* où les nœuds sont étiquetés par des actions et les arcs par des observations. Pour exécuter

un arbre de politique, chaque agent commence à la racine de l'arbre, effectue l'action correspondante, suit la branche de l'arbre étiquetée par l'observation reçue, et répète le processus jusqu'à atteindre l'horizon final. Un exemple de politique à horizon 3 est donné dans le cas du MPOMDP, i.e que les agents partagent leurs actions et observations. Cette politique consiste à écouter deux fois avant de choisir quelle porte ouvrir ou de réécouter si les agents ont reçu une information contradictoire quant à la position du tigre. Cet exemple est donné par la figure A.4. Selon l'équation (A.4), il est possible d'écrire :

$$EU(\Delta^*) = \max_{\delta^1(h^1),\delta^2(h^2),\delta^3(h^3)} \sum_{\{s^t\}_{t=0}^3} \sum_{\boldsymbol{o}^1,\boldsymbol{o}^2} eu_\Delta(w^3) \tag{A.5}$$

Où le monde w^3 correspond à une assignation des variables $(s^0, s^1, s^2, s^3, \boldsymbol{o}^1, \boldsymbol{o}^2)$.

Puisqu'au temps de décision $\boldsymbol{a}^3 = \delta^3(\boldsymbol{o}^1, \boldsymbol{o}^2)$ les agents on déjà observé $\boldsymbol{o}^1$ et $\boldsymbol{o}^2$ et effectué les actions $\boldsymbol{a}^1$ et $\boldsymbol{a}^2$, en utilisant la règle algébrique suivante :

$$\text{If } \pi_x : \mathcal{Y} \mapsto \mathcal{X} \text{ alors } \max_{\pi_x} \sum_{y\in\mathcal{Y}} f(\pi_x(y), y) = \sum_{y\in\mathcal{Y}} \max_{x\in\mathcal{X}} f(x, y) \tag{A.6}$$

Il est possible de transformer l'équation (A.5),en :

$$(\text{A.5}) \Leftrightarrow EU(\Delta^*) = \max_{\boldsymbol{a}^1} \sum_{\boldsymbol{o}^1} \max_{\boldsymbol{a}^2} \sum_{\boldsymbol{o}^2} \max_{\boldsymbol{a}^3} \sum_{\{s^t\}_{t=0}^3} eu_\Delta(w^3) \tag{A.7}$$

La règle A.6 modèle simplement le fait qu'à l'étape de l'application de la fonction π_x, la valeur de la variable y est connue et la maximisation est donc réalisée seulement sur les valeurs possibles de x plutôt que sur toutes les associations possibles $\mathcal{Y} \mapsto \mathcal{X}$, assurant ainsi un gain exponentiel dans la complexité en pire cas. L'intérêt de telles transformations sera explicité un peu plus loin dans cette annexe.

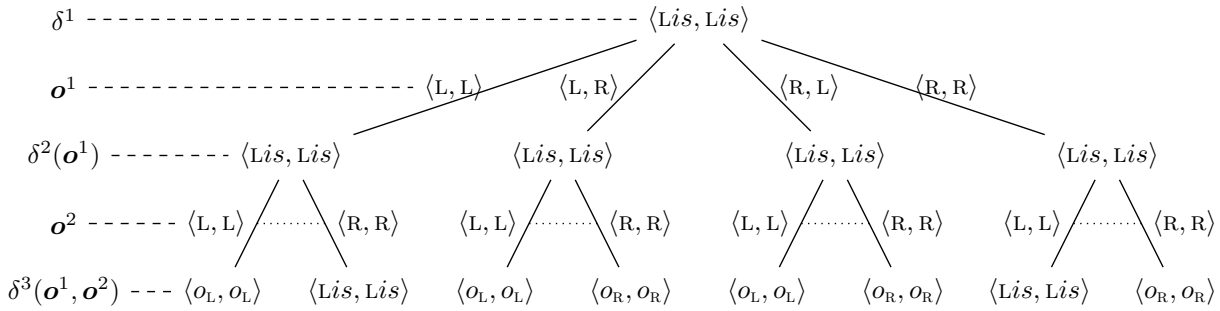


FIGURE A.4 – Exemple d'arbre de politique à horizon 3 dans le problème du tigre.

A.1 Comprendre les modèles de Markov via l'algèbre

Voyons maintenant les conséquences sur la complexité en pire cas de l'ignorance des valeurs passées de toutes les variables de l'environnement au travers de l'étude de l'équation de l'utilité espérée décrite par l'équation (A.4).

A.1.1 MMDP

Dans les MMDPs, puisqu'il n'existe pas d'observation explicite, l'expression de la contribution à l'utilité espérée devient :

$$\begin{aligned} (\text{A.2}) \Rightarrow eu_{\Delta}(w^T) &= \Pr(s^0)\prod_{t=1}^{T}\mathcal{T}(s^t|s^{t-1},\boldsymbol{a}^t)\sum_{t=1}^{T}\mathcal{R}(s^t,\boldsymbol{a}^t) \\ (\text{A.4}) \Rightarrow EU(\Delta^*) &= \max_{\{\delta^t\}_{t=1}^{T}}\sum_{\{s^t\}_{t=0}^{T}} eu_{\Delta}(w^T) \end{aligned} \tag{A.8}$$

Où $w^t = (s^0, ..., s^{t-1})$.

La figure A.1 montre une représentation graphique d'un MMDP à horizon T. Il convient de remarquer ici que bien que les états ne fassent pas partie de la connaissance des agents initialement, ils deviennent totalement observables au travers des liens d'informations. En considérant ce fait, il est également possible d'observer qu'au cours du temps, les agents ont effectivement accès à toutes les valeurs des variables de l'environnement. L'équation (A.8) peut être alors transformée en utilisant la règle (A.6) :

$$(\text{A.8}) \Leftrightarrow EU(\Delta^*) = \sum_{s^0}\max_{\boldsymbol{a}^1}\sum_{s^1}...\max_{\boldsymbol{a}^T}\sum_{s^T} eu_{\Delta}(w^T) \tag{A.9}$$

Où $\boldsymbol{a}^1 = \delta^1(s^0)$, $\boldsymbol{a}^2 = \delta^2(s^0, s^1)$, etc.

A.1.2 MPOMDP

Dans les MPOMDPs, puisque les agents n'ont pas accès à l'état réel du monde, l'historique entier des actions et des observations passées doit être maintenu [Smallwood et Sondik, 1973]. Dans ce cas, l'expression de l'utilité espérée devient :

$$(\text{A.4}) \Rightarrow EU(\Delta^*) = \max_{\{\delta^t\}_{t=1}^{T}}\sum_{\{s^t\}_{t=0}^{T}}\sum_{\{\boldsymbol{o}^t\}_{t=1}^{T-1}} eu_{\Delta}(w^T) \tag{A.10}$$

La figure A.2 montre une représentation graphique d'un MPOMDP à horizon T. Les liens d'information montrent que les agents ont accès à l'ensemble de l'historique des actions et des observations de l'autre agent (mais pas à l'état) et peuvent donc raisonner sur la même information. On observe alors que le nombre de transformations possibles dans l'équation de l'utilité espérée est moindre ; ainsi, de la même manière que nous avons transformé l'équation (A.8) en l'équation (A.9), il est possible de transformer l'équation (A.10) par l'application de la règle (A.6) :

$$(\text{A.10}) \Leftrightarrow EU(\Delta^*) = \max_{\boldsymbol{a}^1} \sum_{\boldsymbol{o}^1} \dots \max_{\boldsymbol{a}^T} \sum_{s^0,\dots,s^T} eu_\Delta(w^T) \tag{A.11}$$

Il convient alors de remarquer que la dernière somme sur les états possibles ne peut être distribuée de la même manière que la somme sur les observations puisque les agents n'ont pas accès à la valeur réelle de ces états passés au moment d'effectuer les décisions et doivent donc maintenir un *état de croyance*. Cette croyance est représentée explicitement par la séquence des observations reçues depuis le début, ou peut être maintenue implicitement au travers d'une *distribution de probabilité sur les états* par l'utilisation de la règle de Bayes :

$$\boldsymbol{b}^{t+1}(s') \propto \mathcal{O}(\boldsymbol{o}|\boldsymbol{a}, s') \int \mathcal{T}(s'|s, \boldsymbol{a}) \boldsymbol{b}^t(s) \, \mathrm{d}s$$

Où $\boldsymbol{b}^t \in \Delta\mathcal{S}$ est une distribution de probabilité sur le simplex sur $\mathcal{S}$ est est appelé *état de croyance*.

A.1.3 DEC-POMDP

Bien plus complexes, les DEC-POMDPs font l'hypothèse que les agents n'ont pas accès à l'historique des autres agents. Chaque agent doit alors maintenir un *état de croyance sur tous les historiques possibles des autres agents* et les transformations faites ci-avant sur l'équation de l'utilité espérée ne fonctionnent plus. La figure A.3 montre une représentation graphique d'un DEC-POMDP à horizon T. Par comparaison aux figures précédentes, il existe beaucoup moins de liens d'informations et les agents n'ont donc pas accès à l'observation ni à l'action des autres agents.

L'utilité espérée d'un DEC-POMDP est originalement la même que celle d'un MPOMDP (équation (A.10)). Cependant, puisque les simplifications ne s'appliquent pas à cause des connaissances partielles des agents. il est seulement possible de décomposer chaque politique jointe et chaque observation jointe pour chaque agent. Par exemple, si l'on considère seulement l'agent 1, la règle (A.6) peut être appliquée sur les décisions de

l'agent 1 :

$$\begin{aligned} EU(\Delta^*) &= \max_{\delta^1,\ldots,\delta^T} \sum_{s^0,\ldots,s^T} \sum_{\boldsymbol{o}^1,\ldots,\boldsymbol{o}^T} eu_\Delta(w^T) \\ &= \max_{\{\delta_i^1,\ldots,\delta_i^T\}_{i=1}^n} \sum_{\{o_i^1,\ldots,o_i^{T-1}\}_{i=1}^n} \sum_{s^0,\ldots,s^T} eu_\Delta(w^T) \\ &= \max_{\{\delta_i^1,\ldots,\delta_i^T\}_{i=2}^n} \max_{a_1^1} \sum_{o_1^1} \ldots \max_{a_1^{T-1}} \sum_{o_1^{T-1}} \max_{a_1^T} \sum_{\{o_i^1,\ldots,o_i^{T-1}\}_{i=2}^n} \sum_{s^0,\ldots,s^T} eu_\Delta(w^T) \end{aligned} \quad \text{(A.12)}$$

Cette dernière équation (A.12) représente à la fois que l'agent 1 doit raisonner sur un *état de croyance* -- au travers de la somme sur les états -- et sur un *état de croyance multiagent* qui consiste en toutes les politiques possibles de tous les autres agents considérant tous les historiques possibles d'observations. Un état de croyance multiagent peut aussi être vu comme un point dans le simplex sur $\mathcal{S} \times Q_{\neq i}^T$, où $Q_{\neq i}^T$ représente l'ensemble de toutes les politiques jointes à horizon T pour tous les agents sauf i.

Voyons maintenant la complexité algorithmique de ces modèles à partir de la représentation algébrique de leur utilité espérée.

A.2 Complexité algorithmique des modèles de Markov

Pour cela considérons deux types d'algorithmes différents : un qui met à profit l'espace mémoire pour améliorer les temps de calcul et un autre qui, au contraire, économise la mémoire au détriment du temps de calcul.

A.2.1 Variable Elimination (VE)

Le premier type algorithme considéré pour résoudre ces problèmes de Markov multiagents et l'algorithme par élimination de variable (VE [Dechter, 1999]). Le principe de VE est d'utiliser la structure algébrique du problème pour calculer l'utilité espérée globale en ne faisant que des calculs locaux du type de la programmation dynamique.

Par exemple, considérons trois fonctions $f(x,y), f(x,z), f(x,u)$ et supposons que l'on veut calculer $C = \max_{x,y,z,u}(f(x,y)+f(x,z)+f(x,u))$. Le principe de VE est d'*éliminer* chaque variable successivement. Éliminer z consiste d'abord à décomposer C comme $C = \max_{x,y,u}(f(x,y) + (\max_z f(x,z)) + f(x,u))$. De fait, z est éliminé en considérant

seulement les fonctions locales qui dépendent de z. Cela crée ainsi une nouvelle fonction $g(x) = \max_z f(x, z)$ qui ne dépend que de x. Les autres variables peuvent ensuite être éliminées de manière similaire pour obtenir la valeur de C. Les valeurs optimales de chacune des variables peuvent être mémorisées pendant les calculs.

Plus généralement, VE peut être utilisé pour calculer l'élimination de variable sur une combinaison de fonctions, i.e sur des quantités de la forme $\oplus_V(\otimes_{f\in F} f)$, où V est l'ensemble des variables à éliminer, F est l'ensemble des fonctions locales (chacune des fonctions dépendant d'un sous-ensemble de V), et $\oplus$ et $\otimes$ sont des opérateurs satisfaisant les propriétés algébriques telles que la commutativité, l'associativité ou la distributivité de $\oplus$ sur $\otimes$. Il est également possible d'adapter VE à des cas incorporant plusieurs opérateurs d'élimination et de combinaison [Ndilikilikesha, 1994]. Cette adaptation est requise pour les modèles de Markov qui impliquent l'opérateur d'élimination max sur les règles de décision et $\sum$ sur les états et les observations (cf. e.g. équation (A.5)), et l'opérateur de combinaison $\times$ pour les probabilités conditionnelles et $\sum$ pour les récompenses.

La complexité temporelle et spatiale de VE est $O(|F|d^{\omega+1})$, où :
- $|F|$ est le nombre de fonctions locales ;
- d est la taille maximum des domaines des variables à éliminer ;
- ω est la *taille induite par l'ordre d'élimination*. Ce paramètre est le nombre maximal de variables impliquées dans une fonction créée pendant les éliminations. Certains ordres d'élimination sont meilleurs que d'autres : dans l'exemple considéré ci-dessus, la taille induite par l'ordre d'élimination z, y, x, u est de 1 alors que que celle induite par l'ordre x, y, z, u est de 3. Si ω est la taille minimum induite quelque soit l'ordre d'élimination, il est simplement appelé *taille induite*. Le trouver est généralement un problème NP-complet [Dechter, 1999].

Lorsque plusieurs opérateurs d'élimination sont impliqués, la complexité théorique n'est pas exponentielle dans la taille induite mais dans la *taille induite contrainte* [Park et Darwiche, 2004; Pralet *et al.*, 2006a], qui prend en compte le fait que la présence de plusieurs opérateurs d'élimination impose généralement des contraintes sur l'ordre d'élimination.

Pour VE, l'intérêt principal de la transformation de quantité de la forme

$$(a)\ \max_{\pi_x} \sum_y f(\pi_x(y), y) \text{ vers } (b)\ \sum_y \max_x f(x, y)$$

est de diminuer la taille du plus grand domaine des variables. Plus précisément, dans (a), la règle de décision π_x peut être considérée comme une variable dont le domaine est l'ensemble des tuples $E = \{(x_1, \ldots, x_{dom(y)|}), |, x_i \in dom(x)\}$. Chaque élément de E définit la valeur prise par π_x pour chaque valeur de y. La taille de E est donc de

$|dom(x)|^{|dom(y)|}$. Pour (b), la taille du plus grand domaine impliqué est seulement de $\max(|dom(x)|, |dom(y)|)$, qui n'est pas exponentiel en $|dom(y)|$. Cela montre l'intérêt de l'utilisation de la règle (A.6) pour la transformation des équations (A.8) et (A.10) en de nouvelles équations (A.9), (A.11) et (A.12).

Il convient toutefois de remarquer que de telles transformations ne sont pas toujours possibles. Par exemple,

$$\max_{\pi_x, \pi_t} \sum_{y,z} f(\pi_x(y), y, z, \pi_t(z))$$

peut possiblement être transformée en

$$\max_{\pi_t} \sum_{y} \max_{x} \sum_{z} f(x, y, z, \pi_t(z)),$$

mais pas dans une forme n'impliquant aucune élimination de règle de décision. Une telle situation apparaît généralement un premier agent connaît seulement la valeur de la variable y avant de prendre la décision x, tandis qu'un autre agent ne connaît que la valeur de z avant de prendre la décision t.

