Contributions à la résolution des processus décisionnels de Markov centralisés et décentralisés: algorithmes et théorie

Jilles Steeve Dibangoye

UNIVERSITÉ de CAEN BASSE-NORMANDIE

U.F.R. : *Sciences*

ECOLE DOCTORALE SIMEM

Cotutelle de thèse
Entre
L'Université de Caen Basse-Normandie *(France)*
Et
L'Université Laval (*Canada)*
Arrêté du 6 janvier 2005

THESE

Présentée par

Mr Jilles Steeve DIBANGOYE

Et soutenue

Le 17 décembre 2009

En vue de l'obtention du

DOCTORAT de l'UNIVERSITE de CAEN
Spécialité : Informatique et Applications

Arrêté du 07 août 2006

Titre : Contributions à la résolution des processus décisionnels de Markov centralisés et décentralisés: Algorithmes et Théorie

MEMBRES du JURY

Mr François CHARPILLET	**Directeur de Recherche**	**LORIA, France** *(rapporteur)*
Mr François LAVIOLETTE	**Professeur Associé**	**Université Laval - Québec, Canada** *(rapporteur)*
Mr Olivier Sigaud	**Professeur**	**Université Pierre et Marie Curie, France**
Mr Danny Dubé	**Professeur Associé**	**Université Laval - Québec, Canada**
Mr Abdel-Illah MOUADDIB	**Professeur**	**Université de Caen Basse-Normandie** *(Directeur de thèse)*
Mr Brahim CHAIB-DRAA	**Professeur**	**Université Laval - Québec, Canada** *(Directeur de thèse)*

Résumé

Cette thèse porte sur les problèmes de prise de décisions séquentielles sous incertitudes dans un système mono ou multi-agents. Les processus décisionnels de Markov offrent un modèle mathématique permettant à la fois de formaliser et de résoudre de tels problèmes. Il existe de multiple travaux proposant des techniques efficaces pour la résolution des processus décisionnels de Markov. Néanmoins, trois facteurs, dits malédictions, limitent grandement le passage à l'échelle de ces techniques. Le premier facteur, la malédiction de la dimension, est le plus connu. Il lie la complexité des algorithmes au nombre d'états du système dont la croissance est exponentielle en fonction des attributs des états. Le second facteur, la malédiction de l'historique, a été identifié plus récemment. Il lie la complexité des algorithmes à la dimension exponentielle de l'espace des historiques à prendre en compte afin de résoudre le problème. Enfin, le dernier facteur, la malédiction de la distributivité, est identifié dans cette thèse. Il lie la complexité des algorithmes à la contrainte du contrôle distribué d'un système, résultant sur une complexité doublement exponentielle.

À travers nos contributions, nous proposons une réponse à chacune des trois malédictions. Nous atténuons à la fois la malédiction de la dimension et la malédiction de l'historique en exploitant les dépendances causales soit entre états soit entre historiques. Suivant cette idée, nous proposons une famille d'algorithmes exacts et approximatifs, nommée programmation dynamique topologique, visant à résoudre les processus décisionnels de Markov complètement ou partiellement observables. Ces algorithmes permettent de réduire considérablement le nombre de mises à jour d'un état ou d'un historique. Ainsi, lorsque les problèmes sont munis d'une structure topologique, la programmation dynamique topologique offre une solution efficace. Pour pallier aux effets de la malédiction de la distributivité, nous avons proposé d'étendre la planification centralisée au cadre du contrôle distribué. Nous proposons une analyse formelle des problèmes de contrôle distribué des processus décisionnels de Markov sous le regard de la planification centralisée. De cette analyse, de nombreux algorithmes de planification centralisée ont vu le jour. Parmi eux, figure point-based incremental pruning (PBIP), l'algorithme approximatif pour la résolution des processus de Markov décentralisés, le plus efficace à ce jour.

Abstract

This thesis addresses the computational issues in sequential decision-making under various sources of uncertainty for either single or multi-agent systems. In particular, we will be concerned in problems that can be modeled as Markov decision processes. In addition, we address extensions of Markov decision process, which involve uncertainty resulting from other agents in the world, and partial observability.

We present several strategies that improve the performances of state-of-the-art techniques for solving Markov decision processes. We suggest a general method, namely topological dynamic programming, that exploits the causal relation between states to deal with two key issues. First, it detects the structure of the problem as a means of overcoming both dimensionality and history curses. Secondly, it circumvents the problem of unnecessary backups and builds optimal and approximate solutions based on a topological order induced by the underlying problem structure. Moreover, we introduce the centralized planning for distributed control of Markov decision processes. We provide theoretical basis for further work in that direction. This analysis results in a number of efficient exact as well as approximate algorithms for solving decentralized Markov decision processes. Among many, *point-based incremental pruning* algorithm turns out to be the most efficient so far. The centralized planning for distributed control opens a number avenues, including a bounded sufficient statistic for a general model of decentralized Markov decision processes.

À ma mère.

Avant-propos

Cela fait tout juste quatre années que j'ai débuté cette thèse, et jusqu'au dernier instant j'ai eu recours aux encouragements et aux supports de l'ensemble de mon entourage pour y parvenir. J'aimerai remercier certaines des personnes qui m'ont aidé à faire découvrir et développer ma passion pour la recherche et m'ont encouragé à la pratiquer du mieux de mes capacités. Je remercie très sincèrement :

- Ma famille élargie, en particulier mon père et ma mère. Mes parents ont été d'un support sans faille durant l'ensemble de ma scolarité. Malgré les difficultés auxquels j'ai fait face durant toutes ces années, j'ai toujours su que leur soutient m'était acquis quoi qu'il advienne. Sans ce support indéfectible, je ne suis pas sûr que cette thèse aurait vu le jour. Je remercie également : Ophélie, Teddy, Gaëlle, Anixe, Josephe, Lili, Stecy, Pa Félicien, Pa Marcelin et tous les autres.
- Mes amis durant mes études au Gabon, en France puis au Canada. Je crois profondément en l'amitié et durant toutes ces années, j'ai pu en mesuré l'importance. Je vous remercie pour l'évasion et les conseils que vous m'avez procuré : Ngome Jean-Paul, Alex Da-Silva, Nzaou Mayft, Mba Ndong Bertran, Alonso, Mouhamadou Sarr, Willy Midongo, Patio, Bekale Jean-Cédric, Ebanet Ghislain, Innoncent, Jerry, Leonce, Marion, Jules, Momo, Batch, Soto, Alpho, Alphano, Cédrique, Aworet Mike, Herve Topieu, Herve Avenot, et Merouane Rachidi avec lequel j'ai fait la tournée des soirées québécoises.
- Mes collègues à l'UCBN, dans le groupe MAD. J'ai particulièrement apprécié les relations sympathiques entre jeunes doctorants, et pour cela je remercie : Mouhamad Elfalou pour nos interminables discussions, Lamia Belouaer pour son immense gentillesse, Arnaud Canu, François Bourdon, Ramzi Gannouni, Laurent Jean-Pierre, Abir Karami, Simon Le Gloannec, Boris Lesner, Laëtitia Matignon, Bruno Mermet, Sébastien Picant, Hugo Pommier, Benoit Romito, Gaëlle Simon, Serge Stinckwich,Maroua Bouzid, Anne France Viet, Brune Zanuttini, Aurelie Beynier, Matthieu Boussard.
- Mes collègues à l'ULaval en particulier ceux du DAMAS. J'ai beaucoup aimé les longues coupures agrémentées de discussions scientifiques sur divers domaines. Ces débats ont attisé ma curiosité dans bon nombre de domaines qui m'étaient inconnus. Je re-

mercie : Abdeslam Boularias, Hamid R. Chinaei, Camille Besse, Andrey Burkov, Patrique Dallaire, Jean-Samuel, Stephane Ross, Sébastien Chouinard, Charles Desjardin, Pierre-Luc Grégoire, Minh Nguyen-duc, Pierrick Plamondon, Julien Laumonier, Olivier Gagne, Patrick Cing-Mars, Maxime Lemonnier.

- Mes collègues, chercheurs et professeurs. De nombreuses personnes ont influencé mes contributions à travers les discussions : Guy Shani (processus de Markov partiellement observables) ; Raghav Aras, Frans Oliehoek, Matthias Spaan, Christopher Amato, Akshat Kumar (contrôle reparti des processus de Markov) ; Pr. François Laviollette (processus de Markov complètement observables) ; Pr. Patrice Boizimaut, Pr. Etienne Grandjean, Dr. Samir Loudni (Problèmes de satisfaction de contraintes) ; Pr. Pascal Lang (Optimisation convexe). Guy Shani a collaboré à l'écriture et la conception de l'algorithme de programmation dynamique topologique. Il aura été pour moi un véritable mentor durant les deux dernières années de ma thèse. J'espère avoir l'opportunité de travailler avec lui à nouveau. Christopher Amato et Akshat Kumar sont d'excellents chercheurs dont les centres d'intérêt coïncident avec les miens. J'ai hâte de travailler à nouveau avec eux. Le professeur P. Lang est un excellent pédagogue, qui m'a permis d'appréhender une discipline barbare pour un chercheur en Intelligence Artificielle, à savoir la recherche opérationnelle et en particulier l'optimisation convexe et ses innombrables techniques. Ces cours m'ont donné le goût de l'analyse formelle et la rigueur mathématique. J'espère en faire un usage très prochainement.
- À ceux qui ont contribué directement à ce document. Les membres de mon jury de thèse : Olivier Sigaud, Français Charpillet, François Laviollette, Danny Dubé ainsi que mes directeurs de thèse, Abdel-Illah Mouaddib et Brahim Chaib-draa. Mes directeurs se sont (je crois) partagés les rôles. Tandis que l'un était ultra optimiste l'autre était plutôt pessimiste. Dans un premier temps cela peut être déconcertant, mais avec le temps je m'y suis fait. J'ai même fini par trouvé une astuce afin d'avoir un avis presque juste, il me suffisait de pondérer les deux avis. L'un et l'autre m'ont permis de faire une thèse sans contrainte aucune. J'ai pu ainsi travaillé sur un grand nombre de sujets allant des systèmes mono agent aux systèmes multi-agents. Cette liberté a assurément ouvert ma curiosité pour des sujets très différents. Leurs conseils m'auront surtout été utiles lors de la rédaction d'articles. Cet exercice loin d'être évident nécessite la maîtrise de grand nombre de paramètres dont je n'avais même pas conscience. Je suis chanceux qu'une telle influence soit perceptible dans mes travaux et pour longtemps après.

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

Chapitre 1

Introduction

« L'essentiel, quand on a un commandement, c'est de prendre une décision, quelle qu'elle soit. On s'effraie au début, puis avec l'expérience, on s'aperçoit que cela revient à peu près au même... quoi qu'on décide. »

Jean Anouilh
extrait de *L'alouette*

Le contrôle d'un système évoluant dans le temps est un problème que nous rencontrons dans diverses situations : qu'il s'agisse pour un pilote d'effectuer une séries de manœuvres afin de procéder à l'atterrissage d'un avion ; pour une meute d'hyènes affamée de s'organiser afin de capturer un zèbre ; pour un courtier de chercher à tirer profit des variations des prix d'une obligation suite aux fluctuations des taux d'intérêts ; ou tout simplement, pour une mère de suivre une recette de cuisine pour la préparation du gâteau préféré de ses enfants. Toutes ces situations sont autant d'exemples qui requièrent la prise de décisions, puis l'observation des effets, suivies d'une nouvelle prise de décisions, de l'observation d'autres effets, et ainsi de suite. Ces situations sont qualifiées de problèmes de prise de décisions séquentielles, certaines d'entre elles peuvent être facilement formalisées, mais leur résolution n'est généralement pas simple. Dans ce contexte, l'objectif de cette thèse est alors double. D'une part, elle présente les modèles de formalisation des problèmes de prise de décisions séquentielles. D'autre part, elle propose des méthodes efficaces exactes ou approximatives de résolution de ces modèles.

L'optimisation des problèmes de prise de décisions séquentielles est au cœur de nombreux domaines de recherche. En économie par exemple, les chercheurs s'intéressent essentiellement aux problèmes de prise de décisions séquentielles, où l'évolution du système est souvent modélisée à l'aide de variables continues représentant des quantités telles que : la vitesse angulaire de l'avion ; la position du zèbre ; ou la valeur en temps réel dune obligation. Les problèmes de contrôle modélisés à l'aide de variables continues sont souvent traités sous couvert de la théorie du contrôle. Au contraire, en recherche opérationnelle et en intelligence artificielle, les recherches partent bien souvent de variables à valeurs discrètes, telles que les décisions suivantes par exemple : procéder au virage à gauche de l'avion ; faire avancer la meute d'hyènes ; ou encore effectuer la vente d'une obligation de son portefeuille. Les problèmes où les décisions de différents acteurs impliquent parfois des influences mutuelles ont été essentiellement étudiés en théorie des jeux. Par exemple, les déplacements du zèbre contraignent la meute d'hyènes à adapter sa stratégie de chasse, et vice-versa.

Quel que soit la discipline, l'essence de la prise de décisions séquentielles est le choix d'une décision dont l'impact est à la fois immédiat et à long terme. En effet, le choix de la meilleure décision est intimement lié aux situations futures et à leur gestion. Dans cette thèse, nous explorons des stratégies algorithmiques afin d'automatiser le choix de la meilleure décision face à divers problèmes de prise de décisions séquentielles, du point de vue des chercheurs en intelligence artificielle. Chaque chapitre de cette thèse présente un modèle général de prise de décisions séquentielles ainsi que les algorithmes exacts et/ou approximatifs de résolution de ce modèle. Dans le reste de ce chapitre, nous définissons le contexte scientifique, puis nous discutons des caractéristiques des problèmes de prise

de décisions séquentielles abordés dans cette thèse.

1.1 Contexte scientifique

Cette section situe la thèse à l'intersection de plusieurs disciples desquelles mes contributions se sont fortement inspirées. Nous y offrons, un bref exposé des interconnections, avantages et inconvénients des différents domaines s'appliquant à résoudre les problèmes de prise de décisions séquentielles, à savoir : la théorie du contrôle ; la recherche opérationnelle ; et l'intelligence artificielle.

1.1.1 Problématique inter-disciplinaire

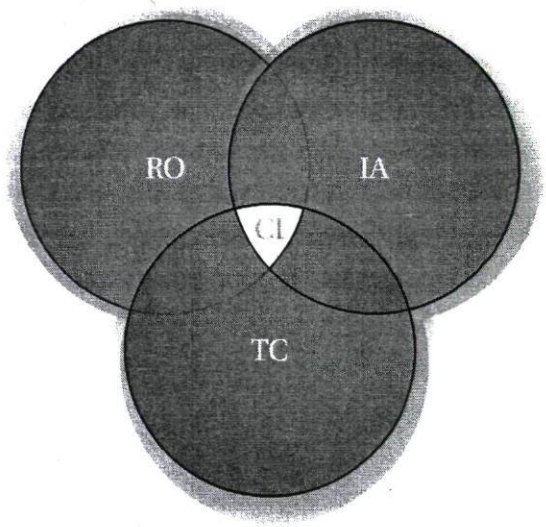


FIGURE 1.1 – Problématique inter-disciplinaire.

Dès les années 80, de nombreux chercheurs se sont consacrés au développement d'une théorie à l'intersection de ces trois domaines de recherche [Albus, 1985, Meystel, 1986, Pao, 1986, Saridis, 1986]. Dans ce contexte, ils ont suggéré la théorie des machines intelligentes – de telles machines étaient originalement conçues pour le traitement des tâches anthropomorphiques avec le minimum d'interactions humaines [Saridis, 1986]. Ces machines requéraient en outre des mécanismes de contrôle dits intelligents, offrant des prestations telles que : l'utilisation de la mémoire ; l'apprentissage ; et la prise de décisions en réponse à des requêtes. Un contrôle intelligent était alors considéré comme une décision inspirée à la fois des mathématiques, des méthodes linguistiques, et de l'algorithmique.

La théorie du contrôle intelligent, proposée par [Saridis, 1989, Tou and Fu, 1973], combine ainsi de puissants outils mathématiques et statistiques de prise de décisions et de modélisation, aux techniques linguistiques de prise en compte d'informations imprécises ou incomplètes. Cette théorie est considérée comme l'intersection selon [Cai, 1997, Saridis, 1989] de l'intelligence artificielle, la recherche opérationnelle et la théorie du contrôle, comme cela est illustré en Figure 1.1. Cette idée de contrôle intelligent joue aujourd'hui un rôle majeur dans la conception des systèmes intelligents et autonomes au sein de ces trois disciplines.

1.1.2 Théorie du contrôle

La théorie du contrôle, appelée également l'automatique, est une discipline des mathématiques appliquées. Elle a pour objet l'étude des systèmes et la conception des composants de contrôle, nommés régulateurs. En théorie du contrôle, un système est une modélisation d'un procédé en fonctionnement, impliquant : des variables d'entrée (r, e), des variables de sortie (y), et des variables de contrôle (u). Ainsi, la modélisation de la préparation d'un gâteau, requiert la connaissance de l'intensité du four, de la quantité d'ingrédients initiaux, ou encore des caractéristiques du gâteau désiré. Contrôler un système signifie alors influencer son comportement afin d'atteindre un état désiré. Pour le contrôle de systèmes, les ingénieurs suggèrent des composants qui incorporent diverses techniques mathématiques. Ces composants vont du régulateur à boules de J. Watt conçu en 1788, durant la révolution industrielle anglaise, aux contrôleurs de microprocesseurs inclus dans différents produits de haute technologie tels que les robots industriels ou les pilotes automatiques d'avions. L'étude de ces régulateurs et de leurs interactions avec les éléments contrôlés constituent l'objet de la théorie du contrôle.

Il existe deux grands courants de travaux en théorie du contrôle. Un de ces courants est basé sur l'idée selon laquelle nous disposons d'un bon modèle de l'objet à contrôler et que nous désirons uniquement optimiser son comportement. Par exemple, les principes physiques et les spécifications d'ingénieurs peuvent être utilisés afin de calculer la trajectoire d'un avion en plein atterrissage, de sorte à minimiser le temps de vol et la consommation de carburant. Le résultat final consiste en un plan d'atterrissage. Les techniques utilisées dans ce courant sont très proches de celles employées en théorie de l'optimisation et ses domaines d'application. Le contrôle, illustré en Figure 1.2, qui en résulte est qualifié de contrôle en boucle ouverte. Il s'agit d'un contrôle qui ignore à la fois la réaction de l'objet contrôlé mais aussi celle de l'environnement dans lequel l'objet évolue. Par exemple, le régulateur de l'intensité du four est incapable de s'ajuster selon l'humidité du gâteau à moins d'une intervention humaine. Un pilote automatique guidant la phase d'atterrissage d'un avion et fonctionnant avec un contrôle en boucle ouverte ne réagira pas si le plan est

défaillant, l'avion pourrait dévier et conduire à une catastrophe. Le contrôle en boucle ouverte est néanmoins utile pour des systèmes bien définis où des formules mathématiques peuvent synthétiser la relation entre les variables d'entrée et les variables de sortie. Ce type de contrôle est également utilisé pour sa simplicité et son moindre coût. Ceci est notamment vrai pour des applications où la besoin de prendre en compte les réactions du système est jugée négligeable.

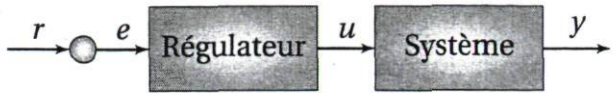


FIGURE 1.2 – Contrôle en boucle ouverte.

Afin d'éviter des catastrophes telles que celles qui pourraient se produire si un pilote automatique fonctionnait avec un contrôle en boucle ouverte, un deuxième type de contrôle a été proposé. Il fait l'objet du second courant majeur de la théorie du contrôle. Ce courant est basé sur la prise en compte des contraintes imposées par l'incertitude du modèle de l'objet à contrôler ou encore de l'environnement dans lequel l'objet évolue. L'idée centrale de ce courant est la prise en compte de la rétroaction (ym) afin de corriger les déviations de l'objet par rapport à son comportement désiré. Par rétroaction, nous entendons la prise en compte dans le contrôle courant de la réponse du système suite au contrôle (u) effectué précédemment. Le contrôle qui en résulte est qualifié de contrôle en boucle fermée. Un tel contrôle est illustré à la Figure 1.3. Ainsi, notre mère de famille jettera un coup d'œil régulier à la recette de cuisine afin de prendre en compte l'état de progression de la préparation du gâteau et les instructions de la recette. De même, le pilote d'avion prendra compte des instructions de la tour de contrôle de l'aéroport afin d'assurer un atterrissage sans craintes. Comme nous le verrons par la suite, ces schémas de contrôle, loin d'être propres à la théorie du contrôle, sont également utilisés en prise de décisions pour le contrôle de systèmes complexes dans d'autres disciplines y compris l'intelligence artificielle.

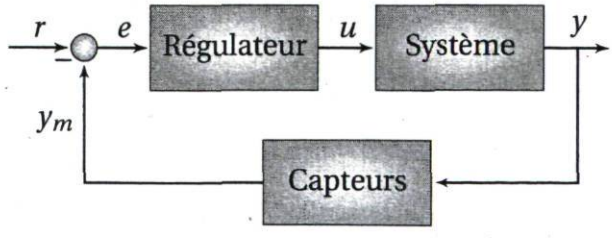


FIGURE 1.3 – Contrôle en boucle fermée.

Dès les premières heures des développements industriels et la production de matériaux et processus nécessitant un contrôle, des techniques du contrôle automatiques furent suggérées. Ainsi, les besoins dans l'industrie de la téléphonie conduirons respectivement Nyquist (1932), Bode (1945) et bien d'autres à développer les mécanismes de réponses

et rétroactions, dont une large application ne se ferra que bien plus tard. C'est durant la seconde guerre mondiale et après, avec la conception des avions, des missiles et autres armes, que le besoin de nouvelles techniques de contrôle bien plus efficaces s'est fait sentir. Dans la période 1945-1965, des techniques de contrôle dites standards basées sur des représentations graphiques et des fonctions mathématiques complexes permettront de résoudre bon nombre de problèmes dans l'industrie pétrolière et chimique. Au cours de la période 1960-1990, la théorie des matrices à variables multiples, munie de fondations théoriques telles que l'algèbre linéaire et la théorie des opérateurs, aboutira au développement de la théorie du contrôle linéaire. Ces avancées théoriques permettront une meilleure représentation et compréhension des problèmes de contrôle, et cela de façon plus appropriée. Certains de ces problèmes sont encore aujourd'hui au centre des préoccupations des chercheurs en automatique, comme en témoigne les travaux sur le contrôle optimal d'un modèle « pilote véhicule » que menait parallèlement le département d'astrophysique de la NASA et le centre de recherche Bolt Beranek and Neumann Inc. [Kleinman and Braon, 1970, Levison, 1972, 1970]. La théorie du contrôle trouve de nombreuses applications dans divers problèmes de prise de décisions, tels que :

- le *contrôle d'un engin.* Le contrôle autonome d'un engin est une application récurrente en théorie du contrôle. Cette dernière offre les outils mathématiques de prise de décisions, comme l'illustre de nombreux exemples : que ce soit le contrôle d'un hélicoptère afin qu'il soit de façon autonome capable de décoller, atterrir, maintenir une position et effectuer des trajectoires simples [Abbeel et al., 2005, 2006] ; ou encore le contrôle d'un véhicule que l'on doit munir de capteurs intelligents capables automatiquement d'ajuster la vitesse en fonction des réponses de l'environnement et des perceptions extérieures [Hebert et al., 1997, Kiencke et al., 2006].
- le *contrôle de multiples engins.* La croissance de l'automatisation du contrôle des véhicules pourrait cantonner les hommes à un rôle de supervision. Cela pourrait en outre permettre une réduction considérable des embouteillages interminables aux heures de pointe.

Cependant, la rigueur mathématique de la théorie du contrôle ne garantie pas le succès commercial des applications. La complexité des applications réelles restreint considérablement l'application des techniques mathématiques extrêmement rigoureuses mais à coûts exorbitants. L'échec dans de nombreuses applications réelles de l'encodage de ces techniques mathématiques hautement estimables conduira à l'émergence de deux disciplines souvent opposées à tord, à savoir la recherche opérationnelle et l'intelligence artificielle.

1.1.3 Recherche opérationnelle

La recherche opérationnelle s'est considérablement développée durant la seconde guerre mondiale grâce : d'une part, aux innombrables tâches militaires pour lesquelles elle a été originellement conçue ; et d'autre part, à l'explosion des capacités de calculs des ordinateurs. La recherche opérationnelle, ou encore nommée *l'aide à la décision*, est une branche interdisciplinaire des mathématiques et des sciences formelles. Elle fait appel aux méthodes de modélisation mathématique, aux statistiques et à l'algorithmique, afin d'aboutir à des solutions optimales ou presqu'optimales pour des problèmes complexes. L'objectif premier de la recherche opérationnelle est de proposer des modèles conceptuels pour analyser des systèmes complexes. Cela permet ainsi d'offrir des critères objectifs et qualitatifs sur lesquels les décideurs doivent fonder leurs stratégies. Les domaines d'application des techniques de la recherche opérationnelle vont des problèmes *combinatoires, stochastiques,* aux problèmes *concurrentiels.* Ces problèmes, loin d'être la panacée de la recherche opérationnelle, trouvent également des solutions, parmi les meilleures, en intelligence artificielle. Nous proposons un rappel succinct de ces problèmes et des difficultés qu'il peut y avoir à les résoudre.

Optimisation combinatoire

On appelle problème d'optimisation combinatoire, un problème comprenant un très grand ensemble de solutions discrètes possibles, parmi lesquelles l'on recherche un optimum. Dans sa forme la plus générale, un problème d'optimisation combinatoire consiste à trouver, dans un ensemble discret, la meilleure solution, parmi toutes les solutions possibles. La notion de meilleure solution est définie par une fonction, appelée fonction objectif. Cette fonction est utilisée en optimisation mathématique. Concrètement, une fonction objectif associe une valeur à une instance d'un problème d'optimisation. Le but du problème d'optimisation combinatoire est alors de déterminer l'instance (l'ensemble des instances) qui minimise ou maximise cette fonction.

Notre pilote en phase d'atterrissage pourrait chercher le plus court chemin entre sa position courante 1 et le point d'atterrissage 4, comme illustré par exemple en Figure 1.4. En effet, l'ensemble des solutions discrètes possibles correspond à l'ensemble des trajectoires allant de 1 à 4. En outre, la fonction objectif correspond à la longueur d'une trajectoire donnée, allant de 1 à 4.

Trouver une solution optimale dans un ensemble discret et fini est un problème facile en théorie : il suffit d'énumérer toutes les solutions possibles ; puis de définir la fonction

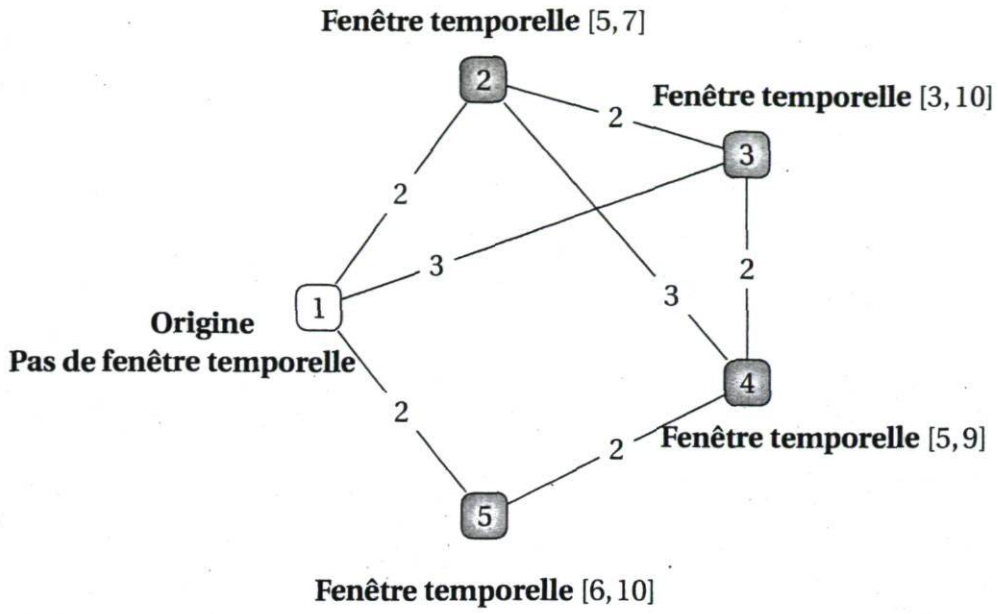


FIGURE 1.4 – Problème de plus court chemin.

objectif qui associe une valeur à chaque solution possible ; et enfin d'identifier celle qui minimise ou maximise cette fonction. En pratique, cependant, l'énumération de toutes les solutions possibles requière parfois un temps non négligeable. Or, le temps d'une énumération exhaustive par un ordinateur est un facteur très contraignant. Il s'élève parfois à plusieurs dizaines voire plusieurs centaines d'années de calculs. C'est pour cette raison que les problèmes d'optimisation combinatoires sont réputés si difficiles.

Optimisation stochastique

Un problème consistant à trouver une solution optimale, face à un système dont la dynamique est incertaine, est appelé problème d'optimisation stochastique [Doob, 1953]. Dans cette thèse, nous associons à tout problème d'optimisation stochastique deux facteurs : (1) un processus stochastique pouvant être perçu en informatique comme une tâche en pleine exécution ou simulation ; et (2) un critère objectif par rapport auquel on tente de contrôler le processus et d'influencer son évolution notamment au travers de la fonction objectif. Un problème d'optimisation stochastique consiste ainsi au contrôle d'un processus stochastique, tel qu'un processus de Poisson, ou un processus de Markov, ...

Par exemple, connaissant la distribution décrivant le nombre d'avions entrant dans un aéroport à une certaine heure et la distribution décrivant la durée de leur escale, il est

possible de déterminer le nombre minimum de pistes à ouvrir afin qu'un avion ait moins de 5% de chances de devoir attendre plus de 15 minutes avant d'amorcer son atterrissage.

Nous distinguons un processus stochastique d'un processus déterministe. Un processus déterministe ne peut transiter que vers un seul et unique état du système. A la différence, un processus stochastique transite vers un état quelconque du système avec une certaine probabilité. En d'autres termes, bien que l'on sache précisément l'état initial du processus stochastique, il est impossible de prévoir avec certitude quelle trajectoire suivra ce processus, même si certaines sont plus probables que d'autres. Bien qu'il ait eu de nombreux modèles de processus stochastiques introduits afin de formaliser et résoudre les problèmes de prise de décisions séquentielles, les modèles de contrôle des processus décisionnels de Markov se sont imposés comme des modèles de référence.

Optimisation multi-agents

Un problème consistant à la recherche d'une solution optimale ou un équilibre, face à un système dont l'évolution dépend des interactions entre les décisions de différents agents du système, est appelé problème d'optimisation multi-agents. La fixation d'une politique de prix de vente, sachant que les résultats d'une telle politique dépendent de la politique que les concurrents adopteront en est un exemple classique. Ces problèmes sont le plus souvent étudiés dans le cadre de la théorie des jeux.

La théorie des jeux constitue une branche des mathématiques appliquées. Elle est étudiée aussi bien en recherche opérationnelle, en économie, qu'en intelligence artificielle [John von Neumann and Morgenstern, 1944]. La théorie des jeux s'intéresse à l'étude mathématique des comportements dans un problème de stratégie, où les choix d'un agent dépendent des choix d'autres agents. Bien qu'initialement introduite afin d'étudier les interactions entre deux protagonistes, elle a été étendue afin de prendre en compte un plus large nombre de protagonistes et différents types d'interactions (coopération, compétition, etc). Les jeux étudiés en théorie des jeux sont simplement des objets mathématiques.

Un jeu est un tuple constitué d'un ensemble de joueurs, un ensemble de coups ou stratégies par joueur, ainsi que les récompenses associées pour chaque combinaison de stratégies. Un jeu est caractérisé par un équilibre : une solution disposant de propriétés satisfaisantes. Cette notion d'équilibre a été introduite afin de caractériser des solutions de plus ou moins bonne qualité à défaut d'une solution optimale. Dans un équilibre, chaque joueur adopte une stratégie de sorte qu'il n'existe aucun joueur qui ait une récompense à gagner en déviant unilatéralement de sa stratégie courante tandis que les autres maintiennent la leur. Parmi plusieurs équilibres, l'équilibre de Nash tente de saisir ce concept

[Nash, 1950]. Selon les jeux, les joueurs disposent d'information parfaite ou imparfaite, et complète ou incomplète.

Un jeu à information parfaite correspond à une situation où chaque joueur connaît toutes les stratégies effectuées par les autres joueurs avant de faire son choix. Un jeu est dit à information complète si lors de la prise de décision, chaque joueur connaît : ses actions possibles ; les stratégies possibles des autres joueurs ; les récompenses qui en résultent ; ainsi que les motivations des autres joueurs. Un jeu est en revanche dit à information incomplète si l'une des ces hypothèses n'est pas vérifiée. Malheureusement, les applications réelles sont rarement à information complète. Cette propriété sert néanmoins en pratique à établir un estimé des comportements optimistes. Afin d'étudier et résoudre les jeux à information incomplète, un modèle générique a été introduit. Il s'agit des jeux stochastiques à information incomplète [Hansen et al., 2004, Kuhn, 1953].

1.1.4 Algorithmes et heuristiques

Afin de combattre la complexité liée à l'énumération exhaustive de l'ensemble des solutions possibles d'un problème, plusieurs méthodes génériques ont vu le jour. La recherche heuristique en est un exemple majeur. La recherche heuristique a été introduite à l'origine par Herbert Simon et fût développée par la suite par [Heath and Tversky, 1991, Tversky and Kahneman, 1974]. Il s'agit de règles simples et efficaces, acquises avec l'expérience, permettant de réduire considérablement l'ensemble des solutions possibles à énumérer. En effet, selon [Pearl, 1984], une heuristique est une méthode capable de produire une solution satisfaisante pour les utilisateurs dans de nombreux cas pratiques, mais pour laquelle il n'existe pas de preuve formelle de son optimalité. Alternativement, une heuristique peut être optimale mais incapable de trouver une telle solution sans faire appel à une quantité démesurée de ressources. Les heuristiques sont souvent utilisées lorsqu'il n'y a aucune méthode connue pour le déterminisme d'une solution optimale sous certaines contraintes en ressources, telles que le temps. Malgré la relative simplicité des concepts qui caractérisent la plupart des heuristiques, leur adaptation efficace à un problème particulier nécessite un grand effort de modélisation et une bonne connaissance des propriétés du problème à traiter. Les heuristiques peuvent se classer en quatre catégories :

1. Les heuristiques constructives se contentent de construire pas à pas une seule solution, c'est le cas des méthodes dites gloutonnes. Elles se caractérisent par une grande rapidité de construction d'une solution satisfaisante mais dont la valeur, selon la fonction objectif, est loin d'être optimale.
2. Les heuristiques de recherche locale travaillent quant à elles sur une solution qu'elles tentent d'améliorer itérativement. C'est le cas d'un algorithme tabou [Glover, 1989],

ou de l'algorithme Monté Carlo. Lors d'une itération, la solution courante est légèrement modifiée afin d'obtenir une solution voisine. Ces heuristiques obtiennent en général des solutions de valeurs supérieures à celles des heuristiques constructives. Malheureusement, elles n'ont pas toujours une grande capacité à explorer des régions très différentes les unes de autres. Leur recherche progresse de régions voisines en régions voisines, jusqu'à ce que l'espace entier de toutes les solutions possibles soit exploré.

3. Les heuristiques évolutives agissent sur une population d'individus (des solutions ou des morceaux de solutions) qui coopèrent et s'adaptent individuellement. C'est par exemple le cas des algorithmes à inspiration biologique, tels que les algorithmes génétiques [Goldberg, 1989, Holland, 1975] ou des fourmis [Dorigo, 1992]. Elles ont la plupart du temps une grande capacité à construire des solutions très différentes les unes des autres. Cependant, elles manquent souvent d'agressivité, car malgré la diversité des solutions rencontrées, celles-ci ne sont pas toujours de grande qualité.
4. Les heuristiques les plus puissantes connues à ce jour consistent à intégrer une heuristique de recherche locale (souvent un algorithme tabou [Glover, 1989]) à une méthode évolutive. Ceci afin de leur confiner l'agressivité qui fait souvent défaut. Ces méthodes sont appelées heuristiques évolutives hybrides.

Certaines de ces heuristiques se sont avérées non seulement extrêmement efficaces mais aussi optimales pour un très large ensemble d'applications, si bien qu'elles sont aujourd'hui considérées comme des algorithmes à part entière. Parmi les algorithmes les plus réputés dans la résolution des problèmes d'optimisation, on compte l'algorithme de séparation et évaluation (ou encore selon le terme anglo-saxon branch-and-bound) [Land and Doig, 1960] ; l'algorithme A* [Peter E. Hart and Raphael, 1968] ; et la *programmation dynamique* [Bellman, 1957].

Recherche heuristique

L'algorithme de séparation et évaluation est une méthode générique de résolution des problèmes d'optimisation et plus particulièrement des problèmes d'optimisation combinatoire. Il s'agit d'une méthode d'énumération implicite : toutes les solutions possibles du problème peuvent être énumérées mais, l'analyse des propriétés du problème permet d'éviter l'énumération de larges sous-ensembles de solutions sous-optimales. L'exploration des solutions possibles du problème se fait suivant une fonction heuristique. Cette fonction heuristique est d'une importance majeure. En effet, lorsque la fonction heuristique ne sous-estime (respectivement surestime) jamais la valeur réelle d'une solution, l'algorithme retourne à terme une solution optimale. Elle s'apparente en ce sens à l'algorithme A*.

L'algorithme A* se distingue de l'algorithme de séparation et évaluation de part le mode d'exploration des solutions. L'algorithme A* énumère les solutions dans l'ordre du meilleure d'abord suivant la fonction heuristique. Cet algorithme tire sa popularité des propriétés dont elle dispose. En particulier, aucun autre algorithme de recherche muni de la même fonction heuristique n'est garanti d'énumérer moins de nœuds que A*.

Programmation dynamique

La programmation dynamique est une méthode d'optimisation mathématique pour la résolution des problèmes d'optimisation. Cette méthode requiert que toute solution optimale puisse être construite à partir de sous solutions optimales afin de garantir l'optimalité de la solution globale : c'est le principe d'optimalité de Bellman. Lorsque ce principe est vérifié, cet algorithme nécessite très peu de temps pour résoudre le problème en comparaison aux méthodes classiques telles que celles associées au principe variationnel [Goldstein et al., 2002]. La programmation dynamique permet de résoudre des problèmes combinatoires, sans générer explicitement l'espace entier de recherche. Il permet également de résoudre efficacement les problèmes d'optimisation stochastique (y compris déterministe). En particulier, il existe un algorithme dérivé de la programmation dynamique, pour la résolution des problèmes d'optimisation à temps discret et horizon fini, nommé induction rétrograde [Putterman, 1994]. Cet algorithme a servi de fondation aux algorithmes d'approximation itérative de politiques ou de valeurs dans les problèmes de prise de décisions séquentielles à horizon infini [Howard, 1960, Putterman, 1994]. Les problèmes de recherche de plus court chemin de toute paire de nœuds d'un graphe peuvent également être résolus en utilisant l'algorithme de Floyd basé sur la programmation dynamique [Cormen et al., 2001].

La programmation dynamique souffre néanmoins de trois facteurs qui limitent considérablement sont utilisation même pour des applications vérifiant le principe d'optimalité de Bellman. Le premier facteur, identifié par [Bellman, 1957] en personne, et certainement le plus connu, est la malédiction de la dimension. Il s'agit d'un phénomène selon lequel la complexité de la résolution d'un problème d'optimisation croît avec leur nombre de paramètres et la taille de leurs domaines respectifs. Le second facteur, identifié par [Pineau et al., 2003b],correspond à la malédiction de l'historique. Ce facteur s'observe essentiellement lorsque le système à contrôler fournit en rétroaction une information incomplète. Ce facteur caractérise le fait que le contrôle d'un tel système ne dépend plus de l'état du système mais des historiques des choix d'actions et des informations perçues jusqu'ici. Malheureusement, ces données croissent de façon exponentielle dans le temps. Le dernier facteur, introduit et motivé dans cette thèse, est nommé la malédiction de la distributivité [Dibangoye et al., 2009b]. Il apparaît lors du contrôle distribué d'un système multi-agents.

C'est le moins connu de tous les facteurs de limitation de l'usage de la programmation dynamique, et de loin le plus contraignant. Dans le contexte du contrôle distribué, nous démontrons que le choix des actions à un instant donné est conditionné par l'ensemble des historiques accessibles à cet instant. A la décharge de la programmation dynamique, ces facteurs sont des difficultés des problèmes à résoudre. Tout autre méthode peinerait tout autant à les surmonter. Fort de ces observations, il est naturel de se demander : « comment surmonter ces facteurs tout en utilisant un outil dont l'efficacité sur un très grand nombre d'applications n'est plus à démontrer ? »

1.1.5 Intelligence Artificielle

Cette section introduit l'ensemble des définitions élémentaires en intelligence artificielle qui offrirons un cadre cohérent pour la formalisation et la résolution des problèmes de prise de décisions séquentielles. Le lecteur intéressé par une introduction historique de l'intelligence artificielle peut se référer à l'ouvrage de [Russell and Norvig, 2003]. Notre présentation se focalise essentiellement sur le problème de la prise de décisions séquentielles comme le voient les chercheurs en intelligence artificielle. Le cadre dans lequel nous nous situons tout au long de cette thèse est celui représentant un processus d'apprentissage par l'interaction afin d'atteindre un but [Sutton and Barto, 1998].

L'agent

L'apprenti [1], renvoie à une entité physique, par exemple le robot de nettoyage d'une salle ; le décideur d'une entreprise ; le pilote d'un avion en pleine atterrissage ; la mère préparant un gâteau, ou une entité abstraite, par exemple un programme informatique de gestion des achats sur internet. Cette entité interagit avec son environnement. Lors de ses interactions séquentielles, l'agent exécute des actions auxquelles l'environnement répond via une rétroaction en présentant de nouvelles situations, auxquelles l'agent doit réagir, et ainsi de suite. La rétroaction de l'environnement inclut deux signaux, le premier est associé à une nouvelle situation présentée à l'agent et le second correspond à un nombre réel nommé récompense (respectivement coût) que l'agent aura à maximiser (respectivement minimiser) au cours du temps.

L'agent est caractérisé d'une part, par sa capacité à percevoir son environnement, pouvant contenir d'autres agents, aux travers de capteurs. D'autre part, il est caractérisé par

1. dit acteur en économie, dit décideur en recherche opérationnelle, dit contrôleur en automatique ou encore « *agent* » en intelligence artificielle.

sa capacité à agir sur l'environnement via ses actionneurs. Les agents sont des entités autonomes ou semi-autonomes et situés. C'est à dire que chaque agent a un certain nombre de responsabilités dans la résolution du problème et une certaine connaissance soit de ce que font les autres agents, s'il y en a, soit de comment ils le font. Chaque agent accomplit une sous tâche du problème entier indépendamment et il peut soit en percevoir le résultat tout seul ou avec l'ensemble des autres agents. Chaque agent n'est sensible qu'à son environnement immédiat et bien souvent il est « inconscient » de la présence d'autres agents ou même de la totalité de l'environnement. Ainsi, la connaissance d'un agent se limite bien souvent à la tâche en cours sans aucune « conscience » de l'ensemble des tâches à résoudre afin de résoudre le problème tout entier.

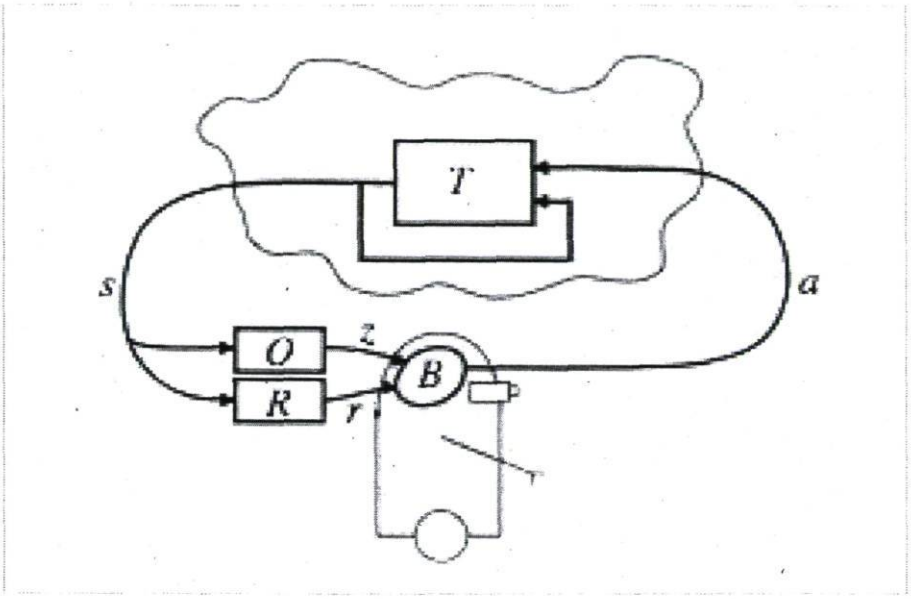


FIGURE 1.5 – Agent interagissant avec son environnement.

Lorsque plusieurs agents interagissent ensemble, il est dit qu'ils sont interactionnels. Cela signifie qu'ils forment une collection d'entités qui coopèrent ou sont en compétition pour la satisfaction d'une certaine tâche. Cette notion signifie qu'un tel groupe d'agents peut dans une certaine mesure être perçu comme une microsociété disposant de ses connaissances, capacités, responsabilités, et cela même lorsque les agents en question sont coopératifs. Un groupe d'agents est également dit structuré. En effet, dans presque la totalité des stratégies de résolution d'un problème multi-agents, chaque agent se coordonnera avec les autres agents dans le processus de résolution du problème tout entier. Et cela bien que chaque agent individuellement ne perçoit que son environnement immédiat et ne dispose que de ses propres capacités. Enfin, cet ensemble de propriétés intrinsèques aux agents conduit à un comportement global cohérent : c'est l'émergence.

Plusieurs définitions sont proposées afin d'expliquer ce concept, les deux suivantes

me semblent des plus pertinentes dans le cadre de cette thèse : « l'ensemble ne fait pas plus que la somme des ses parties. Cela signifie qu'il n'est pas nécessairement possible de prédire le comportement de l'ensemble des agents par la seule analyse de ses parties. » Fort de toutes ces observations, il est possible de définir un agent comme un élément de la société qui peut percevoir parfois de façon limitée les aspects de l'environnement et affecter cet environnement soit directement soit au travers d'une coopération avec les autres agents. Bon nombre de méthodes de résolution de problèmes très complexes requièrent une variété d'agents. Cela est en particulier vrai pour les agents centraux se contentant de recevoir les informations partielles en provenance de certains agents ; les agents de coordination facilitant les interactions entre agents ; agents de recherche permettant d'examiner l'ensemble des informations et d'en extraire des concepts ou des généralisations utiles ; et les agents de décisions qui sont capables à la fois de dispatcher les tâches mais aussi d'apporter de solutions quant à la manière de satisfaire ces tâches.

La planification

Il existe deux approches distinctes d'apprentissage par interaction. La première est appelée planification. Elle s'applique lorsque nous disposons d'un modèle d'interaction. C'est à dire de règles qui relient les actions exécutées par l'agent aux rétroactions de l'environnement. La deuxième est dite apprentissage par renforcement, elle s'applique lorsqu'au contraire nous ne disposons d'aucun modèle d'interaction. Il faut alors construire ce modèle puis procéder à la planification. Au cours de cette thèse, nous nous intéressons dans un premier temps à la planification, nous n'aborderons l'apprentissage par renforcement que très brièvement. En effet, qu'il s'agisse d'apprentissage par renforcement ou de planification, certaines opérations élémentaires sont essentiellement les mêmes. Bien que cette thèse n'aborde pas de façon frontale la question de l'apprentissage par renforcement, bon nombre de nos contributions peuvent être très utiles dans ce contexte. Afin d'illustrer l'exposé sur la planification, nous utiliserons des exemples issues de la robotique. En effet, la robotique a été longtemps un des domaines de recherche en intelligence artificielle qui fournissait le plus d'intuitions dans la consolidation du concept de la résolution orientée agent d'un problème.

La recherche en planification a été le fruit d'un effort concerté afin de concevoir des robots qui pourraient réaliser des tâches avec un certain degré de flexibilité et une certaine capacité de réponse au monde extérieur. En somme, planifier suppose qu'un robot soit capable de réaliser des actions atomiques. Ce domaine tente de trouver une séquence d'actions qui permettra d'accomplir une tâche de haut niveau, telle que traverser une pièce pleine d'obstacles.La planification est un problème difficile pour un certain nombre de raisons, parmi lesquelles nous citerons la taille de l'ensemble des séquences de mouve-

ments possibles pour un robot. Même un robot extrêmement simple est capable de générer un vaste nombre de séquences de mouvements potentiels. Imaginons, par exemple, un robot qui soit capable de se déplacer en avant, en arrière, à droite, ou à gauche, et considérons le nombre de moyens qu'a ce robot pour se déplacer dans une pièce. Supposons aussi qu'il y ait des obstacles et que le robot soit obligé de sélectionner un chemin qui lui permettrait de se déplacer autour de ces mêmes obstacles de manière efficace. Ecrire un programme qui soit capable de découvrir le meilleur chemin sous ces circonstances, requiert des techniques sophistiquées pour la représentation de la connaissance spatiale et le contrôle de la recherche à travers divers types d'environnements.

Une des méthodes utilisée en planification est la décomposition hiérarchique du problème [Dietterich, 2000, Erol et al., 1994]. Si vous planifiez un voyage de la ville de Caen à la ville de Québec, vous aurez notamment à suivre séparément les étapes suivantes : réserver le vol ; vous rendre à l'aéroport ; procéder au changement d'avions lors des escales ; et trouver un moyen de transport dans Québec. Chacun de ces sous problèmes pourrait être encore décomposé en sous problèmes de plus petites tailles et ainsi de suite. Non seulement cette approche permet de réduire considérablement l'espace de recherche, mais en plus elle permet la mémorisation fréquente de sous plans utiles pour un usage futur. Bien que les humains planifient sans un réel effort, créer un programme qui soit capable de faire de même est une tâche très complexe. Une tâche telle que celle qui consiste à décomposer un problème en sous problèmes plus simples et indépendants requiert des heuristiques très sophistiquées et une grande connaissance du domaine d'application. Déterminer quels sous plans devraient être mémorisés et comment ils pourraient être généralisés pour un usage futur constituent autant de problèmes difficiles.

Outre la capacité à réduire l'espace de recherche, le robot doit également être capable d'organiser ses plans afin de répondre au mieux aux rétroactions imprévues de l'environnement. Souvent, un robot aura à construire un plan sur la base d'informations incomplètes et corriger par la suite son comportement lors de l'exécution de ce plan. Un robot peut ne pas avoir les capteurs adéquats afin de localiser tous les obstacles sur son chemin. Un tel robot doit se déplacer à travers la pièce sur la base de se qu'il aura perçu puis il doit corriger par la suite son chemin lors de la détection des autres obstacles.

Frustrer à la fois par la complexité de la représentation de l'espace de recherche, et par la définition des techniques de recherche adéquates pour la planification traditionnelle, les chercheurs, y compris [Agre and Chapman, 1987] et [Brooks, 1991], ont redéfini le problème sous la forme d'un problème d'interactions de plusieurs agents semi-autonomes ou autonomes. Chacun des agents ayant la responsabilité de sa propre portion du problème, de sorte qu'à travers leur coordination une solution du problème originale émerge. Aujourd'hui, la recherche en planification a largement dépassée les frontières de la robo-

tique. Elle inclut désormais la coordination de plusieurs ensembles complexes de tâches ou buts. Les planificateurs modernes sont appliqués aux agents logiciels [Nilsson, 1994] aussi bien qu'aux agents physiques.

1.2 Prise de décisions séquentielles

Compte tenu de la diversité des problèmes abordés dans cette thèse, nous nous sommes inspirés de [Putterman, 1994] dans la conception de notre formalisme. Le caractère abstrait de ce formalisme est nécessaire afin d'offrir un exposé cohérent aussi bien pour la présentation de l'état de l'art que pour celle des contributions de la thèse.

Cette section introduit la problématique de prise de décisions séquentielles. Cette problématique est représentée symboliquement au travers de la Figure 1.6. À un instant quelconque $(\tau - 1)$ dans le temps, un agent perçoit une information $\omega_{\tau-1}$ sur l'état $s_{\tau-1}$ du système (ce qu'on voit). Muni de l'historique des paires d'états et d'actions $h_{\tau-1} = (s_0, a_0, \omega_1, \cdots, a_{\tau-2}, \omega_\tau$ (ce qu'on infère), le décideur choisit une action $a_{\tau-1}$ (ce qu'on voit). Le choix de cette action produit deux effets : d'une part, le décideur reçoit une récompense immédiate $r_{\tau-1}$ (respectivement un coût immédiat) et d'autre part, le système transite vers un nouvel état en tenant compte de l'information ω_τ recueillie sur cet état (ce qui se passe). Ce problème se répète tant que l'agent le désire.

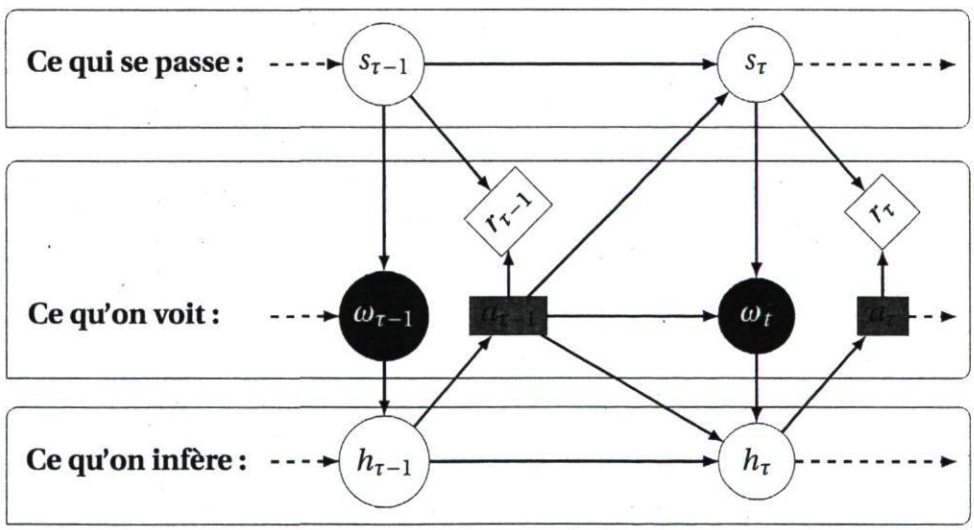


FIGURE 1.6 – Processus de prise de décisions séquentielles [Pineau, 2004].

1.2.1 Différents types de modèles

Les problèmes de prise de décisions séquentielles recouvrent un vaste éventail de problématiques. Ces dernières décrivent les caractéristiques et propriétés de l'environnement ; des contraintes de planification ou d'exécution ; et des propriétés spécifiques des agents intervenant dans la résolution de la tâche globale. Toutes ces propriétés peuvent être vues comme des propriétés de l'environnement ou des agents individuellement [Russell and Norvig, 2003].

Décisions séquentielles

Les problèmes de prise de décisions séquentielles sont décomposés en instants de décisions – ces derniers correspondent à la division du temps opérée par le décideur quant aux instants où un agent interagit avec l'environnement en exécutant une action et percevant ensuite les informations sur l'état suivant du système. Lorsqu'un même problème de prise de décisions séquentielles se répète sur plusieurs épisodes, il est dit épisodique. Comme il s'agit essentiellement du même problème, nous ne traitons dans cette thèse qu'un seul épisode :

1. *continu - discret*
   Les décisions sont prises à différents instants dans le temps. Nous utiliserons la variable N N afin de représenter le nombre de décisions à prendre. L'ensemble des instants de décisions correspond à un sous-ensemble de réels non négatifs. On en distingue de deux catégories, les ensembles d'instants de décisions continus ou discrets. On parle alors de problèmes de prise de décisions séquentielles à temps continu ou discret, respectivement. S'il s'agit d'un ensemble discret, nous dirons que les décisions sont prises aux instants $\tau = 0, 1, \cdots, N-1$ fixés. Si au contraire il s'agit d'un continuum, les décisions sont prises soit suivant des instants choisis aléatoirement, soit selon la détection d'un certain nombre d'événements prédéfinis, ou encore de façon continuelle dans le temps. Dans cette thèse, nous n'aborderons que des problèmes de prise de décisions séquentielles à temps discrets. Lorsque que nous serons confrontés à un problème à temps continu, nous procèderons à une discrétisation du temps.
2. *horizon fini - horizon infini*
   L'ensemble des instants de décisions peut être fini, si $\tau = 0, 1, \cdots, N-1$ pour tout entier $N < \infty$, ou infini dans le cas où $\tau = 0, 1, \cdots$ pour tout entier $N = \infty$. Lorsque Nest fini, tout problème de prise de décisions séquentielles est appelé problème à horizon fini ; le cas échant, il sera dit à horizon infini. La grande partie de mes

contributions concerne les problèmes à horizon infini, cependant nous aborderons au Chapitre 5 les problèmes à horizon fini comme base pour la résolution des problèmes à horizon infini aborder au Chapitre 6.

Propriétés du système

1. *Markovien – non Markovien*
   Un système dont l'évolution futur ne dépend de l'historique qu'au travers de l'état courant : c'est la propriété de Markov. De tels systèmes sont dits Markoviens. Un système est dit non Markovien s'il est souvent nécessaire de se souvenir de tout ou partie des historiques afin d'en estimer avec exactitude l'information suffisante afin de construire un plan des décisions futures. Dans cette thèse, tous les systèmes étudiés sont munis de la propriété de Markov. Nous démontrons au Chapitre 5 que dans une certaine mesure tout problème de prise de décisions séquentielles peut être modélisé comme un système Markovien.
2. *statique - dynamique*
   Lorsque le système est capable de transiter d'un état à un autre alors même que l'agent est en pleine délibération, le système est dit dynamique. Le cas échéant, le système est dit statique, il s'agit des situations où le décideur est capable de délibérer suffisamment vite de sorte que le système ne transite d'un état à un autre qu'à la suite de l'exécution d'au moins une action.
3. *stochastique - déterministe*
   Lorsque le système transite toujours vers un unique état successeur de l'état courant suite à l'exécution d'une action donnée, le système est dit déterministe. Le cas échéant, il est dit stochastique. Dans un tel système, les transitions sont dictées par une distribution de probabilités sur l'espace des états possibles du système, et cela pour chaque action possible.
4. *synchrone - asynchrone*
   Dans le cas de systèmes synchrones, la mesure du temps est l'exécution par l'agent d'une action. En d'autres termes, la transition d'un état du système à un autre ne se produit qu'à la suite de l'exécution d'une action. Lorsque plusieurs agents sont considérés, nous supposerons qu'ils sont synchrones c'est à dire qu'à chaque instant de décision, ils exécutent tous une action individuelle – éventuellement l'action « ne rien faire ». Nous ne considérons pas les systèmes asynchrones, c'est à dire les systèmes pour lesquels la transition d'un état à un autre n'est pas nécessairement due à l'exécution d'une action : le système n'attend pas que l'agent ait exécuté son action avant de transiter vers un nouvel état. Dans cette thèse, nous n'abordons pas non plus le problème de synchronisation des agents, nous supposerons que les agents exécutent leurs actions de façon synchrone.

5. *mono agent - multi-agents*

   Afin de simplifier l'exposé d'un algorithme ou d'un problème, nous nous restreindrons souvent au cas d'un ou de deux agents. Cependant, les modèles utilisés prennent en compte plus de deux agents en général. La résolution de problèmes multi-agents requiert bien souvent la prise en compte des caractéristiques du systèmes, par exemple les interactions entre agents, ou de tout autre restriction permettant de réduire la complexité. Ces aspects ne sont pas abordés dans cette thèse. Néanmoins nos contributions peuvent s'imbriquer aux méthodes d'exploitation des interactions entre agents.

6. *compétitif - coopératif*

   Lorsqu'un système inclut plusieurs agents en compétition dans le but de minimiser les gains d'un adversaire et de maximiser les siens : le problème est dit compétitif. Lorsque les agents doivent s'entre aider afin de maximiser les gains de l'ensemble des agents : le problème est dit coopératif. Bien que les approches proposées dans cette thèse puissent très facilement s'étendre dans le cas compétitif, nous nous concentrons exclusivement aux problèmes coopératifs.

7. *contrôle centralisé - contrôle distribué*

   Lorsque le système est contrôlé par plus d'un agent, ces derniers peuvent prendre leurs décisions individuelles d'un commun accord. Par exemple, ils peuvent s'en remettre aux choix d'un agent central. Il s'agit alors d'un contrôle centralisé. Lorsqu'au contraire les agents exécutent leurs actions séparément et indépendamment des décisions prises par les autres agents, on parle alors de contrôle distribué.

1.2.2 États, Actions et Observations

À chaque instant de décision, le système est dans une configuration donnée, appelée état. L'ensemble des états du système est noté $S.$. Si à un instant de décision, le décideur observe complètement l'état du système $(\omega = s),)$, il peut d'après la propriété de Markov choisir d'exécuter une action a parmi toutes les actions possibles. On note A des actions possibles. Il arrive parfois que l'agent soit incapable d'observer complètement l'état du système $(\omega \neq s)$, mais seulement une information partielle ω. On appelle observation une telle information, notée ω. L'ensemble des observations perçues par le décideur est noté Ω. Nous nous intéresserons essentiellement aux ensembles S, A et Ω finis. Parfois l'agent doit baser le choix de l'action à exécuter sur ses observations et actions passées. C'est à dire que le choix de l'action a_τ est conditionné par l'historique $h_{0:\tau} = (a_0, \omega_1, a_1, \omega_2, \cdots, a_{\tau-1}, \omega_\tau)$, où ω_τ et a_τ dénotent l'observation perçue et l'action exécutée par l'agent à l'instant τ. Lorsque l'agent perçoit complètement l'état s du système, l'historique s'écrit $h_{0:\tau} = (a_0, s_1, a_1, s_2, \cdots, a_{\tau-1}, s_\tau)$. L'historique $h_{0:\tau}$ est alors donné récursivement par $h_{0:\tau} = (h_{0:\tau-1}, a_{\tau-1}, \omega_\tau)$. On note $H_{0:\tau}$ l'ensemble de tous les historiques $h_{0:\tau}$ de

longueur $\tau+1$. Notons que $H_{0:1} = A\times\Omega$, $H_{0:2} = A\times\Omega\times A\times\Omega$ et $H_{0:\tau} = \Omega\times A\times\Omega\times\cdots\times A\times\Omega$, enfin $H_{0:\tau}$ satisfait l'induction $H_{0:\tau} = H_{0:\tau-1}\times A\times\Omega$.Les actions peuvent être choisies de façon aléatoire ou déterministe. On note $\mathscr{P}(A)$ l'espace des distributions de probabilités sur A. Choisir une action aléatoirement revient à sélectionner une distribution de probabilités dans $\mathbb{P}(\cdot)\in\mathscr{P}(A)$, puis sélectionner l'action a avec la probabilité $\mathbb{P}(a)$. Un cas dégénéré des sélections aléatoires d'actions est le cas déterministe, c'est à dire : soit $\mathbb{P}(a) = 1$ soit $\mathbb{P}(a) = 0$.

États du système

À chaque instant de décision le système est dans un état. Il s'agit des différentes configurations dans lesquelles le système peut se retrouver. Par exemple, la position dans l'espace d'un avion en phase d'atterrissage, ou encore les caractéristiques du gâteau en cours de cuisson.

1. *continu - discret*

   Bien que ces configurations correspondent parfois à des espaces continus, il existe des méthodes de discrétisation d'espaces continus afin de les rendre discrets [Scherrer, 2005]. Nous nous intéresserons plus particulièrement aux ensembles d'états finis, aux Chapitres 2 et 4. Lorsque nous aborderons des modèles où l'information suffisante pour planifier appartient à un continuum, nous procéderons à des stratégies de planification sur un ensemble discrets du continuum. C'est notamment le cas aux Chapitres 2,3,4, 5 et 6. Cette information suffisante est encore appelée la statistique suffisante. C'est une distribution de probabilités qui est suffisante pour sélectionner les actions, ou politiques à chaque instant de décision. Souvent la statistique suffisante est vue comme l'état courant du système

2. *complètement observable - partiellement observable*

   Si les capteurs de l'avion en phase d'atterrissage fournissent la position exacte de l'avion dans l'espace ; si les perceptions de la mère quant à l'état de cuisson du gâteau sont exactes, il est alors dit que ces perceptions ou encore observations sont complètes. Cela signifie que ces observations fournissent toute l'information utile en vue de déterminer l'état courant du système. Lorsque ces perceptions permettent de déterminer sans incertitude l'état du système, le problème de prise de décisions est dit complètement observable. Le cas échéant, il est dit partiellement observable.

Actions disponibles

L'ensemble des actions A décrit les différents effets des décisions sur l'état du système.

1. *continu - discret*

   L'ensemble des actions peut être soit discret soit continu. Une fois de plus nous ne nous intéresserons qu'au cas discret. Lorsqu'un environnement dispose d'un ensemble continu d'actions, nous supposerons qu'une méthode de discrétisation lui aura été appliquée afin de nous ramener dans le cas discret.

2. *fini - infini*

   Dans cette thèse, nous n'aborderons que le cas des ensembles finis d'actions, le cas général des espaces infinis d'actions accroît considérablement la complexité des problèmes.

En résumé, nous aborderons dans cette thèse des problèmes, à temps discrets et horizon fini ou infini, de contrôle centralisé ou distribué d'environnements statiques, stochastiques, synchrones, coopératifs, stationnaires ou non stationnaires, markoviens, et mono agent ou multi-agents.