A.2.2 Recherche dans un arbre

À l'inverse de l'algorithme VE, la recherche dans un arbre est une méthode de haut en bas qui cherche exhaustivement une solution optimale sur toutes les instanciations possibles des variables.

l'idée principale de cet algorithme est d'instancier les variables et de calculer la valeur de chaque séquence de décisions considérant tous les mondes possibles. L'algorithme 14 est un algorithme de recherche en profondeur d'abord (DFS pour *Depth First Search*) qui explore récursivement toutes les assignations possibles des nœuds. Selon le type de nœud, DFS choisira de maximiser (pour les nœuds de décision -- ligne 5) ou de sommer (pour les nœuds d'observations ou d'états -- ligne 12) les valeurs espérées des différentes instanciations de chacun des nœuds fils. Dès qu'il rencontre un nœud feuille (une récompense -- ligne 19), DFS calcule la valeur espérée $eu_{\Delta_A}(w_A^T)$ étant donné l'instanciation A correspondant au chemin parcouru depuis la racine jusqu'à la feuille.

La complexité de DFS dépend principalement de la profondeur maximale de l'arbre σ et du facteur de branchement β à chaque nœud. En fait, la complexité temporelle de l'algorithme est en $\mathcal{O}(\beta^\sigma)$: pour chaque nœud de l'arbre, DFS doit évaluer chacun des nœuds fils. Cependant, la complexité spatiale reste polynômiale en σ et β : DFS doit seulement mémoriser l'assignation courante des variables et la liste des variables

Algorithm 14 Depth First Search (DFS)

Entrée : L : Une liste ordonnée de nœuds
A : Une assignation des nœuds qui ne sont pas dans L
Retourne : La valeur espérée maximale $EU(\Delta^*)$
$n \leftarrow first(L)$
si n est un nœud de décision **alors**
  $val \leftarrow -\infty$
  pour tout $v \in domain(n)$ **faire**
    $val \leftarrow \max(val, \text{DFS}(queue(L), A \cup \{n \leftarrow v\}))$
  fin pour
  retourner val
fin si
si n est un nœud d'observation ou d'état **alors**
  $val \leftarrow 0$
  pour tout $v \in domain(n)$ **faire**
    $val \leftarrow val + \text{DFS}(queue(L),\ A \cup \{n \leftarrow v\})$
  fin pour
  retourner val
fin si
Si n est un nœud récompense **alors retourner** $eu_{\Delta_A}(w_A^T)$

non assignées et leurs valeurs non testées. Cela a des répercussions importantes sur la complexité en pire cas des algorithmes pour les modèles de Markov multiagent.

A.2.3 Complexité des algorithmes

Si l'on se base sur l'équation (A.9) qui alterne des nœuds *max* et des nœuds *somme*, et si l'on décompose chaque action jointe comme une séquence de n actions unitaires, la complexité de DFS pour un MMDP à horizon T peut-être facilement calculée. La profondeur de l'arbre est égale au nombre de variables à instancier : $T+1$ états et nT décisions. La profondeur est donc de $\sigma = T(n+1)+1$. En ce qui concerne le facteur de branchement, la plus grand domaine d'une variable est le maximum entre le nombre d'état et le nombre d'actions. Puisque la profondeur croît linéairement en l'horizon, la complexité spatiale de l'algorithme est polynômiale.

Si l'on regarde l'algorithme VE dans les MMDPs, la taille des domaines est la même. La taille induite contrainte est toutefois polynômiale en fonction du nombre d'agents puisqu'à chaque étape de temps t une fonction des décisions des agents a_i^t est induite

VE	ω	d	Time Comp.	Space Comp.
MMDP	$n+1$	$\max(\lvert\mathcal{S}\rvert, \lvert\mathcal{A}\rvert)$	$\text{poly}(T)\exp(n)$	$\text{poly}(T)\exp(n)$
MPOMDP	$2nT+1-n$	$\max(\lvert\mathcal{S}\rvert, \lvert\mathcal{A}\rvert, \lvert\Omega\rvert)$	$\exp(n,T)$	$\exp(n,T)$
DEC-POMDP	$2nT+1-n$	$\lvert\Omega\rvert^{T-1}$	$\exp(n)\exp^2(T)$	$\exp(n)\exp^2(T)$
DEC-POMDP$_K$	$2nT+1-n$	$\lvert\Omega\rvert^{K}$	$\exp(n,T)$	$\exp(n,T)$
DFS	$\sigma \leqslant$	β	Time Comp.	Space Comp.
MMDP	$T(n+1)+1$	$\max(\lvert\mathcal{S}\rvert, \lvert\mathcal{A}\rvert)$	$\exp(n,T)$	$\text{poly}(n,T)$
MPOMDP	$T(2n+1)+1-n$	$\max(\lvert\mathcal{S}\rvert, \lvert\mathcal{A}\rvert, \lvert\Omega\rvert)$	$\exp(n,T)$	$\text{poly}(n,T)$
DEC-POMDP	$T+1+n(2T-1-T\lvert\Omega\rvert^{T-1})$	$\max(\lvert\mathcal{S}\rvert, \lvert\mathcal{A}\rvert, \lvert\Omega\rvert)$	$\exp(n)\exp^2(T)$	$\text{poly}(n)\exp(T)$
DEC-POMDP$_K$	$T+1+n(2T-1-T\lvert\Omega\rvert^{K})$	$\max(\lvert\mathcal{S}\rvert, \lvert\mathcal{A}\rvert, \lvert\Omega\rvert)$	$\exp(n,T)$	$\text{poly}(n,T)$

TABLE A.1 – Complexité de VE et DFS dans les modèles de Markov multiagents. $\exp^2$ signifie doublement exponentiel.

par l'élimination de la variable d'état s^t. Les complexités spatiales et temporelles de l'algorithme sont donc exponentielles dans le nombre d'agents, mais polynômiales en l'horizon puisque $|F|$ croît linéairement avec l'horizon. On peut remarquer ici que VE considère naturellement s^{t-1} comme un historique suffisant pour prendre la décision a_i^t.

Le plus grand domaine d pour les MPOMDPs est similaire à σ dans l'algorithme DFS, La taille induite contrainte ω devient toutefois polynômiale en l'horizon. Ceci est dû à la somme sur tous les états à la fin de l'équation (A.11) qui induit une large fonction de toutes les variables de décision et d'observation provoquant l'explosion de la complexité.

Dans le cas des DEC-POMDPs, l'Arbre de recherche est plutôt différent. De fait, selon l'équation (A.12), les agents doivent raisonner sur toutes les règles de décision possibles des autres agents sans aucune connaissance à priori de leurs observations. Ils doivent donc prendre des décisions pour tous leurs historiques possibles. Par exemple, un algorithme de recherche dans les arbres par profondeur d'abord pour résoudre le problème du tigre à horizon 2 est donné par la figure A.5. La profondeur de l'arbre de recherche n'est alors plus polynômiale en fonction de l'horizon de planification puisque la taille des entrées des règles de décision croît linéairement en fonction de l'horizon, créant ainsi une croissance exponentielle du nombre de règles de décision possibles. Ce phénomène est principalement dû à l'ignorance des agents à propos du monde w^T. La profondeur de l'arbre de recherche σ devient alors exponentielle en fonction de l'horizon ainsi que la complexité spatiale.

Ce problème de la croissance exponentielle du nombre de règles de décision possibles produit des conséquences différentes sur l'algorithme VE mais sans changer l'interprétation finale sur la complexité : la taille induite contrainte ω dans les DEC-POMDPs reste identique à celle dans les MPOMDPs. Néanmoins, la taille maximale des domaines d croît dramatiquement à cause de l'élimination de variables de décision spéciales ayant des

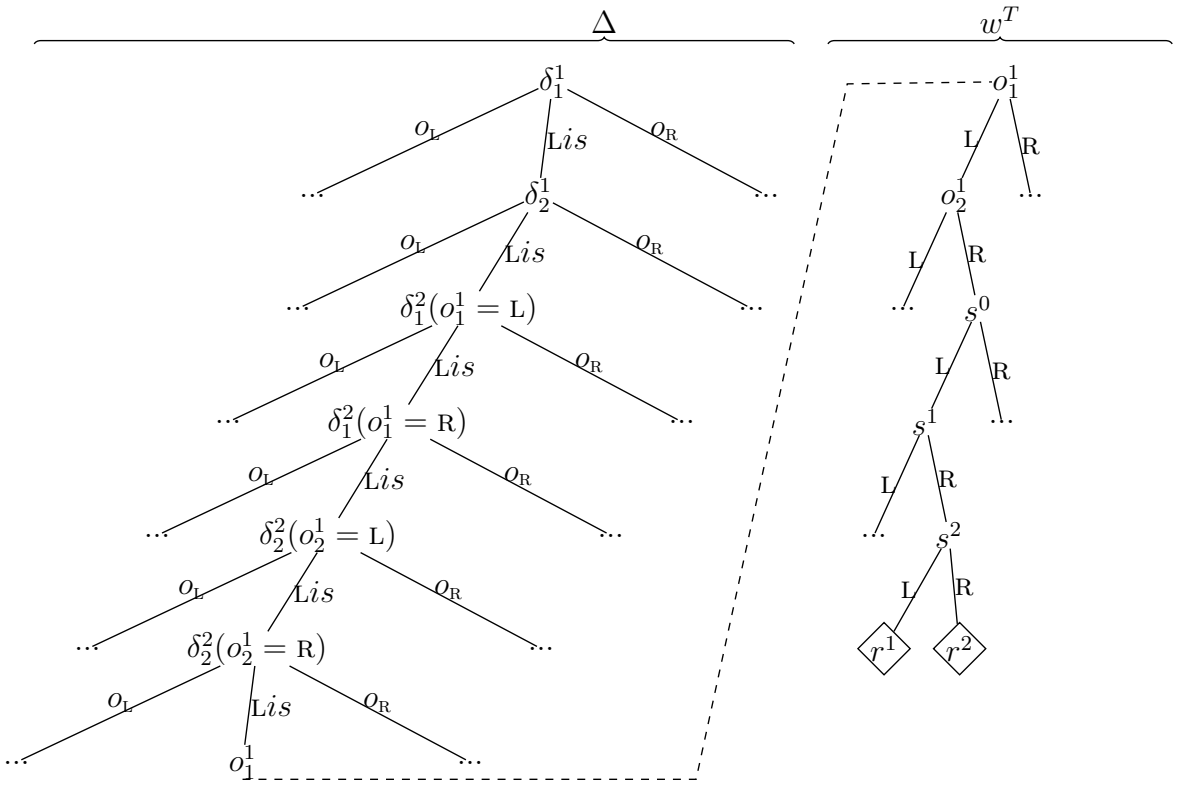


FIGURE A.5 – Exemple d'arbre de recherche pour le problème du tigre à horizon 2 en DEC-POMDP.

domaines dont la taille dépend de l'historique au complet. d devient alors exponentiel en l'horizon.

On retrouve donc résumée dans la table A.1 la complexité des algorithmes VE et DFS qui sont en fait les résultats classiques de la littérature sur la complexité des modèles de Markov multiagent lorsque l'on fixe le nombre d'agents [Papadimitriou et Tsisiklis, 1987; Bernstein *et al.*, 2000]. L'approche proposée ici explique néanmoins beaucoup mieux la complexité de ces modèles que les réductions polynômiales que l'on peut trouver dans la littérature.

Cette approche permet également de mettre en avant une autre classe de problèmes de Markov multiagent. En supposant que les agents n'ont qu'une mémoire limitée, il est possible de réduire significativement cette complexité. Si l'on définit par exemple un historique K-limité comme $\hat{h}_i^t = (o_i^{t-K}, ..., o_i^{t-1})$, alors en utilisant ce type d'historique seulement, la profondeur de l'arbre pour DFS devient $\sigma = T + 1 + n(2T - 1 - T|\Omega|^K)$ qui reste linéaire en T et donc :

Théorème 4. *Decider que $EU(\Delta^*) > \alpha$ pour un* DEC-POMDP *à historique K-limité est* PSPACE.

Démonstration. L'algorithme DFS utilise seulement un espace polynomial pour résoudre un DEC-POMDP à horizon K-limité. □

Ce résultat est également la source des travaux réalisés au chapitre 4. Il convient toutefois de remarquer que même si le problème de décision dans ce cas est PSPACE, mémoriser la politique Δ^* telle que $EU(\Delta^*) > \alpha$ reste un problème exponentiel en complexité spatiale.

Annexe B

Les différentes approches de planification

Dans cette annexe les différentes approches de planification sont présentées. L'approche initiale classique est tout d'abord expliquée en profondeur, depuis la planification STRIPS *jusqu'à Graphplan en passant par la planification à base de plans partiels. L'approche hiérarchique est ensuite survolée avant de présenter les grandes classes de planification probabilistes.*

La recherche sur la planification en Intelligence Artificielle s'est souvent concentrée sur des problèmes incluant un grand nombre d'actions interagissant de manière complexe. Au contraire de la recherche en ordonnancement qui s'est plutôt attachée à résoudre des problèmes impliquant un petit nombre de choix d'actions, mais où l'ordre de ces actions dans le temps et l'utilisation des ressources pour chacune d'entre elles rendaient ces problèmes tout aussi difficiles.

Smith *et al.* [2000] ont proposé une taxinomie de la planification donnée par la figure B.1. Ces auteurs décomposent la planification en deux champs : d'une part un champ propre à la planification dans un monde modélisé, où le temps est discrétisé, comprenant la planification classique, la planification à base de probabilités, etc. D'autre part un champ consacré à la planification dans le monde réel plus proche de l'ordonnancement, où le temps est continu, et où la résolution est basée sur des méthodes telles que la satisfaction de contraintes (CSP) ou encore la logique propositionnelle (SAT).

Dans cette annexe, nous allons étudier les différentes méthodes proposées par les articles de Zhang et Dietterich [2000] et de Dearden *et al.* [2003] au travers de la taxinomie présentée par Smith *et al.* [2000] (figure B.1). Nous allons donc présenter

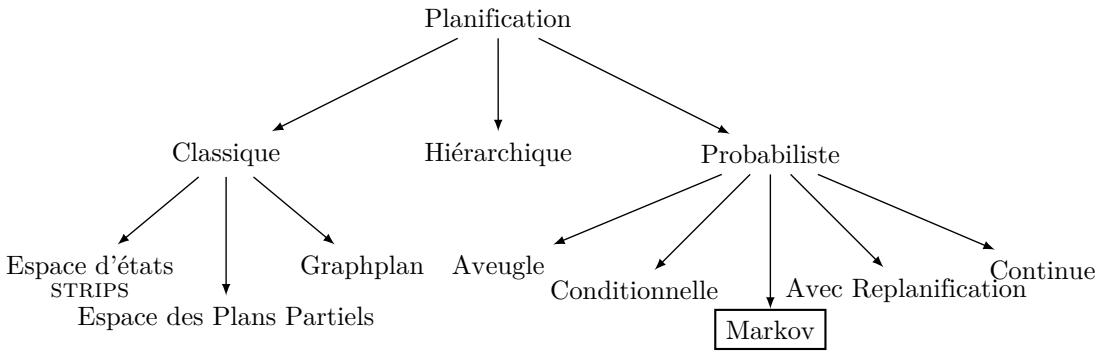


FIGURE B.1 – Taxinomie de la planification inspirée de Smith *et al.* [2000].

dans un premier temps les méthodes de planification décrites dans [Smith *et al.*, 2000] et discuter de leur capacité à gérer le temps et les ressources, puis, dans un second temps nous positionnerons les stratégies employées par Zhang et Dietterich [2000], Dearden *et al.* [2003] ainsi que celle proposée par Smith *et al.* [2000].

B.1 Planification classique

La plupart des travaux en planification des trente dernières années se positionnent essentiellement dans le cadre la planification classique. Dans un problème de planification classique, l'objectif est d'atteindre un ensemble donné de buts, usuellement représentés par des littéraux (positifs ou négatifs) de la logique propositionnelle. L'état initial du monde est, quant à lui, également représenté par des littéraux. La planification consiste alors à trouver une succession d'actions possibles faisant évoluer le monde d'un état initial à un état but. Le langage de représentation du monde généralement utilisé en planification classique est le langage STRIPS[1] basé sur les hypothèses restrictives suivantes :

- l'agent est omniscient, au sens ou l'environnement est totalement observable ; toute information est disponible et certaine à tout instant.
- les actions sont atomiques et déterministes : aucune action de l'agent ne peut être interrompue et les effets des actions sont connus avec certitude.
- l'univers est statique, au sens ou si l'agent n'agit pas, l'environnement n'évolue pas.