1.2.3 Politiques et règles de décisions

Une politique, un plan contingenté, ou encore une stratégie, est une application qui spécifie la procédure de sélection d'une action à chaque instant de décision. Une politique propose aux agents l'action à exécuter pour tout état ou historique du système. Il existe un large éventail de classes de politiques allant des politiques Markoviennes déterministes aux politiques aléatoires dépendant de l'historique, comme illustrer à la Figure 1.1. Plus formellement, une politique π est une séquence de règles de décisions, $(d_0, d_1, \cdots, d_{N-1})$, où $d_\tau \in D_\tau^\chi$ représente une règle de décisions pour instant de décision $\tau = 0, 1, \cdots, N-1$, et $N \leqslant \infty$, avec χ une des classes de politiques. Soit Π^χ l'ensemble de toutes les politiques de la classe χ, c'est à dire $\Pi^\chi = D_0^\chi \times D_1^\chi \times \cdots \times D_{N-1}^\chi$, où D_τ^χ représente l'ensemble des règles de décisions de la classe χ à l'instant $\tau < N \leqslant \infty$.

Le tableau 1.1 donne une classification des règles de décisions. Un règle de décisions est une application qui associe à tout rétroaction de l'environnement une action. Dans la classe des règles de décisions stationnaires et déterministes $D^{\mathbf{SD}}$, une règle de décisions est une application $d\colon S \to A$ invariante par rapport au temps, et qui associe à tout état $s \in S$, une action $a \in A$. Une politique dans l'espace des politiques stationnaires et déter-

	Choix de l'action	
Dépendance	Déterministe	Aléatoire
Stationnaire	$D^{\mathbf{SD}}$ $d: S \to A$	$D^{\mathbf{SA}}$ $d: S \to \mathscr{P}(A)$
Markovien	$D^{\mathbf{MD}}$ $d: S_\tau \to A_\tau$	$D^{\mathbf{MA}}$ $d: S_\tau \to \mathscr{P}(A_\tau)$
Non-markovien	$D^{\mathbf{HD}}$ $d_\tau: H_{0:\tau} \to A_\tau$	$D^{\mathbf{HA}}$ $d_\tau: H_{0:\tau} \to \mathscr{P}(A_\tau)$

TABLE 1.1 – Classification des règles de décisions

ministes $\Pi^{\mathbf{SD}}$, est alors donnée par $\pi = (d, d, d, \cdots)$ où la règle de décisions est la même quelque soit l'instant de décision. C'est dans ce cas précis, qu'il est admis d'identifier abusivement une politique π à une règle de décisions d.

Malheureusement, la diversité des problèmes traités dans cette thèse nous oblige à différencier la politique π des règles de décisions associées, sauf contre indication. En effet, dans la classes des règles de décisions non Markoviennes et déterministes, par exemple $D^{\mathbf{HD}}$„ une règle de décisions est une application $d_\tau: H_{0:\tau} \to A$ qui associe à un historique $h_{0:\tau} \in H_{0:\tau}$ une action $a \in A$. Une politique dans la classe des politiques non Markovienne et déterministe est quant à elle une séquence de règles de décisions $(d_0, d_1, \cdots, d_{\tau-1})$, différentes les unes des autres.

On appelle politique Markovienne toute politique dont la sélection de l'action à exécuter ne dépend que de l'état courant du système et non de l'historique. Une politique est déterministe si elle choisit toujours la même action pour le même état ou historique, quelque soit l'instant de décision. Nous disons qu'une politique est dépendante de l'historique si la sélection de l'action à exécuter dépend de l'historique. De façon équivalente nous dirons qu'elle est non Markovienne le cas échéant. Enfin, une politique est aléatoire si la sélection de l'action à exécuter à instant τ dépend non seulement de l'état ou l'historique, mais aussi d'une distribution de probabilités sur le choix de l'action ou de la règles de décisions à exécuter. Pour tout état s_τ (resp. historique $h_{0:\tau}$), l'action a est exécutée avec une probabilité $\mathbb{P}(a = a_\tau | s_\tau)$ (respectivement $\mathbb{P}(a = a_\tau | h_{0:\tau})$). On peut dès lors classifier les politiques en quatre catégories de politiques : les politiques markoviennes et déterministes $\Pi^{\mathbf{MD}}$; les politiques markoviennes aléatoires $\Pi^{\mathbf{MA}}$; les politiques non Markoviennes et déterministes $\Pi^{\mathbf{HD}}$; et les politiques non Markoviennes et aléatoires $\Pi^{\mathbf{HA}}$. La synthèse de cette discussion est fournie au Tableau 1.1. Nous appelons politique station-

naire toute politique $\pi_{0:\tau}$ telle que $d_\tau = d$ pour tout $\tau = 1, 2, \cdots, N-1$. On peut voir une politique déterministe comme un cas dégénéré de politique aléatoire telle que $\mathbb{P}(a = a_\tau|\cdot) = 1$ ou $\mathbb{P}(a = a_\tau|\cdot) = 0$. Une politique stationnaire est de la forme $\pi_{0:\tau} = (d, d, \cdots)$, nous la noterons pour simplifier d sans distinction de l'instant de décision. Une politique est dite pure si elle est stationnaire et déterministe. Il en résulte deux autres classes de politiques : politiques stationnaires et déterministes $D^{\mathbf{SD}}$ et les politiques stationnaires et aléatoires $D^{\mathbf{SA}}$.

1.3 Contributions de la thèse

De nombreux modèles de prise de décisions dans l'incertain ont été proposés par des chercheurs en planification, théorie de la décision, recherche opérationnelle, théorie du contrôle, ou encore en économie. Bon nombre de ces formulations s'inscrivent dans un modèle plus général appelé modèle de contrôle des processus décisionnels de Markov [Bertsekas, 2005, Putterman, 1994]. Ce modèle prend en compte non seulement les effets des actions courantes mais aussi ceux des actions à venir. Cela permet en outre de modéliser une large panoplie d'applications réelles. La famille des modèles de contrôle des processus décisionnels de Markov peut être vue comme une généralisation des modèles classiques de planification, y compris STRIPS [Fikes and Nilsson, 1971] et GraphPlan [Blum and Furst, 1995]. En effet, les planificateurs en Intelligence Artificielle se sont initialement cantonnés à la recherche de séquences d'actions permettant d'atteindre un but prédéfini suivant une représentation logique [Hoffmann et al., 2006] ou symbolique [**?**] d'un système déterministe : c'est la planification déterministe. Les modèles de contrôle des processus décisionnels de Markov généralisent ces modèles de planification déterministe. D'une part, ils offrent une plus grande diversité sur la nature des actions possibles. Certaines des actions sont en effet stochastiques : les effets de l'exécution de ces actions sont incertains. D'autre part, les modèles de contrôle des processus décisionnels de Markov offrent de plus riches critères de sélection d'un plan. Plutôt que de sélectionner un plan quelconque capable d'atteindre le but prédéfini, ils différentient les plans selon une fonction objectif minimisant des coûts ou maximisant des récompenses. De plus les modèles de contrôle des processus décisionnels de Markov permettent de formaliser les applications dont la rétroaction du système offre une information partielle sur l'état courant du système [Lovejoy, 1991, Sondik, 1978, **?**]. Enfin, ils peuvent intégrer l'influence d'un ou de plusieurs agents à la fois. En particulier, ils offrent la possibilité d'étudier le contrôle distribué d'un système par un groupe d'agents. Ce dernier modèle est nommé modèle de contrôle distribué des processus de Markov (DEC-POMDPs/DEC-MDPs) [Bernstein et al., 2000, 2002, Pynadath and Tambe, 2002].

Les modèles de Markov tombent ainsi dans une classe plus vaste des modèles de planification probabiliste. Il s'agit des modèles dont les transitions d'un état à un autre se font suivant une distribution de probabilités. Il est à noter que la classe des modèles probabilistes inclus :

1. la planification conditionnelle avec actions non probabilistes : ce type de modèles produit une séquence d'actions formant un plan dont le succès est garanti quelque soit les conditions rencontrées [**?**].
2. la planification contingentée avec actions non probabilistes : contrairement au cas précédent, ce type de modèles produit des plans contingentés. Il s'agit de plans dont les effets et le choix des actions sont conditionnés par les rétroactions du système. Comme les actions sont déterministes, le planificateur doit alors rechercher un plan dont le succès est assuré quelque soit les circonstances. On parle alors de plan universel.
3. la planification conditionnelle avec actions probabilistes : ces planificateurs sont similaires à ceux introduits au cas (1), à l'exception les actions sont stochastiques. Dès lors, le planificateur doit rechercher une séquence d'actions dont la probabilité de succès est la plus grande [Nicola et al., 2001].
4. la planification contingentée avec actions probabilistes : comme dans le second cas, le planificateur produit des plans contingentés. De façon similaire au cas (3), les actions probabilistes permettent au planificateur de sélectionner le plan dont la probabilité de succès est la plus grande. D'autres critères peuvent être utilisés lors de la sélection de ces plans contingentés. Parmi de nombreux critères possibles, on note : la minimisation de la longueur du plan ; ou encore la maximisation de l'espérance des récompenses du plan [Blum and Langford, 1998].

Les processus décisionnels de Markov et ses extensions tombent dans la catégorie des modèles de planification contingentée avec actions probabilistes. Cette dernière inclut toutes les autres catégories. Malheureusement, la caractère générique des processus décisionnels de Markov et ses extensions s'accompagne de très grandes difficultés quant à la résolution des applications réelles qu'ils parviennent à modéliser.

Dans cette thèse, nous faisons références à ces difficultés au travers des trois facteurs suivants : la malédiction de la dimension [Bellman, 1957] ; la malédiction de l'historique [Pineau et al., 2003b] ; et la moins connue de toutes et sans doute la pire la malédiction de la distributivité. Nous proposons en particulier de réduire l'impact de toutes ces malédictions au travers d'algorithmes efficaces. D'une part, nous réduisons la complexité liée à la fois à la malédiction de la dimension et la malédiction de l'historique via une approche nommée planification topologique pour le contrôle centralisé des processus de

Markov (Section 1.3.1). Dans sa version courante cette approche propose une famille d'algorithmes très performants pour la résolution des problèmes de contrôle centralisé des processus décisionnels de Markov. D'autre part, nous démontrons qu'une planification centralisée pour le contrôle distribué des processus de Markov est possible. Et surtout, nous offrons une analyse formelle d'une telle approche (Section 1.3.2). Les algorithmes de planification centralisée pour le contrôle distribué offrent une bonne réponse à la malédiction de la distributivité. Ils sont, à ce jour, les meilleurs pour la résolution des problèmes de contrôle distribué des processus décisionnels de Markov.

1.3.1 Planification topologique pour le contrôle centralisé

L'une des méthodes en planification pour le contrôle centralisé des processus de Markov consiste à rechercher une politique en maximisant une fonction de valeurs. La fonction de valeurs fait office de fonction objectif dans les processus décisionnels de Markov. L'approche construit efficacement une politique optimale ou presqu'optimale. En effet, elle permet de comparer numériquement différents choix d'actions permettant d'atteindre ou de satisfaire un but tout en optimisant un critère objectif. Cependant, rechercher le plus rapidement et précisément la fonction de valeurs requiert des efforts conséquents en calculs. De nombreuses tentatives ont été proposées afin d'accélérer les algorithmes classiques, y compris l'algorithme d'itération de valeurs [Howard, 1960] et l'algorithme d'itération de politiques [Putterman, 1994]. Parmi ces approches, les algorithmes dits prioritisés suggèrent des solutions aux problèmes de l'ordre des mises à jour des entrées de la dite fonction de valeurs. Les algorithmes prioritisés permettent de réduire considérablement le nombre requis de mises à jour. Néanmoins, ils rajoutent un coût additionnel parfois prohibitif. Se pose alors la question suivante : « est-il possible de réduire substantiellement le nombre de mises à jour de la fonction de valeurs tout en ne rajoutant qu'un coût additionnel négligeable ? »

C'est à cette question que tente de répondre la planification topologique que nous proposons dans le cadre du contrôle centralisé des processus décisionnels de Markov. Selon la classe des processus de Markov à laquelle on est confronté, l'approche est sensiblement différente mais le principe général est invariant.

Dans le cas le plus simple des problèmes de contrôle centralisé des processus décisionnels de Markov complètement observables (MDPs), la fonction de valeurs associe à tout état du système une valeur réelle. La planification topologique dans ce contexte procède d'abord à l'étude des dépendances causales entre les états du système. Puis elle extrait la structure des dépendances causales associée aux états, aussi nommée structure topologique. Enfin, elle effectue des mises à jour de la fonction de valeurs suivant l'ordre inverse

de l'ordre topologique sur cette structure.

Dans le cas le plus complexe des problèmes de contrôle centralisé des processus décisionnels de Markov partiellement observables (POMDPs), la fonction de valeurs associe à tout historique d'actions et d'observations une valeur réelle. La planification topologique se décline dans ce cadre d'abord en l'étude des dépendances causales entre les historiques. Puis, elle procède à l'extraction de la structure topologique associée. Et enfin, elle réalise la mise à jour des historiques suivant l'ordre inverse de l'ordre topologique induit par la dite structure.

L'efficacité de ce raisonnement réside dans le fait qu'en regroupant et ordonnant les états ou historiques sélon leur degré de dépendance, nous réduisons considérablement le nombre de mises à jour inutiles. En effet, chaque groupe d'états ou d'historiques est virtuellement mise à jour une et une seule fois. De façon similaire aux MDPs et POMDPs, ce raisonnement peut s'étendre au cadre général de la prise de décisions séquentielles. De plus, nous montrons que cette approche s'intègre facilement dans des paradigmes antérieurs de calcul de la fonction de valeurs, y compris la recherche par chaînage avant et la recherche par chaînage arrière. Enfin, cette approche a été validée sur différents domaines de la littérature des MDPs et POMDPs. Elle a montré des performances remarquables sur l'ensemble de ces domaines. Elle offre de meilleures performances lorsque le domaine est naturellement structuré. Certaines applications sont tout naturellement plus appropriées à cette approche. En particulier, les applications liées aux chaînes d'approvisionnement, où la tâche principale peut se décomposer en plusieurs tâches avec des dépendances causales strictes entre elles.

1.3.2 Planification centralisée pour le contrôle distribué

À la différence des problèmes de contrôle centralisé, le contrôle distribué des processus décisionnels de Makov (DEC-POMDPs) offre un challenge de taille. En effet, l'analyse de complexité de ce modèle révèle que est NEXP-difficile ou indécidable. Outre cette complexité, la recherche en DEC-POMDPs est très recente et explique en partie l'écart tant du point de vu théorique que du point de vu algorithmique en comparaison aux sous-classes MDPs et POMDPs. Les approches proposées à ce jour peuvent-être classées comme des approches de planification distribuée pour le contrôle distribué des processus décisionnels de Markov. Nous nommons ainsi ces approches car elles consistent en la distribution du problème général en plusieurs sous-problèmes représentant chacun un problème mono-agent. Cela de sorte à pouvoir utiliser les techniques de planification developées dans le cadre du contrôle centralisé. Ainsi, la résolution du problème entier correspond en la résolution de l'ensemble des problèmes mono-agent

À la différence des problèmes de contrôle centralisé, le contrôle distribué des processus décisionnels de Markov (DEC-POMDPs/DEC-MDPs) offre un challenge de taille. En effet, l'analyse de complexité de ce modèle révèle que le problème de décision associé est indécidable dans le cas à horizon infini et NEXP-difficile dans le cas à horizon fini. Outre cette complexité, la recherche en DEC-POMDPs est très récente et explique en partie l'écart tant du point de vu théorique que du point de vu algorithmique en comparaison aux sous-classes MDPs et POMDPs. Les approches proposées à ce jour peuvent être classées comme des approches de planification distribuée pour le contrôle distribué des processus décisionnels de Markov. Nous nommons ainsi ces approches car elles consistent en la distribution du processus de résolution du problème en plusieurs processus de résolution. Ils représentent chacun un problème mono agent. Cela de sorte à pouvoir utiliser les techniques de planification développées dans le cadre du contrôle centralisé. Ainsi, la résolution du problème entier correspond en la résolution d'un ensemble des problèmes mono agent [Hansen, 1998, Kaelbling et al., 1995, Putterman, 1994].

Contrairement à la planification distribuée pour le contrôle distribué des processus décisionnels de Markov, nous proposons une approche fondamentalement différente. Elle est nommée la planification centralisée pour le contrôle distribué des processus décisionnels de Markov. La différence réside dans le fait que l'ensemble des agents est vu maintenant comme un et un seul agent central. Parce que lors du contrôle distribué des processus décisionnels de Markov, un agent n'a aucune connaissance des perceptions des autres agents, il est difficile d'envisager une planification centralisée pour le contrôle distribué des processus décisionnels de Markov. Néanmoins, nous décrivons et démontrons qu'il est non seulement possible de planifier de façon centralisée pour le contrôle distribué des processus décisionnels de Markov, mais surtout qu'il est possible de le faire de manière exacte. Nous généralisons, pour y parvenir, un grand nombre de résultats théoriques connus dans le cadre des MDPs et POMDPs au cadre général des processus de prise de décisions séquentielles, y compris au cadre des DEC-MDPs et DEC-POMDPs. Nous réintroduisons et démontrons tous les concepts de base, y compris le critère d'optimalité, les équations d'optimalité, le principe d'optimalité, et la statistique suffisante, pour la résolution des problèmes de contrôle des processus de Markov à temps discrets. Nous offrons en outre une nouvelle analyse sur les origines de la complexité des DEC-POMDPs à travers la notion de la malédiction de la distributivité.

Fort de cette étude théorique du problème, nous généralisons les algorithmes d'itération de valeurs et politiques pour des DEC-POMDPs à horizon infini. De même, nous proposons un algorithme d'induction rétrograde pour la résolution des DEC-POMDPs à horizon fini. Puis nous dérivons de ces algorithmes des versions approximatives bien plus efficaces, à mémoire limitée ou spécialisées aux sous-classes de DEC-POMDPs. Sur l'ensemble des benchmarks, ces algorithmes approximatifs sont de loin les meilleurs. La

planification centralisée pour le contrôle distribué des processus décisionnels de Markov est une approche prometteuse permettant d'exploiter au mieux l'ensemble des propriétés des systèmes à contrôler. Elle ouvre la voie à une nouvelle famille d'algorithmes exactes et approximatifs. L'une des contributions des plus importantes de cette thèse est la preuve d'une grande similarité entre l'ensemble des problèmes de prise de décisions séquentielles. Cette observation est de taille car cela signifie qu'une connaissance acquise dans un modèle peut toujours être exploitée dans l'ensemble des autres modèles. Elle s'oppose de ce fait à l'idée communément admise selon laquelle un POMDP serait très différent d'un DEC-POMDP : « de mon point de vue le premier correspond au second plus la contrainte de distributivité. »

1.4 Sommaire de la thèse

Cette thèse inclut à la fois une analyse rigoureuse des problèmes de prise de décisions séquentielles ainsi que de nouveaux algorithmes efficaces pour la résolution de ces problèmes. Les chapitres de cette thèse sont partagés en deux grandes parties. L'une établit l'état de l'art sur les modèles de contrôle des processus décisionnels de Markov, y compris : les MDPs ; les POMDPs ; et les DEC-POMDPs. L'autre partie décrit nos contributions dans chacun de ces modèles et plus généralement dans la résolution des processus de prise de décisions séquentielles :

Chapitre 1 **— Introduction**

Chapitre 2 **— Contôle centralisé des processus décisionnels de Markov.** Nous y discutons des systèmes mono agent, à savoir les processus décisionnels de Markov complètement observable (MDPs) ou partiellement observable (POMDPs). Nous revenons succinctement sur ces modèles ainsi que sur les algorithmes dont les plus performants à ce jour. Ce chapitre permet au lecteur d'avoir un regard général sur l'ensemble des méthodes de résolution des MDPs et POMDPs.

Chapitre 3 **— Contôle distribué des processus décisionnels de Markov.** Nous y discutons des systèmes multi-agents. En particulier, nous motivons et expliquons l'intérêt du contrôle distribué. Nous introduisons par la suite le modèle des DEC-POMDPs ainsi que ses innombrables sous-classes. Nous rappelons les algorithmes classiques pour la résolution des DEC-POMDPs à horizon fini aussi bien qu'à horizon infini. Nous discutons en profondeur des approches approximatives tout en soulignant leurs différences, forces et faiblesses.

Chapitre 4 **— Résolution topologique pour le contrôle centralisé.** Nous abordons les contributions de cette thèse dans la résolution des problèmes de contrôle centralisé des

processus décisionnels de Markov. Nous introduisons l'idée de la planification topologique ainsi que les motivations théoriques et ses applications. Nous décrivons l'ensemble des algorithmes exactes et approximatifs pour la résolution des MDPs mais aussi des POMDPs, dans ce cadre.

Chapitre 5 — **DEC-POMDPs à Horizon fini.** Ce chapitre expose mes contributions aux problèmes du contrôle distribué des processus décisionnels de Markov partiellement observables, à temps discret et horizon fini. Nous y refondons, introduisons et démontrons tous les concepts de base, y compris : le critère d'optimalité ; les équations d'optimalité ; le principe d'optimalité ; et la statistique exhaustive pour la résolution des problème de contrôle des processus de Markov à temps discrets et horizon fini, dont les MDPs, POMDPs, DEC-MDPs et DEC-POMDPs. Ces résultats sont soit des extensions des résultats existants dans le cadre des MDPs et POMDPs soit des résultats spécifiques au contrôle distribué, c'est à dire les DEC-MDPs et DEC-POMDPs et ses sous-classes. Outre l'étude théorique de la planification centralisée, nous proposons également une famille d'algorithmes de résolution des problème de prise de décisions séquentielles et plus particulièrement des DEC-POMDPs à temps discrets et horizon fini.

Chapitre 6 — **DEC-POMDPs à Horizon infini.** À l'instar du chapitre 5, ce chapitre expose des contributions aux problèmes de contrôle distribué des processus décisionnels de Markov à temps discrets et à horizon infini.

Chapitre 7 — **DEC-MDPs.** Ce chapitre introduit la planification centralisée pour le contrôle distribué des processus de Markov munis de l'hypothèse d'observabilité conjointe totale. Cette hypothèse additionnelle distingue un DEC-POMDP d'un DEC-MDP. Sous cette hypothèse, nous montrons que la statistique suffisante pour la planification exacte un DEC-MDP est la distribution des probabilités sur l'ensemble des observations conjointes. Ce résultat tranche radicalement avec l'état de l'art. En effet, jusqu'ici dans le cadre du contrôle distribué des processus de Markov, il n'existait pas de statistique suffisante. Il fallait alors considérer l'ensemble des historiques possibles de tous les agents afin de résoudre les problèmes modéliser sous forme de DEC-MDPs. Ce résultat offre des perspectives prometteuses, dont une plus grande capacité d'applicabilité du modèle au applications réelles.

Chapitre 8 — **Conclusion globale.** Nous y rappelons l'ensemble des contributions de la thèse. Nous discutons en outre des perspectives possibles des contributions de cette thèse. D'une part, nous soutenons que la planification topologique s'applique tout naturellement au cadre du contrôle distribué des processus de Markov (DEC-POMDP, DEC-MDP, ND-POMDP, . . .). D'autre part, nous soulignons que l'effort réalisé dans l'analyse des DEC-POMDPs (Chapitre 5) servira à établir des résultats très prometteur dans le cadre des DEC-MDPs, offrant ainsi des perspective jusqu'ici inespérées. Nous y discutons également les questions toujours ouvertes à ce jour.

Première partie

État de l'art

« Intuition and concepts constitute ... the elements of all our knowledge, so that neither concepts without an intuition in some way corresponding to them, nor intuition without concepts, can yield knowledge. »

Immanuel Kant

Chapitre 2

Contrôle centralisé des processus décisionnels de Markov

« Now comes the "Funny Quote", written in italics. »

CONSIDÉRONS le problème de construction d'une politique permettant de guider un robot dans sa mission d'exploration de la planète Mars. L'échantillonnage de roches puis le déplacement d'une roche à une autre, requiert l'usage des capteurs du robot. Lorsque ces capteurs sont complètement fiables, le robot perçoit complètement l'état du système. Malheureusement, dans la grande majorité des cas, les capteurs des robots ne sont que partiellement fiables, ainsi le perception de l'état du système est partielle.

Ce chapitre rappelle les modèles mono-agent de contrôle des processus décisionnels de Markov complètement ou partiellement observables, utiles respectivement en cas de capteurs complètement fiables – Section 2.1 – ou partiellement fiables – Section 2.3. En outre, nous y discutons des méthodes de contruction de politiques de contrôle de ces processus décisionnels de Markov en Section 2.2,2.4 et 2.5.

le problème de construction d'une politique permettant de guider un robot dans sa mission d'exploration de la planète Mars. L'échantillonnage de roches puis le déplacement d'une roche à une autre, requiert l'usage des capteurs du robot. Lorsque ces capteurs sont complètement fiables, le robot perçoit complètement l'état du système. Malheureusement, dans la grande majorité des cas, les capteurs des robots ne sont que partiellement fiables, ainsi le perception de l'état du système est partielle.

Ce chapitre rappelle les modèles mono agent de contrôle des processus décisionnels de Markov complètement ou partiellement observables. Ces derniers sont utiles respectivement en cas de capteurs complètement fiables – Section 2.1 – ou partiellement fiables – Section 2.3. En outre, nous y discutons des méthodes de construction de politiques de contrôle de ces processus décisionnels de Markov en Section Section 2.2,2.4 et 2.5.

Hypothèses 1. *Nous supposons tout au long de ce chapitre que :*

1. *S, A, et Ω sont discrets et finis,*
2. *T, et R sont des distributions de probabilités stationnaires,*
3. *les récompenses sont bornées, $|R(s,a)| \leqslant M < \infty$, pour tout $(s,a) \in S \times A$.*

2.1 Processus décisionnels de Markov

Dans cette section, nous abordons la modélisation et la résolution des problèmes mono agent de prise de décisions séquentielles à temps discrets, à horizon $\tau = 0, 1, \cdots, N$ fini ($N < \infty$) ou infini ($N = \infty$), pour les environnements complètement observables, comme illustrer à la Figure 2.1. L'hypothèse d'observabilité complète signifie que lors de l'exécution d'une action $a_{\tau-1}$ dans un tel environnement, le décideur central perçoit de l'environnement une information ω_τ qui révèle entièrement l'état courant du système s_τ. Plus formellement, nous avons $\mathbb{P}(s_\tau|\omega_\tau) = 1$. Cela lui permet par la suite de sélectionner une nouvelle action a_τ à exécuter, pour tout horizon $\tau = 0, 1, \cdots, N-1$. Un tel processus d'observation complète de l'état courant du système, puis de sélection d'une action à exécuter , est appelé processus décisionnel de Markov (MDP).

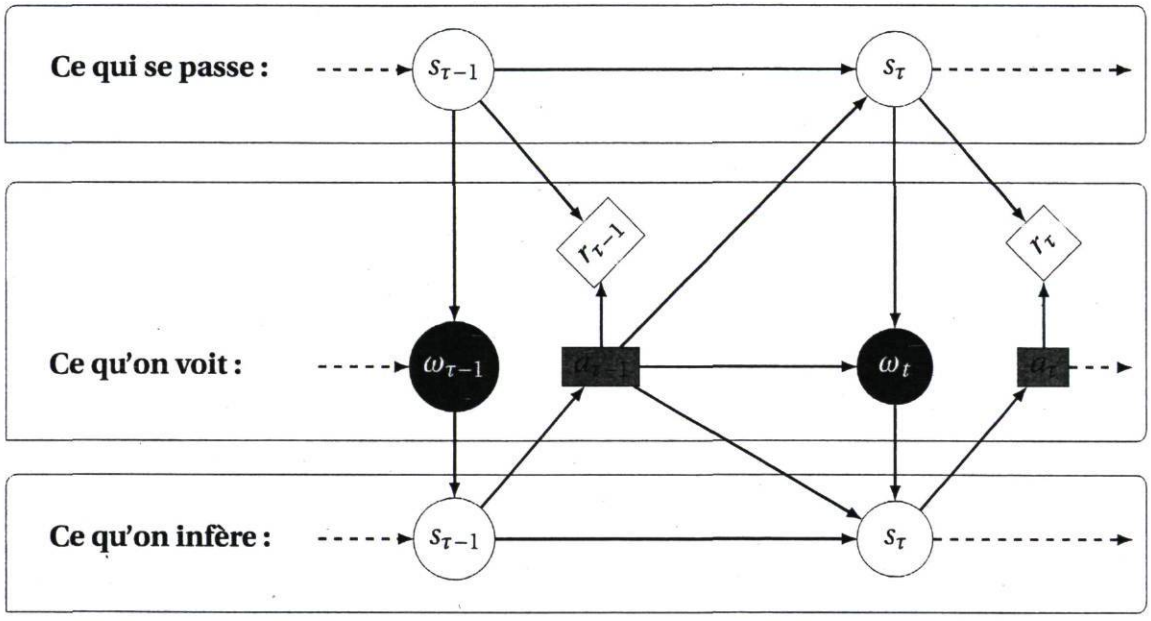


FIGURE 2.1 – Processus décisionnel de Markov – adapté de Pineau [2004].

La Figure 2.1 illustre le cycle de contrôle d'un processus décisionnel de Markov complètement observable. Nous y distinguons trois principales composantes. La première composante « ce qui se passe » décrit l'évolution du système dans le temps, c'est à dire la transition d'un état $s_{\tau-1}$ à un autre du système : c'est le point de vu d'un observateur extérieur. La seconde composante « ce qu'on voit » représente l'interaction du décideur central avec son environnement : c'est la perception de la rétroaction de l'environnement $(\omega_\tau, r_{\tau-1})$ suite à l'exécution d'une action $a_{\tau-1}$. Enfin, la dernière composante « ce qu'on infère » illustre l'information courante extraite par le décideur central suite aux interactions avec son environnement. En conséquence de l'hypothèse d'observabilité complète, le décideur central peut extraire de la rétroaction $(\omega_\tau, r_{\tau-1})$ issue de l'environnement, l'état

courant s_τ du système. Ce formalisme permet tout aussi bien de modéliser un problème général de planification qu'une application issue du monde réel et vérifiant l'hypothèse d'observabilité complète [Powell, 2006, Putterman, 1994, Si et al., 2004, Zilberstein et al., 2001].

2.1.1 Le formalisme d'un MDP

Définition 1. *Un processus décisionnel de Markov est défini par un* 4*-uplet* (S, A, T, R)*, où :*

- $S = \{s_0, s_1, \cdots, s_N\}$ *dénote l'ensemble fini des états* s_τ *possibles du système, pour tout horizon* $\tau = 0, 1, \cdots, N-1$.
- $A = \{a_0, a_1, \cdots, a_{N-1}\}$ *représente l'ensemble fini des actions* a_τ *disponibles, pour tout horizon* $\tau = 0, 1, \cdots, N-1$.
- *La loi de transition d'un état à un autre est définie par* $T(s'|s, a)$ *et telle que :*

$$T(s'|s,a) = \mathbb{P}(s_\tau = s'|s_0, a_0, s_1, a_1, \cdots, s_{\tau-1} = s, a_{\tau-1} = a) \quad (2.1)$$

$$= \mathbb{P}(s_\tau = s'|s_{\tau-1} = s, a_{\tau-1} = a) \quad (2.2)$$

  pour tout horizon $\tau = 0, 1, 2, \cdots, N-1$. *Le passage de l'équation (2.1) à l'équation (2.2) est dû à la propriété de Markov.* $T(s'|s, a)$ *est une distribution de probabilités caractérisant la probabilité de transiter dans l'état* $s_\tau = s'$*, suite à l'exécution de l'action* $a_{\tau-1} = a$ *dans l'état* $s_{\tau-1} = s$*, et cela pour tout triplet* (s, a, s').
- $R(s, a) : S \times A \to \Re$ *associe à toute paire état action* (s, a) *une valeur numérique réelle quantifiant l'utilité d'exécuter l'action a dans l'état s. La fonction de récompenses peut également s'écrire comme* $R(s) : S \to \Re$*. Il s'agit alors d'associer à tout état de l'agent une valeur réelle.*

2.1.2 Problème décisionnel de Markov

On appelle problème décisionnel de Markov, tout processus décisionnel de Markov muni d'un critère d'optimisation. Ce problème a pour hypothèses, une connaissance complète et exacte des lois qui régissent la dynamique du système. En particulier, il est muni d'une loi stochastique T de transition d'un état à un autre et une fonction de récompenses ou coûts R. Il est communément choisi comme critère d'optimisation l'espérance mathématique sur la somme cumulée des récompenses r_τ, pour tout $\tau = 0, 1, \cdots, N$. En effet, étant donné un MDP, l'objectif de l'agent est de sélectionner une politique π lui permettant de maximiser le total espéré des récompenses cumulées au cours du temps.

Propriétés des politiques

D'après la propriété de Markov, une politique π d'un tel processus ne dépend de l'historique h_τ qu'au travers de l'état courant s_τ. Ainsi, pour T et R stationnaires, il existe toujours une politique $\pi \in \Pi^{\mathbf{MA}}$ dans l'ensemble des politiques Markoviennes et aléatoires qui soit optimale pour le contrôle d'un MDP [Blackwell, 1965]. De plus, Putterman [1994] démontre qu'il existe toujours au moins une politique pure. C'est à dire une politique déterministe et Markovienne, dont le total espéré des récompenses cumulées soit au moins aussi élevé que celui d'une quelconque politique Markovienne et aléatoire. Ces propriétés permettent de simplifier considérablement la représentation des politiques à rechercher. Ceci est possible en les restreignant dans la classe des politiques pures $\pi \equiv (d, d, \cdots, d)$, où $d \in D^{\mathbf{MD}}$ est une règle de décisions. En d'autres termes, plutôt que de rechercher des politiques aléatoires et non Markoviennes $\pi \equiv (d_0, d_1, \cdots, d_{N-1})$, il nous est possible de limiter notre recherche dans les règles de décisions pures $d \in D^{\mathbf{MD}}$. Nous rappelons que dans ce contexte, il est possible de remplacer (abusivement) la règle de décisions d par sa politique pure associée $\pi = (d, d, \cdots, d)$.

Critères d'optimisation

Afin de différencier deux règles de décisions, nous utilisons l'espérance mathématique, $v^d_{0:N}$. Elle représente le total espéré des récompenses cumulées au cours des instants de décisions $\tau = 0, 1, \cdots, N-1$. Mathématiquement, cela s'écrit différemment selon que nous soyons dans le cas fini $N < \infty$,

$$v^d_{0:N}(s_0) = \mathbb{E}_d \left\{ \sum_{\tau=0}^{N-1} \lambda^\tau R(s_\tau, a_\tau) + \lambda^\tau R(s_N) \mid d(s_\tau) = a_\tau \right\} \tag{2.3}$$

ou dans le cas infini $N = \infty$,

$$v^d_{0:N}(s_0) = \mathbb{E}_d \left\{ \sum_{\tau=0}^{\infty} \lambda^\tau R(s_\tau, a_\tau) \mid d(s_\tau) = a_\tau \right\} \tag{2.4}$$

Dans ce dernier cas, il est nécessaire d'introduire un facteur de décompte noté $\lambda \in [0, 1)$. Ce facteur permet de garantir que le total espéré des récompenses cumulées reste borné. Ce paramètre, essentiel dans le cas à horizon infini, peut être également introduit dans le cas à horizon fini. Il peut s'expliquer par le poids attribué aux récompenses futures. Le critère d'optimisation est alors défini comme le total espéré des récompenses cumulées et décomptées. Ce critère n'affecte pas les performances des algorithmes. Cependant, il peut, dans tous les cas, modifier les préférences du décideur ou de l'agent quant à la politique optimale

Dans la théorie des processus décisionnels de Markov et ses algorithmes, l'objectif premier est : d'une part, la recherche d'une règle de décisions $d^* \in D^{\mathbf{MD}}$ dont le total espéré des récompenses cumulées est maximal ; et d'autre part, la caractérisation de la valeur de cette règle de décisions pour chaque état du système. En d'autres termes, cela revient à rechercher une règle de décisions d^* telle :

$$v_{0:N}^{d^*}(s) \geqslant v_{0:N}^{d}(s), \quad \forall s \in S, \forall d \in D^{\mathbf{MD}}. \tag{2.5}$$

Une telle règle de décisions d^* ainsi que la politique $\pi^* \equiv (d^*, d^*, \cdots, d^*)$ sont dites *optimales.* Dans certain cas, soit une telle règle de décisions n'existe pas soit les ressources nécessaires pour sa construction sont exorbitantes. Il est alors préférable de rechercher une règle de décisions à ε de l'optimum. Plus formellement, cela revient à rechercher, pour tout réel $\varepsilon > 0$, une règle de décisions d_ε^* telle que :

$$v_{0:N}^{d_\varepsilon^*}(s) + \varepsilon \geqslant v_{0:N}^{d}(s), \quad \forall s \in S, \forall d \in D^{\mathbf{MD}}. \tag{2.6}$$

Une telle règle de décisions d_ε^* ainsi que la politique $\pi_\varepsilon^* \equiv (d_\varepsilon^*, d_\varepsilon^*, \cdots, d_\varepsilon^*)$ sont dites ε-*optimales.* Dans les modèles à horizon infini, nous recherchons une règle de décisions d dont la valeur est la plus large :

$$v^d(s) = \lim_{N \to \infty} v_{0:N}^{d}(s), \quad \forall s \in S \, \forall d \in D^{\mathbf{MD}}. \tag{2.7}$$

Grâce au facteur de décompte λ, la valeur $v^d(s)$ ne diverge pour aucun état $s \in S$. Les vecteurs de valeurs associés aux règles de décisions sont appelés *fonctions de valeurs.*

Sachant un MDP, l'objectif premier est : d'une part, la recherche d'une règle de décisions $d^* \in D^{\mathbf{MD}}$ dont le total espéré des récompenses cumulées est maximal ; et d'autre part, la caractérisation de la valeur de cette règle de décisions pour chaque état du système. En d'autres termes, cela revient à rechercher une règle de décisions d^* telle :

$$v_{0:N}^{d^*}(s) \geqslant v_{0:N}^{d}(s), \quad \forall s \in S, \forall d \in D^{\mathbf{MD}}. \tag{2.8}$$

La règle de décisions d^* et la politique $\pi^* \equiv (d^*, d^*, \cdots, d^*)$ sont dites optimales. Dans certain cas, soit une telle règle de décisions n'existe pas soit les ressources nécessaires pour sa construction sont exorbitantes. Il est alors préférable de rechercher une règle de décisions à ε de l'optimum. Plus formellement, cela revient à rechercher, pour tout réel $\varepsilon > 0$, une règle de décisions d_ε^* telle que :

$$v_{0:N}^{d_\varepsilon^*}(s) + \varepsilon \geqslant v_{0:N}^{d}(s), \quad \forall s \in S, \forall d \in D^{\mathbf{MD}}. \tag{2.9}$$

Une telle règle de décisions d_ε^* ainsi que la politique $\pi_\varepsilon^* \equiv (d_\varepsilon^*, d_\varepsilon^*, \cdots, d_\varepsilon^*)$ sont dites ε-optimales. Dans les modèles à horizon infini, nous recherchons une règle de décisions d dont la valeur est la plus large :

$$v^d(s) = \lim_{N \to \infty} v_{0:N}^{d}(s), \quad \forall s \in S \, \forall d \in D^{\mathbf{MD}}. \tag{2.10}$$

Grâce au facteur de décompte λ, la valeur $v^d(s)$ne diverge pour aucun état $s \in S$. Les vecteurs de valeurs associés aux règles de décisions sont appelés fonctions de valeurs.

2.1.3 Fonction de valeurs

Nous caractérisons la fonction de valeurs $v^*_{0:N}$ d'un problème décisionnel de Markov comme la valeur satisfaisant :

$$v^*_{0:N}(s) = \max_{d \in D^{\mathbf{MD}}} v^d_{0:N}(s), \quad \forall s \in S. \tag{2.11}$$

Ainsi le total espéré des récompenses cumulées d'une règle de décisions d^* satisfait :

$$v^{d^*}_{0:N}(s) = v^*_{0:N}(s), \quad \forall s \in S \tag{2.12}$$

et la valeur d'une règle de décisions ε-optimales satisfait :

$$v^{d^*_\varepsilon}_{0:N}(s) + \varepsilon \geqslant v^*_{0:N}(s), \quad \forall s \in S \tag{2.13}$$

Passons à la limite infinie les équations (2.11),(2.12), et (2.13). Pour les modèles à horizon infini munis du facteur de décompte ?, les équations résultantes caractérisent respectivement la fonction de valeurs optimales,

$$v^*(s) = \max_{d \in D^{\mathbf{MD}}} v^d(s), \quad \forall s \in S, \tag{2.14}$$

le total espéré des récompenses cumulées et décomptées d'une règle de décisions d^* :

$$v^{d^*}(s) = v^*(s), \quad \forall s \in S \tag{2.15}$$

et la fonction de valeurs d'une règle de décisions ε-optimales :

$$v^{d^*_\varepsilon}(s) + \varepsilon \geqslant v^*(s), \quad \forall s \in S \tag{2.16}$$

Le lecteur notera qu'il n'est pas toujours nécessaire de construire une règle de décisions optimales d^* pour tout état initial $s_0 \in S$. En effet, parfois le décideur à connaissance de l'état initial de l'agent. Dès lors, il lui suffit de construire une règle de décisions dont le total espéré des récompenses cumulées à long terme est maximal uniquement pour cet état. Alternativement, le décideur peut ne pas connaître avec exactitude l'état initial de l'agent. Il se peut qu'il ne dispose que d'une distribution de probabilités $\mathbb{P}(s_0 = s)$ sur l'ensemble des états du système. Il est alors facile de montrer qu'une règle de décisions optimales dans ce cas est donnée par la recherche d'une règle de décisions dont la valeur $v^*_{0:N}(s)$ est maximale pour tout état s dans lequel l'agent pourrait être initialement ($\mathbb{P}(s_0 = s) \neq 0$). Dans tous les cas, une règle de décisions satisfaisant l'équation (2.12) est optimale.

2.1.4 Prise de décisions optimales

Cas à horizon fini

Nous introduisons maintenant les équations d'optimalité (aussi appelées équations de Bellman) ainsi que le principe d'optimalité de Bellman [Bellman, 1957, Putterman, 1994]. Soit

$$v^*_{\tau:N}(s) \quad = \quad \sup_{d \in D^{\mathbf{MD}}} v^d_{\tau:N}(s), \quad \forall s \in S, \tag{2.17}$$

Le supremum est défini sur l'ensemble des politiques Markoviennes et déterministes possibles des totaux espérés des récompenses cumulées à long terme. Les équations d'optimalité sont alors définies par :

1. pour tout horizon $\tau = 0, 1, \cdots, N-1$,

$$v_{\tau:N}(s) \quad = \quad \sup_{a \in A} \left\{ R(s,a) + \sum_{s' \in S} T(s'|s,a) \cdot v_{\tau+1:N}(s') \right\} \tag{2.18}$$

2. pour l'horizon $\tau = N$, pour tout état $s \in S$, nous avons la condition limite,

$$v_{N:N}(s) \quad = \quad R(s) \tag{2.19}$$

Le supremum, équation (2.18), est toujours atteint car les ensembles A et S sont finis. Dès lors, il est possible de remplacer l'opérateur « sup » par l'opérateur « max » comme suit,

$$v_{\tau:N}(s) \quad = \quad \max_{a \in A} \left\{ R(s,a) + \sum_{s' \in S} T(s'|s,a) \cdot v_{\tau+1:N}(s') \right\} \tag{2.20}$$

Une solution au système d'équations (2.18) et (2.19) est une séquence de fonctions de valeurs optimales $v^*_{\tau:N} \colon S \to \Re$, pour tout horizon $\tau = 0, 1, \cdots, N$, avec la propriété que $v^*_{N:N}$ satisfait l'équation (2.19), $v^*_{N-1:N}$ satisfait la (N?1)-ème équation, $v^*_{N-2:N}$ satisfait la (N?2)-ième équation, et ainsi de suite [Bellman, 1957, Putterman, 1994]. Parfois le déterminisme des fonctions de valeurs optimales est laborieux. Pour cette raison, nous nous contentons des fonctions de valeurs ε-optimales. Putterman [1994] a montré qu'en construisant une règle de décisions telle que : pour chaque état s_τ, la valeur $v_{\tau:N}(s_\tau)$ soit à $(N-\tau)\frac{\varepsilon}{N}$ de la valeur optimale $v^*_{\tau:N}(s_\tau)$, on est alors garantie d'obtenir une règle de décisions d^*_ε ε-optimales. Plus formellement, cela peut se réécrire,

$$v^{d^*_\varepsilon}_{\tau:N}(s) + (N-\tau)\frac{\varepsilon}{N} \quad \geqslant \quad v^*_{\tau:N}(s) \tag{2.21}$$

pour tout horizon $\tau = 0, 1, \cdots, N$, et pour tout état $s \in S$.

Cas à horizon infini $N = \infty$

Nous pouvons alors reformuler les équations d'optimalité, en passant à la limite infinie :

$$v(s) = \max_{a \in A} \left\{ R(s,a) + \lambda \sum_{s' \in S} T(s'|s,a) \cdot v(s') \right\} \tag{2.22}$$

Nous définissons alors $\mathbb{H}$: $(S \to \Re) \to (S \to \Re)$ comme un opérateur qui à toute fonction de valeurs ? associe une nouvelle fonction de valeurs, donnée sous forme matricielle par :

$$\mathbb{H}v \equiv \max_{d \in D^{\mathbf{MD}}} \{R_d + \lambda T_d \cdot v\} \tag{2.23}$$

où R_d est le vecteur de récompenses pour tout état s lorsque la règle de décisions d est exécutée $R_d(s) = R(s, d(s))$; T_d est la fonction de transitions d'un état s à un état s' lorsque l'agent suit la règle de décisions s', de sorte que $T_d(s'|s) = T(s'|s, d(s))$. Cet opérateur est dit opérateur de mises à jour. Nous noterons par abus $v_0, v_1, \cdots, v_\tau, \cdots$, la séquence V des fonctions de valeurs produite par l'application successive de l'opérateur $\mathbb{H}$ en partant d'une fonction de valeurs v_0 quelconque. On montre alors que l'opérateur $\mathbb{H}$ muni du facteur de décompte $\lambda \in [0,1)$, est un opérateur de contraction sur V pour les modèles à horizon infini : pour toute paire de fonctions de valeurs $v, u \in V$,

$$\|\mathbb{H}v - \mathbb{H}u\| \leqslant \lambda \|v - u\| \tag{2.24}$$

où l'opérateur $\|\cdot\|$ indique la norme max. Par conséquent, il existe toujours une fonction de valeurs optimales v^* au terme des application de l'opérateur $\mathbb{H}$. Elle est caractérisée par la propriété suivante :

$$\mathbb{H}v^* = v^* \tag{2.25}$$

Les deux équations précédentes (2.24) et (2.25) peuvent s'interpréter comme suit : L'application successive de l'opérateur $\mathbb{H}$, partant d'une fonction de valeurs initiale v_0 quelconque, permet des approximations $v_1, v_2, \cdots$ successives de la fonction de valeurs optimales v^*. Ce processus continu jusqu'à ce que $\mathbb{H}v_\tau = v_\tau$. Dès lors, la fonction de valeurs v_τ est égale à la fonction de valeurs optimales v^*.

2.2 Résolution exacte de MDPs

Dans cette section, nous rappelons les méthodes classiques de résolution de MDPs. Nous exposons les unes après les autres, les principales approches de résolution de MDPs,

y compris : par des approximations successives de la fonction de valeurs optimales ; puis par approximation de la politique optimale, en 2.2.1 ; et enfin par le déterminisme du temps passer par le système dans un état, en 2.2.3. Certains de ces algorithmes sont dits synchrones car les mises à jour des valeurs des états sont effectuées en parallèle, c'est à dire sans préférence ou priorité d'une partie des états sur les autres.

2.2.1 La programmation dynamique synchrone

La programmation dynamique est une méthode générale de résolution des problèmes d'optimisation séquentielle à temps discret [Bellman, 1957]. En particulier, elle définit le cadre général de deux des méthodes classiques de résolution des MDPs à savoir : l'algorithme d'itération de valeurs ; et l'algorithme d'itération de politiques.

Algorithme d'itération de valeurs

Cet algorithme se décline en deux versions dédiées respectivement au cas à horizon fini et au cas à horizon infini. Dans le cas horizon fini, l'algorithme d'itérations de valeurs consiste en la résolution des équations d'optimalité de Bellman (2.18) ainsi que la condition limite (2.19). L'algorithme 1 qui en résulte est alors appelé algorithme d'induction rétrograde (ou selon l'acronyme anglais backward induction) et noté BI [Bellman, 1957].

Algorithme 1 Algorithme d'induction rétrograde pour les MDPs.

1: **procedure** BI(S, A, T, R, N)
2: Posons $\tau \leftarrow N$ et $v^*_{N:N}(s) \leftarrow R(s)$ pour tout état $s \in S$.
3: **pour tout** $\tau = N-1, \cdots, 0$ **faire**
4: Calculons $v^*_{\tau:N}(s)$ pour tout état $s \in S$ par :

$$v^*_{\tau:N}(s) \leftarrow \max_{a \in A} \left\{ R(s,a) + \sum_{s'} T(s'|s,a) v^*_{\tau+1:N}(s') \right\} \tag{2.26}$$

5: **fin pour**
6: Retourner la politique optimale,

$$d^*(s) \leftarrow \operatorname*{argmax}_{a \in A} \left\{ R(s,a) + \sum_{s'} T(s'|s,a) v^*_{0:N}(s') \right\} \tag{2.27}$$

7: **fin procedure**

L'attrait de cet algorithme réside dans sa simplicité et son efficacité. L'étude de sa com-

plexité révèle qu'il ne requiert que $N|A||S|^2$ opérations élémentaires afin de déterminer la fonction de valeurs optimales $v^*_{0:N}$, et par conséquent la règle de décisions optimales. Bien qu'il y a $(|A|^{|S|})^{N-1}$ politiques déterministes et Markoviennes possibles, la complexité $\mathcal{O}(N|A||S|^2)$ en fait un algorithme d'une efficacité remarquable. En outre, il n'est pas nécessaire de mémoriser l'ensemble des fonctions de valeurs successives, il suffit de garder en mémoire la dernière fonction de valeurs $v_{\tau+1:N}$ afin de calculer la fonction de valeurs courantes $v_{\tau:N}$. Soit une complexité en mémoire de $\mathcal{O}(|S|)$.

Dans le cas à horizon infini, l'algorithme d'induction rétrograde est modifié comme décrit Algorithme 2. Il se nomme alors l'algorithme d'itération de valeurs (VI). Cet algorithme détermine une règle de décisions ε-optimales, d^*_ε, et une approximation de sa fonction de valeurs.

Algorithme 2 Algorithme d'itération de valeurs pour les MDPs.

1: **procedure** VI(S, A, T, R, λ)
2: Sélectionnons une fonction de valeurs initiales $v_0 \in V$, $\varepsilon > 0$, et $\tau = 1$.
3: **répéter**
4: Calculons $v_{\tau+1}(s)$ pour tout états $s \in S$ par :

$$v_{\tau+1}(s) \leftarrow \max_{a \in A} \left\{ R(s,a) + \lambda \sum_{s'} T(s'|s,a) \cdot v_\tau(s') \right\} \tag{2.28}$$

5: Poser $\tau = (\tau + 1)$.
6: **jusqu'à** $\|v_{\tau+1} - v_\tau\| \leqslant \frac{\varepsilon(1-\lambda)}{2\lambda}$
7: Si retourner la règle de décisions ε-optimales,

$$d^*_\varepsilon(s) \leftarrow \operatorname*{argmax}_{a \in A} \left\{ R(s,a) + \lambda \sum_{s'} T(s'|s,a) \cdot v_{\tau+1}(s') \right\} \tag{2.29}$$

8: **fin procedure**

Cet algorithme possède de nombreuses propriétés dont celles relatives à sa convergence vers une règle de décisions ε-optimales. En particulier, on montre que $\|v^{d^*_\varepsilon} - v^*\| \leqslant \varepsilon$ [Putterman, 1994]. Comme dans le cas fini, la complexité en temps de l'algorithme d'itération de valeurs est de $\mathcal{O}(|A||S|^2)$, tandis que celle en mémoire est de $\mathcal{O}(|S|)$.

Algorithme d'itération de politiques

L'algorithme d'itération de valeurs permet de déterminer une règle de décisions ε-optimales par une succession d'approximations de la fonction de valeurs optimales. Bien qu'efficace en général, il arrive parfois que le taux de convergence de cet algorithme vienne

à décevoir. Certaines applications réelles disposent de propriétés qui préconisent une recherche dans l'espace des règles de décisions. Il est alors préférable d'utiliser l'algorithme d'itération de politiques plutôt que l'algorithme d'itération valeurs. De l'aveux de son auteur, l'algorithme d'itération de politiques (PI) ne peut-être efficace que dans le cas infini ($N = \infty$).

Algorithme 3 Algorithme d'itération de politiques pour la résolution des MDPs.

1: **procedure** PI(S, A, T, R, λ)
2: Soient une règle de décisions quelconque d_0, $\varepsilon > 0$, et $\tau = 0$.
3: **répéter**
4: **Évaluation de la règle de décision :** calcul de v^{d_τ} en résolvant,

$$(\mathbf{1} - \lambda T_{d_\tau}) \cdot v \quad \leftarrow \quad R_{d_\tau} \tag{2.30}$$

5: **Amélioration de la règle de décision :** choix de $d_{\tau+1}$ satisfaisant,

$$d_{\tau+1} \quad \leftarrow \quad \underset{d \in D^{\mathbf{MD}}}{\arg\max} \left\{ R_d + \lambda T_d \cdot v^{d_\tau} \right\} \tag{2.31}$$

définir $d_{\tau+1} \leftarrow d_\tau$ si possible.
6: Poser $\tau \leftarrow (\tau + 1)$.
7: **jusqu'à** $d_{\tau+1} \leftarrow d_\tau$
8: $d^* \leftarrow d_\tau$.
9: **fin procedure**

L'algorithme d'itération de politiques décrit Algorithme 3 se décompose en deux sous routines : d'une part, l'évaluation de la fonction de valeurs associée à la règle de décisions courantes ; d'autre part, la modification de la règle de décisions courantes. L'évaluation de la règle de décisions courantes d est donnée par :

$$v^d(s) \quad = \quad R_d(s) + \lambda \sum_{s'} T_d(s'|s) v^d(s') \tag{2.32}$$

ou sous forme matricielle,

$$v^d \quad = \quad R_d + \lambda T_d v^d. \tag{2.33}$$

Il s'ensuit que ?d satisfait le système d'équations,

$$(\mathbf{1} - \lambda T_d) v \quad = \quad R_d \tag{2.34}$$

Le calcul de la fonction de valeurs d'une règle de décisions étant assez coûteux, on se contente parfois d'une simple approximation. Pour y parvenir, il suffit de mettre à jour successivement v^d comme indiqué équation 2.32. Ce processus se reproduit jusqu'à ce que deux fonctions de valeurs issues de mises à jour successives soient assez proches. On

pourrait par exemple utiliser le critère d'arrêt de l'algorithme d'itération de valeurs. Lors de l'amélioration de la règle de décisions courantes d_τ, il suffit de choisir la règle $d_{\tau+1}$ comme une règle de décisions telle que :

$$R_{d_{\tau+1}} + \lambda T_{d_{\tau+1}} v^{d_\tau} \geqslant R_{d_\tau} + \lambda T_{d_\tau} v^{d_\tau} \tag{2.35}$$

avec une inégalité stricte dans au moins une des composantes. L'algorithme d'itération de politiques est garanti de converger vers une politique optimale.

$$v^{d_0} \leqslant v^{d_1} \leqslant \cdots \leqslant v^{d_{N-1}} \tag{2.36}$$

En pratique, on observe que le nombre d'itérations utiles est considérablement réduit en comparaison de celles requises par l'algorithme d'itération de valeurs. Cependant l'étape d'évaluation de la règle de décisions courantes est si prohibitive qu'en somme le coût total n'est pas garanti d'être en faveur de l'algorithme d'itération de politiques.

Exemple 1. *Considérons le problème de la grille* 4×4 *illustré à la Figure 2.3.*

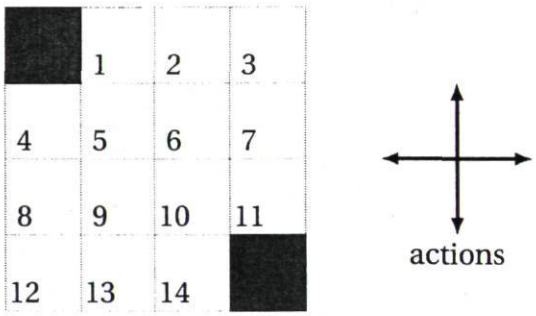


FIGURE 2.2 – Problème de la grille 4×4.

L'ensemble des états non terminaux est donné par l'ensemble $S \equiv \{1, \cdots, 14\}$. *Il existe 4 actions* $A \equiv \{haut, bas, gauche, droite\}$. *Ces dernières causent la transition déterministe (pour simplifier) d'un état vers un autre. Les actions qui mèneraient à la sortie de l'agent hors de la grille, laissent l'agent dans l'état courant. Ainsi, nous avons* $T(6|5, droite) = 1$ *tandis que* $T(10|5, droite) = 0$, *et* $T(7|7, droite) = 1$. *Il s'agit ici d'un problème épisodique à récompenses décomptées. La récompense est de ?1 pour toute transition. Et cela jusqu'à ce que l'agent atteigne l'état final. L'état final est représenté par la cellule en rouge.*

Illustrons les différences entre les algorithmes d'itération de valeurs et de politiques, Figure 2.3. La colonne de gauche décrit une séquence d'approximations successives de la fonction de valeurs v_τ, pour tout horizon $\tau = 0, 1, 2, \cdots, N$. La colonne de droite illustre une séquence de politiques gourmandes. Il s'agit de politiques extraites en sélectionnant

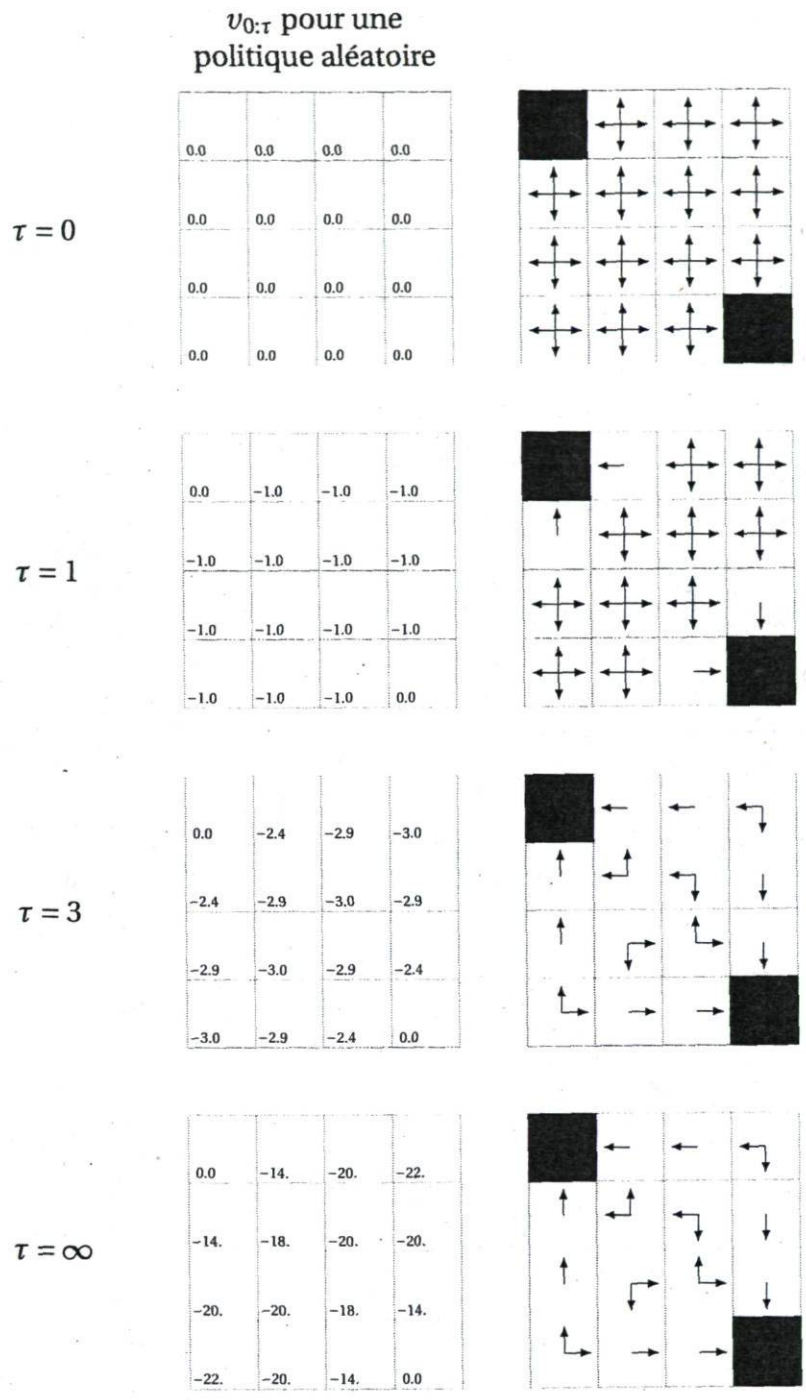


FIGURE 2.3 – Convergence de l'algorithme d'itération de politiques.

l'action qui maximise la fonction de valeurs pour un état donné. Les flèches indiquent les actions maximisant la fonction de valeurs v_τ à l'horizon $\tau = 0, 1, 2, \cdots, N$. Le lecteur observera qu'à l'itération $\tau = 3$, l'algorithme d'itération de politiques a déjà convergé vers une politique optimale, tandis que l'algorithme d'itération de valeurs doit procéder à des itéra-

tions additionnelles. Cette convergence de l'algorithme d'itération de politiques peut être liée : soit à la structure quasi acyclique du problème ; soit à la nature quasi déterministe de la fonction de transitions ; etc. La convergence vers une politique optimale de l'algorithme d'itération de politiques n'est bien sûr pas toujours aussi rapide. Mais dans tous les cas elle requiert toujours un nombre d'itérations moindre en comparaison à l'algorithme d'itération de valeurs. □

2.2.2 Programmation dynamique asynchrone

Le principal inconvénient des algorithmes de programmation dynamique présentés précédemment réside dans le fait que les opérations de mises à jour s'appliquent sur l'ensemble des états du système. Ainsi si l'ensemble S des états est très large, une seule mise à jour requiert des ressources considérables. Par exemple, l'ensemble des états du jeu Backgammon s'élève à plus de 10^{20} états. Même si nous pouvions mettre à jour un million d'états en une seconde, il nous faudrait plus d'un million d'années pour effectuer une seule mise à jour [Sutton and Barto, 1998]. La programmation dynamique asynchrone, en opposition à la programmation dynamique synchrone, correspond à des algorithmes de mises à jour partielles de la fonction de valeurs : seule une partie des états de la fonction de valeurs est mise à jour. Ces algorithmes mettent à jour les valeurs des états dans un ordre quelconque, en utilisant les valeurs les plus récentes des autres états comme base. Certains états peuvent voir leur valeur mise à jour plusieurs fois tandis que d'autres valeurs d'états ne seront mises à jour qu'une seule fois. Néanmoins, afin de garantir la convergence, tout algorithme de programmation dynamique asynchrone doit mettre à jour tous les états du système suffisamment souvent.

Algorithmes de mises-à-jour prioritisées

Les algorithmes prioritisés, tels que prioritized value iteration (PVI) [Wingate and Seppi, 2005] décrit Algorithme 4, se caractérisent par la mémorisation d'un certain nombre d'information supplémentaires. En particulier, ils mémorisent la fonction de valeurs $v^a : S \rightarrow \Re$ associée à chacune des actions $a \in A$. Cette dernière fonction de valeurs est encore appelée Q-valeurs.

Les premiers algorithmes de mises à jour prioritisées ont été développés simultanément et indépendamment par Moore and Atkeson [1993] et Peng and Williams [1993]. Il s'agit de deux exemples d'algorithmes de programmation dynamique asynchrones. L'idée est d'effectuer une propagation des valeurs des états buts vers les états initiaux. Comme il

Algorithme 4 Algorithme à mises à jour prioritisées

1: **procedure** PVI(S, A, T, R, λ)

2: Soient, $\varepsilon > 0$, $s = s_0$, $Q = \emptyset$

3: **répéter**

$$a^* \leftarrow \arg\max_{a \in A} \left\{ R(s,a) + \lambda \sum_{s'} T(s'|s,a) \max_{a' \in A} v^{a'}(s') \right\} \tag{2.37}$$

$$\mathbf{e}(s) \leftarrow \left| \left\{ R(s,a^*) + \lambda \sum_{s'} T(s'|s,a^*) \max_{a \in A} v^{a}(s') \right\} - v^{a^*}(s) \right| \tag{2.38}$$

4: Si $\mathbf{e}(s) > \varepsilon$, insérer (s, a^*) dans une pile Q de priorité $\mathbf{e}$.

5: Sélectionner (s, a) en-tête de la pile Q puis calculer $v^a(s)$ comme suit :

$$v^a(s) \leftarrow R(s,a) + \lambda \sum_{s'} T(s'|s,a) \max_{a' \in A} v^{a'}(s') \tag{2.39}$$

6: Pour tout état-action $(\bar{s}, \bar{a}) \in S \times A$ tel que $T(s|\bar{s}, \bar{a}) > 0$:

$$\mathbf{e}(\bar{s}) \leftarrow \left| \left\{ R(\bar{s},\bar{a}) + \lambda \sum_{s'} T(s'|\bar{s},\bar{a}) \max_{a \in A} v^{a}(s') \right\} - v^{\bar{a}}(\bar{s}) \right| \tag{2.40}$$

si $\mathbf{e}(\bar{s}) > \varepsilon$ insérer $(\bar{s}, \bar{a})$ dans la pile Q avec la priorité $\mathbf{e}$.