Ces hypothèses peuvent généralement être étendues pour permettre :

- la prise en compte du temps (synchronisation d'actions, durée des actions, ...),
- la prise en compte des ressources (énergie, materiel, ...),

1. Pour STanford Research Institute Problem Solver

– la prise en compte de l'incertitude (dynamique de l'environnement, incertitudes sur les effets des actions ou sur des états de l'environnement, ...).

Sous les hypothèses restrictives nous pouvons définir :

L'état : Tout d'abord un état e du monde de la planification est représenté par un ensemble fini de formules atomiques sans symbole de variable. Une formule atomique de base est aussi appelée un *fluent*.

L'action : Ensuite, un opérateur o représente un modèle d'action. Un tel opérateur est généralement défini par son nom et un triplet $< Prec, Add, Del >$ où $Prec$, Add et Del sont des ensembles finis de fluents. $Prec(o)$ représente les préconditions de l'opérateur o, i.e. un ensemble fini de fluents qui doivent être vérifiés pour que l'opérateur puisse être appliqué. $Add(o)$ représente les ajouts de o i.e. un ensemble fini de fluents qui sont ajoutés à l'état du monde. Finalement, $Del(o)$ représente les retraits de o, i.e. un ensemble fini de fluents qui sont retirés de l'état du monde. Une action, dénotée par a, est une instance de base d'un opérateur o (toutes les variables de o sont instanciées).

Dans ce contexte, un problème de planification est un triplet $< O, I, B >$ où :

– O dénote un ensemble fini d'opérateurs utilisables dans le domaine de la planification considéré,
– I est l'état initial du problème, il est représenté par un ensemble fini de fluents,
– B est le but du problème, il est représenté par un ensemble fini de fluents.

Par exemple, dans le monde des blocs constitué de deux blocs $B1$ et $B2$, un opérateur pourrait être $Saisir(x)$. Les actions seraient alors instanciées en $Saisir(B1)$ et $Saisir(B2)$.

Il existe dans la littérature un grand nombre de méthodes et techniques qui se sont concentrées sur la résolution de problèmes de planification classique. Notre objectif ici n'est pas de toutes les expliciter en détail, mais de couvrir les approches les plus répandues en se concentrant sur les idées principales, les avantages et les défauts de chacune d'entre elles.

B.1.1 Espaces d'états

La recherche dans les espaces d'états consiste à explorer l'ensemble des états du monde atteignables par les actions disponibles pour trouver un état où tous les buts sont vérifiés. Cette recherche correspond en fait à l'exploration d'un arbre ou les nœuds sont les états du monde et les arcs les actions appliquées pour passer d'un état à un autre (i.e les opérateurs instanciés). La recherche dans cet arbre peut se faire de deux manières : soit en partant de l'état initial du monde et en appliquant successivement les

actions possibles jusqu'a atteindre un état but, soit en régressant un état but jusqu'a atteindre l'état initial :

- Application d'une action a à un état e : $e \uparrow a$ (chaînage avant) (cf. algorithme 15) :
  - une action a est applicable sur un état e ssi $Prec(a) \subseteq e$,
  - le nouvel état est l'ensemble de fluents :
    $e \uparrow a = (e - Del(a)) \cup add(a)$
- Régression d'un état e par une action a : $e \downarrow a$ (chaînage arrière) (cf. algorithme 16) :
  - la régression d'un état (partiel) b à travers une action a est possible ssi :
    - a est une action pertinente : $add(a) \cap b \neq \emptyset$ et
    - a est une action consistante avec b : $Del(a) \cap b = \emptyset$,
  - le nouvel état (partiel) est l'ensemble de fluents :
    $b \downarrow a = (b - add(a)) \cup Prec(a)$

Ce type de recherche engendre un très grand nombre d'états à explorer pour trouver un état satisfaisant les buts. Il existe cependant un bon nombre d'algorithmes des plus naïfs (profondeur d'abord, largeur d'abord, ...) aux plus efficaces (A, A*, WA*, A** ...), ces derniers nécessitant des heuristiques plus ou moins complexes souvent basées sur la connaissance du domaine du problème, mais permettant de guider la recherche. Les planificateurs utilisant ce type d'algorithmes de recherche avec heuristique sont à ce jour les plus performants en terme de vitesse pour trouver une solution en comparaison des algorithmes décrits un peu plus loin, bien que les plans solution trouvés soient souvent plus coûteux que ceux générés par les autres algorithmes.

Un des problèmes majeurs de la recherche dans les espaces d'états est qu'elle ne permet de trouver que des plans séquentiels, totalement ordonnés, n'autorisant pas la parallélisation d'actions indépendantes [2] et ne profitant pas de la décomposition du problème.

B.1.2 Espace des Plans Partiels

La recherche dans l'espace des plans partiels a été popularisée par McAllester et Rosenblitt [1991]. Elle se base sur une recherche par chaînage arrière vue dans la section précédente : L'idée est de choisir une action qui puisse établir un des buts et de l'ajouter au plan. Le but est alors retiré de l'ensemble de buts à atteindre et remplacé par des sous-buts correspondant aux préconditions de l'action. Ce processus est répété jusqu'a ce que tous les sous-buts restants soit une sous-partie de l'ensemble des conditions initiales. Ainsi, cette approche permet de travailler sur plusieurs sous-buts indépendamment, de

2. Une définition de l'indépendance de deux actions peut être trouvée dans la section B.1.3

Algorithm 15 *ChainageAvant*($EtatCourant, PlanSolution$).

si $buts \subseteq EtatCourant$ **alors**
    retourner ($PlanSolution$)
fin si
Choisir une action A (instance d'un opérateur) t.q. :
        $Prec(A) \subseteq EtatCourant$
        et suivant une heuristique
si A n'existe pas **alors**
    Échec
fin si
Construire un nouvel état $EtatSuivant$:
        $EtatSuivant \leftarrow (EtatCourant \setminus Del(A)) \cup Add(A)$
ChainageAvant($EtatSuivant, PlanSolution \mid A$)

Algorithm 16 *ChainageArriere*($Buts, Contraintes, PlanSolution$).

si $\square \models Contraintes$ **alors** ▷ Si on ne peut satisfaire les contraintes
    Échec
fin si
si $Buts \subseteq EtatInitial$ **alors**
    retourner ($PlanSolution$)
fin si
Sélectionner un but $b \in Buts$
Choisir une action A (instance d'un opérateur) t.q. :
        $b \in Add(A)$
        et suivant une heuristique
si $A \in PlanSolution$ **alors**
    $ChainageArriere(Buts \setminus \{b\}, ContraintesMAJ, PlanSolution)$
sinon*ChainageArriere*($(Buts \cup Prec(A)) \setminus \{b\}, ContraintesMAJ, PlanSolution \mid A$)
fin si

les résoudre à l'aide de plusieurs sous-plans puis de combiner ces sous-plans.

Un plan partiel est défini comme un triplet $< OP, CO, CI >$ où :
- OP est l'ensemble des opérateurs du plan partiel,
- CO est un ordre partiel sur OP, autrement dit, c'est l'ensemble des contraintes de précédence qui lient les opérateurs deux à deux,
- CI est l'ensemble des contraintes d'instanciation des variables associées au plan.

L'espace de recherche est donc encore une fois un arbre où les nœuds sont des plans partiels et ou les arcs sont les opérations de modification de ces plans partiels :
- *Choix d'un établisseur* : On appelle une action E un établisseur ssi $\exists A$, $Add(E) \cap Prec(A) \neq \emptyset$. À partir de là, choisir un établisseur, c'est placer avant une action A une autre action E pour établir une précondition de A ;
- *Promotion d'un casseur* : On appelle une action C un casseur ssi $\exists E$, $Del(C) \cap Add(E) \neq \emptyset$. Promouvoir un casseur C, c'est le contraindre à s'exécuter avant l'établisseur E : $C \prec E \prec A$ (l'établissement de la précondition de A par E n'est pas remis en cause par C) ;
- *Séparation casseur/demandeur* : On appelle une action A un demandeur ssi un fluent des préconditions de A n'a pas encore été établi. Séparer un casseur C et un demandeur A, c'est contraindre l'instanciation de C pour qu'il ne nuise pas à A par la suite (C ne peux empêcher d'établir la précondition de A qui est concernée) ;
- *Démotion d'un casseur* : Démettre un casseur C, c'est contraindre C à s'exécuter après A : $A \prec C$ (C ne détruira pas la précondition de A créée par E) ;
- *Choix d'un white knight* : Choisir un white knight W, c'est placer entre C et A une action W pour rétablir les préconditions de A détruites par C. W n'a généralement pas de préconditions ni de retraits ($Prec(W) = \emptyset$ et $Del(W) = \emptyset$).

À l'aide de ces opérations sur les plans partiels, il devient aisé de construire un plan en instanciant les opérateurs de OP. Chaque instanciation d'opérateur est ajoutée à l'ensemble CI. Les contraintes de précédence, que ce soit pour les choix d'établisseurs ou la promotion/démotion de casseurs sont ajoutés au cours de la résolution dans l'ensemble CO. Des exemples de planification dans les espaces de plans partiels peuvent être trouvés dans [Russell et Norvig, 2009] pp. 391--393.

B.1.3 Graphplan

Graphplan est un algorithme à part des deux autres approches vues précédemment. Il se base sur un graphe de planification. Ce graphe de planification permet, en plus d'extraire un plan à l'aide de Graphplan, de fournir des informations utiles à la création d'heuristiques efficaces pour les algorithmes décrits auparavant.

Un graphe de planification est, selon Russell et Norvig [2009], une séquence de niveaux correspondant aux étapes de temps dans le plan, où le niveau zéro est l'état initial. Chaque niveau contient un ensemble de fluents et un ensemble d'actions (des instances des opérateurs). Les fluents sont ceux qui sont *possiblement* présents à cette étape du plan, les actions, celles qui ont leurs préconditions *possiblement* satisfaites à cette étape du plan. Il peut néanmoins arriver que des interactions négatives apparaissent entre les actions les empêchant de s'exécuter simultanément. On définit les interactions de la manière suivante :

Définition 27. *Interactions* positives *: Deux actions a1, a2 sont en interaction positive ssi :*

- $Add/Add : \exists f, f \in Add(a1) \cap Add(a2)$
- $Add/Prec : \exists f, f \in Add(a1) \cap Prec(a2)$

Définition 28. *Interactions* négatives *: Deux actions a1, a2 sont en interaction négative ssi :*

- *Effets antagonistes :* $\exists f, f \in Add(a1) \cap Del(a2)$ *ou*
- *Interactions croisées :* $\exists f, f \in Del(a2) \cap Prec(a1)$

Définition 29. *Interactions* indépendantes *: Deux actions a1, a2 sont indépendantes (noté a1 # a2) ssi elles n'ont pas d'interactions négatives i.e.,*

- $Del(a1) \cap (Prec(a2) \cup Add(a2)) = \emptyset$ *et*
- $Del(a2) \cap (Prec(a1) \cup Add(a1)) = \emptyset$

De la, on peut définir des fluents et des actions mutuellement exclusives :

Définition 30. *Actions* mutuellement exclusives *: Deux actions d'un même niveau dans le graphe de planification sont mutuellement exclusives (*mutex*) ssi :*

- *elles ne sont pas indépendantes ou,*
- *elles ont des préconditions* mutex *au niveau précédent (elles ne peuvent donc pas être déclenchées en même temps) (voir Définition 31) :*
  $\exists (p, q) \in Prec(a1) \times Prec(a2)$*, telles que p et q sont* mutex.

Définition 31. *Fluents* mutuellement exclusifs *: Deux fluents p et q sont* mutex *au niveau i ssi tous les couples d'actions qui les produisent à ce même niveau sont* mutex *:* $\forall a1, a2$ *t.q.* $p \in Add(a1)$, $q \in Add(a2)$, *a1 et a2* mutex.

Graphplan possède plusieurs propriétés importantes : En dehors de la possibilité de rechercher directement un plan à l'intérieur avec Graphplan, il fournit suffisamment d'informations qui permettent de produire des heuristiques très efficaces pour la recherche dans les espaces d'états ou de plans partiels. De plus il se construit de manière polynômiale en temps et en espace par rapport à la taille des données (le nombre d'actions possibles

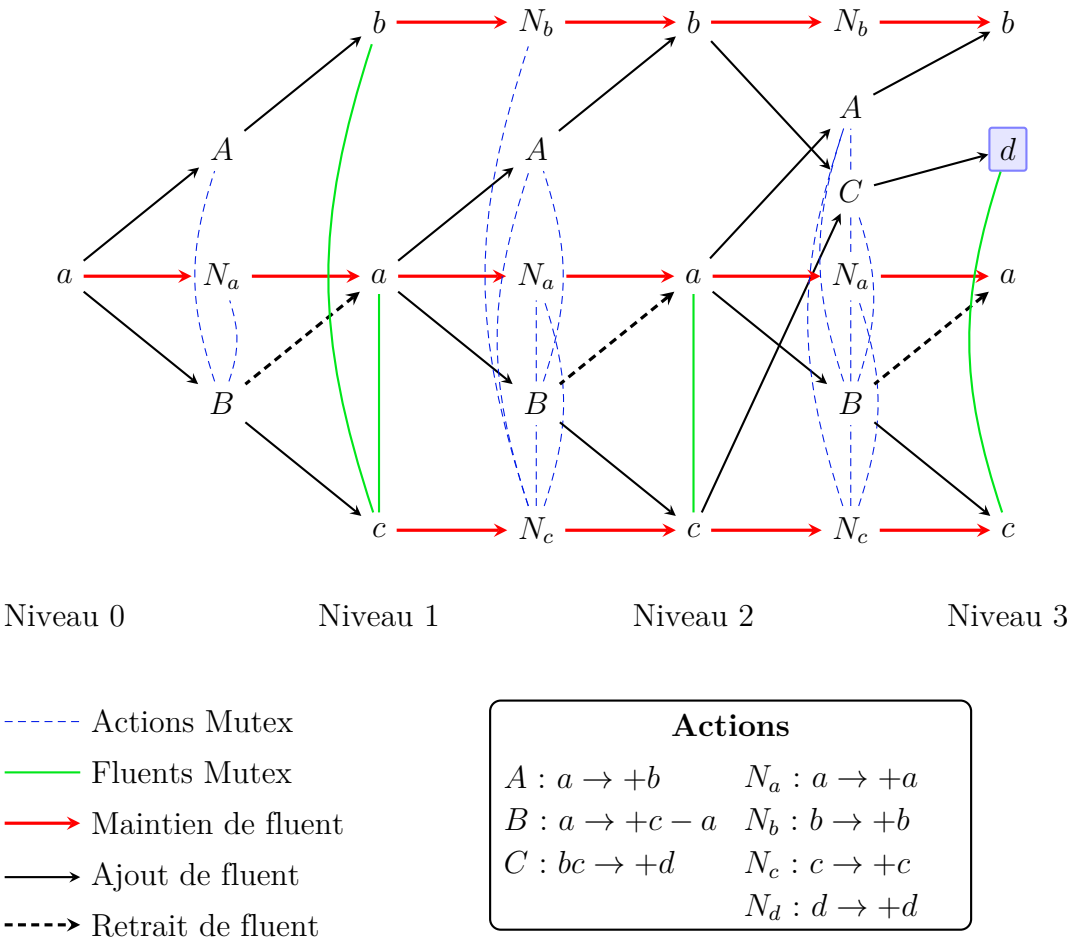


FIGURE B.2 – Un exemple de graphe de planification.

dans le plan), une propriété très intéressante dans ce domaine. Le problème est que la recherche d'un plan solution dans ce graphe de planification est, elle, exponentielle.

Un exemple de graphe de planification est fourni par la figure B.2. Le niveau 0 comprend les fluents existant à l'état initial. Au niveau 1 apparaissent les actions pouvant être exécutées ainsi que les ajouts et retraits qu'elles effectuent. Dès lors, on voit apparaître à ce niveau des actions *mutex* dues au fait que l'action B retire le fluent a de l'environnement, nécessaire à l'exécution de l'action A et à l'action N_a (action qui consiste juste à simuler que a n'a pas disparu entre deux niveaux). Par conséquent, le fluent c ajouté par B au niveau 1 se retrouve *mutex* avec les ajouts de A et N_a (resp. les fluents b et a). Ensuite, au niveau 2, toutes les actions ayant des préconditions *mutex*, se retrouvent également *mutex*. Et ainsi de suite jusqu'a ce qu'apparaisse le fluent d (notre but, encadré figure B.2) au niveau 4. Le fluent d n'a pas pu être obtenu avant car l'action C qui l'ajoute au monde possède comme précondition les fluents b et c qui

étaient *mutex* jusqu'au niveau 3. Enfin, à l'aide de l'algorithme Graphplan (Algorithme non donné ici mais basé sur un chaînage arrière donné par l'algorithme 16), on peut extraire un plan de ce graphe de planification comme indiqué en gras/encadré dans la figure B.3.

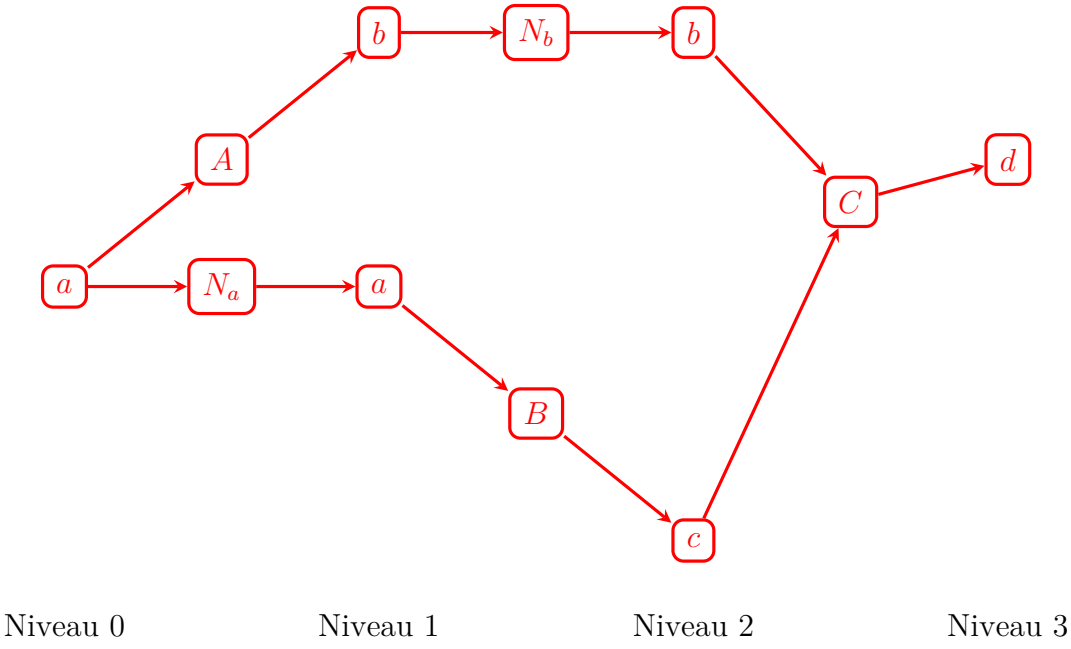


FIGURE B.3 – Extraction de plan d'un graphe de planification.

B.1.4 Discussion

Malgré leur efficacité, les algorithmes classiques précédemment évoqués, utilisables dans des environnements déterministes et totalement observables, ont de sérieuses limitations attribuées à STRIPS :

Temps : Il n'existe pas de représentation explicite du temps dans la représentation STRIPS. Il n'est pas possible de spécifier la durée d'une action ou des contraintes de temps entre les buts ou les actions.

Ressources : Rien n'est prévu dans la représentation pour spécifier les besoins en ressources ou pour modéliser la consommation de ressources.

Incertitude : STRIPS n'a pas la capacité de modéliser l'incertitude. L'état initial doit être connu entièrement et les buts, une fois atteints, sont considérés comme certains.

Néanmoins, de nombreuses extensions de STRIPS existent. Tant pour la modélisation du temps [Smith et Weld, 1999; Penberthy et Weld, 1994] que pour la gestion de

nombreux types de ressources [Kautz et Walser, 1999; Koehler, 1998] ou de l'incertitude [Weld *et al.*, 1998; Draper *et al.*, 1994]. Seulement, ce type d'extensions dégénèrent significativement les performances [Smith *et al.*, 2000].

Ceci dit, si l'on ne considère que les méthodes intrinsèquement, depuis le système TWEAK de Chapman [1987] et jusqu'aux formalisations de Kambhampati et Srivastava [1996], l'approche de la planification était essentiellement fondée sur les stratégies de raffinements dans les espaces de plans partiels, les autres approches étant très marginalisées au sein de la communauté.

En 1995 cependant, l'apparition du planificateur Graphplan fut à l'origine d'un bouleversement de cet ordre bien établi. Les idées qui guidaient sa conception ont initié des travaux qui ont permis aux algorithmes de planification d'augmenter leurs performances de manière si importante que l'on peut maintenant commencer à envisager des applications réelles. Par un curieux retour des choses, la planification par recherche heuristique dans les espaces d'états s'est (re)trouvée être très performante avec l'apparition de planificateurs utilisant des heuristiques inspirées par Graphplan. Elle permet actuellement de résoudre des problèmes qui étaient très largement hors de portée des planificateurs il y a seulement quelques années, mais toujours dans le cadre des limitations de STRIPS.

Plusieurs méthodes algorithmiques, issues d'autres domaines classiques de l'IA sont maintenant en concurrence : recherche heuristique dans les espaces d'états, dans les espaces de plans partiels, Graphplan, planification SAT, planification CSP, recherche locale, etc. Les différents planificateurs qui en découlent font l'objet d'évaluations comparatives régulières dans le cadre des compétitions IPC (*International Planning Competition*) des conférences AIPS, et maintenant ICAPS.

Il est à noter que nous avons délibérément omis de citer les méthodes de planification SAT pour la simple mais excellente raison que la représentation de contraintes temporelles en logique propositionnelle augmente considérablement le nombre de clauses SAT et rend la résolution proprement impossible. Il en est tout de même fait état dans [Smith *et al.*, 2000] car ce type de modélisation permet aisément la représentation des ressources et l'optimisation. Dans un contexte d'utilisation de la planification sur des horizons infinis ou en temps continu, nous ne pouvons pas prendre en compte une approche complexifiant à ce point la représentation du temps.

B.2 Planification hiérarchique

La plupart des planificateurs développés dans le cadre d'applications réelles utilisent les réseaux de hiérarchisation de tâches (HTN pour *Hierarchical Task Network*). La principale différence avec la planification classique et est que cette dernière essaie de décomposer des tâches de haut niveau en tâches de plus bas niveau là où la planification classique cherche juste à assembler des actions pour atteindre des buts. De plus, un but est plutôt spécifié, dans la planification hiérarchique, comme une tâche de haut niveau, que comme un ensemble de littéraux à obtenir.

L'algorithme de planification consiste donc à décomposer chaque tâche en tâches élémentaires, tout en vérifiant qu'elles n'aient pas d'interactions négatives entre elles. La planification se termine lorsque le réseau ne contient que des tâches élémentaires et lorsque l'ensemble des contraintes d'ordre relatives à la gestion des conflits est consistant (i.e. qu'il n'existe pas de boucles). L'algorithme 17 est une simplification en pseudo-code de la planification hiérarchique.

Algorithm 17 HTN $- Plan(Plan)$

si $Plan$ contient des conflits **alors**
    si il n'existe pas de manière de résoudre les conflits **alors**
        Échec
    sinonChoisir une manière de résoudre les conflits et l'appliquer
    fin si
fin si
si $Plan$ ne contient que des tâches élémentaires **alors**
    retourner $(Plan)$
fin si
Sélectionner une tâche non élémentaire $t \in Plan$
Choisir une décomposition E de t
$NouveauPlan \leftarrow$ Remplacer t par E dans $Plan$
HTN $- Plan(NouveauPlan)$

Le temps et la gestion de données métriques ne posent pas de réelles difficultés pour les planificateurs HTN. Ces contraintes peuvent être directement spécifiée à l'intérieur des tâches ou alors évaluées pas le biais de test de consistance de l'algorithme. Seulement, selon Smith *et al.* [2000], les planificateurs HTN essuient pour le moment trois critiques importantes :

Sémantique : Il n'existe pas de sémantique bien définie pour la décomposition de tâche, cela a pour conséquence de rendre difficile tout jugement de la complétude ou de la consistance d'un plan.

Conception : La conception de ce type de planificateur nécessite de prévoir et d'analyser toutes les tâches possiblement existantes et décomposables. Il est vraiment difficile d'une part, de produire la liste exhaustive de toutes les décompositions d'une tâche, mais d'autre part, à chaque ajout de fonctionnalité au système, il faudra prévoir et lister toutes les nouvelles décompositions à ajouter pour utiliser au mieux les fonctionnalités du système.

Fragilité : Les planificateurs HTN sont loins d'être robustes. En effet, ils sont incapables de prendre en compte des tâches non explicitement prévues par le concepteur, même si les tâches élémentaires sont suffisantes pour construire un plan correct.

B.3 Planifications probabilistes

Jusqu'à présent nous avons considéré que des domaines de la planification classique totalement observables, statiques et déterministes. De plus nous avons considéré que la description des actions était correcte et complète. Dans ces circonstances, un agent peut, après avoir élaboré un plan, l'exécuter sans aucune surprise d'aucunes sorte. En revanche, dans un environnement incertain, cet agent devra utiliser ses moyens de perception pour analyser et éventuellement anticiper en modifiant ou en adaptant son plan en cours d'exécution. Les méthodes utilisées pour réagir dans ces types d'environnements utilisant les probabilités d'occurrence d'événements incertains pour les prévoir et les modéliser, sont qualifiées de probabilistes. Une autre classe de méthodes utilisant la logique pour modéliser l'incertitude ne sera pas étudiée ici puisqu'elle n'intervient pas dans la catégorisation des techniques intéressantes pour notre thèse.

Il existe selon Russell et Norvig [2009] quatre méthodes de planification probabilistes, les deux premières sont plus attachées à résoudre les problèmes à indétermination limitée (dans le sens où les actions ont un nombre fixe d'effets définis ayant chacun sa probabilité d'occurrence), les deux suivantes s'attachent plus à l'indétermination pure (dans le sens ou les actions ont des effets en trop grand nombre ou inconnus) :

- *Planification Aveugle* : Aussi connue sous le nom de planification *conformante*, cette méthode construit un plan standard séquentiel qui doit pouvoir être exécuté sans perceptions. Le plan construit doit pouvoir assurer d'atteindre les buts voulus dans toutes les circonstances possibles en regard des certitudes sur l'état initial et des sorties actuelles des actions.
- *Planification Conditionnelle* : Aussi connue sous le nom de planification contingente, cette approche ajoute des branches conditionnelles au plan aux endroits ou pourraient survenir des contingences. L'agent choisit quelle branche du plan exécuter en utilisant ses percepts comme test conditionnel.

- *Supervision d'exécution et replanification* : Dans cette approche, un agent peut utiliser toutes les techniques vues précédemment (classique, aveugle ou contingente) pour produire un plan, mais ajoute à cela une supervision de l'exécution. La replanification intervient lorsque l'exécution ne se passe pas comme prévu.
- *Planification continue* : Tous les planificateurs vus ci-dessus cont conçus pour atteindre un but et s'arrêter. Un planificateur continu est conçu pour durer indéfiniment. Il est capable de supporter les imprévus et même jusqu'a l'abandon des buts courants en utilisant des mécanismes de reformulation des buts.

Il existe également toute la planification sous incertain représentée essentiellement par les modèles de Markov et détaillée au chapitre 2 que nous ne présenterons donc pas ici.

Annexe C

Filtres à particules appliqués aux POMDPs

Dans cette annexe nous présentons un peu plus en détail les filtres a particules et leur application aux POMDPs *pour la représentation des états de croyance. Chaque étape de filtrage est présentée selon les travaux détaillés sur le sujet par Thrun [1999].*

Les simulations Monte-Carlo, bien qu'efficaces, sont surtout reconnues pour consommer beaucoup de temps de calcul. Cependant, de récents travaux ont considérablement réduit les temps de simulation. Ces techniques, appelées *filtres à particules*, permettent la simulation en parallèle de plusieurs essais Monte-Carlo séquentiels tout en utilisant une taille fixe de la mémoire et tout en assurant des garanties de performance sur l'estimation de la distribution de probabilité postérieure.

Thrun [1999] a appliqué la technique des filtres à particules avec succès aux POMDPs. Ceux-ci étaient utilisés comme variante à base d'échantillons des filtres de Bayes pour récursivement estimer la densité postérieure sur un état s -- l'état de croyance $\boldsymbol{b}$ -- du système dynamique [Fox *et al.*, 2001] :

$$\boldsymbol{b}^{t+1}(s') \propto \mathcal{O}(o|a,s') \int \mathcal{T}(s'|s,a)\boldsymbol{b}^{t}(s)\,\mathrm{d}s$$

Où, o est l'observation reçue et a l'action effectuée.

C.1 Représentation à base d'échantillons

La représentation usuelle d'un état de croyance sur un espace d'état discret est généralement un vecteur $\boldsymbol{b}$ qui décrit pour chaque état s la probabilité de se retrouver dans celui-ci. Lorsque l'espace d'état est trop grand, voire continu, cette représentation consomme beaucoup trop d'espace et nécessite alors d'être approximée.

Dans ce contexte, le filtre à particules représente l'état de croyance par un ensemble S de N particules (potentiellement pondérées). Chaque $s_{(i)}$ est un échantillon représentant un état tirée depuis la distribution originale $\boldsymbol{b}$.

Comme montré par la figure C.1, il existe deux types populaires d'approximations à base d'échantillons : *l'échantillonnage pondéré par la vraisemblance* pour lequel les points sont tirés directement de la distribution à approximer (notée f dans la figure C.1(a)), et *l'échantillonnage par importance*, où les échantillons sont tirés depuis une autre distribution, telle que celle notée g dans la figure C.1(b). Dans ce dernier cas, un facteur d'importance est associé aux échantillons x

$$p(x) = \frac{f(x)}{g(x)}$$

pour prendre en compte la différence entre la distribution échantillonnée, g, et la distribution approximée f (dans la figure C.1, les hauteurs des barres indiquent ce facteur d'importance). Il est possible de montrer que ces approximations convergent vers la distribution souhaitée à un taux de $1/\sqrt{N}$ [Tanner, 1993].

Dans le chapitre 5 de cette thèse, nous avons choisi d'utiliser la représentation à base de facteur d'importance puisque c'est celui qui a été le plus étudié dans la littérature robotique [Kwok *et al.*, 2003]. L'état de croyance est donc représenté par un ensemble S de N particules pondérées $\langle s^{(i)}, w^{(i)} \rangle$, où les $w^{(i)}$ sont des réels non négatifs, appelés *facteurs d'importance*, qui somment à un. Voyons maintenant en détail fonctionnement d'un filtre particulaire.