7: **jusqu'à** $Q \neq \emptyset$

8: Retourner la fonction de valeurs ε-optimale v^*_ε donné par :

$$v^*_\varepsilon(s) \leftarrow \max_{a \in A} v^a(s) \tag{2.41}$$

9: **fin procedure**

n'est pas toujours possible d'identifier les états buts des états initiaux, les auteurs utilisent les valeurs courantes des états pour effectuer leurs mises à jour. Imaginons que la fonction de valeurs soit invariante par application de l'opérateur de mise à jour. Puis, suite à une modification dans l'environnement, un état voit sa valeur modifiée. Il paraît évident que plusieurs autres états doivent également voir leur valeur modifiée. Mais seule les paires « états prédécesseurs et actions » menant à l'état dont la valeur a été modifiée, sont directement concernées. Lors de la mise à jour de ces paires d'états et d'actions, les valeurs associées à aux états prédécesseurs pourraient également être modifiées. Si cela s'avère être le cas, les paires « états prédécesseurs et actions » menant aux états dont les valeurs ont été précédemment modifiées, doivent être mises à jour, et ainsi de suite. De cette manière, il est possible d'effectuer des mises à jour par propagation arrière. Et ainsi, il est possible de réduire les mises à jour inutiles. D'autres algorithmes prioritisés, inspirés par cet algorithme, ont vu le jour par la suite [Andre et al., 1998, Wingate and Seppi, 2005].

Tous sont munis de cette idée d'ordonner les mises à jour des valeurs des états suivant une erreur entre deux mises à jour successives. L'erreur entre deux fonctions de valeurs successives est nommée l'erreur de Bellman (notée $\mathbf{e}(\cdot)$) et s'écrit :

$$\mathbf{e}(s) = \left| \left\{ R(s,a^*) + \lambda \sum_{s'} T(s'|s,a^*) \max_{a \in A} v^a(s') \right\} - v^{a^*}(s) \right| \tag{2.42}$$

où a^* dénote l'action optimale pour l'état s sachant une fonction de valeurs. Cependant, il n'est pas garantie qu'en ordonnant les mises à jour suivant l'erreur de Bellman, le gain dû à la réduction des mises à jour domine le coût occasionné par le maintient de ses erreurs pour les états prédécesseurs de l'état mis à jour.

Algorithme topologique d'itération de valeurs

La recherche d'ordres de mise à jour des états est reconnue comme un moyen utile pour accélérer la vitesse de convergence vers une fonction de valeurs optimales [Moore and Atkeson, 1993, Peng and Williams, 1993]. Cependant, cette recherche peut nécessiter des efforts non négligeables. Cela est essentiellement dû à la nature dynamique des ordres de mises à jour calculés dynamiquement à chaque mise à jour des algorithmes prioritisés tels que generalized prioritized value iteration [Moore and Atkeson, 1993, Peng and Williams, 1993]. Ainsi, bien que le nombre de mises à jour est réduit, dans la majorité des approches basées sur cette idée, les bénéfices ne se manifestent pas à travers le temps total de calcul de la fonction de valeurs optimales. En conséquence, d'autres approches ayant recours aux ordres stationnaires de mises à jour ont été proposées [Abbad and Boustique, 2003, Bonet and Geffner, 2003a].

C'est dans ce contexte que l'algorithme topologique d'itération de valeurs a vu le jour [Dai and Goldsmith, 2007]. Il s'agit d'un algorithme de recherche d'un ordre stationnaire de mises à jour de la fonction de valeurs afin de pallier aux inconvénients des approches prioritisées. L'idée des algorithmes topologiques est d'exploiter la dépendance causale entre les états. Considérons par exemple trois états s, s' et s''. Si s précède l'état s' et s' précède s'', alors la valeur $v(s)$ est influencée par toute modification de la valeur $v(s')$ et de même la valeur $v(s')$ est influencée par toute modification de la valeur $v(s'')$:

$$v(s) = \max_{a \in A} \left\{ R(s,a) + \lambda \sum_{s' \in \mu(s,a)} T(s'|s,a) v(s') \right\} \tag{2.43}$$

où $\mu(s,a) = \{s' | T(s'|s,a) > 0\}$. Suite à ces observations, il est légitime de procéder aux mises à jour des états s, s' et s'' suivant l'ordre s'' puis s' et enfin s. Malheureusement, dans les processus décisionnels de Markov, il est courant que l'état s'' précède l'état s' qui précède l'état s et inversement. On dit alors que ces états sont mutuellement dépendants. Un

MDP dont tout ou partie de ses états sont mutuellement dépendants, est dit cyclique. Le cas échéant on parle de MDP acyclique. Pour tout MDP, les auteurs proposent de grouper l'ensemble S_k des états mutuellement dépendants, pour tout $k = 0, 1, \cdots K$ où $K \leqslant |S|$. Pour y parvenir, on considère le graphe orienté dont les nœuds correspondent aux états du MDP et les arcs (s, si) reliant l'état s à l'état s' existent si et seulement si $\sum_{a \in A} T(s'|s, a) \geqslant 0$, pour toute paire d'états $(s, s') \in S^2$. Le MDP réduit dont les états, appelés aussi macro états, correspondent aux composantes fortement connexes $\{S_k\}_k$, est acyclique. L'ordre des mises à jour de la fonction de valeurs est donné par l'inverse du tri topologique du graphe acyclique. Les macro états S_k sont mis à jour virtuellement une seule fois suivant l'ordre inverse du tri topologique. La mise à jour d'un macro état S_k correspond à l'application de l'algorithme d'itération de valeurs jusqu'à convergence mais uniquement sur S_k, pour tout $k = 0, 1, \cdots, K$.

Malgré les performances de cet algorithme, il est utile de noter qu'il ne produit un ordre optimal que pour les MDPs acycliques. Le cas échéant, l'algorithme se réduit à l'application de l'algorithme d'itération de valeurs sur tout ou partie des états du MDP. Dès lors, se pose la question de la démarche à suivre lorsque tous les états sont mutuellement dépendants. En somme cet algorithme montrera de très bonnes performances si le MDP est acyclique. Autrement, il n'est pas meilleur que l'algorithme d'itération de valeurs. Outre l'exploitation de la dépendance causale, cet algorithme topologique exploite également l'analyse d'accessibilité afin d'élaguer les états inaccessibles d'un état initial donné. Le Chapitre 4 étant cette idée au cas général des processus décisionnels de Markov. Nous montrons comment les dépendances causales inter états peuvent être utilisées afin de réduire le taux de convergence des algorithmes de programmation dynamique.

Autres algorithmes

Les idées de recherche d'ordres de mise à jour et d'exploitation de l'analyse d'accessibilité ont également été proposés dans d'autres paradigmes algorithmiques. Que se soit, la recherche heuristique par chaînage avant (selon le terme anglo-saxon forward search) y compris RTDP [Barto et al., 1993], LRTDP [Bonet and Geffner, 2003b], LAO* [Hansen and Zilberstein, 2001] et HDP [Bonet and Geffner, 2003a] ; ou la recherche heuristique par chaînage arrière (selon le terme anglo-saxon backward search) telle que IPS [McMahan and Gordon, 2005]. Tous tentent de calculer de bonnes séquences d'états à mettre à jour tout en évitant les sous-espaces d'états inaccessibles. Les heuristiques de recherche par chaînage avant sont duales aux heuristiques de recherche par chaînage arrière. Les heuristiques de recherche par chaînage avant correspondent à la simulation de l'exécution $(s_0, a_0, s_1, a_1, \cdots, a_{N-1}, s_N)$ d'une règle de décisions d en partant d'un état initial s_0, telle que $a_\tau = d(s_\tau)$. La séquence d'états $(s_0, s_1, \cdots, s_N)$ ainsi construite est par la suite mise à

jour dans l'ordre inverse $v(s_N)$, $v(s_{N-1})$, $\cdots$, $v(s_0)$. La règle de décisions d est souvent initialisée par la règle de décisions optimales pour le MDP déterministe associé au MDP à résoudre [Bonet and Geffner, 2003b] ou par tout autre règle de décisions associée à une fonction de valeurs supérieures à celles de la fonction de valeurs optimales. Le cadre général des heuristiques de recherche par chaînage avant est décrit à la Figure 5. À noter que les performances de ces heuristiques sont intimement liées à la qualité de la fonction de valeurs initiales ou de la règle de décisions initiales. Plus la distance $\|v^{d_0} - v^{d^*}\|$ entre la règle de décisions initiales d_0 et la règle de décisions optimales d^* est petite plus la convergence est rapide et vice versa. Les résultats empiriques de LRTDP sont par exemple extrêmement impressionnants lorsque le MDP à résoudre est presque déterministe et que la règle de décisions utilisées est la règle de décisions du MDP déterministe extrait du MDP à résoudre.

Algorithme 5 Heuristique de recherche par chaînage avant.

procedure FS
  Soient $\bar{v}$, d et s_0. Posons $\tau \leftarrow 0$.
  MISES À JOUR À BASE DE TRAJECTOIRES(s_τ)
  si $\bar{v}$ n'a pas convergée **alors**
    Poser $\tau \leftarrow (\tau + 1)$.
    MISES À JOUR À BASE DE TRAJECTOIRES(s_τ).
  sinon
    Retourner d.
  fin si
fin procedure

Algorithme 6 Mises à jour à base de trajectoires.

procedure MISES À JOUR À BASE DE TRAJECTOIRES(s_τ)
  si s_τ n'est pas un état but **alors**
    Choisissons l'action $a_\tau \leftarrow d(s_\tau)$,
    Échantillonnons $s_{\tau+1}$ de $T(\cdot|s_\tau, a_\tau)$,
    MIS À JOUR À BASE DE TRAJECTOIRES($s_{\tau+1}$).
  fin si
  Mettre à jour $\bar{v}(s_\tau)$ et d.
fin procedure

2.2.3 Programmation linéaire

Cette section aborde la résolution des processus décisionnels de Markov complètement observables via la programmation linéaire (PL), alternative à la programmation dynamique. À ce jour, il y a pas d'évidence que la programmation linéaire est une méthode

efficace dans le cas général des MDPs à récompenses décomptées. Cependant, les recherches en PL pourrait changer cet état de fait.

Nous proposons un bref exposé de la formulation d'un programme linéaire pour la résolution des MDPs, car la PL offre une élégante théorie, permettant d'incorporer des contraintes de façon aisée. De plus, elle permet l'analyse de sensibilité et robustesse de la solution trouvée.

Formulation du problème

Si une fonction de valeurs $v \in V$ satisfait l'inégalité

$$v > r_d + \lambda \mathbb{P}_d \cdot v \tag{2.44}$$

pour toute règle de décisions $d \in D$, alors v est une borne supérieure sur la fonction de valeurs optimales v^*. Cette observation donne les bases de la forme primale du PL pour la résolution d'un MDP.

Programme primal

Programme linéaire primal

1. Minimiser ζ,

$$\zeta \leftarrow \sum_{s \in S} b(s) v(s) \tag{2.45}$$

avec $\sum_s b(s) \leftarrow 1$ et $b(s) \geqslant 0$, pour tout état $s \in S$, et tel que :

$$v(s) \geqslant R(s,a) + \lambda \sum_{s' \in S} T(s'|s,a) v(s') \tag{2.46}$$

pour tout état $s \in S$.

FIGURE 2.4 – Programme linéaire primal.

Bien que cette formulation soit valide, sa forme duale décrite à la Figure 2.5 offre un meilleur cadre algorithmique pour l'analyse et la compréhension du modèle. En observant le primal, on constate qu'il possède $|S|$ colonnes et $|S||A|$ lignes, tandis que le dual

Programme linéaire dual

1. Maximiser ζ,

$$\zeta = \sum_{s \in S} \sum_{a \in A} x(s,a) \cdot R(s,a) \tag{2.47}$$

tel que :

$$\sum_{a \in A} x(s',a) - \lambda \sum_{s \in S} \sum_{a \in A} T(s'|s,a) x(s,a) = b(s') \tag{2.48}$$

avec $x(s,a) \geqslant 0$ pour toute action $a \in A$ et tout état $s \in S$.

♣

FIGURE 2.5 – Programme linéaire dual.

possède $|S|$ lignes et $|S||A|$ colonnes. Par conséquent, il est préférable de résoudre le dual. De récents travaux sur la résolution des MDPs suggérèrent que la programmation linéaire peut dominer la programmation dynamique notamment lorsqu'une partie des attributs des états est définie sur des espaces continus [Altman, 1998, Dolgov and Durfee, 2006b].

Lien entre les solutions du dual et celles du primal

Rappelons que $x(s,a)$ correspond à une solution réalisable du programme dual en Figure 2.5 si et seulement si il est non négatif et satisfait l'équation (2.48) pour tout état $s \in S$.

1. Pour toute règle de décisions $d \in D^{\mathbf{MA}}$, pour tout $s \in S$, et pour toute $a \in A$, si $x_d(s,a)$ est donné par :

$$x_d(s,a) \equiv \sum_{s \in S} b(s) \sum_{\tau=1}^{\infty} \lambda^{\tau-1} \mathbb{P}_d(s_\tau = s, a_\tau = a | s_1 = s). \tag{2.49}$$

   alors $x_d(s,a)$ est une solution réalisable du programme dual.

2. Si nous disposons d'une solution réalisable du programme dual $x(s,a)$, alors nous pouvons définir une règle de décisions stationnaires et aléatoires d_x comme suit :

$$\mathbb{P}\{d_x(s) = a\} = \frac{x(s,a)}{\sum_{\bar{a} \in A} x(s,\bar{a})} \tag{2.50}$$

   alors $x_{d_x}(s,a)$ défini par l'équation (2.49) est une solution réalisable de programme dual et $x_{d_x}(s,a) = x(s,a)$ pour tout $a \in A$ et tout $s \in S$.

représentent le produit total des probabilités décomptées partant de la distribution initial $b \in \mathscr{P}(S)$, déterminant la probabilité que l'agent soit dans l'état s et choisisse d'exécuter l'action a. En multipliant cette valeur par $R(s,a)$ et en sommant sur l'ensemble des pairs états et actions, nous parvenons à calculer le total espéré des récompenses cumulées et décomptées à long terme pour la règle de décision d_x . C'est à dire,

$$\sum_{s \in S} b(s) \cdot v^{d_x}(s) \quad = \quad \sum_{s \in S} \sum_{a \in A} x(s,a) \cdot R(s,a). \tag{2.51}$$

En résumé, le premier point identifie les solutions réalisables à travers les règles de décisions stationnaires et aléatoires. Le second point, montre comment générer une politique stationnaire et aléatoire à partir d'une solution réalisable, et établit ainsi une équivalence entre la quantité x_{d_x} et x. C'est à dire une relation qui associe à toute règle de décisions stationnaires et aléatoires, une solution.

2.3 Les MDPs partiellement observables

Le modèle de contrôle centralisé des processus décisionnels de Markov complètement observables (MDPs) est incapable de formaliser les interactions d'un robot d'exploration dont les capteurs sont partiellement fiables. Plus précisément, la politique d'un tel robot ne peut se réduire en une fonction qui à tout état associe une action car l'état n'est pas directement perçu par le robot. Le robot doit alors se souvenir d'informations antérieures afin de construire « une croyance » sur son état courant. La totalité des informations dont dispose un agent correspond à l'ensemble des historiques possibles d'actions et d'observations. En effet, lors de la simulation des interactions entre un tel robot et son environnement, suite à l'action a_τ sélectionnée, l'environnement renvoie une information sur l'état courant – l'observation $\omega_{\tau+1}$ – et une récompense $r_{\tau+1}$. Ce processus est modélisé par le modèle de contrôle centralisé des processus décisionnels de Markov partiellement observables (POMDPs). Outre cette capacité à formaliser les problèmes de navigation pour les robots dont les capteurs ont une fiabilité partielle, le modèle des POMDPs est un formidable outil de modélisation d'un large éventail d'applications réelles, y compris des applications d'aide aux patients à mobilité réduite [Pineau et al., 2003a] ou encore d'aide à l'extraction des préférences d'un utilisateur [Boutilier et al., 2009].

La Figure 2.6 illustre le cycle de contrôle d'un processus décisionnel de Markov partiellement observable. Comme dans le cas de MDPs, nous distinguons trois principales composantes. Les deux premières composantes sont similaires à celles rencontrées lors du contrôle d'un MDP. La troisième composante « ce qu'on infère » illustre la « croyance » de l'agent central quant à l'état courant du système. Comme les POMDPs ne supposent

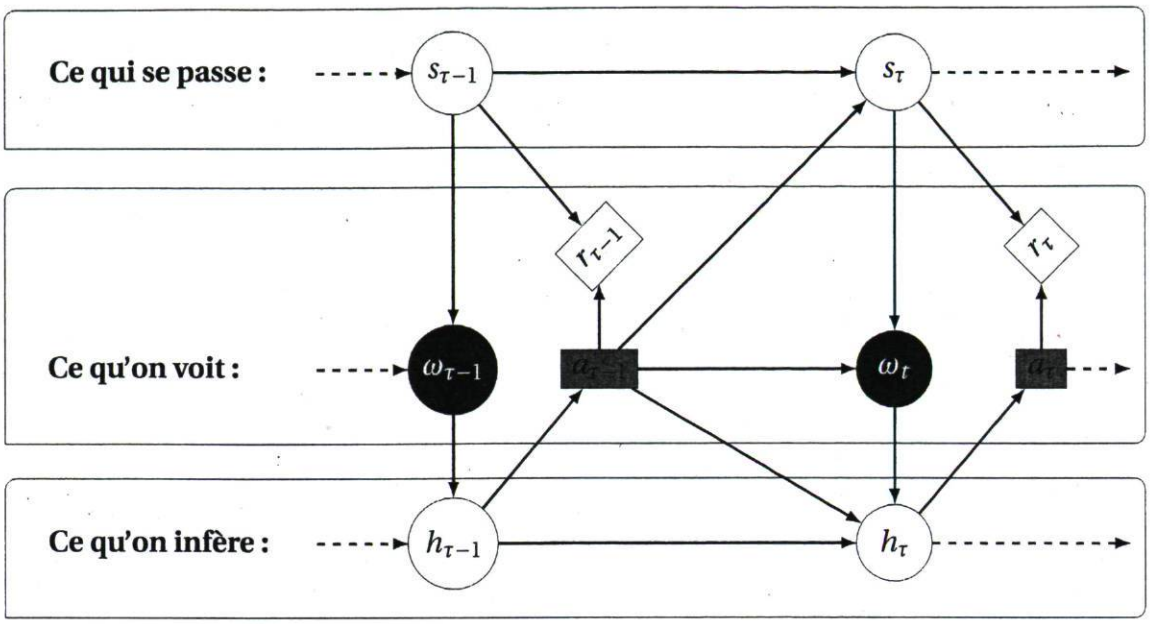


FIGURE 2.6 – Cycle de contrôle d'un POMDP – adpaté de Pineau [2004].

pas l'observabilité complète à la différence des MDPs, l'état courant s_τ du système dépend de l'historique $h_\tau \equiv (s_0, a_0, \cdots, a_{\tau-1}, \omega_\tau)$ entier. Plus formellement, l'état courant s_τ du système est donné par la probabilité $\mathbb{P}(s_\tau|h_\tau)$. La distribution de probabilités $\mathbb{P}(s|h)$ sur l'ensemble des états $s \in S$ pour tout historique h, est appelé la croyance de l'agent central.

2.3.1 Cadre formel

Définition 2. *Un processus décisionnel de Markov partiellement observable est un* 6*-uplet* (S, A, Ω, T, R, O) *donné par :*

- Les observations. $\Omega = \{\omega_0, \omega_1, \cdots, \omega_N\}$ *dénote l'ensemble fini des observations* ω_τ *possibles, pour tout horizon* $\tau = 0, 1, \cdots, N$.
- La fonction d'observations. O: $S \times A \to \mathscr{P}(\Omega)$ *est la distribution de probabilités*

$$O(s, a, \omega) = \mathbb{P}(\omega_{\tau+1} = \omega | s_\tau = s, a_\tau = a) \quad \forall \tau = 0, 1, \cdots, N-1 \tag{2.52}$$

  définissant la probabilité que l'agent perçoive l'observation ω *après avoir exécuté l'action* a *dans l'état* s. *Il est utile de noter que* $O(s, a, \omega)$ *étant une probabilité conditionnelle,* $\sum_\omega O(s, a, \omega) = 1$ *pour toute paire d'état et d'action* (s, a).
- (S, A, T, R) *est un* 4*-uplet identique à celui défini dans le cadre des processus décisionnels de Markov complètement observables.*

2.3.2 Prise de décisions optimales

De manières similaires aux MDPs, dans le modèle des POMDPs, l'objectif est de construire une politique aussi proche que possible de l'optimum afin de guider l'agent dans la satisfaction de sa tâche. Nous choisissons une fois encore le critère relatif au total espéré des récompenses cumulées dans le cas fini et le critère relatif au total espéré des récompenses cumulées et décomptées dans le cas infini.

Historique

Afin de différencier les politiques, il est utile de pouvoir les évaluer. Hors cela n'est possible que si l'on parvient à lier l'information collectée lors des interactions de l'agent avec l'environnement aux états du système, et par conséquent aux récompenses à maximiser. Par chance, à chaque réalisation d'un historique $h_N = (s_0, a_0, \omega_1, \cdots, a_{N-1}, \omega_N)$ correspond une séquence de récompenses $\{r_0(h_0, a_0), r_1(h_1, a_1), \cdots, r_N(h_N)\}$. Notons qu'un historique $h_{\tau+1}$ est donné par $h_{\tau+1} \equiv (h_\tau, \omega_{\tau+1})$, pour tout horizon $\tau = 0, 1, \cdots, N-1$ et par $h_0 = (s_0)$ à l'horizon $\tau = 0$. La récompense immédiate pour tout horizon $\tau = 0, 1, \cdots, N-1$, est alors donnée par :

$$r_{\tau+1}(h_{\tau+1}, a_{\tau+1}) = \mathbb{P}(s_{\tau+1}|h_\tau, a_\tau, \omega_{\tau+1}) \cdot R(s_{\tau+1}, a_{\tau+1}) \tag{2.53}$$

Le lecteur notera que le caractère stationnaire de la fonction R n'est pas important dans l'équation (2.53). En effet, la récompense $r_{\tau+1}(h_{\tau+1}, a_{\tau+1})$ est non stationnaire quand bien même R serait stationnaire. Pour cette raison, dans le cadre des processus décisionnels de Markov partiellement observables, la nature stationnaire des fonctions R et T peut-être relaxée. Néanmoins, sans perte de généralité, nous considérons uniquement le cas où R et T sont stationnaires.

Dans le cadre du contrôle centralisé des processus décisionnels de Markov partiellement observables, il est démontré que la statistique suffisante pour un historique h_τ est résumée par une distribution de probabilités sur l'ensemble des états du système. Cette distribution de probabilités est appelée l'état de croyance (ou croyance) [Aström, 1965].

Elle est donnée par l'équation suivante : pour tout horizon $\tau = 1, 2, \cdots, N$,

$$
\begin{aligned}
b_\tau(s') &= \mathbb{P}(s_\tau = s' | h_\tau) && (2.54)\\
&= \mathbb{P}(s_\tau = s' | h_{\tau-1}, a_{\tau-1}, \omega_\tau) && (2.55)\\
&= \frac{\mathbb{P}(h_{\tau-1}, a_{\tau-1}, \omega_\tau), s_\tau = s')}{\mathbb{P}(h_{\tau-1}, a_{\tau-1}, \omega_\tau)} && (2.56)\\
&= \frac{\mathbb{P}(s_0, a_0, \omega_1, \cdots, \omega_{\tau-1}, a_{\tau-1}, \omega_\tau, s_\tau = s')}{\sum_{s_\tau} \mathbb{P}(s_\tau | h_{\tau-1}, a_{\tau-1}, \omega_\tau)} && (2.57)\\
&= \frac{\sum_{s_{\tau-1}} O(s_{\tau-1}, a_{\tau-1}, \omega_\tau) T(s_{\tau-1}, a_{\tau-1}, s') b_{\tau-1}(s_{\tau-1})}{\sum_{s_\tau} \mathbb{P}(s_\tau | h_{\tau-1}, a_{\tau-1}, \omega_\tau)} && (2.58)\\
&= \frac{\sum_{s_{\tau-1}} O(s_{\tau-1}, a_{\tau-1}, \omega_\tau) T(s_{\tau-1}, a_{\tau-1}, s') b_{\tau-1}(s_{\tau-1})}{\sum_{s_\tau} \sum_{s_{\tau-1}} O(s_{\tau-1}, a_{\tau-1}, \omega_\tau) T(s_{\tau-1}, a_{\tau-1}, s_\tau) b_{\tau-1}(s_{\tau-1})} && (2.59)
\end{aligned}
$$

À l'horizon $\tau = 0$, le décideur dispose d'un état de croyance initial $b_0(s) = \mathbb{P}(s_0 = s)$, pour tout état $s \in S$. Soit, pour tout horizon $\tau = 1, 2, \cdots, N$ et pour tout état $s' \in S$,

$$
b_\tau(s') = \frac{\sum_s O(s, a, \omega) T(s, a, s') b_{\tau-1}(s)}{\sum_{s'} \sum_s O(s, a, \omega) T(s, a, s') b_{\tau-1}(s)}, \qquad (2.60)
$$

où le dénominateur est également nommé facteur de normalisation. Muni de l'état de croyance, il est facile d'estimer la récompense pour tout horizon $\tau = 0, 1, \cdots, N-1$,

$$
\begin{aligned}
r_\tau(h_\tau, a) &= R(h_\tau, a) && (2.61)\\
&= \sum_s b_\tau(s) \cdot R(s, a) && (2.62)
\end{aligned}
$$

et pour l'horizon $\tau = N$, $r_N(h_N) = \sum_s b_N(s) \cdot R(s)$. La première équation (2.61) s'explique par le fait que R soit stationnaire. La seconde est déduite d'après la définition de l'état de croyance. Ainsi, l'état de croyance est suffisant pour conditionner la sélection des actions, donc la construction de la politique optimale. Seulement contrairement aux états du système, l'ensemble des états de croyance est un continuum décrivant l'ensemble des distributions de probabilités sur l'ensemble fini S des états du système. Ainsi lorsque le nombre d'états du système croît, maintenir et mémoriser l'ensemble des états de croyance nécessite d'énormes ressources en temps et en mémoire. D'où l'intérêt grandissant pour des méthodes permettant de déterminer les états de croyance utiles pour une planification optimale ou presqu'optimale [Pineau, 2004, Pineau et al., 2003b, Smith and Simmons, 2004]. Bien que le calcul des états de croyance sur lesquels sera effectué la sélection des actions est un facteur crucial comme nous le verrons plus bas, la complexité de calcul de la politique optimale est la principale raison de la difficulté de résolution des POMDPs. Nous distinguons pour ce faire deux cas, le cas à horizon fini $N < \infty$ et le cas à horizon infini $N = \infty$.

2.3.3 Cas à horizon fini

Tout comme dans le cas des processus décisionnels de Markov complètement observables, sachant un POMDP le décideur central doit construire une politique qui guidera le choix des actions de l'agent. De façon similaire aux MDPs, nous avons choisi de différencier les politiques selon le critère du total espéré des récompenses cumulées à long terme. Mais contrairement aux MDPs, l'agent ne perçoit pas l'état dans lequel il se trouve à chaque instant de décision. Dès lors, les actions sont conditionnées par l'historique ou l'état de croyance correspondant.

Représentations des politiques

À défaut de l'hypothèse d'observabilité complète du système, la classe des politiques d'un POMDP est restreinte aux politiques non Markoviennes et aléatoires $\Pi^{\mathbf{HA}}$. En d'autres termes, une politique π est donnée par une séquence de règles de décisions $(d_0, d_1, \cdots, d_{N-1})$, une pour chaque instant $\tau = 0, 1, \cdots, N-1$. De plus, chaque règle de décisions d_τ : $H_{0:\tau} \rightarrow \mathscr{P}(A)$ est une application qui à tout historique $h_\tau \in H_{0:\tau}$ associe une distribution de probabilités sur les actions $a \in A$, pour tout $\tau = 0, 1, \cdots, N-1$. Cependant, on démontre qu'il existe toujours une politique non Markovienne et déterministe $\pi^* \in \Pi^{\mathbf{HD}}$ dont le total espéré des récompenses cumulées est au moins aussi élevé que celui de tout autre politique non Markovienne et aléatoire [Putterman, 1994]. Pour cette raison, il est possible de restreindre la recherche des politiques optimales dans la classe des politiques et règles de décisions non Markoviennes et déterministes $\Pi^{\mathbf{HD}}$ et $D^{\mathbf{HD}}$ respectivement. La recherche d'une telle politique requiert des ressources en temps et en mémoire considérables. En effet, à chaque horizon τ, il faudra déterminer une règle de décision d_τ définie sur un ensemble d'historiques $H_{0:\tau}$ potentiellement infini.

Lorsque le décideur central est muni d'une information sur l'état initial du système b_0 associé à un historique initial h_0, la classe des politiques recherchées peut-être réduite. En effet, seules les croyances accessibles de la croyance initiale b_0 requièrent le calcul d'une action. En d'autres termes, la règle de décisions d_0 ne doit être définie que pour l'historique initiale h_0 ; la règle de décisions d_1 ne doit être définie que pour les croyances b_1 extraite de toute séquence (h_0, a_0, ω_1) ; la règle de décisions d_τ ne doit être définie que pour les croyances b_τ extraites des séquences $(h_{\tau-1}, a_{\tau-1}, \omega_\tau)$; et ainsi de suite jusqu'à l'horizon $N-1$. Une telle politique équivaut à un arbre de décisions où les nœuds sont étiquetés par des actions et les arcs par des observations, comme illustré à la Figure 2.7.

Définition 3. *Un arbre de décisions noté $\delta_{\tau:N}$ correspond à un arbre de profondeur $(N-\tau)$, pour tout horizon $\tau = 0, 1, \cdots, N-1$. La racine de cet arbre prescrit l'action initiale à exécuter*

pour un historique h_0 donné. Par la suite, selon l'observation perçue par l'agent, un arc de l'arbre étiqueté par cette observation pointe sur un sous-arbre, la racine de ce dernier est l'action suivante à exécuter, et ainsi de suite. Cet arbre de décisions définit la politique pour tout historique de longueur $\ell = 0, 1, \cdots, \tau$.

La différence entre une politique définie sans connaissance de l'historique initial h_0 et celle construite avec cette information est de taille. En effet, l'espace de définition de règles de décisions de la première est gigantesque, tandis que l'espace des politiques définies sous forme d'arbres de décisions est réduit aux nombre de croyances accessibles. Néanmoins l'espace des politiques sous forme d'arbres de décisions croît de façon exponentielle avec le nombre d'observations et l'horizon de planification comme nous le verrons par la suite. Notons enfin qu'il est possible de déterminer la politique optimale dans le cas où l'on ne dispose d'aucune information initiale. Pour cela, il suffit de conserver pour chaque croyance initial possible $b_0 \in \mathscr{P}(S)$ un arbre de décisions.

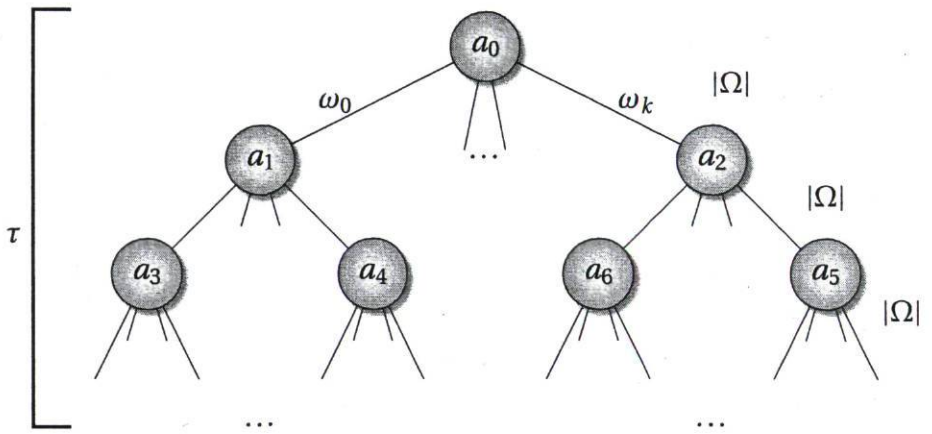


FIGURE 2.7 – Arbre de décisions de profondeur τ.

Critères d'optimisation

Afin d'identifier la politique optimale parmi l'ensemble de toutes les politiques, il est nécessaire de les différencier les unes des autres. La section suivante détaille les différents critères de différenciation des politiques sous forme d'arbres de décisions. De façon similaire au cas des MDPs, on associe à chaque politique π un fonction de valeurs v^π donnant l'espérance mathématique des récompenses cumulées à long terme si l'agent suit cette politique. La fonction de valeurs v^π est donnée par :

$$v^\pi_{0:N}(b_0) = \mathbb{E}_\pi \left\{ \sum_{\tau=0}^{N-1} r_\tau(b_\tau, a_\tau) + r_N(h_N) \mid d_\tau(b_\tau) = a_\tau \right\} \tag{2.63}$$

où $b_\tau(s) = \mathbb{P}(s_\tau = s \mid b_{\tau-1}, a_{\tau-1}, \omega_\tau)$ correspond à l'état de croyance associé à l'historique h_τ. Déterminer la politique optimale, c'est chercher une politique $\pi^* \in \Pi^{\mathbf{HD}}$ telle que :

$$v^{\pi^*}_{0:N}(b_0) \geqslant v^{\pi}_{0:N}(b_0), \quad \forall b_0 \in \mathscr{P}(S) \tag{2.64}$$

pour toute politique $\pi \in \Pi^{\mathbf{HD}}$. Lorsqu'une telle politique n'existe pas ou alors que les ressources disponibles ne permettent pas de la construire, on a recours à la construction d'une politique ε-optimale notée π^*_ε et telle que :

$$v^{\pi^*_\varepsilon}_{0:N}(b_0) + \varepsilon \geqslant v^{\pi}_{0:N}(b_0), \quad \forall b_0 \in \mathscr{P}(S) \tag{2.65}$$

pour toute politique $\pi \in \Pi^{\mathbf{HD}}$. La fonction de valeurs optimales $v^*_{0:N}$ d'un problème décisionnel de Markov partiellement observable satisfait :

$$v^*_{0:N}(b_0) = \max_{\pi \in \Pi^{\mathbf{HD}}} v^{\pi}_{0:N}(b_0), \quad \forall b_0 \in \mathscr{P}(S). \tag{2.66}$$

Ainsi, la fonction de valeurs de la politique optimale satisfait :

$$v^*_{0:N}(b_0) = v^{\pi^*}_{0:N}(b_0), \quad \forall b_0 \in \mathscr{P}(S). \tag{2.67}$$

et la fonction de valeurs de la politique ε-optimale satisfait :

$$v^{\pi^*_\varepsilon}_{0:N}(b_0) + \varepsilon \geqslant v^*_{0:N}(b_0), \quad \forall b_0 \in \mathscr{P}(S). \tag{2.68}$$

Selon Sondik [1978], on peut ainsi voir un POMDP comme un cas particulier de MDP où l'état est une distribution de probabilités sur les états sous-jacents du système. Dans ce contexte, le maximum sur l'ensemble des politiques non Markoviennes et déterministes est donné par :

$$v^*_{\tau:N}(b_\tau) = \max_{\pi \in \Pi^{\mathbf{HD}}} v^{\pi}_{\tau:N}(b_\tau), \quad \forall b_\tau \in \mathscr{P}(S), \forall \tau = 0, 1, \cdots, N. \tag{2.69}$$

Les équations d'optimalité peuvent alors se décliner comme suit : $\forall b \in \mathscr{P}(S)$,

$$v^*_{\tau:N}(b) = \max_{a \in A} \left\{ R(b,a) + \sum_{\omega \in \Omega} \mathbb{P}(b' | b, a) v^*_{\tau+1:N}(b') \right\}, \tag{2.70}$$

où $b' = \mathbb{P}(b, a, \omega)$. La fonction de valeurs $v^*_{\tau:N}$ est la fonction de valeurs optimales pour tout historique allant de l'horizon τ à l'horizon N. On peut extraire la règle de décisions optimales d^*_τ correspond à la fonction de valeurs optimales $v^*_{\tau:N}$ comme suit : $\forall b \in \mathscr{P}(S)$,

$$d^*_\tau(b) = \operatorname*{argmax}_{a \in A} \left\{ R(b,a) + \sum_{\omega \in \Omega} \mathbb{P}(b' | b, a) v^*_{\tau+1:N}(b') \right\} \tag{2.71}$$

La mise à jour d'une telle fonction de valeurs est très laborieuse car l'espace des états de croyance est le continuum $\mathscr{P}(S)$. Dans ce contexte, Sondik [1978] a montré que les fonctions de valeurs $v_{\tau:N}$ peuvent être représentées comme un ensemble de fonctions de valeurs : $\Lambda_\tau = \{v^\delta_{\tau:N}\}$, pour tout $\tau = 0, 1, \cdots, N-1$. Chacune des fonctions de valeurs $v^\delta_{\tau:N}$ représente alors la fonction de valeurs d'un arbre de décisions δ. L'ensemble Λ_τ correspond plus

précisément à un ensemble d'hyperplans $v^{\delta}_{\tau:N}$ de dimension $|S|$, un pour chaque arbre de décisions $\delta_{\tau:N}$. Il est alors possible de déterminer la valeur d'une croyance quelconque, comme suit :

$$v_{\tau:N}(b) = \max_{v^{\delta}_{\tau:N} \in \Lambda_{\tau}} v^{\delta}_{\tau:N}(b) \tag{2.72}$$

Cela est notamment possible car la fonction de valeurs $v^{*}_{\tau:N}$ est prouvée convexe et linéaire par morceaux [Sondik, 1978], pour tout horizon $\tau = 0, 1, \cdots, N$. Ainsi, nous associons à chaque fonction de valeurs $v^{\delta}_{\tau:N} \in \Lambda_{\tau}$ une action correspondant à l'action en racine de l'arbre de décisions $\delta_{\tau:N}$. Il est facile de démontrer que l'ensemble Λ_{τ} est borné car le nombre d'arbres de décisions possibles pour chaque horizon τ est fini.

Pour tout arbre de décisions δ, le total espéré des récompenses cumulées à long terme en suivant un tel arbre de décisions s'écrit :

$$v^{\delta}(s) \quad = \quad R(s, \alpha(\delta)) + \lambda \sum_{s' \in S} \sum_{\omega \in \Omega} T(s'|s, \alpha(\delta)) O(\omega|s, \alpha(\delta)) \cdot v^{\mu(\delta,\omega)}(s') \tag{2.73}$$

où $\alpha(\delta)$ correspond à l'action en racine de l'arbre δ ; $\mu(\delta, \omega)$ dénote le sous arbre de δ accessible de l'arc étiqueté par l'observation ω. La fonction de valeurs v^{δ} correspond à l'hyperplan associé à l'arbre de décisions δ. Un arbre de décisions δ est dit dominé s'il n'existe aucun état de croyance $b \in \mathscr{P}(S)$ pour lequel la fonction de valeurs v^{δ} est maximale. Plus formellement, si pour tout état de croyance $b \in \mathscr{P}(S)$, il existe un tout autre arbre de décisions $\hat{\delta}$, tel que :

$$\sum_{s \in S} v^{\delta}(s) \cdot b(s) \leqslant \sum_{s \in S} v^{\hat{\delta}}(s) \cdot b(s) \tag{2.74}$$

alors δ est un arbre de décisions dominés. On démontre plus généralement que toute politique π est une politique sous optimale si et seulement si tout ou partie de l'ensemble d'arbres de décisions associés est dominé. Le programme linéaire Algorithme 7, permet d'éliminer les arbres de décisions dominés.

L'ensemble des résultats énoncé ici s'étend par passage à la limite infinie, lorsque l'horizon de planification est infini $N = \infty$.

2.3.4 Cas à horizon infini

Dans cette section, nous nous intéressons au cas du contrôle centralisé des processus décisionnels de Markov partiellement observables à horizon infini. Nous y discutons de la représentation des politiques et des critères d'optimisation. En outre, nous soulignons les principales différences ou similitudes avec le cas à horizon fini.

Algorithme 7 Algorithme d'élagage des arbres de décisions dominés (POMDP).

procedure ÉLAGAGE($\mathcal{Q}$)
  pour tout $\delta \in \mathcal{Q}$ **faire**
    Maximisons ε, sachant que $\forall \hat{\delta} \in \mathcal{Q} \setminus \{\delta\}$,

$$v^{\hat{\delta}}(b) + \varepsilon \leqslant v^{\delta}(b) \tag{2.75}$$

    avec $\sum_s b(s) \leftarrow 1$ et $b(s) \geqslant 0$, pour tout état $s \in S$.
    si $\varepsilon \leqslant 0$ **alors**
      Éliminer δ de $\mathcal{Q}$.
    fin si
  fin pour
  Retourner $\mathcal{Q}$.
fin procedure

Représentations des politiques

Nous avons vu qu'il était possible de représenter une politique par un ensemble fini d'arbres de décisions dans le cas fini. Malheureusement, lorsque le nombre de décisions à prendre est potentiellement infini, les arbres de décisions considérés doivent alors être de profondeurs infinies. Il est possible, en utilisant un facteur de décompte, de déterminer une profondeur d'arbres de décisions au delà de laquelle les gains escomptés sont négligeables. Ainsi, nous pouvons borner la taille des arbres de décisions recherchées. Néanmoins, cela ne suffit pas à valider l'encodage sous forme d'arbres de décisions. En effet, non seulement cette représentation permet d'obtenir une politique approximative, mais en plus la profondeur de ces arbres de décisions pourrait être beaucoup trop élevée. Nous devons alors adopter une nouvelle représentation compacte pour toute politique à horizon infini. Hansen [1997] a proposé de représenter une politique π comme un automate d'états fini.

Définition 4. *Un automate d'états fini δ est donné par un tuple (X, α, μ, x_0), où :*

1. *X dénote un ensemble fini d'états possibles ;*
2. *$\alpha : X \to A$ définit une fonction qui associe à tout état $x \in X$ une action $a \in A$;*
3. *$\mu : X \times \Omega \to X$ décrit la fonction de transitions d'un état à un autre, selon l'observation perçue ;*
4. *et x_0 est l'état initial.*

L'exemple Figure 2.3.4 illustre une telle représentation de politiques. On constate que cette représentation de la politique permet d'extraire une séquence infinie d'actions. Par

exemple, pour une séquence d'observations $\langle \omega_1, \omega_1, \omega_2 \rangle$, l'automate produit l'historique $\langle \alpha(x_0), \omega_1, \alpha(x_1), \omega_1, \alpha(x_0), \omega_2 \rangle$, et l'action associée à cette historique est $\alpha(x_2)$.

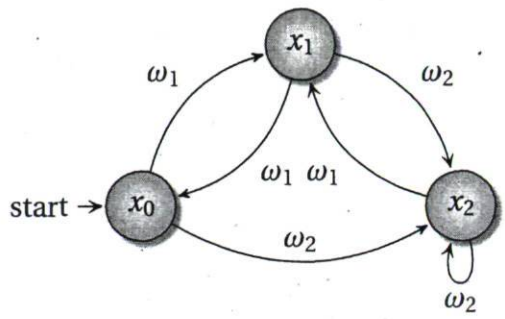


FIGURE 2.8 – Exemple de politique sous forme d'un automate.

Cette représentation est différente de celle des arbres de décisions. Par exemple, lorsqu'on ne dispose pas d'information initiale sur le système, il faut conserver plusieurs arbres de décisions afin de construire une politique optimale. Or, en utilisant les automates, un seul automate suffit à représenter une politique optimale, si elle existe. En effet, il est possible d'extraire autant d'automates différents d'un seul automate que de nœuds dans cet automate. En particulier, en changeant l'état initial d'un automate, ce dernier est considérablement modifié alors même que ni les nœuds ni les transitions de l'automate ne sont modifiés. Ces différents automates sont optimaux pour certaines croyances. Comme le nombre de croyances est potentiellement infini, il faut alors un nombre potentiellement infini de nœuds dans l'automate afin de garantir l'optimalité. Or l'automate est par définition muni d'un nombre fini de nœuds. Cette contradiction permet de conclure qu'il n'existe pas toujours de politique optimale représentable sous forme compacte.

Critères d'optimisation

S'il n'est pas possible d'avoir un nombre infini de nœuds dans de tels automates, il est cependant possible augmenter le nombre de nœuds tant que le manque à gagner n'est pas en dessous d'un certain seuil $\varepsilon > 0$. L'objectif est donc de calculer une politique ε-optimale. Pour y parvenir, nous reformulons les équations de Bellman (2.70) en passant à la limite infinie :

$$v(b) = \max_{a \in A} \left\{ R(b,a) + \lambda \sum_{\omega \in \Omega} \mathbb{P}(b'|b,a) v(b') \right\}, \quad \forall b \in \mathscr{P}(S), \tag{2.76}$$

où $\lambda \in [0,1)$ est le facteur de décompte. De manière similaire, il est possible d'étendre l'ensemble des résultats établis dans le cas à horizon fini.

La connaissance des critères d'optimisation permet de définir des méthodes de construction des solutions optimales, ε-optimales, ou approximatives.

2.4 Résolution de POMDPs

Cette section propose un survol des méthodes de résolution exacte des POMDPs. Compte tenu de la similitude des critères d'optimisation dans le cas fini et infini, la description dans le cas horizon fini sert de base pour le cas horizon infini. Comme d'une part, la sélection d'un arbre de décisions est identique quelque soit la croyance initial, et d'autre part, tout politique π est composée d'un ensemble d'arbres de décisions δ, nous nous contentons de décrire le processus de construction d'un arbre de décisions optimales ou ε-optimales pour une seule croyance.

Une des premières idées quant à la construction d'une politique optimale ou ε-optimale d'un POMDP à horizon fini consiste à énumérer toutes les politiques puis à les évaluer afin d'en extraire la meilleure.

2.4.1 Algorithme d'énumération de politiques

Nous présentons un simple algorithme de calcul d'un arbre de décisions optimales $\delta_{0:N-1}$ d'un POMDP à horizon fini muni d'un état de croyance initial $b_0 \in \mathscr{P}(S)$, voir Algorithme 8. L'idée intuitive de cet algorithme est la suivante : l'algorithme commence par construire tous les arbres de décisions $\mathscr{Q}_{N-1:N-1} = \{\delta_{N-1:N-1}\}$ correspondants aux dernières décisions ; il construit ensuite les arbres de décisions $\mathscr{Q}_{N-2:N-1} = \{\delta_{N-2:N-1}\}$ correspondants aux deux dernières décisions $\tau = N-2$ et $\tau = N-1$; et ainsi de suite jusqu'à la construction de l'ensemble des arbres de décisions $\mathscr{Q}_{0:N-1} = \{\delta_{0:N-1}\}$ correspondants aux décisions de l'horizon initial à l'horizon final $\tau = 0, 1, \cdots, N-1$.

L'algorithme d'énumération des arbres de décisions décrit, Algorithme 8, consiste en deux principales étapes :

1. l'énumération des arbres de décisions $\mathscr{Q}_{\tau:N-1}$, à chaque horizon $\tau = 0, 1, \cdots, N$,
2. l'élagage de arbres de décisions *dominés*, à chaque horizon $\tau = 0, 1, \cdots, N$.

L'énumération des arbres de décisions se fait de façon incrémentale. Elle débute par la construction de l'ensemble des arbres de décisions possibles $\mathscr{Q}_{N-1:N-1}$ que l'agent pourrait suivre pour le choix de sa dernière action. Il s'agit d'arbres de profondeur 1 c'est à dire d'actions, $\mathscr{Q}_{N-1:N-1} = A$. L'ensemble des arbres de décisions ainsi construit peut être

Algorithme 8 Algorithme d'énumération des arbres de décisions pour les POMDPs.

procedure ÉNUMÉRATION DES ARBRES DE DÉCISIONS
  Soit $\tau \leftarrow N$, $\mathcal{Q}_{N-1:N-1} \leftarrow A$
  répéter
    $\tau \leftarrow (\tau - 1)$,
    Générer l'ensemble des arbres de décisions $\mathcal{Q}_{\tau:N-1}$
    Élaguer les arbres de décisions dominés

$$\mathcal{Q}_{\tau:N-1} \leftarrow \text{ÉNUMÉRATION DE POLITIQUES}(\mathcal{Q}_{\tau:N-1}) \tag{2.77}$$

  jusqu'à $\tau = 0$
  Retourner $\mathcal{Q}_{0:N}$.
fin procedure

élagué afin d'éliminer les actions dominées pour tout état de croyance. Le programme linéaire permettant de réaliser cela est décrit à la Figure 7. L'état suivant de l'algorithme d'énumération des arbres de décisions consiste en l'énumération des arbres de décisions $\mathcal{Q}_{N-2:N-1}$. Pour ce faire, on procède à une mise à jour exhaustive des arbres de décisions $\mathcal{Q}_{N-1:N-1}$. C'est à dire que pour chaque action, on associe à toute observation possible un arbre de décisions issue de $\mathcal{Q}_{N-1:N-1}$. Ainsi si l'on dispose initialement de $|\mathcal{Q}_{\tau:N-1}|$ arbres de décisions, $|A|$ actions disponibles, et $|\Omega|$ observations, il y aura $|A||\mathcal{Q}_{\tau:N-1}|^{|\Omega|}$ arbres de décisions dans l'ensemble $\mathcal{Q}_{\tau-1:N-1}$, pour tout horizon $\tau = 0, 1, \cdots, N-1$. Après cette énumération exhaustive des arbres de décisions, une étape d'élagage est effectuée. Ces sous routines de génération puis élagage d'arbres de décisions se poursuivent jusqu'à ce que l'horizon souhaité soit atteint. Bien qu'inefficace en pratique, cet algorithme sert de base pour la construction d'algorithmes plus efficaces dans le cas mono et multi-agents.

Afin de comprendre la complexité de l'algorithme d'énumération exhaustive des politiques, il convient d'analyser la taille de l'espace de recherche des arbres de décisions non dominées $\mathcal{Q}_{\tau:N-1}$. Comme nous l'avons souligné précédemment, la génération exhaustive des arbres de décisions $\delta_{\tau:N-1} \in \mathcal{Q}_{\tau:N-1}$ partant des arbres de décisions $\delta_{\tau-1:N-1} \in \mathcal{Q}_{\tau-1:N-1}$, requiert la construction de $|A||\mathcal{Q}_{\tau-1:N-1}|^{|\Omega|}$ arbres de décisions avec $|\mathcal{Q}_{N-1:N-1}| = |A|$. Dès lors, le nombre d'arbres de décisions possibles à horizon τ est donné par :

$$\begin{aligned}
|\mathcal{Q}_{\tau:N-1}| &= |A||\mathcal{Q}_{\tau-1:N-1}|^{|\Omega|} && (2.78)\\
&\leqslant |A|\left(|A||\mathcal{Q}_{\tau-2:N-1}|^{|\Omega|}\right)^{|\Omega|} && (2.79)\\
&\leqslant |A|\left(|A|\left(|A||\mathcal{Q}_{\tau-3:N-1}|^{|\Omega|}\right)^{|\Omega|}\right)^{|\Omega|} && (2.80)\\
&\leqslant \cdots && (2.81)\\
&\leqslant |A|^{\sum_{\ell=0}^{\tau}|\Omega|^{\ell}} && (2.82)\\
&= |A|^{\frac{|\Omega|^{\tau+1}-1}{|\Omega|-1}} && (2.83)
\end{aligned}$$

pour tout horizon $\tau = 0, 1, \cdots, N-1$. Au regard de l'espace de recherche de l'algorithme d'énumération exhaustive des arbres de décisions, les recherches se sont tournées vers des approches susceptibles de distinguer les arbres de décisions non dominées des politiques dominées sans au préalable avoir à les énumérer de façon exhaustive. Une des approches la plus réputée pour cela est l'approche préconisée dans le cadre de la résolution des MDPs : l'algorithme d'itération de valeurs.

2.4.2 Algorithme d'itération de valeurs

Cet algorithme est une généralisation de l'algorithme d'itération de valeurs pour la résolution des MDPs. Contrairement aux MDPs, en POMDPs la statistique suffisante résumant tout historique est l'état de croyance. Partant d'un POMDP, il est possible de mettre à jour l'état de croyance à l'aide du théorème de Bayes sur le calcul des probabilités conditionnelles. Comme l'état de croyance constitue la statistique suffisante pour la prise de décisions optimales, il est possible l'utiliser afin de définir un type particulier de MDP avec un ensemble infini d'états [Aström, 1965]. Dans ce contexte, les équations d'optimalité peuvent se réécrire comme suit :

1. pour tout horizon $\tau = 0, 1, \cdots, N-1$,

$$v_{\tau:N}(b) = \max_{a \in A} \left\{ R(b,a) + \lambda \sum_{\omega \in \Omega} \mathbb{P}(\omega|b,a) v_{\tau+1:N}(b') \right\} \tag{2.84}$$

   avec $b'(s) = \mathbb{P}(s'|b,a,\omega)$, $\mathbb{P}(s',\omega|b,a) = O(\omega|s,a)\,T(s'|s,a)$.

2. pour $\tau = N$ on a la condition limite :

$$v_{N:N}(b) = \max_{a \in A} R(b,a) \tag{2.85}$$

La valeur $v_{\tau:N}(b)$ ne peut être calculée directement pour chaque état de croyance $b \in \mathscr{P}(S)$, car il y a un nombre infini d'états de croyance. Cependant, il est possible de calculer l'ensemble des fonctions de valeurs $\Lambda_{\tau:N}$ par une séquence d'opérations sur l'ensemble des fonctions de valeurs $\Lambda_{\tau+1:N}$, pour tout horizon $\tau = 0, 1, \cdots, N-1$. Il est utile de noter que l'ensemble des fonctions de valeurs, à l'horizon $\tau = N$, est donné par :

$$\Lambda_{N:N} \leftarrow R(a), \quad \forall a \in A. \tag{2.86}$$

De plus, l'équation (2.84) peut se réécrire comme suit :

$$v_{\tau:N}(b) = \max_{a \in A} \left\{ R(b,a) + \lambda \sum_{\omega \in \Omega} \max_{v \in \Lambda_{\tau+1:N}} \mathbb{P}(\omega|b,a) v(b') \right\} \tag{2.87}$$

Mise à jour de $v_{\tau:N} = \mathbb{H}v_{\tau+1:N}$

1. La première opération consiste en la génération des ensembles intermédiaires $\Lambda^{a,*}_{\tau:N}$ et $\Lambda^{a,\omega}_{\tau:N}$, pour toute action $a \in A$, pour toute observation $\omega \in \Omega$ et pour tout horizon $\tau = 0, 1, \cdots, N-1$:

$$\Lambda^{a,*}_{\tau:N} \leftarrow v^{a,*}(s) = R(s,a) \tag{2.88}$$

$$\Lambda^{a,\omega}_{\tau:N} \leftarrow v^{a,\omega}_i(s) = \lambda \sum_{s \in S} T(s'|s,a) O(\omega|a,s') v_i(s'), \tag{2.89}$$

   $\forall v_i \in \Lambda_{\tau+1:N}$ où $v^{a,\omega}_i$ et v_i sont des fonctions de valeurs sur S.

2. Par la suite, nous créons un ensemble $\Lambda^a_{\tau:N}$ pour toute action $a \in A$, par la somme croisée sur les observations. Ceci de sorte que l'ensemble résultant ♣ contienne des fonctions de valeurs construites en sommant $|\Omega|$ fonctions de valeurs, une pour chaque observation :

$$\Lambda^a_{\tau:N} = \Lambda^{a,*}_{\tau:N} \oplus \Lambda^{a,\omega_1}_{\tau:N} \oplus \Lambda^{a,\omega_2}_{\tau:N} \oplus \cdots \oplus \Lambda^{a,\omega_{|\Omega|}}_{\tau:N} \tag{2.90}$$

3. Enfin, nous prenons l'union des ensembles $\Lambda^a_{\tau:N}$:

$$\Lambda_{\tau:N} = \cup_{a \in A} \Lambda^a_{\tau:N} \tag{2.91}$$

   La fonction de valeurs $v_{\tau:N}$ peut alors être extraite de $\Lambda_{\tau:N}$ comme prescrit à l'équation (2.87).

FIGURE 2.9 – Opérateur $\mathbb{H}$ de mises à jour pour les POMDPs

Malheureusement, dans le pire des cas, l'algorithme de mise à jour de la fonction de valeurs $v_{\tau:N}$, décrit à la Figure 2.9, requiert un temps doublement exponentiel selon le nombre d'observations et l'horizon de planification. Afin de comprendre ce résultat, observons tout d'abord que l'étape (1) de l'algorithme 2.9 génère $|A||\Omega||\Lambda_{\tau+1:N}|$ fonctions de valeurs Λ. En outre, l'étape (2) opère $|A||\Lambda_{\tau+1:N}|^{|\Omega|}$ sommes croisées [1]. Ainsi, dans le pire cas, l'opération de mise à jour génère :

$$|\Lambda_{\tau:N}| = \mathcal{O}(|A||\Lambda_{\tau+1:N}|^{|\Omega|}) \tag{2.92}$$

$$= \mathcal{O}(|A|^{\frac{|\Omega|^{\tau+1}-1}{|\Omega|-1}}) \tag{2.93}$$

$$\tag{2.94}$$

1. La somme croisée de deux ensembles $A = \{1,2\}$ et $B = \{3,4\}$ notée $A \oplus B$ est égale à $\{(1+3),(1+4),(2+3),(2+4)\}$ soit l'ensemble $\{4,5,5,6\}$.

fonctions de valeurs à l'horizon τ. De plus, ces fonctions de valeurs peuvent être calculées en temps $\mathcal{O}(|S|^2|A||\Lambda_{\tau+1:N}|^{|\Omega|})$. Il est utile de noter la similarité des dimensions, entre l'espace des arbres de décisions, équation (2.83), et celui des fonctions de valeurs à horizon τ, équation (2.94). Cela s'explique du fait que toute fonction de valeurs de l'ensemble $\Lambda_{\tau:N}$ correspond à la fonction de valeurs d'un arbre de décisions.

Heureusement, bon nombre de fonctions de valeurs $v \in \Lambda_{\tau:N}$ générées sont complètement dominées par une autre fonction de valeurs ou par une combinaison linéaire de fonctions de valeurs :

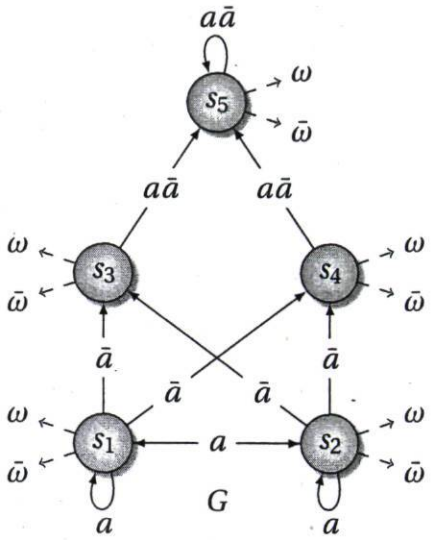
$$v_i(b) \quad < \quad v_j(b), \quad \forall b \in \mathscr{P}(S) \tag{2.95}$$

Ces fonctions de valeurs dominées peuvent alors être supprimées sans avoir aucun impact sur la fonction de valeurs $v_{\tau:N}$. Une fois de plus, comme toute fonction de valeurs de $\Lambda_{\tau:N}$ est la fonction de valeurs d'un arbres de décisions, le programme linéaire 7 permet également d'éliminer les fonctions de valeurs dominées.

La recherche des fonctions de valeurs $v \in \Lambda_{\tau:N}$ complètement dominées est coûteuse en temps. En effet, vérifier si une fonction de valeurs est complètement dominées requiert la résolution d'un programme linéaire avec $|S|$ variables et $|\Lambda_{\tau:N}|$ contraintes. Seulement cette tâche d'élagage des fonctions de valeurs complètement dominées est utile car elle peut permettre de ne conserver qu'un nombre raisonnable de fonctions de valeurs. Malgré de nombreuses méthodes d'élagage des fonctions de valeurs dominées, dans la grande majorité des cas pratiques les ensembles de fonctions de valeurs $\Lambda_{\tau:N}$ croissent de façon exponentielle avec l'horizon de planification $\tau = 0, 1, \cdots, N$. L'une des principales raisons de la difficulté à appliquer les POMDPs comme modèle de formalisation de certains problèmes réels réside dans cette complexité inhérente à l'espace mémoire nécessaire pour le résoudre.

Exemple 2. *Considérons un problème, illustré à la Figure 2.10, à 5 états. L'agent est initialement dans un état quelconque avec une probabilité uniforme. Dans ces états, la fonction d'observations offre une connaissance floue de l'état courant de l'agent. En prenant l'action a, l'agent se déplace de façon stochastique entre l'état s_1 et l'état s_2. Tandis qu'en exécutant l'action $\bar{a}$, l'agent se déplace vers les états s_3 et s_4. L'état s_5 est un état absorbant. La fonction de récompenses est telle qu'il est bon* $(+100)$ *de se déplacer vers l'état s_3, et néfaste* (-100) *de se déplacer vers l'état s_4. La récompense est nulle partout ailleurs. La fonction de transitions $T^a(s, s')$ et la fonction d'observations $O^\omega(s)$ (dépendant exclusivement de la paire d'état et d'observation (s, ω)) sont illustrées à la Figure 2.10.*

En observant ce POMDP, il apparaît que seul deux états dans les fonctions valeurs seront améliorés par application de l'opérateur de mises à jour $\mathbb{H}$, à savoir les états s_1 et s_2.



$$T^a(s,s') = \begin{array}{c} \\ s \end{array}\begin{array}{c} s' \\ \begin{bmatrix} .1 & .9 & 0 & 0 & 0 \\ .8 & .2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

$$T^{\bar{a}}(s,s') = \begin{bmatrix} 0 & 0 & .9 & .1 & 0 \\ 0 & 0 & .1 & .9 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$O^{\omega}(s) = \begin{array}{c} \\ s \end{array}\begin{array}{c} \omega \\ \begin{bmatrix} .7 & .3 \\ .4 & .6 \\ .5 & .5 \\ .5 & .5 \\ .5 & .5 \end{bmatrix} \end{array} \qquad R(s) = \begin{array}{c} \\ s \end{array}\begin{bmatrix} 0 \\ 0 \\ +100 \\ -100 \\ 0 \end{bmatrix}$$

FIGURE 2.10 – Exemple de POMDP simple, et sa représentation graphique G.

Cela s'explique par le fait que les états s_3, s_4 et s_5 obtiennent leurs valeurs optimales, au travers de leur récompense immédiate $+100$, -100 et 0, respectivement. Pour cette raison, nous focaliserons notre attention uniquement sur les deux composantes représentant les états s_1 et s_2.

Afin de commencer la résolution de ce problème, un ensemble d'hyperplans initiaux Λ_0 est extrait de la fonction de récompenses, en incluant un hyperplan $v^{a,*}$ par action $a \in A$:

$$\Lambda_0 \leftarrow v^{a,*}(s) = R(s,a), \quad \forall a \in A. \tag{2.96}$$

La Figure 2.11 montre la fonction de valeurs initiales v_0. Cette figure montre uniquement les deux dimensions représentant les états s_1 et s_2. En effet, seules les valeurs correspondantes aux états $\{s_1, s_2\}$ peuvent varier.

La Figure 2.12 décrit les étapes qui permettent de construire l'ensemble des hyperplans Λ_1 à horizon $\tau = 1$. La première étape consiste en la projection de l'ensemble Λ_0 suivant chaque paire « action / observation », comme décrit à l'équation (2.89). La seconde étape décrit le produit croisé, équation (2.90). Dans ce cas, parce que chaque ensemble $\Lambda_0^{a,\omega}$ contient un seul hyperplan, le produit croisé se réduit alors en une somme. L'étape

finale procède à l'union des deux ensembles Λ_0^a et $\Lambda_0^{\bar{a}}$ comme décrit dans l'équation (2.91). Cela produit la solution à horizon $\tau = 1$ pour le problème à 5 états. La fonction de valeurs correspondantes est illustrée à la Figure 2.13.

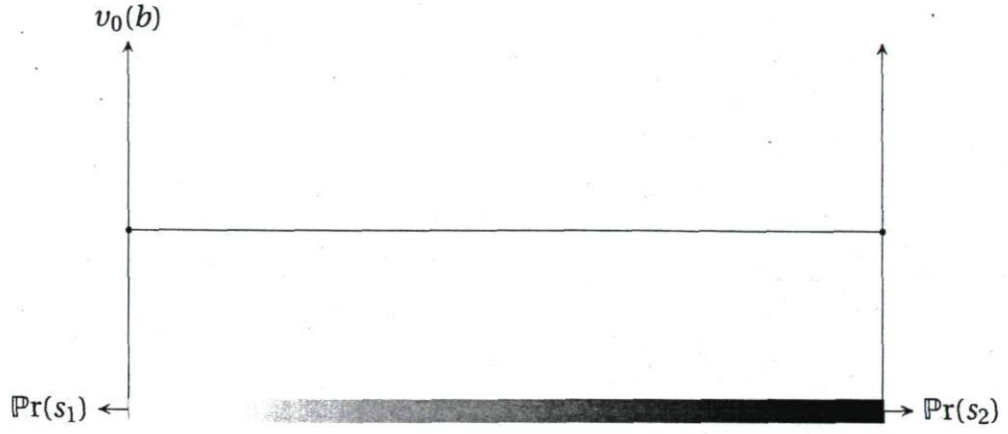


FIGURE 2.11 – Fonctions de valeurs initiales v_0.