C.2 Filtrage bayésien

Dans sa forme basique, le filtre à particule à échantillonnage d'importance réalise un filtrage bayésien récursif selon une procédure d'échantillonnage souvent référée par l'*échantillonnage séquentiel par importance avec rééchantillonnage* (SISR). Cette

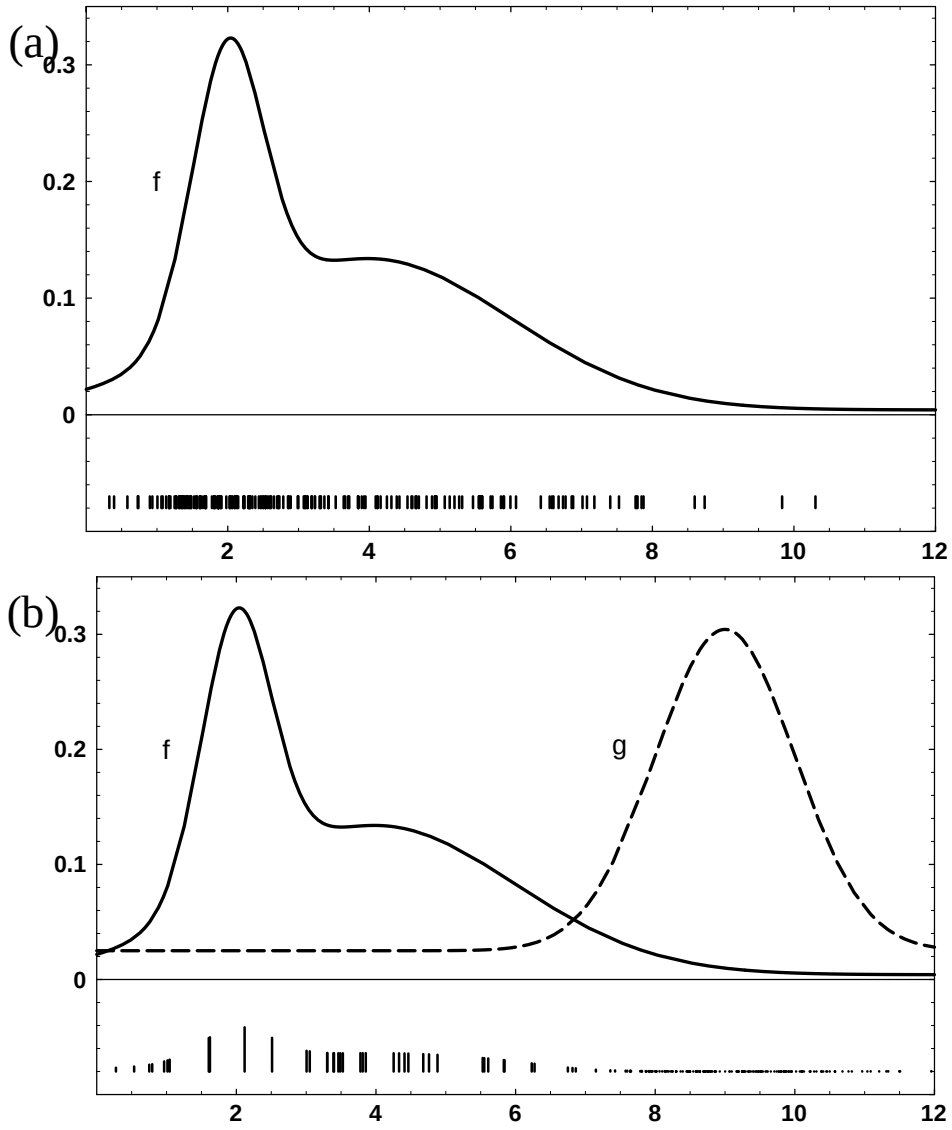


FIGURE C.1 – Échantillonnage : (a) pondéré sur la vraisemblance et (b) pondéré par l'importance. En bas de chaque courbe sont représentés les échantillons qui approximent la distribution f. La hauteur des échantillons exprime leur *facteur d'importance* [Thrun, 1999].

méthode se décompose en trois étapes illustrées par la figure C.2 et que nous allons détailler par la suite : (1) la prédiction, (2) la correction, et (3) le rééchantillonnage.

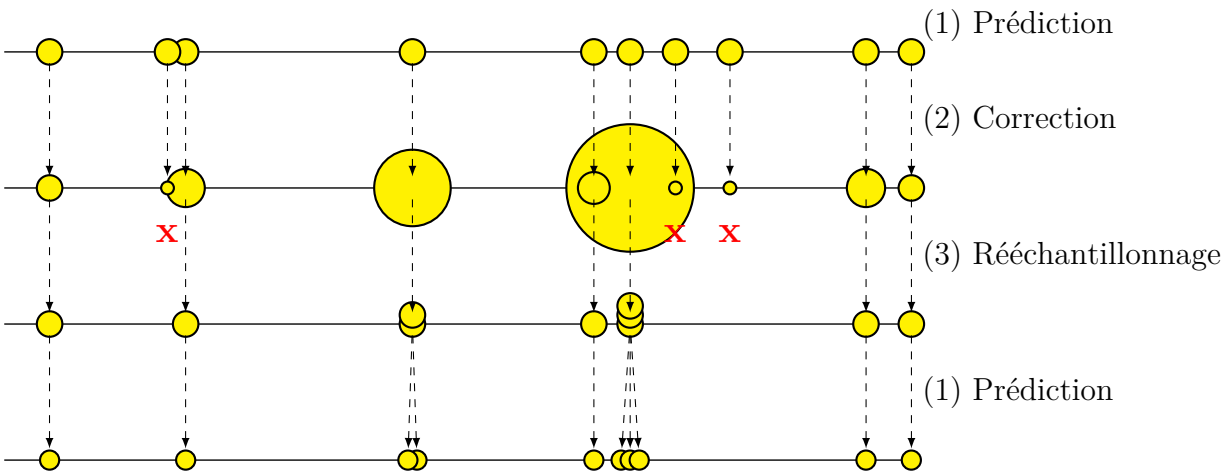


FIGURE C.2 – Filtrage bayésien par échantillonnage d'importance. L'étape (1) est la *prédiction*, la (2) est la correction, et la (3) le rééchantillonnage.

C.2.1 Prédiction

Cette étape, utilise l'état courant s et l'action[1] a pour échantillonner l'état suivant s' selon la fonction de transition $\mathcal{T}(s'|s,a)$, qui décrit la dynamique du système selon l'interaction de l'agent.

Dans notre cas de POMDP, puisque seulement la distribution courante $\boldsymbol{b}^t$ sur les états est disponible au travers d'une approximation, cette prédiction est faite pour chaque particule s_i^t en échantillonnant la distribution de probabilité de l'état futur s^{t+1} étant donné l'état s_i^t représenté par la particule i et l'action entreprise par l'agent a^t à l'étape t :

$$s_i^{t+1} \sim \mathcal{T}(s_i^{t+1}|a^t, s_i^t)$$

C.2.2 Correction

En plus de l'estimation qu'à l'agent d'où peut le conduire son action, celui reçoit également une observation lui donnant un peu plus d'information quant à l'état sous-jacent de son environnement. À l'aide de cette observation l'agent peut alors corriger sa prédiction. Cette étape pondère donc l'échantillon s' par la vraisemblance de l'observation $w' = \mathcal{O}(o|s,a,s')$. Cette vraisemblance est extraite de la fonction d'observation de l'agent (e.g. du modèle de ses capteurs et de l'environnement).

1. Il est également possible d'utiliser une distribution sur les actions plutôt que l'action seule.

Encore une fois, dans le cas du POMDP, le facteur d'importance w_i^{t+1} de la particule prédite s_i^{t+1} est calculé directement à partir de la vraisemblance d'obtenir l'observation réellement perçue o^t, sachant que l'état précédent était s_i^t et que l'action effectuée est a^t à l'étape t :

$$w_i^{t+1} = \mathcal{O}(o^t | s_i^t, a^t, s_i^{t+1})$$

C.2.3 Rééchantillonnage

Une fois la correction effectuée, il faut maintenant prendre en compte cette correction pour permettre à nouveau de faire une prédiction à partir de cette nouvelle distribution. Il est donc nécessaire de représenter la nouvelle distribution avec des échantillons ayant tous un facteur d'importance identique.

Pour cela, l'algorithme effectue N tirages selon la distribution discrète définie par la pondération $w^{(i)}$ appliquée à l'étape précédente puis remplace l'ancien ensemble de particules par le nouveau. Le nouvel état de croyance est alors représenté par un nouvel ensemble de particules, certaines ayant un grand facteur d'importance étant représentées plusieurs fois (comme illustré à l'étape (3) de la figure C.2). Cela permet de renforcer la croyance selon la pondération de l'étape précédente tout en gardant un nombre fixe d'échantillons, garantissant ainsi l'espace utilisé par la représentation.

Annexe D

Processus gaussiens appliqués aux POMDPs

Dans cette annexe nous détaillons plus avant les recherches effectuées en collaboration avec l'étudiant à la maîtrise Patrick Dallaire dans le cadre de l'apprentissage de POMDP*s continus quasi-déterministes. Dans une première section les* POMDP*s continus sont détaillés avant de présenter les* POMDP*s à base de processus gaussiens. Des expérimentations sont ensuite présentées démontrant la faisabilité de l'approche.*

D.1 POMDP continu

Les POMDPs continus sont l'extension naturelle des POMDPs où l'espace des états, des actions et des observations sont des espaces continus à respectivement m, n, et p dimensions. De fait les fonctions de transitions et d'observations sont alors des fonctions continues définissant des densités de probabilité plutôt que des distributions. Plus formellement, un POMDP continu est défini par un tuple $(\mathcal{S}, \mathcal{A}, \Omega, \mathcal{T}, \mathcal{O}, \mathcal{R}, b_1, \gamma)$, avec :

- $\mathcal{S} \subset \mathbb{R}^m$: L'espace d'états, continu et potentiellement multidimensionnel.
- $\mathcal{A} \subset \mathbb{R}^n$: L'espace d'actions, continu et potentiellement multidimensionnel. Nous supposons ici que $\mathcal{A}$ est un sous-ensemble fermé de $\mathbb{R}^n$, afin que le contrôle optimal défini plus bas existe.
- $\Omega \subset \mathbb{R}^p$: L'espace d'observations, continu et potentiellement multidimensionnel.
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, \infty]$: La fonction de transition qui détermine la densité de probabilité conditionnelle $\mathcal{T}(\mathbf{s}, \mathbf{a}, \mathbf{s}') = p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ de se déplacer vers l'état $\mathbf{s}'$, lorsque l'agent exécute l'action $\mathbf{a}$ dans l'état $\mathbf{s}$.

- $\mathcal{O} : \mathcal{S} \times \mathcal{A} \times \Omega \to [0, \infty]$: La fonction d'observation qui détermine la densité de probabilité conditionnelle $O(\mathbf{s}', \mathbf{a}, \mathbf{z}') = p(\mathbf{z}|\mathbf{s}', \mathbf{a})$ d'observer $\mathbf{z}'$ lorsque l'agent atteint l'état $\mathbf{s}'$ suite à l'exécution de l'action $\mathbf{a}$.
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathbb{R} \to [0, \infty]$: La fonction de récompense qui détermine la densité de probabilité conditionnelle $R(\mathbf{s}', \mathbf{a}, r') = p(r'|\mathbf{s}', \mathbf{a})$ de recevoir la récompense r', lorsque l'agent atteint l'état $\mathbf{s}'$ suite à l'exécution de l'action $\mathbf{a}$.
- $b_1 \in \Delta\mathcal{S}$: La distribution de l'état initial.
- γ : Le facteur d'escompte.

L'état de croyance b, qui est la densité de probabilité a posteriori sur l'état courant $\mathbf{s}$ de l'agent, peut toujours être maintenu à jour par l'utilisation de la règle de Bayes comme suit :

$$b^{\mathbf{az}}(\mathbf{s}') \propto \mathcal{O}(\mathbf{s}', \mathbf{a}, \mathbf{z}') \int_{\mathcal{S}} \mathcal{T}(\mathbf{s}, \mathbf{a}, \mathbf{s}') b(\mathbf{s}) d\mathbf{s} \tag{D.1}$$

Le comportement de l'agent est ainsi déterminé par une politique π qui, pour tout état de croyance b possible, précise l'action à exécuter. La politique optimale, noté π^*, est celle qui permet de maximiser la somme des récompenses escomptées espérées sur un horizon infini. Une telle politique optimale est obtenue en résolvant l'équation de Bellman :

$$V^*(b) = \max_{\mathbf{a} \in A} \left[g(b, \mathbf{a}) + \gamma \int_Z f(\mathbf{z}|b, \mathbf{a}) V^*(b^{\mathbf{az}}) d\mathbf{z} \right] \tag{D.2}$$

où V^* est la fonction de valeur de la politique optimale et est aussi le point fixe de l'équation de Bellman. De plus,

$$g(b, \mathbf{a}) = \int_S b(\mathbf{s}) \int_S T(\mathbf{s}, \mathbf{a}, \mathbf{s}') \int_{\mathbb{R}} r R(\mathbf{s}', \mathbf{a}, r) dr d\mathbf{s}' d\mathbf{s}$$

est la récompense espérée lorsque l'action $\mathbf{a}$ est exécutée dans l'état de croyance b ; et

$$f(\mathbf{z}|b, \mathbf{a}) = \int_S O(\mathbf{s}', \mathbf{a}, \mathbf{z}) \int_S T(\mathbf{s}, \mathbf{a}, \mathbf{s}') b(s) d\mathbf{s} d\mathbf{s}'$$

est la densité de probabilité conditionnelle d'observer $\mathbf{z}$ suite à l'exécution de l'action $\mathbf{a}$ dans l'état de croyance b. Pour obtenir l'état de croyance suivant, noté $b^{\mathbf{az}}$, il suffit de faire une mise à jour de b en appliquant la règle de Bayes (précédemment introduite par l'équation (D.1)) avec l'observation $\mathbf{z}$ et l'action $\mathbf{a}$.

D.2 GP-POMDP

Dans cette annexe, nous considérons le problème d'apprendre la politique optimale dans un POMDP continu, tel que décrit à la section précédente, lorsque les fonctions $\mathcal{T}$, $\mathcal{O}$ et $\mathcal{R}$ ne sont pas connues a priori. Pour envisager la prise de décisions optimales, le

modèle est appris par modélisation via les processus gaussiens (GPs) [O'Hagan, 1992; Williams et Barber, 1998], et ce, uniquement à partir de séquences d'action-observation. Les GPs sont une classe de modèle probabiliste qui met l'emphase sur les points où une fonction est instanciée, en utilisant la distribution gaussienne sur l'espace des fonctions. Généralement, la distribution gaussienne est paramétrée par un vecteur de moyenne et une matrice de covariance, mais dans le cas des GPs, ces deux paramètres sont des fonctions de l'espace sur lesquels ils opèrent.

Pour notre problème de contrôle optimal, nous proposons d'utiliser un *Modèle Dynamique par Processus gaussien* (GPDM) afin d'apprendre les fonctions de transition, observation et récompense, et proposons ensuite un algorithme de planification en ligne choisissant les actions qui maximisent à long terme les récompenses espérées en fonction du modèle courant.

Afin d'utiliser une modélisation par processus gaussien pour apprendre les fonctions $\mathcal{T}$, $\mathcal{O}$ et $\mathcal{R}$ du modèle POMDP, il faut d'abord faire l'hypothèse que les dynamiques du système peuvent être exprimées sous la forme quasi-déterministe suivante :

$$\begin{aligned} \mathbf{s}_t &= \mathcal{T}'(\mathbf{s}_{t-1}, \mathbf{a}_{t-1}) + \epsilon_T \\ \mathbf{z}_t &= \mathcal{O}'(\mathbf{s}_t, \mathbf{a}_{t-1}) + \epsilon_O \\ r_t &= \mathcal{R}'(\mathbf{s}_t, \mathbf{a}_{t-1}) + \epsilon_R \end{aligned} \tag{D.3}$$

où ϵ_T, ϵ_O et ϵ_R sont des variables de bruit blanc gaussiens à moyenne nulle. Les fonctions $\mathcal{T}'$, $\mathcal{O}'$ et $\mathcal{R}'$ sont des fonctions déterministes qui retournent respectivement l'état suivant, l'observation associée à ce dernier et la récompense obtenue. L'objectif est donc d'apprendre ce modèle et de maintenir une estimation par maximum de vraisemblance sur la trajectoire des états en utilisant un GPDM avec des méthodes d'optimisation.

Pour simplifier la lecture, nous utiliserons une notation matricielle où $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_{N+1}]^\top$ représente la matrice de séquence d'état, $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N]^\top$ la matrice de séquence d'action, $\mathbf{Z} = [\mathbf{z}_2, \mathbf{z}_3, \ldots, \mathbf{z}_{N+1}]^\top$ pour les observations et $\mathbf{r} = [r_2, r_3, \ldots, r_{N+1}]^\top$ le vecteur contenant les récompenses obtenues.

D.2.1 Modèle Dynamique par Processus gaussien

Un GPDM consiste en une fonction dont l'ensemble de départ est un espace latent et l'espace d'arrivée est celui des observations. Ce modèle probabiliste, dans le contexte du POMDP, est défini tel que $\mathcal{S} \times \mathcal{A} \to \Omega \times \mathcal{R}$. Il représente donc la fonction d'observation-récompense où l'action est complètement observable et l'état est une variable latente. Une

seconde fonction permet de régir les dynamiques, qui sont par hypothèse Markovienne du premier ordre, dans cet espace latent. Le modèle des dynamiques est défini tel que $\mathcal{S} \times \mathcal{A} \to \mathcal{S}$ et tel qu'il corresponde à la fonction de transition du POMDP. Ces deux fonctions du GPDM sont définies par des combinaisons linéaires de fonctions de base non linéaires :

$$\mathbf{s}_t = \sum_i \mathbf{b}_i \phi_i(\mathbf{s}_{t-1}, \mathbf{a}_{t-1}) + \mathbf{n}_s \tag{D.4}$$

$$\mathbf{y}_t = \sum_j \mathbf{c}_j \psi_j(\mathbf{s}_t, \mathbf{a}_{t-1}) + \mathbf{n}_y \tag{D.5}$$

où $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \dots]$ et $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \dots]$ sont les poids des fonctions des base ϕ_i et ψ_j, $\mathbf{n}_s$ et $\mathbf{n}_y$ sont des bruits blancs gaussiens. L'espace d'observation-récompense conjoint est noté par $Y = \Omega \times \mathcal{R}$ et, par conséquent, $\mathbf{y}_t$ est le vecteur d'observation $\mathbf{z}_t$ augmenté de la récompense r_t. Pour suivre la méthodologie bayésienne, les paramètres inconnus doivent être marginalisés par intégration. Ceci peut être fait sous forme analytique [Mackay, 2003; Neal, 1996] en appliquant une distribution gaussienne isotropique a priori sur les colonnes de $\mathbf{C}$ et ainsi mener à la fonction de vraisemblance gaussienne multivariée :

$$p(\mathbf{Y} \mid \mathbf{S}, \mathbf{A}, \bar{\alpha}) = \frac{|\mathbf{W}|^N}{\sqrt{(2\pi)^{N(p+1)} |\mathbf{K}_Y|^{(p+1)}}} \exp\left(-\frac{1}{2}\mathrm{tr}(\mathbf{K}_Y^{-1}\mathbf{Y}\mathbf{W}^2\mathbf{Y}^\top)\right) \tag{D.6}$$

où $\mathbf{Y} = [\mathbf{y}_2, \mathbf{y}_3, \dots, \mathbf{y}_{N+1}]^\top$ est la séquence des observations-récompenses conjointes, $\mathbf{K}_Y$ est une matrice de noyau calculé à partir des hyperparamètres $\bar{\alpha} = \{\alpha_1, \alpha_2, \dots, \mathbf{W}\}$. La matrice $\mathbf{W}$ est diagonale et contient $p+1$ facteurs de mise à l'échelle afin de tenir compte des différentes variances entre les dimensions d'observation. En utilisant une seule fonction de noyau pour l'état et l'action, les éléments de la matrice de noyau sont $(\mathbf{K}_Y)_{i,j} = k_Y([\mathbf{s}_i, \mathbf{a}_{i-1}], [\mathbf{s}_j, \mathbf{a}_{j-1}])$. Notez que pour le cas de l'observation, l'action correspond à celle du pas de temps précédent. Le noyau choisi pour cette fonction est, avec $\mathbf{x}_i = [\mathbf{s}_i, \mathbf{a}_{i-1}]$, la fonction radiale de base (RBF) standard :

$$k_Y(\mathbf{x}, \mathbf{x}') = \alpha_1 \exp\left(-\frac{\alpha_2}{2}\|\mathbf{x} - \mathbf{x}'\|^2\right) + \alpha_3^{-1}\delta_{\mathbf{x}\mathbf{x}'} \tag{D.7}$$

où l'hyperparamètre α_1 représente la variance en sortie, α_2 est l'inverse de la largeur de la RBF qui représente à quel point la fonction est lisse et α_3^{-1} est la variance du bruit additif $\mathbf{n}_y$.

Pour ce qui est de la modélisation des dynamiques dans l'espace latent, la méthode est similaire, mais nécessite l'application de la propriété de Markov. En utilisant une distribution gaussienne isotropique a priori sur les colonnes de $\mathbf{B}$, l'intégration peut se faire sous forme analytique. Il en résulte la densité de probabilité suivante sur les trajectoires latentes :

$$p(\mathbf{S} \mid \mathbf{A}, \bar{\beta}) = \frac{p(\mathbf{s}_1)}{\sqrt{(2\pi)^{N(m+n)} |\mathbf{K}_X|^{m+n}}} \exp\left(-\frac{1}{2}\mathrm{tr}(\mathbf{K}_X^{-1}\mathbf{S}_{out}\mathbf{S}_{out}^\top)\right) \tag{D.8}$$

où $p(\mathbf{s}_1)$ est la distribution initiale sur l'état b_1 qui est par hypothèse isotropique gaussienne, $\mathbf{S}_{out} = [\mathbf{s}_2, \ldots, \mathbf{s}_{N+1}]^\top$ est la matrice des coordonnées latentes correspondant aux états cachés. La matrice de noyaux $\mathbf{K}_X$ de taille $N \times N$ est construite à partir de $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top$ où $\mathbf{x}_i = [\mathbf{s}_i, \mathbf{a}_i]$ avec $1 \leqslant i \leqslant t-1$. La fonction de noyau choisi pour la modélisation des dynamiques est :

$$k_X(\mathbf{x}, \mathbf{x}') = \beta_1 \exp\left(-\frac{\beta_2}{2} \|\mathbf{x} - \mathbf{x}'\|^2\right) + \beta_3 \mathbf{x}^\top \mathbf{x}' + \beta_4^{-1} \delta_{\mathbf{x}\mathbf{x}'} \tag{D.9}$$

où la mise à l'échelle du terme linéaire est représentée par le paramètre additionnel β_3. Comme tous les hyperparamètres sont inconnus, nous suivons la démarche de Wang *et al.* [2005] et appliquons les distributions non informatives $p(\bar{\alpha}) \propto \prod_i \alpha_i^{-1}$ et $p(\bar{\beta}) \propto \prod_i \beta^{-1}$ a priori sur les hyperparamètres. Il en résulte une interprétation complètement probabiliste des séquences d'action-observation :

$$p(\mathbf{Z}, \mathbf{r}, \mathbf{S}, \bar{\alpha}, \bar{\beta} | \mathbf{A}) = p(\mathbf{Y} | \mathbf{S}, \mathbf{A}, \bar{\alpha}) p(\mathbf{S} | \mathbf{A}, \bar{\beta}) p(\bar{\alpha}) p(\bar{\beta}) \tag{D.10}$$

Pour réaliser l'apprentissage d'un GPDM, Wang *et al.* ont proposé de minimiser la log distribution jointe a posteriori négative $-\ln p(\mathbf{S}, \bar{\alpha}, \bar{\beta}, \mathbf{W} | \mathbf{Z}, \mathbf{r}, \mathbf{A})$ par rapport aux paramètres inconnus et qui est défini par [Wang *et al.*, 2008] :

$$\begin{aligned} \mathcal{L} = &\frac{m+n}{2} \ln |\mathbf{K}_X| + \frac{1}{2} \mathrm{tr}(\mathbf{K}_X^{-1} \mathbf{S}_{out} \mathbf{S}_{out}^\top) + \frac{1}{2} \mathbf{s}_1^\top \mathbf{s}_1 - N \ln |\mathbf{W}| \\ &+ \frac{p+1}{2} \ln |\mathbf{K}_Y| + \frac{1}{2} \mathrm{tr}(\mathbf{K}_Y^{-1} \mathbf{Y} \mathbf{W}^2 \mathbf{Y}^\top) + \sum_i \ln \alpha_i + \sum_i \ln \beta_i + C \end{aligned} \tag{D.11}$$

Pour plus de détails sur les méthodes d'apprentissage du GPDM, nous référons le lecteur intéressé à Wang *et al.* [2008]. Le résultat de la phase d'apprentissage permet d'obtenir une estimation du maximum de vraisemblance a posteriori (MAP) de $\{\mathbf{S}, \bar{\alpha}, \bar{\beta}, \mathbf{W}\}$. Celui-ci est ensuite utilisé pour estimer les fonctions de transition et d'observation-récompense :

$$\mathbf{s}_{t+1} = \mathbf{k}_X([\mathbf{s}_t, \mathbf{a}_t])^\top \mathbf{K}_X^{-1} \mathbf{S}_{out} \tag{D.12}$$

$$\mathbf{y}_{t+1} = \mathbf{k}_Y([\mathbf{s}_{t+1}, \mathbf{a}_t])^\top \mathbf{K}_Y^{-1} \mathbf{Y} \tag{D.13}$$

où $\mathbf{k}_X$ et $\mathbf{k}_Y$ sont des vecteurs de covariance entre la donnée test et les données contenues dans leurs ensembles d'entraînement respectifs. Les vecteurs de covariances sont calculés en utilisant les hyperparamètres $\bar{\alpha}$ pour le modèle d'observations et $\bar{\beta}$ pour le modèle de transition. De plus, comme $\mathbf{S}$ fait partie intégrante de l'estimation MAP, la séquence d'états obtenus est considérée la plus probable en fonction du modèle courant. Par conséquent, le dernier élément de cette matrice correspond à la meilleure estimation que l'on a de l'état courant. L'état de croyance complet de l'agent, conditionné uniquement par les observations, actions et récompenses, est donc déterminé par l'estimation MAP du modèle ainsi que la trajectoire $\mathbf{S}$ dans l'espace latent.

Une fois le modèle $\mathcal{T}$, $\mathcal{O}$ et $\mathcal{R}$ appris, une simple planification approximative en ligne à partir de ce modèle est effectuée pour estimer quelle est la meilleure action à entreprendre. Voyons maintenant quels sont les résultats expérimentaux.

D.3 Expérimentations

Pour valider notre approche, en particulier l'apprentissage du modèle GP-POMDP, nous avons convenu de contrôler un dirigeable en ligne sans connaître le modèle physique de celui-ci. Nous avons choisi le dirigeable, car, comparativement aux autres appareils, il présente l'avantage d'opérer à une vitesse relativement lente et peut conserver son altitude sans nécessairement avoir à se déplacer.

Le but des expérimentations que nous avons faites est de valider l'utilisation des GPDMs pour des fins d'identification de modèle en ligne et d'estimation de l'état. L'ajout d'un algorithme de planification en ligne nous permet d'évaluer la qualité du modèle dans un contexte de prise de décisions. L'agent doit maintenir autant que possible un dirigeable à une hauteur nulle en utilisant le minimum d'énergie. Chaque épisode débute à une hauteur de 0 ainsi qu'à une vélocité nulle et dure pendant 100 étapes de temps. Concernant le simulateur de dynamique utilisé, la discrétisation du temps est de une seconde. Les observations sont constituées de la hauteur (m) et de la vélocité (m/s), toutes deux dégradées par un bruit blanc gaussien à moyenne nulle et d'écart-type de $1cm$. Les récompenses sont aussi corrompues par le même bruit blanc. Les dynamiques du dirigeable sont simulées par une fonction de transition déterministe dont l'état en sortie, contenant la hauteur et la vélocité, est ensuite dégradé par un bruit blanc gaussien de moyenne nulle et d'écart-type de $0.5cm$. L'ensemble des actions est continu et défini par $A = [-1, 1]$ où les bornes représentent respectivement les poussées maximales vers le bas et vers le haut.

En vue d'apprendre et de planifier à partir des observations et des récompenses, nous avons fait l'apprentissage de GPDM à chaque pas de temps afin de fournir à l'algorithme de planification un modèle et une séquence d'états probables. Pour assurer un bon compromis entre l'exploration et l'exploitation, l'agent-dirigeable choisit une action aléatoire suivant une distribution de probabilité décroissante au cours du temps, favorisant ainsi l'exploration au début de chaque épisode.

Lors de nos expérimentations, la fonction de récompense fut cruciale en raison du peu de temps que l'agent-dirigeable avait pour apprendre et de la faible profondeur de l'arbre de l'algorithme de planification. Par conséquent, nous l'avons défini par la

somme de deux gaussiennes. La première gaussienne ayant un écart-type de 1 mètre a pour rôle de donner un demi-point lorsque l'agent-dirigeable est aux alentours de l'altitude cible. La seconde gaussienne possédant un faible écart-type de $5cm$ donne un autre demi-point lorsque l'agent-dirigeable est à seulement quelques centimètres de son but. Un coût sur les actions est aussi appliqué afin de forcer l'agent-dirigeable à atteindre son but avec une quantité d'énergie minimale. De plus, toutes les récompenses observées sont corrompues par un bruit blanc gaussien d'écart-type de 0.01

La figure D.3 montre la récompense moyenne reçue par l'agent-dirigeable. Notons que la courbe se stabilise autour d'une récompense 0.8. Cette valeur correspond à la récompense octroyée lorsque le dirigeable est à une distance de $5cm$ de l'altitude cible et qu'aucune action significative n'est exécutée. Sur la figure D.3, nous observons que la distance moyenne du dirigeable par rapport à l'altitude cible se stabilise autour de $10cm$. La figure D.3 montre l'erreur de prédiction moyenne de la séquence d'observation-récompense lorsque l'agent utilise l'équation (D.13) et l'estimation du dernier état. Nous avons défini l'erreur comme étant la somme des erreurs absolues sur la prédiction de l'observation et la récompense bruitée. La figure D.3 montre que la majorité des trajectoires du dirigeable ont une grande variance au début de l'épisode et que celle-ci diminue au fur et à mesure que l'agent reçoit des observations. Chaque boîte représente les 25ième et 75ième percentiles, la marque centrale est la médiane et les moustaches englobent les données non aberrantes. Pour comparaison avec la politique aléatoire, qui n'est pas rapportée dans cet article, cette stratégie diverge rapidement, et ce, jusqu'à 3 mètres du l'altitude cible.