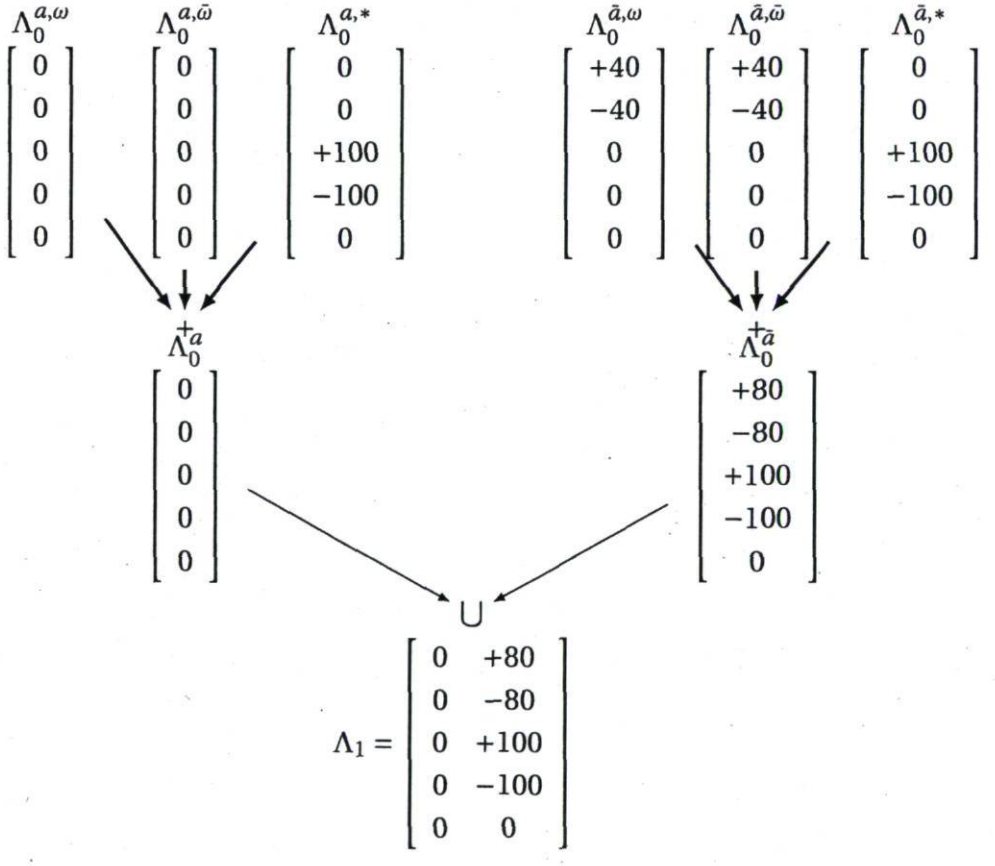


FIGURE 2.12 – Ensemble d'hyperplans à horizon $\tau = 1$, Λ_1.

La Figure 2.14 décrit les étapes menant à la solution à horizon $\tau = 2$. C'est à dire la fonction de valeurs $v_2 \equiv \Lambda_2$. Cela commence par la projection de l'ensemble Λ_1 suivant

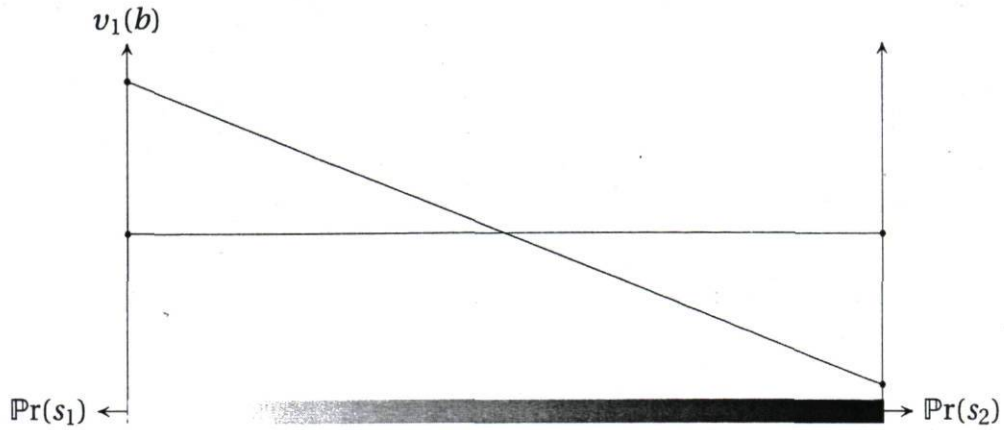


FIGURE 2.13 – Fonction de valeurs à horizon $\tau = 1$, v_1.

$$\Lambda_1^{a,\omega} = \begin{bmatrix} 0 & 47 \\ 0 & -14 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad \Lambda_1^{a,\bar{\omega}} = \begin{bmatrix} 0 & 17 \\ 0 & -34 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad \Lambda_1^{a,*} = \begin{bmatrix} 0 \\ 0 \\ +100 \\ -100 \\ 0 \end{bmatrix} \quad \Lambda_1^{\bar{a},\omega} = \begin{bmatrix} +40 & +40 \\ -40 & -40 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad \Lambda_1^{\bar{a},\bar{\omega}} = \begin{bmatrix} +40 & +40 \\ -40 & -40 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad \Lambda_1^{\bar{a},*} = \begin{bmatrix} 0 \\ 0 \\ +100 \\ -100 \\ 0 \end{bmatrix}$$

$$\Lambda_1^{\oplus a} = \begin{bmatrix} 0 & 17 & 47 & 64 \\ 0 & -34 & -14 & -48 \\ 100 & 100 & 100 & 100 \\ -100 & -100 & -100 & -100 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \Lambda_1^{\oplus \bar{a}} = \begin{bmatrix} +80 & +80 & +80 & +80 \\ -80 & -80 & -80 & -80 \\ 100 & 100 & 100 & 100 \\ -100 & -100 & -100 & -100 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\Lambda_2 = \Lambda_1^{\oplus a} \cup \Lambda_1^{\oplus \bar{a}} = \begin{bmatrix} 0 & 17 & 47 & 64 & 80 \\ 0 & -34 & -14 & -48 & -80 \\ 100 & 100 & 100 & 100 & \\ -100 & -100 & -100 & -100 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

FIGURE 2.14 – Ensemble d'hyperplans à horizon $\tau = 2$, Λ_2.

chaque paire « action / observation ». Dans ce cas, il y a deux hyperplans Λ_1, donc il y aura deux hyperplans dans chaque ensemble $\Lambda_2^{a,\omega}$. Puis, le produit croisé est réalisé sur les ensembles $\Lambda_2^{a,\omega}$ et $\Lambda_2^{a,\bar{\omega}}$. Enfin, on les somme, en combinaison avec $\Lambda_2^{a,*}$. Dans le cas de l'ensemble $\Lambda_2^{\bar{a}}$, cela conduit à quatre hyperplans identiques. En effet, chaque ensemble $\Lambda^{\bar{a},\omega}$ et

$\Lambda^{\tilde{a},\tilde{\omega}}$ contient deux copies d'un même hyperplan. La dernière étape procède à l'union des ensembles Λ_2^a et $\Lambda_2^{\tilde{a}}$. Dans ce cas, il suffit de préserver une seule copie de cet hyperplan dans $\Lambda_2^{\tilde{a}}$. L'ensemble Λ_2 contient alors cinq hyperplans, comme illustré à la Figure 2.15. Des itérations supplémentaires peuvent être réalisées de sorte à déterminer une fonction de valeurs pour des horizons plus élevés. Ceci met fin à la discussion sur cet exemple. □

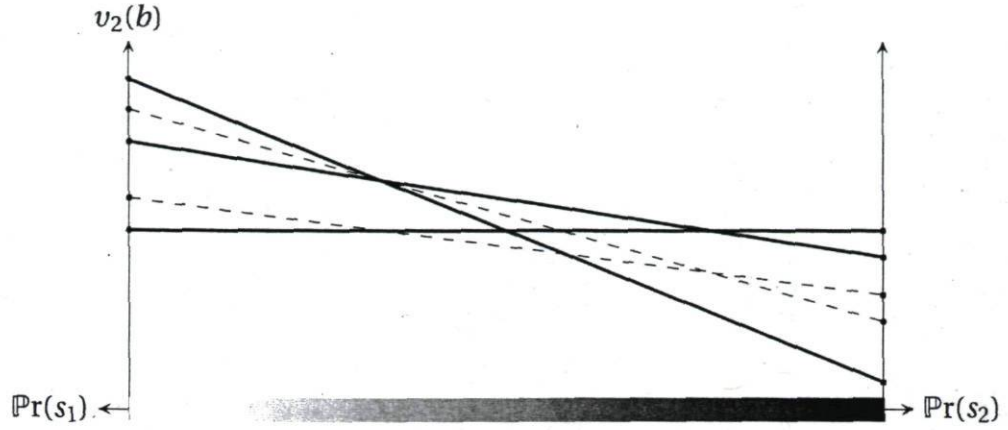


FIGURE 2.15 – Fonctions de valeurs à horizon $\tau = 2$, v_2.

2.4.3 Algorithme d'itération de politiques

Outre l'algorithme d'itération de valeurs, il existe également des approches d'itération de politiques pour la résolution des POMDPs [Hansen, 1997, 1998, Ji et al., 2007, Sondik, 1971, 1978]. La meilleure implémentation de l'algorithme exacte d'itération de politiques présentée ici est dûe à [Hansen, 1997]. Cet algorithme se décompose en deux étapes : d'une part, une étape d'évaluation de la politique courante est effectuée ; d'autre part, une étape d'amélioration de cette dernière est mise en œuvre. Nous nous contentons par la suite du déterminisme d'une politique ε-optimale car nous avons la certitude qu'il existe toujours un automate fini d'états pouvant représenter une telle politique.

L'étape (1) de l'évaluation de la politique procède au calcul de la fonction de valeurs $v^\pi \equiv \Lambda_\tau$ associée à la politique π courante. La résolution de l'équation (2.97) permet de déterminer la fonction de valeurs v^π représentée par l'ensemble fini de fonctions de valeurs Λ_τ. Lorsque la politique π est représentée sous forme d'un automate d'états finis δ, on dénote v^x la fonction associée à chaque état $x \in X$. L'évaluation d'une politique représentée sous forme d'un automate d'états finis consiste alors en la résolution du système d'équations linéaires suivant :

$$v^x(s) = R(s,\alpha(x)) + \lambda \sum_{s',\omega} T(s'|s,\alpha(x))O(\omega|s',\alpha(x))v^{\mu(x,\omega)}(s')$$

Algorithme 9 Algorithme d'itération de politiques.

1: **procedure** PI
2: Soit $\pi \leftarrow \pi_0$ un arbre de décisions donné, et $\tau \leftarrow 0$,
3: **répéter**
4: ÉVALUATION DE LA POLITIQUE : calculer Λ_τ en résolvant,

$$(\mathbf{1} - \lambda T_\pi)v \leftarrow R_\pi \tag{2.97}$$

5: MISE À JOUR DE LA FONCTION DE VALEURS :

$$\Lambda_{\tau+1} \leftarrow \mathbb{H}\Lambda_\tau \tag{2.98}$$

6: TANSFORMATION DE LA POLITIQUE :

$$\pi \leftarrow \text{TRANSFORMER}(\Lambda_\tau, \pi) \tag{2.99}$$

7: **jusqu'à** $\|\Lambda_\tau - \Lambda_{\tau+1}\| \leqslant \varepsilon \frac{1-\lambda}{2\lambda}$
8: Retourner $\pi^*_\varepsilon \leftarrow \pi$.
9: **fin procedure**

pour tout état $x \in X$ et tout état $s \in S$. L'ensemble des fonctions de valeurs Λ_τ est donné par l'ensemble des fonctions de valeurs $\{v^x\}_{x \in X}$. L'étape (2) de mise à jour de la fonction de valeurs correspond à l'utilisation de l'opérateur de mises à jour $\mathbb{H}$, introduit dans le cadre de l'algorithme d'itération de valeurs. Mettre à jour la fonction de valeurs v^π, représentée par Λ_τ, revient à transformer l'ensemble de fonctions de valeurs Λ_τ par un ensemble de fonctions de valeurs améliorées $\Lambda_{\tau+1}$, comme indiqué à l'équation (2.98). La dernière étape procède à la transformation de la politique courante π afin de prendre en compte les dernières modifications de la fonction de valeurs $v^{\pi'}$, représentée par $\Lambda_{\tau+1}$. Pour ce faire, une simple comparaison des ensembles Λ_τ et $\Lambda_{\tau+1}$ est suffisante. La transformation de la politique courante π représentée sous forme d'un automate δ se fait en utilisant les ensembles Λ_τ et $\Lambda_{\tau+1}$. Elle se base sur une représentation arborescente de la fonction de valeurs d'un état $x \in X$ comme illustré à la Figure 2.16. On appelle dans cette représentation arborescente de la fonction de valeurs v^x , état successeur de x noté $\mu(x, \omega)$, l'ensemble des états accessibles en partant de x puis en percevant une observation ω. Dans l'exemple Figure 2.16, les successeurs de l'état x_1 sont les états x_0 et x_2 pour les observations ω_1 et ω_2, respectivement. Nous somme dès lors prêt à présenter les règles de transformation de la politique courante.

Premièrement, notons que certaines des fonctions de valeurs $v^x \in \Lambda_{\tau+1}$ sont dupliquées dans Λ_τ, c'est à dire que leurs actions, leurs états successeurs et leurs valeurs pour tout état $s \in S$ sont identiques. Tout état de l'automate courant δ pour lequel il existe

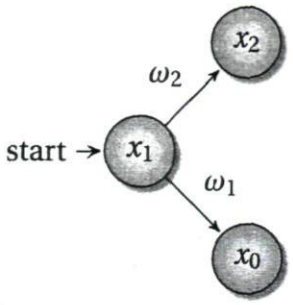


FIGURE 2.16 – Représentation arborescente de la fonction de valeurs v^{x_1} d'un état x_1.

Transformation de la politique courante

Soit une politique π représentée par un automate fini d'états δ de fonction de valeurs v^π représentée par Λ_τ, et $\varepsilon > 0$ un entier arbitraire.

1. Pour toute fonction de valeurs $v^x \in \Lambda_{\tau+1}$:
   (a) Si l'action $\alpha(x)$ et les états successeurs $\mu(x,\omega)$ associés à v^x dupliquent ceux de δ_τ, alors *conserver* l'état x inchangé dans $\delta_{\tau+1}$.
   (b) Si v^x domine une fonction de valeurs $v^y \in \Lambda_\tau$ pour tout état $s \in S$ et tout état y de l'automate δ, *remplacer* l'action ainsi que les états successeurs de y par ceux utilisés dans la construction de x. ♣
   (c) Sinon *ajouter* un état ayant l'action et les états successeurs de x à l'automate $\delta_{\tau+1}$.
2. Finalement, élaguer tout état n'ayant pas de fonction de valeurs dans $\Lambda_{\tau+1}$, tant qu'il n'est pas accessible par un état x dont la fonction de valeurs v^x est inclut dans $\Lambda_{\tau+1}$.

FIGURE 2.17 – Algorithme de transformation de la politique courante.

une fonction de valeurs dupliquées dans $\Lambda_{\tau+1}$ est conservé comme tel dans l'automate amélioré δ'. Les fonctions de valeurs de $\Lambda_{\tau+1}$ qui ne sont pas dupliquées conditionnent les transformations de la politique courante. Si une fonction de valeurs non dupliquées $v^x \in \Lambda_{\tau+1}$ domine une fonction de valeurs $v^y \in \Lambda_\tau$, l'état y est modifié de sorte que son action et ses états successeurs correspondent à ceux de l'état x. Si une fonction de valeurs non dupliquées $v^x \in \Lambda_{\tau+1}$ ne domine pas une fonction de valeurs $v^y \in \Lambda_\tau$, l'état x est ajouté dans δ'. S'il existe des états n'ayant de fonction de valeurs correspondantes dans $\Lambda_{\tau+1}$, elles devraient être supprimées, à moins que leurs états y soient accessibles partant de certains états x dont les fonctions de valeurs sont stockées dans $\Lambda_{\tau+1}$. Cet algorithme d'itération de politiques garantie non seulement une croissance monotone de la qualité

de la fonction de valeurs, $v_0 \leqslant v_1 \leqslant \cdots \leqslant v_N$, mais aussi une convergence vers un politique ε-optimale, en un nombre fini d'itérations [Hansen, 1997].

En outre l'analyse de complexité indique que l'algorithme d'itération de politiques n'a pas une complexité si différente de celle de l'algorithme d'itération de valeurs. En effet, outre les étapes d'évaluation de la politique (étape 1) et de transformation de la politique (étape 3), cet algorithme est identique à l'algorithme d'itération de valeurs. Hors, l'étape d'évaluation de la politique correspond à la résolution de l'équation (2.97) pour tout état x de l'automate représentant la politique courante. Cette procédure nécessite $\mathcal{O}(|\Lambda_\tau||S|^3)$ opérations élémentaires au temps τ. L'étape de la transformation de la politique requiert simplement $\mathcal{O}(|\Lambda_\tau| + |\Lambda_{\tau+1}|)$ opérations élémentaires. En somme, la complexité de l'algorithme d'itération de politiques est dominée par celle de la mise à jour de la fonction de valeurs, soit $\mathcal{O}(|A|^{\frac{|\Omega|^{\tau+1}-1}{|\Omega|-1}})$ comme dans le cas de l'algorithme d'itération de valeurs.

S'il est possible de calculer la fonction de valeurs optimales v^*, ou ε-optimales v^*_ε, sur la totalité des états de croyance, le nombre de fonctions de valeurs Λ issue des mises à jour exactes sera exponentiel par rapport à l'horizon. C'est pour cette raison que les approches exactes présentées précédemment ne peuvent résoudre que des problèmes jouets. Ces dernières observations motivent la recherche d'algorithmes approximatifs capables de faire face à des applications réelles, en particulier au travers du calcul de la fonction de valeurs sur un sous-ensemble représentatif des états de croyance.

2.5 Approximation de POMDPs

De nombreuses approches ont été proposées afin de surmonter la complexité de résolution exacte des POMDPs. Cette section propose un survol des grandes familles d'approches approximatives.

2.5.1 Méthodes à base d'états de croyance

La fonction de valeurs ε-optimales, v^*_ε, peut être approximée progressivement en procédant à des mises à jour sur un sous-ensemble représentatif des états de croyance $B \subseteq \mathscr{P}(S)$. Ce type de mises à jour de la fonction de valeurs est nommée mises à jour à base de points [Lovejoy, 1991, Pineau, 2004, Smith and Simmons, 2005, Spaan and Vlassis, 2005]. Pour tout état de croyance $b \in B$, la mise à jour de la fonction de valeurs représentée par l'ensemble des fonctions de valeurs Λ est définie par $\Lambda' \leftarrow \tilde{\mathrm{H}}\Lambda$, tel que l'opérateur $\tilde{\mathrm{H}}$ soit

défini par :

$$v^{a\omega}(s) = \sum_{s' \in S} O(\omega|a,s')T(s'|s,a)v(s') \tag{2.100}$$

$$v^{a} = R(a) + \lambda \sum_{\omega \in \Omega} \underset{v_{ao}:\, v \in \Lambda}{\operatorname{argmax}} (b \cdot v_{ao}) \tag{2.101}$$

$$v^{b} = \underset{v_a:\, a \in A}{\operatorname{argmax}} (b \cdot v_a) \tag{2.102}$$

pour tout état de croyance $b \in B$, où $\Lambda' = \{v^b \mid b \in B\}$, $\Lambda = \{v\}$. Les principaux attraits de cette approche résident d'une part, dans la complexité polynomiale $\mathcal{O}(|B||A||\Omega||\Lambda|)$ de l'opérateur $\tilde{\mathbb{H}}$ et d'autre part dans le fait que la dimension de la fonction de valeurs approximatives $v_N \equiv \Lambda$ soit bornée à terme par le nombre $|B|$ des états de croyance, soit $|\Lambda| \leqslant |B|$. La politique extraite d'une fonction de valeurs ε-optimales sur B est de bonne qualité si elle se généralise bien aux états de croyance $b \notin B$ à l'extérieur de l'ensemble B. Une fonction de valeurs v se généralisa d'autant plus facilement à un état de croyance $b \notin B$ que sa valeur $v(b)$ n'est pas très éloignée de la valeur optimale $v^*(b)$. Cette propriété explique l'importance d'une sélection pertinente des états de croyance. Sachant que seuls les états de croyance accessibles et noté $\bar{\mathscr{P}}(S)$ conditionnent les actions d'une politique, il est clair que le calcul exact d'une fonction de valeurs sur l'ensemble des états de croyance accessibles produit une politique ε-optimale à l'exécution. Fort de ce constat, de nombreux algorithmes de mises à jour à base de points ne considèrent qu'un sous-ensemble $B \subseteq \bar{\mathscr{P}}(S)$ des états de croyance accessibles à partir de l'état de croyance initial b_0 [Pineau, 2004]. Les techniques de mises à jour à base des états de croyance diffèrent néanmoins de part leur routine individuelle de sélection des états de croyance B mais aussi de part l'ordre suivant lequel ces états de croyance sont mises à jour.

Algorithme PBVI [Pineau et al., 2003b]

L'algorithme d'itération de valeurs à base d'états de croyance (PBVI) a été un des premiers algorithmes d'approximation de la fonction de valeurs sur un ensemble sélectionné d'états de croyance parmi tous les états de croyance accessibles [Pineau et al., 2003b]. En particulier, PBVI calcule alternativement l'ensemble des états de croyance B_τ et la fonction de valeurs ε-optimales, $v_\tau \equiv \Lambda_\tau$, sur l'ensemble B_τ. L'algorithme débute en considérant l'ensemble des états de croyance initiaux $B_0 = \{b_0\}$ comprenant l'état de croyance initial b_0, il calcule la fonction de valeurs ε-optimales Λ_0^* sur l'ensemble B_0, puis étend l'ensemble B_0 à l'ensemble B1 en y incorporant certains des états de croyance successeurs des états de croyance de B0 et calcule une fois de plus la fonction de valeurs Λ_1 ε-optimale sur B_1. Cette procédure se répète tant qu'un critère d'arrêt n'est pas atteint.

En plus d'hériter des propriétés de complexité en temps et mémoire des algorithmes

Algorithme 10 Algorithme PBVI.

1: **procedure** PBVI
2: Soient Λ_0, $B_0 \leftarrow \{b_0\}$, $\tau \leftarrow 0$.
3: **répéter**
4: Ajouter $\Lambda_\tau \leftarrow \mathbb{H}\Lambda_\tau(b)$, pour tout $b \in B_\tau$.
5: Étendre l'ensemble des états de croyance,

$$B_{\tau+1} \leftarrow B_\tau \tag{2.103}$$

$$B_{\tau+1} \leftarrow \underset{b'=(b,a,\omega)}{\arg\max} \min_b \|b' - b\|_1 \tag{2.104}$$

6: Poser $\tau \leftarrow (\tau + 1)$.
7: **jusqu'à** un critère d'arrêt est atteint
8: Retourner Λ_τ.
9: **fin procedure**

à base de points, PBVI se distingue par sa borne sur l'erreur résultant de l'approximation de la fonction de valeurs ε-optimales sur un ensemble fini d'états de croyance. Soit $r_\infty = \max_{a,s} R(s,a)$ la récompense maximale possible, l'algorithme PBVI retourne une fonction de valeurs Λ_N telle que :

$$\|\Lambda_N - \Lambda_N^*\|_\infty \leqslant \frac{2\varepsilon_B r_\infty}{(1-\lambda)^2} + \lambda^N \|\Lambda_0^* - \Lambda^*\|_\infty \tag{2.105}$$

où $\varepsilon_B = \max_{b' \in B \times A \times \Omega} \min_{b \in B} \|b' - b\|_1$.

2.5.2 Méthodes à base de trajectoires d'états de croyance

Une des approches de mises à jour à base d'états de croyance, parmi les plus répandues, sélectionne les états de croyance par simulation de l'exécution d'une politique π. Une telle simulation produit un historique $h_{0:N}$ pour tout $N < \infty$. De cet historique, il est possible d'extraire $(N+1)$ états de croyance comme suit : pour tout historique $h_{0:\tau}$, pour tout temps $\tau = 0, 1, \cdots, N-1$,

$$b_\tau(s) = \mathbb{P}(s | h_{0:\tau}) \tag{2.106}$$

avec $h_{0,\tau+1} = (h_{0,\tau}, a_\tau, \omega_{\tau+1})$ où $a_\tau = \pi(h_{0:\tau})$ et $\omega_{\tau+1}$ est échantillonné à partir de la distribution de probabilités $O(\cdot | a_{\tau+1}, s_{\tau+1})$ et l'état $s_{\tau+1}$ est également échantillonné à partir de la distribution de probabilités $T(\cdot | a_\tau, s_{\tau+1})$. En particulier s_0 est échantillonné à partir de b_0 . Ainsi la simulation d'une politique produit une trajectoire $b_0, b_1, \cdots, b_N$ d'états de croyance, et les méthodes qui génèrent ainsi B sont dites méthodes à base de trajectoires [Bonet and Geffner, 1998, Shani et al., 2007, Smith and Simmons, 2004, 2005].

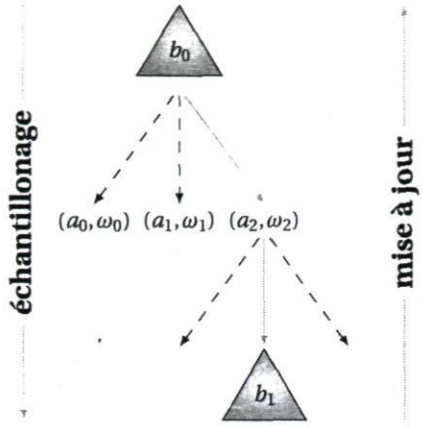


FIGURE 2.18 – Illustration des approches de mises à jour à base de trajectoires.

Algorithme FSVI [Shani et al., 2007]

Par exemple, l'algorithme d'itération de valeurs par chaînage avant (FSVI) utilise la politique π_{MDP} du MDP sous-jacent[2] au POMDP à résoudre afin de sélectionner les actions permettant par la suite de générer l'ensemble des états de croyance.

Algorithme 11 Algorithme FSVI.

1: **procedure** FSVI
2: Soient Λ, π_{MDP} et b_0.
3: Échantillonner s_0 de b_0, et poser $\tau \leftarrow 0$.
4: **répéter**
5: MISES À JOUR À BASE DE TRAJECTOIRES(b_τ, s_τ)
6: **jusqu'à** Λ n'a pas convergé
7: Retourner Λ.
8: **fin procedure**

De plus, les algorithmes à base de trajectoires procèdent aux mises à jour de leurs états de croyance par chaînage arrière. Considérons la trajectoire d'états de croyance $b_0, b_1, \cdots, b_N$. Les mises à jour se font dans le sens inverse de la trajectoire. C'est à dire $v^{b_N}, v^{b_{N-1}}, \cdots, v^{b_0}$. De cette manière la dépendance causale entre les états de croyance est respectée et la propagation des valeurs est plus grande. Par conséquence, la convergence est plus rapide en général. Bien que FSVI soit une approche efficace en pratique elle ne dispose à ce jour d'aucune garantie théorique quant à la qualité de la politique construite.

L'avantage de l'algorithme FSVI réside dans le fait que le calcul de la politique π_{MDP} requiert un temps négligeable. Il en est de même pour la sélection des actions a_τ, pour tout

2. Le MDP sous-jacent d'un POMDP correspond aux attributs (S, A, P, R) d'un MDP dans le modèle (S, A, P, R, O, Ω) du POMDP.

Algorithme 12 Routine de mises à jour dans l'algorithme FSVI.

1: **procedure** MISES À JOUR À BASE DE TRAJECTOIRES(b_τ, s_τ)
2: **si** s_τ n'est pas un état but **alors**
3: Choix de l'action $a_\tau \leftarrow \pi_{MDP}(s_\tau)$.
4: Échantillonner $s_{\tau+1}$ de $T(\cdot|s_\tau, a_\tau)$.
5: Échantillonner $\omega_{\tau+1}$ de $O(\cdot|a_\tau, s_{\tau+1})$.
6: **Mises à jour à base de trajectoires**($b_{\tau+1}, s_{\tau+1}$).
7: **fin si**
8: Ajouter la fonction de valeurs v^{b_τ} à Λ.
9: **fin procedure**

$\tau = 0, 1, \cdots, N$. Néanmoins, un des inconvénients de FSVI est son incapacité à reconnaître les trajectoires capables d'améliorer l'information concernant l'état du système. En effet, en utilisant la politique π_{MDP}, FSVI limite sa capacité à construire des trajectoires susceptibles de visiter des états susceptibles de produire des observations utiles. De plus, la politique π_{MDP} étant stationnaire, les trajectoires générées sont identiques quelque soit la fonction de valeurs courantes. En conséquence, dans certaines applications, FSVI retourne une fonction de valeurs sous-optimales sur l'ensemble des états de croyance visités.

Algorithme HSVI [Smith and Simmons, 2004, 2005]

Dans la famille des algorithmes à base de trajectoires, l'algorithme de construction heuristique de la fonction de valeurs, HSVI, a démontré d'impressionnantes performances sur un grand nombre de benchmarks. À la différence de FSVI, cet algorithme construit ses trajectoires en sélectionnant les actions a_τ suivant une borne supérieure $\bar{\Lambda}_\tau$ et une borne inférieure $\underline{\Lambda}_\tau$ sur la fonction de valeurs Λ^*_τ recherchée, pour tout $\tau = 0, 1, \cdots, N$ avec $N \leqslant \infty$. Le lecteur notera que bien qu'il soit possible de résoudre des POMDPs à horizon infini en utilisant HSVI, la longueur des trajectoires considérées est toujours finie.

Algorithme 13 Algorithme HSVI.

1: **procedure** HSVI
2: Initialiser $\underline{\Lambda}_0$ et $\bar{\Lambda}_0$, et $\tau = 0$.
3: **répéter**
4: EXPLORER($b_\tau, \bar{\Lambda}_\tau, \underline{\Lambda}_\tau$).
5: **jusqu'à** $\bar{\Lambda}_\tau(b_\tau) - \underline{\Lambda}_\tau(b_\tau) \leqslant \varepsilon$
6: **fin procedure**

Chacune des trajectoires d'états de croyance $b_0, b_1, \cdots, b_N$ commence par un état de

Algorithme 14 Routine d'exploration de l'algorithme HSVI.

procedure EXPLORER
  si $\bar{\Lambda}_\tau(b_\tau) - \underline{\Lambda}_\tau(b_\tau) > \frac{\varepsilon}{\lambda^\tau}$ **alors**
    Choix de l'action $a_\tau \leftarrow \arg\max_{a \in A} \bar{\Lambda}_\tau(b_\tau, a)$,
    Choix de l'observation $\omega_{\tau+1}$

$$\omega_{\tau+1} \leftarrow \arg\max_{\omega \in \Omega} \left(\bar{\Lambda}_\tau(b_{\tau+1}) - \underline{\Lambda}_\tau(b_{\tau+1})\right)$$

où $b_{\tau+1} \leftarrow (b_\tau, a_\tau, \omega_{\tau+1})$.
    EXPLORER$(b_{\tau+1}, \bar{\Lambda}_\tau, \underline{\Lambda}_\tau)$
    Mettre à jour la borne inférieure : $\underline{\Lambda}_\tau \leftarrow v^{b_\tau}$.
    Mettre à jour la borne supérieure : $\bar{\Lambda}_\tau \leftarrow v^{b_\tau}(b_\tau)$.
  fin si
fin procedure

croyance initial b_0. Les actions a_τ simulées correspondent toujours à l'action la meilleure prescrite par la borne supérieure $\bar{\Lambda}_\tau$. L'états de croyance $b_{\tau+1}$ succédant à b_τ est l'état de croyance pour lequel la distance $\bar{\Lambda}_\tau(b_{\tau+1}) - \underline{\Lambda}_\tau(b_{\tau+1})$ entre les deux bornes est maximale. Lorsque la trajectoire est complètement générée, les états de croyance sont mises à jour dans l'ordre inverse de leur génération. Les trajectoires générées sont d'autant plus bonnes que la distance entre les bornes est négligeable. Malheureusement, le maintient des ces bornes requiert des efforts considérables. De fait, le gain en temps résultant du nombre réduit des mises à jour n'est que partiellement perceptible par rapport au temps total de résolution du POMDP.

2.5.3 Méthodes de réduction de la dimension

Le domaine de recherche sur les processus décisionnels de Markov partiellement observables est un domaine extrêmement dynamique, comme le prouve le nombre des approches proposées ces dernières années. Sa complexité PSPACE-difficile a orienté les chercheurs vers des méthodes approximatives capables de faire face aux énormes besoins en mémoire.

Famille Exponentielle de PCAs En particulier, [Roy et al., 2005] proposent une méthode de planification dans des sous-espaces de croyances de dimensions réduites. Ceci est possible notamment à travers une technique similaire à celle de l'analyse en composantes principales (PCA) [Jolliffe, 2002]. Les auteurs proposent également une méthode de pla-

nification étant donnée ces sous-espaces de croyances. Les résultats expérimentaux sont assez impressionnants. En effet, la complexité de résolution dans certains exemples de POMDPs est réduite de façon exponentielle.