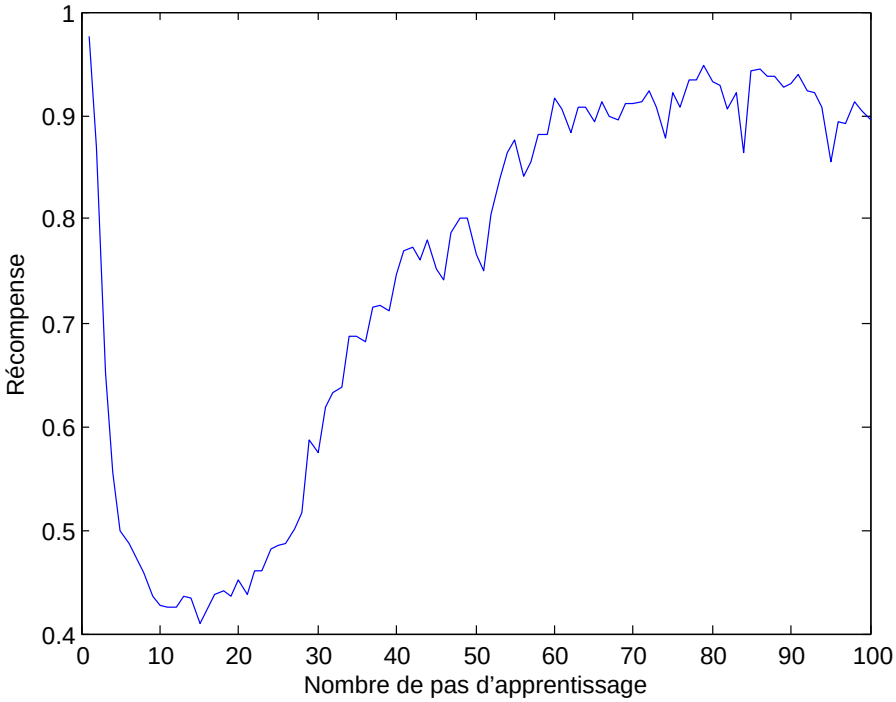


FIGURE D.1 – Récompense moyenne reçue

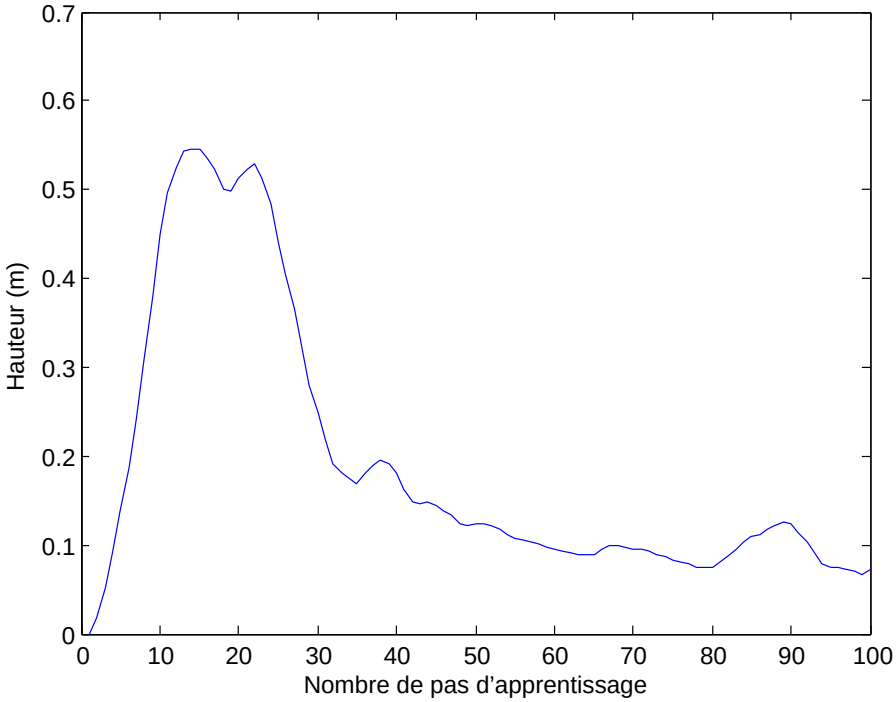


FIGURE D.2 – Distance moyenne à la hauteur requise

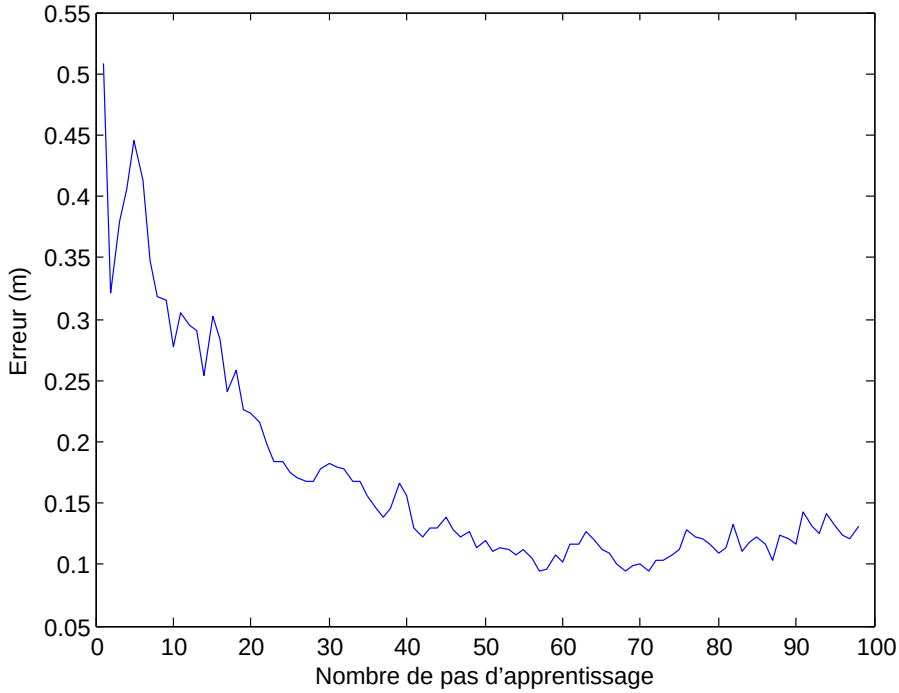


FIGURE D.3 – Erreur moyenne de prédiction

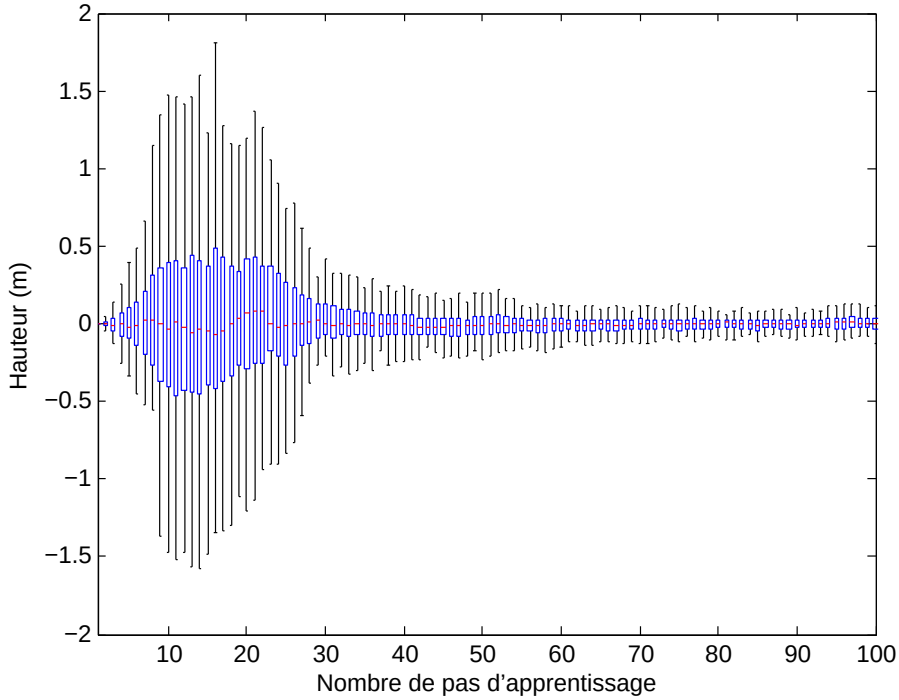


FIGURE D.4 – Quantile à 2.5%, 25%, 50%, 75%, 97.5% de la distribution des trajectoires

Bibliographie

Abbeel, P., Koller, D. et Ng, A. Y. (2006). Learning factor graphs in polynomial time and sample complexity. *J. Mach. Learn. Res.*, 7:1743--1788.

Ahuja, R. K., Kumar, A., Jha, K. et Orlin, J. (2003). Exact and heuristic methods for the weapon target assignment problem. Working papers, Massachusetts Institute of Technology (MIT), Sloan School of Management, MIT, Massachusetts.

Aldous, D. et Fill, J. A. (2002). *Reversible Markov Chains and Random Walks on Graphs*, chapitre 3. Draft.

Astrom, K. J. (1965). The Optimal Control of Markov Decision Processes with Incomplete State Estimation. *Journal of Mathematical Analysis and Applications*, 10:174--205.

Athans, M. et Hosein, P. (1990). An asymptotic result for the multi-stage weapon-target allocation problem. *In Proceedings of the 29th IEEE Conference on Decision and Control*.

Bellman, R. et Dreyfus, S. (1959). Functional Approximations and Dynamic Programming. *Mathematical Tables and Other Aids to Computation*, 13(68):247--251.

Bellman, R. E. (1957). *Dynamic Programming*. Princeton University Press.

Bernstein, D. S., Givan, R., Immerman, N. et Zilberstein, S. (2002). The Complexity of Decentralized Control of Markov Decision Processes. *Math. Oper. Res.*, 27(4):819--840.

Bernstein, D. S., Zilberstein, S. et Immerman, N. (2000). The Complexity of Decentralized Control of Markov Decision Processes. *In Proc. of Uncertainty in Artificial Intelligence*, pages 32--37.

Bertsekas, D. P. (2000). *Dynamic Programming and Optimal Control*. Athena Scientific, 2nd edition édition.

Bertsekas, D. P. et Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA.

Bertsekas, D. P., Tsitsiklis, J. N. et Wu, C. (1997). Rollout Algorithms for Combinatorial Optimization. *Journal of Heuristics*, 3(3):245--262.

Besse, C. et Chaib-draa, B. (2007). An Efficient Model for Dynamic and Constrained Resource Allocation Problems. *In Proc. of the 2nd Inter. Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems*.

Besse, C. et Chaib-draa, B. (2008). Parallel Rollout for Online Solution of Dec-POMDP. *In Proc. of 21st Inter. FLAIRS Conf.*

Besse, C. et Chaib-draa, B. (2009). Quasi-Deterministic Partially Observable Markov Decision Processes. *In Proc. of 16th Inter. Conf. On Neural Information Processing*, pages 237--246.

Besse, C. et Chaib-draa, B. (2010a). Quasi-Deterministic POMDPs and Dec-POMDPs. *In Proc. of 9th Inter. Conf. On Autonomous Agents and MultiAgent Systems (Extended Abstract)*.

Besse, C. et Chaib-draa, B. (2010b). Quasi-Deterministic POMDPs and Dec-POMDPs. *In Proc. of 5th Inter. Workshop On Multiagent Sequential Decision Making in Uncertain Domains*.

Besse, C., Plamondon, P. et Chaib-draa, B. (2007). R-FRTDP. A Real-Time DP Algorithm with Tight Bounds for a Stochastic Resource Allocation Problem. *In Proc. of the 20th Canadian Conf. on Artificial Intelligence*.

Bloch, A. (1977). *Murphy's Law and Other Reasons Why Things Go Wrong*. Price Stern Sloan Adult, New York.

Bonet, B. et Geffner, H. (2003). Labeled RTDP. Improving the Convergence of Real-Time Dynamic Programming. *In Proc. of the Inter. Conf. on Automated Planning and Scheduling*, pages 12--31.

Bonnet, B. (2009). Deterministic POMDPs Revisited. *In Proc. of Uncertainty in Artificial Intelligence*.

Boutilier, C. (1999). Sequential Optimality and Coordination in Multiagent Systems. *In Proc. of the Inter. Joint Conf. on Artificial Intelligence*, pages 478--485.

Boutilier, C., Dearden, R. et Goldszmidt, M. (2000). Stochastic Dynamic Programming with Factored Representations. *Artificial Intelligence*, 121(1-2):49--107.

Brosh, I. et Gerchak, Y. (1978). Markov Chains with Finite Convergence Time. *Stochastic Proc. and their Apps.*, 7:247--253.

Carlin, A. et Zilberstein, S. (2008). Value-Based Observation Compression for Dec-POMDPs. *In Proc. of the Inter. Joint Conf. on Autonomous Agents & Multiagent Systems*.

Cassandra, A., Kaelbling, L. et Littman, M. (1994). Acting Optimally in Partially Observable Stochastic Domains. *In Proc. of Assoc. for the Adv. of Artificial Intelligence*, pages 1023--1028.

Castanon, D. A. et Wohletz, J. M. (2002). Model predictive control for dynamic unreliable resource allocation. *In Proceedings of the 41st IEEE Conference on Decision and Control*, volume 4, pages 3754--3759.

Castanon, D. A. et Wu, C. C. (2004). Decomposition techniques for temporal resource allocation. *In Proceedings of the 43rd IEEE Conference on Decision and Control*, volume 4, pages 3798--3803.

Chang, H. S., Givan, R. et Chong, E. K. P. (2004). Parallel Rollout for Online Solution of POMDPs. *Discrete Event Dynamic Systems*, 14(3):309--341.

Chapman, D. (1987). Planning for Conjonctive Goals. *Artificial Intelligence*, pages 32(3) : 333--377.

Cover, T. M. et Thomas, J. A. (1991). *Elements of information theory*. Wiley-Interscience, New York, NY, USA.

Dallaire, P., Besse, C. et Chaib-draa, B. (2009a). Learning Gaussian Process Models from Uncertain Data. *In Proc. of 16th Inter. Conf. On Neural Information Processing*, pages 433--440.

Dallaire, P., Besse, C. et Chaib-draa, B. (201X). Approximate Inference of Latent Functions from Uncertain Data with Gaussian Processes. *Neurocomputing*, page To appear.

Dallaire, P., Besse, C., Ross, S. et Chaib-draa, B. (2009b). Bayesian Reinforcement Learning in POMDPs with Gaussian Processes. *In Proc. of the Inter. Conf. on Intelligent Robots and Systems*.

Dean, T. et Kanazawa, K. (1990). A Model for Reasoning about Persistence and Causation. *Comp. Intel.*, 5(3):142--150.

Dearden, R. et Boutilier, C. (1997). Abstraction and Approximate Decision-Theoretic Planning. *Artificial Intelligence*, 89(1--2):219--283.

Dearden, R., Meuleau, N., Ramakrishnan, S., Smith, D. E. et Washington, R. (2003). Incremental Contingency Planning. *ICAPS-03 Workshop on Planning under Uncertainty, Trento, Italy*.

Dechter, R. (1999). Bucket Elimination. a Unifying Framework for Reasoning. *Artif. Intel.*, 113(1-2):41--85.

Dechter, R. (2003). *Constraint Processing*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Degris, T., Sigaud, O. et Wuillemin, P.-H. (2006). Learning the Structure of Factored Markov Decision Processes in Reinforcement Learning Problems. *In Proc. of the Inter. Conf. on Machine Learning*, pages 257--264, Pittsburgh, Pennsylvania. ACM.

den Broeder, G. G., Ellison, R. E. et Emerling, L. (1959). On optimum target assignments. *Operations Research*, 7(3):322--326.

Drake, A. (1962). *Observation of Markov Process Through a Noisy Channel*. Thèse de doctorat, MIT.

Draper, D., Hanks, S. et Weld, D. (1994). Probabilistic planning with information gathering and contingent execution. *Proc. 2th National Conf. on AI*, pages 31--36.

Dubois, D. et Prade, H. (1988). *Possibility Theory. An Approach to Computerized Processing of Uncertainty*. Plenum Press, New York.

Eckler, A. R. et Burr, S. A. (1972). Mathematical models of target coverage and missile allocation. *Military Operation Research Society*.

Emery-Montemerlo, R., Gordon, G. J., Schneider, J. G. et Thrun, S. (2004). Approximate Solutions for Partially Observable Stochastic Games with Common Payoffs. *In Proc. of the Inter. Joint Conf. on Autonomous Agents and Multi-Agent Systems*.

Fox, D. (2001). KLD-Sampling. Adaptive Particle Filters. *In Proc. of Advances in Neural Information Processing Systems*, pages 713--720.

Fox, D., Burgard, W. et Thrun, S. (1998). Active Markov Localization for Mobile Robots. *Robotics and Autonomous Systems*, 25:195--207.

Fox, D., Thrun, S., Burgard, W. et Dellaert, F. (2001). *Particle Filters for Mobile Robot Localization*, chapitre 19. Springer.

Georgeff, M. P., Pell, B., Pollack, M. E., Tambe, M. et Wooldridge, M. (1998). The Belief-Desire-Intention Model of Agency. *In Proc. of the 5th Inter. Workshop on Agent Theories, Architectures, and Languages*, pages 1--10.

Glazebrook, K. et Washburn, A. (2004). Shoot-look-shoot : A review and extension. *Operations Research*, 52 :3:454--463.

Gmytrasiewicz, P. J. et Doshi, P. (2005). A Framework for Sequential Planning in Multiagent Ssettings. *J. Artif. Intell. Res.*, 24:49--79.

Goldman, C. V., Allen, M. et Zilberstein, S. (2004). Decentralized Language Learning through Acting. *In Proc. of the Inter. Joint Conf. on Autonomous Agents and Multi-Agent Systems*, pages 1006--1013.

Goldman, C. V. et Zilberstein, S. (2004). Decentralized Control of Cooperative Systems. Categorization and Complexity Analysis. *J. Artif. Intell. Res.*, 22:143--174.

Goldman, C. V. et Zilberstein, S. (2008). Communication-Based Decomposition Mechanisms for Decentralized MDPs. *J. Artif. Intell. Res.*, 32:169--202.

Goldman, R. P. et Boddy, M. S. (1996). Expressive planning and explicit knowledge. *In Proc. of the 3rd Inter. Conf. on Artif. Intel. Planning Systems*, pages 110--117.

González-Rodríguez, G., Colubi, A. et Ángeles Gil, M. (2006). A fuzzy representation of random variables : An operational tool in exploratory analysis and hypothesis testing. *Comput. Stat. Data Anal.*, 51(1):163--176.

Guestrin, C. (2003). *Planning under Uncertainty in Complex Structured Environments*. Thèse de doctorat, Stanford University. Adviser-Daphne Koller.

Guestrin, C., Koller, D. et Parr, R. (2001a). Multiagent Planning with Factored MDPs. *In Proc. of Advances in Neural Information Processing Systems*, pages 1523--1530.

Guestrin, C., Koller, D. et Parr, R. (2001b). Solving Factored POMDPs with Linear Value Functions. *In Workshop on Planning under Uncertainty and Incomplete Information*, pages 67 -- 75, Seattle, Washington.

Hamilton, W. R. (1858). Account of the Icosian Calculus. *Proc. of the Royal Irish Academy*, 6.

Hansen, E. A., Bernstein, D. S. et Zilberstein, S. (2004). Dynamic Programming for Partially Observable Stochastic Games. *In Proc. of Assoc. for the Adv. of Artificial Intelligence*, pages 709--715.

Haralick, R. M., Davis, L. S., Rosenfeld, A. et Milgram, D. L. (1978). Reduction Operations for Constraint Satisfaction. *Information Sciences*, 14(3):199--219.

Hentenryck, P. V., Bent, R. et Upfal, E. (2005). Online stochastic optimization under time constraints. Rapport technique, Brown University, Providence, RI, USA.

Hochwald, B. M. et Jelenkovic, P. R. (1999). State Learning and Mixing in Entropy of Hidden Markov Processes and the Gilbert-Elliott Channel. *IEEE Transactions on Information Theory*, 45(1):128--138.

Hosein, P., Athans, M. et Walton, J. (1988). Dynamic weapon-target assignment problems with vulnerable c2 nodes. *In Proceedings of the 1988 Command and Control Symposium*, pages 240--245.

Hosein, P. A. (1989). *A Class of Dynamic Nonlinear Resource Allocation Problems*. Thèse de doctorat, Massachusets Institute of Technology.

Hosein, P. A. et Athans, M. (1990). Some analytical results for the dynamic weapon-target allocation problem. *Naval Research Logistics*.

Howard, R. (1960). *Dynamic Programming and Markov Processes*. MIT Press.

Hsu, D., Lee, W. S. et Rong, N. (2007). What makes some POMDP problems easy to approximate ? *In Proc. of the Conf. on Neural Information Processing Systems*.

Kaelbling, L. P., Littman, M. L. et Cassandra, A. R. (1998). Planning and Acting in Partially Observable Stochastic Domains. *Artif. Intell.*, 101(1-2):99--134.

Kambhampati, S. et Srivastava, B. (1996). Universal Classical Planner : An Algorithm for Unifying State-Space and Plan-Space Planning. *In New Directions in AI Planning*, pages 61--78. IOS Press (Amsterdam).

Katter, J. D. (1986). A solution of the multi-weapon, multi-target assignment problem. Working paper 26957.

Kautz, H. et Walser, J. (1999). State-space Planning by integer optimization. *Knowledge Engineering Review*, page 15(1).

Koehler, J. (1998). Planning under resource constraints. *Proc. 13th National Conf. on AI*, pages 489--493.

Koller, D. et Milch, B. (2003). Multi-Agent Influence Diagrams for Representing and Solving Games. *Games and Economic Behavior*, 45(1):181--221.

Koller, D. et Parr, R. (1999). Computing Factored Value Functions for Policies in Structured MDPs. *In Proc. of the Inter. Joint Conf. on Artificial Intelligence*, pages 1332--1339.

Kwok, C., Fox, D. et Meila, M. (2003). Adaptive Real-Time Particle Filters for Robot Localization. *In Proc. of the IEEE Inter. Conf. on Robotics & Automation*.

Lee, C.-Y., Su, S.-F. et Lee, Z.-J. (2003). Efficiently solving general weapon-target assignment problem by genetic algorithms with greedy eugenics. *Systems, Man and Cybernetics, Part B, IEEE Transactions on*, 33:113--121.

Lindqvist, B. (1981). Ergodic Markov Chains with Finite Convergence Time. *Stochastic Proc. and their Apps.*, 11:91--99.

Littman, M. (1996). *Algorithms for Sequential Decision Making.* Thèse de doctorat, Department of Computer Science, Brown University.

Lloyd, S. P. et Witsenhausen, H. S. (1986). Weapons allocation is np-complete. *In Proceedings of the 1986 Summer Computer Simulation Conference*, Reno, Nevada.

Mackay, D. J. C. (2003). *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press.

Madani, O., Hanks, S. et Condon, A. (1999). On the Undecidability of Probabilistic Planning and Infinite-Horizon Partially Observable Markov Decision Problems. *In Proc. of Assoc. for the Adv. of Artificial Intelligence*, pages 541--548.

Maltin, S. M. (1970). A review of the literature on the missile-allocation problem. *Operation Research*, 18:334--373.

Manne, A. S. (1958). A target-assignment problem. *Operations Research*, 6:346--351.

Manor, G. et Kress, M. (1997). Evaluating the effectiveness of shoot-look-shoot tactics in the presence of incomplete damage information. *Military Operations Research*, 3 :1:79--89.

Marier, J.-S. (2010). A Markov Model for Multiagent Patrolling in Continuous Time. Mémoire de D.E.A., Université Laval.

Marier, J.-S., Besse, C. et Chaib-draa, B. (2009). A Markov Model for Multiagent Patrolling in Continuous Time. *In Proc. of 16th Inter. Conf. On Neural Information Processing*, pages 648--656.

Marier, J.-S., Besse, C. et Chaib-draa, B. (2010). Solving the Continuous Time Multiagent Patrol Problem. *In Proc. of 2010 IEEE Inter. Conf. on Robotics and Automation.*

Marinescu, R. et Dechter, R. (2009). Memory intensive AND/OR search for combinatorial optimization in graphical models. *Artif. Intell.*, 173(16-17):1492--1524.

McAllester, D. et Rosenblitt, D. (1991). Systematic nonlinear planning. *Proc. 9th National Conf. on AI*, pages 634--639.

McMahan, H. B., Likhachev, M. et Gordon, G. J. (2005). Bounded Real-Time Rynamic Programming. RTDP with Monotone Upper Bounds and Performance Guarantees. *In Proc. of the Inter. Conf. on Machine Learning*, pages 569--576.

Melo, F., Ribeiro, M. et Norte, I. (2005). The Use of Transition Entropy in Partially Observable Markov Decision Processes. Rapport technique, Inst. de Sist. e Robot. de Lisboa.

Monahan, G. E. (1982). A survey of partially observable markov decision processes. theory, models, and algorithms. *Management Science*, 28(1):1--16.

Murphey, R. A. (1999). An approximate algorithm for a weapon target assignment stochastic program. *In* Pardalos, P. M., éditeur : *Approximation and Complexity in Numerical Optimization : Continuous and Discrete Problems*, pages 406--421, Dordrecht. Kluwer Academic.

Murphy, K. (2000). A survey of POMDP solution techniques. Rapport technique, U.C. Berkeley.

Nair, R., Tambe, M., Yokoo, M., Pynadath, D. V. et Marsella, S. (2003). Taming Decentralized POMDPs. Towards Efficient Policy Computation for Multiagent Settings. *In Proc. of the Inter. Joint Conf. on Artificial Intelligence*, pages 705--711.

Nau, D., Ghallab, M. et Traverso, P. (2004). *Automated Planning. Theory & Practice*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Ndilikilikesha, P. (1994). Potential Influence Diagrams. *Int. J. of Approximate Reasonning*, 10:251--285.

Neal, R. M. (1996). *Bayesian Learning for Neural Networks (Lecture Notes in Statistics)*. Springer.

O'Hagan, A. (1992). Some Bayesian numerical analysis. *Bayesian Statistics*, 4:345--363.

Oliehoek, F. A., Spaan, M. T. J., Whiteson, S. et Vlassis, N. (2008). Exploiting Locality of Interaction in Factored Dec-POMDPs. *In Proc. of the Inter. Joint Conf. on Autonomous Agents and Multi-Agent Systems*, pages 517--524.

Papadimitriou, C. et Tsisiklis, J. (1987). The Complexity of Markov Decision Processes. *Math. Oper. Res.*, 12(3):441--450.

Paquet, S., Tobin, L. et Chaib-draa, B. (2006). Prise de décision en temps-réel pour des POMDPs de grande taille. *Revue d'Intelligence Artificielle*, 20(2--3):203--233.

Park, J. et Darwiche, A. (2004). Complexity Results and Approximation Strategies for MAP Explanations. *J. Artif. Intell. Res.*, 21:101--133.

Penberthy, J. et Weld, D. (1994). Temporal Planning with continuous change. *Proc. 12th National Conf. on AI*, pages 1010--1015.

Plamondon, P., Chaib-draa, B. et Benaskeur, A. R. (2007). A Q-decomposition and Bounded RTDP Approach to Resource Allocation. *In Proc. of the 6th International Conference on Autonomous Agents and Multiagent Systems*.

Pralet, C., Schiex, T. et Verfaillie, G. (2006a). From Influence Diagrams to Multioperator Cluster DAGs. *In Proc. of Uncertainty in Artificial Intelligence*.

Pralet, C., Verfaillie, G. et Schiex, T. (2006b). Decision with Uncertainties, Feasibilities, and Utilities. Towards a Unified Algebraic Framework. *In Proc. of the European Conf. on Artificial Intelligence*, pages 427--431.

Puterman, M. L. (1994). *Markov Decision Processes. Discrete Stochastic Dynamic Programming*. Wiley Series in Prob. and Math. Stat.. Applied Prob. and Stat. John Wiley & Sons Inc., New York.

Pynadath, D. V. et Tambe, M. (2002). The Communicative Multiagent Team Decision Problem. Analyzing Teamwork Theories and Models. *J. Artif. Intell. Res.*, 16:389--423.

Rabinovich, Z., Goldman, C. V. et Rosenschein, J. S. (2003). The Complexity of Multiagent Systems. the Price of Silence. *In Proc. of the Inter. Joint Conf. on Autonomous Agents & Multiagent Systems*, pages 1102--1103.

Rezaeian, M. (2006). Hidden Markov Process. A New Representation, Entropy Rate and Estimation Entropy. *Computing Research Repository*, abs/cs/0606114.

Rosenberger, J. M., Hwang, H. S., Pallerla, R. P., Yucel, A., Wilson, R. L. et Brungardt, E. G. (2005). The generalized weapon target assignment problem. *In Proceedings of the 10th International Command and Control Research and Technology Symposium*, McLean, VA.

Ross, S. et Chaib-draa, B. (2007). AEMS. An Anytime Online Search Algorithm for Approximate Policy Refinement in Large POMDPs. *In Proc. of the Inter. Joint Conf. on Artificial Intelligence*, pages 2592--2598.

Roth, M., Simmons, R. et Veloso, M. (2005a). Decentralized Communication Strategies for Coordinated Multi-Agent Policies. *In Multi-Robot Systems. From Swarms to Intelligent Automata*, pages 93--106.

Roth, M., Simmons, R. et Veloso, M. (2005b). Reasoning About Joint Beliefs for Execution-Time Communication Decisions. *In Proc. of the Inter. Joint Conf. on Autonomous Agents and Multi-Agent Systems*.

Russell, S. et Norvig, P. (2009). *Artificial Intelligence. A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 3nd édition.

Seuken, S. et Zilberstein, S. (2005). Formal Models and Algorithms for Decentralized Control of Multiple Agents. Rapport technique, Computer Science Department, University of Massachusetts, Amherst.

Seuken, S. et Zilberstein, S. (2007a). Improved Memory-Bounded Dynamic Programming for Dec-POMDPs. *In Proc. of Uncertainty in Artificial Intelligence*.

Seuken, S. et Zilberstein, S. (2007b). Memory-Bounded Dynamic Programming for Dec-POMDPs. *In Proc. of the Inter. Joint Conf. on Artificial Intelligence*, pages 2009--2015.

Seuken, S. et Zilberstein, S. (2008). Formal Models and Algorithms for Decentralized Control of Multiple Agents. *J. Autonomous Agents and Multi Agent Systems (JAAMAS)*, 17:190--250.

Shannon, C. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal*, 27:379--423.

Singh, S. et Cohn, D. (1998). How to Dynamically merge Markov Decision Processes. *In Proc. of Advances in Neural Information Processing Systems*, volume 10, pages 1057--1063. MIT Press.

Smallwood, R. D. et Sondik, E. J. (1973). The Optimal Control of Partially Observable Markov Processes over the Finite Horizon. *Operations Research*, 9(2):149--168.

Smith, D., Franck, J. et Jónsson, A. (2000). Bridging the Gap between Planning and Scheduling. *Knowledge Engineering Review*, pages 15(1) : 61--94.

Smith, D. et Weld, D. (1999). Temporal Planning with mutual exclusion reasoning. *Proc. 15th National Conf. on AI*, pages 889--896.

Smith, T. et Simmons, R. G. (2004). Heuristic Search Value Iteration for POMDPs. *In Proc. of Uncertainty in Artificial Intelligence*, pages 520--527.

Smith, T. et Simmons, R. G. (2006). Focused Real-Time Dynamic Programming for MDPs. Squeezing More Out of a Heuristic. *In Proc. of Assoc. for the Adv. of Artificial Intelligence*.

Sondik, E. J. (1971). *The optimal control of Partially Observable Markov Processes*. Thèse de doctorat, Stanford University.

Stephane Ross, Joelle Pineau, B. C.-d. et Paquet, S. (2008). Online Planning Algorithms for POMDPs. *J. Artif. Intell. Res.*, 32:663--704.

Stockmeyer, L. J. (1976). The Polynomial-Time Hierarchy. *Theor. Comput. Sci.*, 3(1):1--22.

Strehl, A. L., Diuk, C. et Littman, M. L. (2007). Efficient structure learning in factored-state mdps. *In Proc. of Association for the Advancement of Artificial Intelligence*, pages 645--650.

Subelman, E. (1976). On the Class of Markov Chains with Finite Convergence Time. *Stochastic Proc. and their Apps.*, 4:253--259.

Szer, D. (2006). *Contribution a la resolution des problemes de decision markoviens decentralises*. Thèse de doctorat, Université Henri Poincarré. Adviser-J. F. Charpillet.

Szer, D., Charpillet, F. et Zilberstein, S. (2005). MAA*. A Heuristic Search Algorithm for Solving Decentralized POMDPs. *In Proc. of Uncertainty in Artificial Intelligence*, pages 576--590.

Tanner, M. (1993). *Tools for Statistical Inference*. Springer Verlag.

Thrun, S. (1999). Monte Carlo POMDPs. *In Proc. of Advances in Neural Information Processing Systems*, pages 1064--1070.

Verfaillie, G. et Jussien, N. (2005). Constraint Solving in Uncertain and Dynamic Environments. A Survey. *Constraints*, 10(3):253--281.

Wacholder, E. (1989). A neural network-based optimization algorithm for the static weapon-target assignment problem. *ORSA Journal on Computing*, 4:232--246.

Walsh, T. (2002). Stochastic Constraint Programming. *In Proc. of the European Conf. on Artificial Intelligence*, volume 8. IOS Press.

Wang, J., Fleet, D. et Hertzmann, A. (2008). Gaussian Process Dynamical Models for Human Motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:283--298.

Wang, T., Lizotte, D., Bowling, M. et Schuurmans, D. (2005). Bayesian Sparse Sampling for On-line Reward Optimization. *In Proc .of the 22nd Inter. Conf. on Machine learning (ICML)*, pages 956--963.

Watkins, C. (1989). *Learning from Delayed Rewards*. Thèse de doctorat, University of Cambridge, England.

Watkins, C. J. C. H. et Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3-4):279--292.

Weld, D., Anderson, C. et Smith, D. (1998). Extending Graphplan to handle uncertainty & sensing actions. *Proc. 15th National Conf. on AI*, pages 897--904.

Williams, C. et Barber, D. (1998). Bayesian classification with Gaussian processes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(12):1342--1351.

Witwicki, S. J. et Durfee, E. H. (2010). From Policies to Influences : A Framework For Nonlocal Abstraction In Transition-dependent DEC-POMDP Agents. *In Proc. of the 9th Intern. Conf. on Autonomous Agents and Multiagent Systems*, pages 1397--1398.

Wu, F., Zilberstein, S. et Chen, X. (2009). Multi-Agent Online Planning with Communication. *In Proc. of the Inter. Conf. on Automated Planning and Scheduling*.

Yost, K. A. et Washburn, A. R. (2000). Solving large-scale allocation problems with partially observable outcomes. *Operations Research Letters*.

Younes, H. L. S. (2004). *Verification and planning for stochastic processes with asynchronous events*. Thèse de doctorat, Carnegie Mellon University, Pittsburgh, PA, USA. Chair-Simmons, Reid G.

young Kwak, J., Yang, R., Yin, Z., Taylor, M. E. et Tambe, M. (2010). Teamwork and Coordination under Model Uncertainty in DEC-POMDPs. *In Proc. of the Inter. AAAI workshop on Interactive Decision Theory and Game Theory*.

Zhang, W. et Dietterich, T. G. (2000). Solving Combinatorial Optimization Tasks By Reinforcement Learning : A General Methodology applied to Resource-Constrained Scheduling. *Journal of Artificial Intelligence Research 10*, pages 1--31.