Exploitation de l'historique La principale idée derrière les approches visant à tirer partie de l'historique, est de se détacher du concept d'état de croyance. Ceci avec l'objectif de représenter les politiques directement comme des fonctions conditionnées par des séquences d'observations. L'avantage de ces méthodes est qu'elles ne requièrent pas de modèle de la dynamique de système. Elles ne basent le processus d'optimisation que sur les données observées à savoir : actions ; observations ; ou récompenses immédiates.

L'algorithme UTree, proposé par [McCallum, 1995], offre une approche dans laquelle les historiques d'observations sont représentés sous forme d'arbres de suffixes, et où les branches croissent lorsqu'une nouvelle observation apparaît.

La représentation prédictive des états (PSR) en est un autre exemple [Littman et al., 2001]. Elle est basée sur des prémices similaires. Cependant, en lieu et place des historiques, la politique est conditionnée par des tests de prédictions. Un test correspond à une séquence d'observations futures. Dans ce contexte, les états sont exprimés en termes de probabilités sur les séquences d'observations. L'ensemble des tests peut être appris directement par exploration des données [Rosencrantz et al., 2004, Singh et al., 2003].

L'avantage principale de ces approches est qu'elles ne nécessitent pas la connaissance du modèle. Cela peut s'avérer problématique dans des domaines où le coût d'exploration est exorbitant.

Recherche en ligne D'autres approches par contre tentent de faire face à la complexité prohibitive des POMDPs en alternant phase de planification et phase d'exécution : il s'agit de la planification en ligne [Bonet and Geffner, 1998, Paquet et al., 2005, Ross and Chaib-draa, 2007]. L'avantage de ces approches réside dans le fait qu'elles ne sont pas contraintes à résoudre le problème en totalité. Seuls de très petits horizons sont résolus. Puis la fonction de valeurs récoltées est exploitée afin de prendre les décisions courantes lors de la phase d'exécution.

2.6 Conclusion

Dans ce chapitre, nous avons discuté des problèmes de contrôle centralisé des processus de Markov, qu'ils soient complètement observables (MDPs) ou partiellement observables (POMDPs). Nous avons présenté les modèles, les équations de Bellman qui régissent la sélection des politiques, ainsi que les principaux algorithmes exacts et approximatifs. Nous avons, en particulier, décrit les algorithmes d'itération de valeurs et de politiques dans les deux modèles.

Si les algorithmes classiques semblent les plus appropriés en générale, nous avons souligné l'importance des approches spécialisées, soit dans l'organisation des opérations de mises à jour de la fonction de valeurs ou de la politique, soit dans la compression de l'espace mémoire requis par les approches classiques. Nous avons en particulier mis en évidence les performances des approches de recherche heuristique sur la base de trajectoires. Que se soit en MDP ou en POMDP, ces approches ont montré des performances parmi les meilleures et gagneraient à être considérées dans d'autres domaines étendant les modèles MDPs et POMDPs.

La théorie de la planification centralisé exacte ou approximative pour le contrôle centralisé des processus de Markov complètement ou partiellement observables est extrêmement riche. Les principaux travaux se tournent désormais vers l'application de cette théorie sur de problèmes de taille réelle. En POMDPs, la recherche théorique en planification après avoir connue un croissance considérable ces dix dernières années semble désormais s'orienter vers la validation sur des applications : en Médecine ; dans le E-commerce ; et bien d'autres domaines. Ces applications offrent, en effet, des hypothèses additionnelles pouvant contraindre l'espace des croyances ou des politiques et ainsi réduire la complexité théoriques des POMDPs. En MDPs, les applications sont légions, et ne cessent de nourrir la théorie.

La solidité de ces théories tranche avec la jeunesse des modèles Markoviens introduits récemment dans le cadre du contrôle des processus multi-agents. En particulier, lorsqu'il s'agit du contrôle distribué d'un système multi-agents, le modèle des processus décisionnels de Markov partiellement observables et décentralisés est adopté. Les ébauches de théorie de la planification dans ce cadre très générique ont consisté pour l'essentiel à adapter au cadre multi-agents plusieurs des techniques rudimentaires décrites dans ce chapitre.

Chapitre 3

Contrôle distribué de processus décisionnels de Markov

« L'essentiel, quand on a un commandement, c'est de prendre une décision, quelle qu'elle soit. On s'effraie au début, puis avec l'expérience, on s'aperçoit que cela revient à peu près au même... quoi qu'on décide. »

Jean Anouilh
extrait de *L'alouette*

JUSQU'ICI nous avons décrit les problèmes de prise de décisions séquentielles du point de vue d'un seul agent interagissant avec son environnement : qu'il s'agisse du pilote d'avions, du courtier ou des parents préparant le repas du soir. Ce point de vue est une simplification grossière de la modélisation des acteurs influençant l'évolution des problèmes de prise de décisions séquentielles. Par exemple, la poursuite d'une proie par une meute d'hyènes affamées requiert la prise en compte de divers point de vue, un pour chacun des acteurs interagissant au cours de ce processus. De façon générale, la majorité des applications d'envergure font intervenir une multitude d'acteurs, allant parfois jusqu'à des millions.

Dans ce chapitre, nous nous intéressons au problème du contrôle distribué d'un système de prise de décisions séquentielles, multi-agents et coopératif. Dans les problèmes de prise de décisions séquentielles abordés précédemment, le contrôle étant centralisé, quelque soit le nombre d'agents, les interactions de l'ensemble des agents avec l'environnement étaient perceptibles de tous. À la différence, dans le problème général du contrôle distribué, non seulement les effets de actions individuelles sont incertaines et les observations individuelles sont partielles, mais en plus les récompenses sont conjointes. Pour cette raison, un grand nombre de résultats établis dans le cadre du contrôle centralisé (Chapitre 2) sont à réviser.

Ce chapitre introduit d'une part, le modèle général de contrôle distribué des systèmes de prise de décisions séquentielles, multi-agents et coopératifs – en Section 3.1. D'autre part, il passe, Section 3.2, 3.4, 3.3 et 3.5, en revue l'ensemble des méthodes exactes et approximatives. Nous y abordons en outre et de façon brève les modèles des sous-classes munies d'hypothèses simplificatrices – en Section 3.6 et 3.7.

3.1 DEC-POMDPs

Le modèle général de contrôle distribué des processus décisionnels de Markov partiellement observables (DEC-POMDP) offre un cadre général de formalisation et résolution des problèmes de contrôle distribué des systèmes multi-agents et coopératifs, comme illustré à la Figure 3.1. En général, le système multi-agents est soumis à plusieurs contraintes. D'une part, chaque agent perçoit différente information imparfaite sur l'état du système et les effets des actions conjointes sont incertaines, tout en ne percevant les récompenses que conjointement. D'autre part, à l'exécution chaque agent doit disposer de sa propre politique individuelle, lui permettant ainsi d'exécuter cette dernière indépendamment des autres agents. Cette dernière contrainte est une condition nécessaire et suffisante du contrôle distribué dont sont exemptes les modèles dont nous avons parlé jusqu'ici : MDPs

et POMDPs.

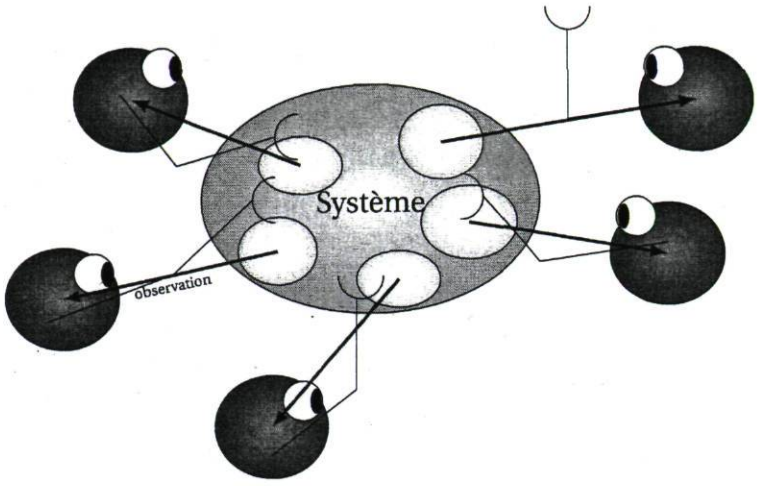


FIGURE 3.1 – Chaque agent perçoit différente information partielle sur l'état du système.

3.1.1 Cadre formel

Un DEC-POMDP est un formalisme multi-agents du contrôle distribué d'un processus décisionnel de Markov. Chaque agent dispose de ses actions et observations individuelles, mais il ne peut les communiquer aux autres agents lors de la phase d'exécution. L'ensemble des fonctions de récompenses, de transitions et d'observations, dépendent quant à elles des actions et parfois des observations conjointes de l'ensemble des agents.

Définition 5. *Un DEC-POMDP est donné par un* 7*-uplet* $(S, A, \Omega, T, R, O, \mathscr{I})$ *où :*

1. Les états. *S est l'ensemble fini des états du système,*
2. Les actions. $A \equiv A^1 \times A^2 \times \cdots \times A^{|\mathscr{I}|}$ *est l'ensemble fini des actions conjointes* $a \in A$ *de l'ensemble des agents* $\mathscr{I}$*. Chaque agent* $i \in \mathscr{I}$ *dispose d'un ensemble* A^i *d'actions individuelles.*
3. Les observations. $\Omega \equiv \Omega^1 \times \Omega^2 \times \cdots \times \Omega^{|\mathscr{I}|}$ *correspond à l'ensemble des observations conjointes* $\omega \in \Omega$ *de l'ensemble des agents. Chaque agent* $i \in \mathscr{I}$ *dispose d'un ensemble* Ω^i *d'observations individuelles.*
4. La fonction de transitions. $T(s, a, \bar{s})$ *représente la probabilité* $\mathbb{P}(\bar{s}|s, a)$ *de transiter vers un état* $\bar{s}$ *après avoir exécuté l'action conjointe* a *dans l'état* s*.*
5. La fonction de récompenses ou de coûts. $R(s, a)$ *définit le modèle de récompenses du système, c'est à dire la récompense (resp. coût) perçue lorsque dans l'état conjoint* s *l'ensemble des agents exécute l'action conjointe* a*.*

6. Les fonctions d'observations. $O(s,a,\omega,\bar{s})$ *décrit la probabilité* $\mathbb{P}(\omega|s,a,\bar{s})$ *que l'ensemble des agents perçoive l'observation conjointe* ω *lorsque dans l'état* s*, ils exécutent l'action conjointe* a *et transitent dans l'état* $\bar{s}$.

3.1.2 Politiques conjointes

Partant d'un DEC-POMDP, l'objectif est de déterminer une politique individuelle $\pi^i \in \Pi^{\mathbf{HA}}$ pour chaque agent $i \in \mathscr{I}$. Les politiques individuelles π^i pour tout $i \in \mathscr{I}$ appartiennent à l'ensemble des politiques non Markoviennes et aléatoires $\Pi^{\mathbf{HA}}$ par défaut. En effet, il n'existe aucun résultat théorique garantissant qu'il existe une sous-classe $\Pi^{\chi} \subset \Pi^{\mathbf{HA}}$, de définition des politiques individuelles π^i, pour tout agent $i \in \mathscr{I}$.

Le problème de déterminisme de la plus petite classe χ des politiques dans laquelle il est garantie de déterminer la politique conjointe $\pi \equiv (\pi^1, \pi^2, \cdots, \pi^{|\mathscr{I}|})$ optimale pour tout DEC-POMDP, est à ce jour un problème ouvert.

Encore une fois, ce problème est fondamental car plus la classe des politiques est restreinte, plus simple est l'encodage de cette politique, plus facile sera la recherche, et par conséquent plus performantes seront les méthodes utilisées à cette fin. Néanmoins, certaines des méthodes que nous décrirons restreignent sans preuves théoriques la recherche dans l'espace des politiques non Markoviennes et déterministes $\Pi^{\mathbf{HD}}$, bien qu'il se pourrait qu'une politique optimale dans l'espace des politiques non Markoviennes et déterministes $\Pi^{\mathbf{HD}}$ soit sous optimale dans l'espace des politiques non Markoviennes et aléatoires $\Pi^{\mathbf{HA}}$.

Comme nous l'avions souligné dans le cadre des POMDPs, il existe deux principales représentations d'une politique individuelle ou conjointe d'un DEC-POMDP, à savoir l'arbre de décisions préconisée dans les cas à horizon fini et la machine à états fini utilisée dans les cas à horizon infini. Nous allons présenter l'une et l'autre de ces représentations de politiques.

Arbres de décisions

Une politique individuelle $\pi^i \in \Pi^{\mathbf{HD}}$ d'un agent $i \in \mathscr{I}$, peut être représentée comme un ensemble d'arbres de décisions de profondeur $N < \infty$, lorsque le nombre de décisions séquentielles à prendre est un nombre fini N. Chacun des arbres de décisions associe à toute séquence d'observations $(\omega_1^i \times \omega_2^i \times \cdots \times \omega_H^i)$ individuelles de longueur $H \leqslant N$, une action

individuelle $a^i \in A^i$. La Figure 3.2 illustre l'exemple d'un arbre de décisions d'une politique individuelle π^i, pour tout $i \in \mathscr{I}$. Cet arbre de décisions décrit un plan contingenté où la décision initiale de l'agent i sera d'exécuter l'action individuelle $\bar{a}$. Puis si l'agent perçoit l'observation individuelle ω (resp. $\bar{\omega}$), il choisira d'exécuter l'action individuelle a (resp. $\bar{a}$). Lorsqu'on dispose d'une information quant à l'état initial du système, une politique individuelle peut être définie partiellement. Ainsi, un seul arbre de décisions peut être suffisant pour représenter une politique individuelle en connaissance de l'état initial ou de l'historique initial du système.

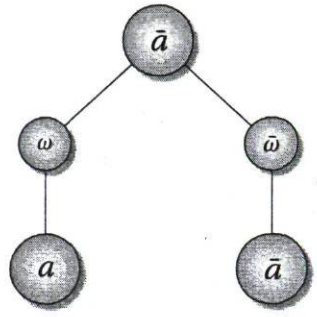


FIGURE 3.2 – Un arbre de décisions de profondeur $N = 2$ d'une politique individuelle.

Machines à états finis

Lorsque le nombre de décisions séquentielles à prendre est infini $N = \infty$, il faudrait un ensemble d'arbres de décisions de profondeur infinie pour représenter une politique individuelle d'un agent. Bien entendu, un tel arbre est impossible à représenter. S'il est impossible de garantir un encodage pour toute politique individuelle quelconque lorsque $N = \infty$, certaines d'entre elles peuvent néanmoins être encodées en utilisant une machine à états finis.

Une politique représentée sous forme d'une machine à états finis est une application qui associe à toute séquence d'observations individuelles (possiblement infinie) un action individuelle. Ainsi, certaines politiques individuelles $\pi^i \in \Pi^{\mathbf{HA}}$ (resp. $\pi^i \in \Pi^{\mathbf{HD}}$) d'un agent i pour tout $i \in \mathscr{I}$ peuvent être représentées par un ensemble de machines aléatoires (respectivement déterministes) à états finis.

Définition 6. *Une machine à états finis est un* 4*-uplet* (X, α, η, x_0)*, où :*

1. *X est un ensemble fini d'états, où chaque état représente un nœud muni d'une action individuelle $a^i \in A^i$.*
2. *$\alpha\colon X \to A^i$ pour tout $i \in \mathscr{I}$, où $\alpha(x)$ est l'action individuelle $a_i \in A^i$ exécutée lorsque la machine se trouve à l'état x ;*

3. η: $X \times \Omega^i \to \mathscr{P}(X)$ *pour tout* $i \in \mathscr{I}$, *où* $\eta(x, \omega^i, \bar{x})$ *est la fonction de transitions. Elle décrit la probabilité de transiter vers l'état successeur* $\bar{x} \in X$ *suite à l'observation* $\omega^i \in \Omega^i$ *en partant de l'état* $x \in X$. *Lorsque la fonction de transitions est déterministe, la machine est dite déterministe, le cas échéant elle est dite stochastique ;*
4. *L'état représenté par* x_0 *est l'état de départ de la machine.*

La Figure 3.3 illustre une de ces machines déterministes à états finis. Cette machine déterministe décrit un plan infini où la décision de départ $\alpha(x_3)$ est celle associée à l'état x_0. Puis pour toute séquence d'observations individuelles de longueur quelconque, il suffit de suivre la fonction de transitions η. L'action individuelle associée à l'état terminal sera celle à exécuter. Considérons par exemple, la séquence d'observations $< \omega, \omega, \omega, \bar{\omega}, \omega, \omega, \omega, \bar{\omega} >$. En partant de l'état de départ, la fonction de transitions nous fournit la séquence de paires d'état et d'observation suivante $< x_0, \omega, x_1, \omega, x_0, \omega, x_1, \bar{\omega}, x_2, \omega, x_1, \omega, x_0, \omega, x_1, \bar{\omega}, x_2 >$. Ainsi, l'état terminal de la séquence $< \omega, \omega, \omega, \bar{\omega}, \omega, \omega, \omega, \omega >$ est l'état x_2 et l'action à exécuter est celle qui y est associée, c'est à dire $\alpha(x_2)$.

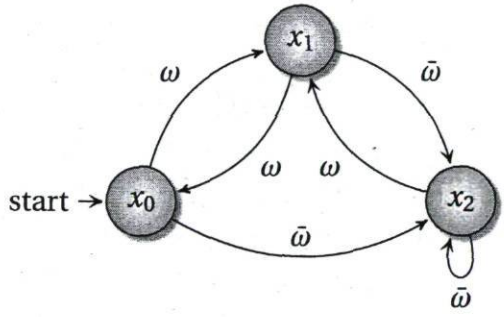


FIGURE 3.3 – Une machine déterministe à états finis.

3.1.3 Fonction de valeurs

Afin de différencier deux politiques conjointes π et $\bar{\pi}$, il nous faut choisir un critère d'optimisation, selon que le nombre de décisions à prendre soit fini $N < \infty$ ou infini $N = \infty$:

1. Si $N < \infty$ (cas à horizon fini) le critère d'optimisation communément admis est celui du total espéré des récompenses cumulées. La fonction de valeurs lorsque l'ensemble des agents I suit la politique conjointe $\pi \equiv (\pi^1, \pi^2, \cdots, \pi^{|\mathscr{I}|})$ est alors donnée par :

$$v^{\pi}_{0:N}(s_0) \quad = \quad \mathbb{E}_{\pi}\left\{ \sum_{\tau=0}^{N-1} R(s_\tau, a_\tau) + R(s_N) \mid s_0, \pi \right\} \tag{3.1}$$

2. Si $N = \infty$ (cas à horizon infini) le critère communément admis est celui du total espéré des récompenses cumulées et décomptées. La fonction de valeurs lorsque l'ensemble des agents suit la politique conjointe $\pi \equiv (\pi^1, \pi^2, \cdots, \pi^{|\mathscr{I}|})$ est alors donnée par :

$$v^{\pi}(s_0) \quad = \quad \mathbb{E}_{\pi}\left\{\sum_{\tau=0}^{\infty} \gamma^{\tau} R(s_{\tau}, a_{\tau}) \mid s_0, \pi\right\} \tag{3.2}$$

où $\gamma \in [0, 1)$ est le facteur de décompte.

3.1.4 Prise de décisions optimales

La théorie de la résolution des DEC-POMDPs est encore à ses balbutiements. Ainsi, il n'existe à ce jour aucun équivalent aux équations de Bellman dans le cas des DEC-POMDPs. Les chapitres 5 et 6 apporterons les équivalents aux équations de Bellman dans le cadre du contrôle distribué. En effet, la majorité des approches proposées jusqu'ici procèdent en deux principales étapes : (1) génération exhaustive de l'ensemble des politiques conjointes possibles ; (2) évaluation de chacune de ces politiques conjointes puis sélection de celle dont la fonction de valeurs est la meilleure selon un état initial s_0 et le critère d'optimisation choisi. Lorsqu'on ne dispose que d'une distribution b_0 sur l'ensemble des états possibles, il suffit de comparer les fonctions de valeurs pondérées, $v^{\pi}(b_0) = \sum_{s \in S} b_0(s) v^{\pi}(s)$, afin de déterminer la politique conjointe dont la fonction de valeurs est la meilleure. Nous exposons une première série d'explications justifiant l'usage récurrent de ces deux étapes.

Lorsqu'on fait référence au contrôle distribué d'un système, l'une des premières idées qui nous vient à l'esprit est de distribuer également la phase de planification. C'est à dire de déterminer pour chacun des agents sa politique individuelle, et cela indépendamment des autres agents. En DEC-POMDPs, cette approche requiert de ressources en temps et en mémoire considérables. Ainsi, à échelle réelle, cette approche reste peu applicable en pratique. Pour mieux comprendre, il faut analyser de plus près les traces d'interactions d'un agent avec son environnement en comparaison de celles issues de l'interaction de l'ensemble des agents avec l'environnement.

Observons la trace issue de l'exécution d'une politique individuelle π^i illustrée à la Figure 3.4, pour tout agent $i \in \mathscr{I}$. Cette trace correspond exclusivement à une séquence d'observations individuelles $(\omega_1^i \times \omega_2^i \times \cdots \times \omega_N^i)$ pour tout agent $i \in \mathscr{I}$. On y observe également, qu'au cours des interactions avec son environnement, un agent ne peut percevoir individuellement les récompenses conjointes. Or, la sélection d'une politique (même individuelle) requiert la connaissance du total espéré des récompenses cumulées lorsque cette politique est suivie. La Figure 3.4 illustre ainsi parfaitement le fait qu'en distribuant le pro-

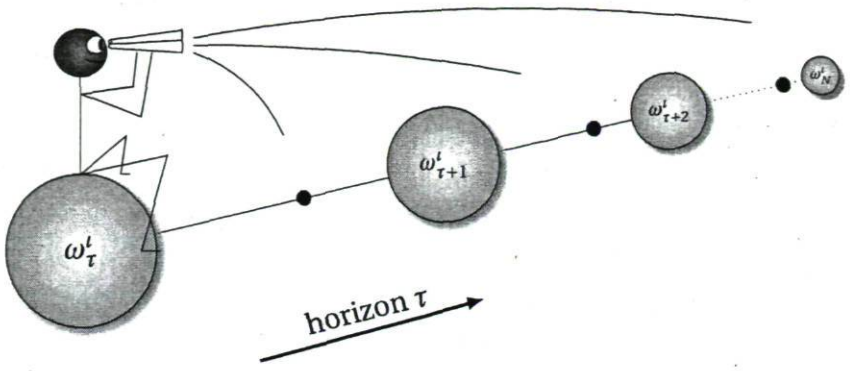


FIGURE 3.4 – Un agent perçoit des informations incomplètes et imparfaites limitant sa capacité à construire indépendamment sa politique individuelle.

cessus de planification et en restreignant le processus de sélection des politiques exclusivement sur les informations collectées par chaque agent indépendamment les uns des autres, il est impossible de construire une politique individuelle π^i optimisant un quelconque critère basé sur les récompenses. En effet, par hypothèse, la fonction de récompenses $R(s, a)$ est définie exclusivement pour un état et une action, tous deux conjoints.

Le choix d'une politique individuelle π^i ne peut donc se faire indépendamment des choix de politiques des autres agents, comme l'illustre la Figure 3.5. Plus précisément, seule la connaissance à priori de l'ensemble $\mathcal{Q}_\tau^{\neq i}$ des politiques individuelles de l'ensemble des agents sauf l'agent i à tout horizon $\tau = 0, 1, \cdots, N$, permet de générer une trace propre à l'agent i composée de paires « observation individuelle – récompense conjointe » $(\omega^i_\tau, r_{\tau+1})$, pour tout agent $i \in \mathcal{I}$, comme illustré à la Figure 3.5. Une telle trace est suffisante pour sélectionner une politique individuelle de façon à optimiser le critère du total espéré des récompenses cumulées. Nous ne surmontons pas pour autant la difficulté à résoudre le problème initial, car ce raisonnement est circulaire. En effet, si nous avons besoin de la politique individuelle de nos coéquipiers afin de construire la notre, eux également ont besoin de notre politique individuelle afin de construire les leurs. Ce raisonnement circulaire explique pourquoi il est souvent utile de générer dans un premier temps l'ensemble des politiques individuelles de chacun des agents avant d'en sélectionner les meilleures – d'où les deux étapes ci-dessus. Bernstein et al. [2002] ont établi que lorsque le nombre de décisions séquentielles à prendre est fini $N < \infty$, un DEC-POMDP peut être résolu de façon optimale en temps au plus doublement exponentiel. En particulier, un DEC-POMDP à horizon fini et à deux agents est NEXP-difficile en général, le cas infini est indécidable en général.

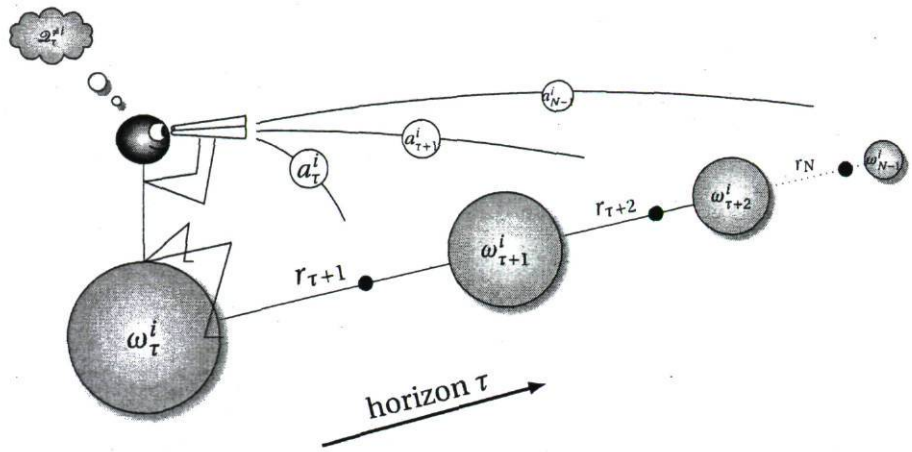


FIGURE 3.5 – Un agent percevant des informations partielles mais muni des politiques individuelles des autres agents, peut construire indépendamment sa politique individuelle.

3.1.5 Statistique suffisante

La statistique suffisante est une donnée essentielle dans la représentation des politiques et la conception des algorithmes. En effet, la connaissance de la statistique suffisante permet de focaliser les efforts de recherche d'une politique optimale dans un espace plus restreint. Par exemple, dans le cadre des processus décisionnels de Markov partiellement observables, nous avons mentionné que la statistique suffisante était la croyance, c'est à dire une distribution de probabilités $b \in \mathscr{P}(S)$ sur l'ensemble des états du système.

La croyance centrale

En simulant conjointement les politiques individuelles de l'ensemble des agents, les traces qui en résulte sont des historiques conjoints. Il convient de se demander si cette information n'offre pas un moyen efficace de calculer la politique conjointe optimale ou ε-optimale d'un DEC-POMDP. Cette question est à ce jour ouverte. Contrairement aux POMDPs, dans le cadre des DEC-POMDPs, il n'est en effet pas prouvé que la croyance conjointe soit suffisante pour la planification exacte. Cette question fondamentale trouve également sa réponse aux chapitres 5 et 6.

La croyance individuelle

La construction de la politique conjointe optimale peut se faire en utilisant la notion de croyance individuelle d'un agent. Cette information a été introduite afin de permettre la sélection d'une politique individuelle d'un seul agent si l'on dispose de l'ensemble des politiques conjointes possibles des autres agents [Hansen et al., 2004, Nair et al., 2003]. En effet, en observant à nouveau la Figure 3.5, on constate que la trace $\langle(\omega_0^i, r_1), (\omega_1^i, r_2), \cdots, (\omega_{N-1}^i, r_N)\rangle$ pour tout agent $i \in \mathcal{I}$ permet de sélectionner une action individuelle a_τ^i pour tout $\tau = 0, 1, \cdots, N$. Par conséquent, cette trace permet de calculer une politique individuelle $\pi i.L'informati$ $\mathcal{I}$ et l'ensemble des politiques individuelles des autres agents $\mathcal{Q}_{\tau:N}^{\neq i}$ est suffisante pour sélectionner les actions individuelles de l'agent $i \in \mathcal{I}$.

Soit $\mathcal{Q}_{0:N}^i$ l'ensemble des politiques individuelles et potentielles de l'agent $i \in \mathcal{I}$. Posons $\mathcal{Q}_{0:N}^{\neq i} \equiv \mathcal{Q}_{0:N}^1 \times \mathcal{Q}_{0:N}^2 \times \cdots \times \mathcal{Q}_{0:N}^{i-1} \times \mathcal{Q}_{0:N}^{i+1} \times \cdots \times \mathcal{Q}_{0:N}^{|\mathcal{I}|}$ l'ensemble des politiques conjointes de l'ensemble des agents $\mathcal{I} \setminus \{i\}$. La croyance individuelle b_i de l'agent $i \in \mathcal{I}$ est donnée par une distribution de probabilités sur l'ensemble $S \times \mathcal{Q}_{0:N}^{\neq i}$, telle que $b_i \in \mathcal{P}(S \times \mathcal{Q}_{0:N}^{\neq i})$.

La dimension de l'espace des croyances individuelles d'un agent est variable suivant l'horizon τ. En effet, elle varie avec la dimension de l'ensemble $\mathcal{Q}_{\tau:N}^{\neq i}$ des politiques possibles des autres agents, pour tout τ. Il faut néanmoins noter que dans le cas d'un processus de prise de décisions séquentielles à horizon fini, l'ensemble des politiques conjointes possibles $\mathcal{Q}_{\tau:N}$ est borné. Par conséquent, la dimension de l'ensemble des croyances individuelles possibles d'un agent est également bornée lorsqu'on a connaissance d'une information initiale. Néanmoins, le nombre de politiques conjointes possibles est bien souvent de dimension doublement exponentielle comme nous le verrons par la suite. De sorte qu'en pratique il ne soit pas possible de les énumérer de façon exhaustive en temps raisonnable. Cependant, lorsque la dimension de l'espace des croyances individuelles n'est pas trop élevée, il est possible de faire usage des croyances individuelles afin de résoudre un DEC-POMDP. Dans ces rares cas la croyances individuelle est en général la statistique suffisante pour la planification distribuée en vue du contrôle distribué des processus de Markov.

3.2 Résolution de DEC-POMDPs à horizon fini

Dans cette section, nous proposons un exposé des méthodes exactes (ou presque) de résolution des problèmes de contrôle distribué des processus de Markov partiellement observables. En particulier, nous nous intéressons dans un premier temps au cas où le

nombre de décisions séquentielles à prendre est fini. Nous discuterons ensuite du cas à horizon fini $N < \infty$. La complexité NEXP-difficile des DEC-POMDPs à horizon fini, suggère que tout algorithme confronté au calcul d'une solution optimale ou même ε-optimale pour un tel problème nécessitera un temps doublement exponentiel dans le pire des cas.

Malgré ce constat accablant, l'intérêt des approches exactes réside dans le fait qu'elles offrent un cadre essentielle au développement de bonnes méthodes approximatives. Ces méthodes permettent d'identifier les routines qui requièrent le plus de ressources en temps et/ou mémoire. Ces routines peuvent alors être remplacées par des méthodes approximatives.

Nous commençons par exposé deux méthodes exactes inspirées de la programmation dynamique – Section 3.2.2. Nous présentons ensuite en Section 3.2.4 une autre approche intuitive mais inspirée cette fois-ci de la recherche heuristique. Enfin, la Section 3.2.5 décrit les méthodes exactes de programmation mathématique permettant de résoudre des DEC-POMDPs à horizon fini.

Hypothèses 2. *Tout au long de cette section, nous supposons que :*

1. *les ensembles A, et Ω sont finis.*
2. *l'horizon de planification est fini, $N < \infty$.*
3. *il existe toujours une politique conjointe déterministe et optimale.*

3.2.1 Énumération exhaustive

Le problème d'identifier une politique conjointe pour l'ensemble des agents d'un DEC-POMDP, est un problème combinatoire. En effet, il s'agit d'identifier la politique conjointe dont le total espéré des récompenses cumulées est le plus élevé parmi l'ensemble fini des politiques conjointes possibles. La nature fini de l'ensemble des politiques conjointes possibles n'en fait pas un problème simple. Pour s'en convaincre, il nous faut tout d'abord noter que par hypothèse une politique conjointe optimale existe toujours dans l'espace des politiques conjointes déterministes et non Markoviennes $\Pi^{\mathbf{HD}}$, pour tout DEC-POMDP. Cette propriété est une hypothèse des méthodes discutées ci-dessous, elle sera formellement prouvée au Chapitre 5. De plus, nous avons mentionné précédemment que toute politique incluse dans l'espace des politiques déterministes et non Markoviennes $\Pi^{\mathbf{HD}}$, pouvait être représentée par un ensemble fini d'arbres de décisions de profondeurs $N < \infty$. En particulier, une politique conjointe d'un DEC-POMDP peut être représentée par un vecteur d'ensembles finis d'arbres de décisions, où chaque ensemble d'arbres de décisions correspond à la politique individuelle d'un agent $i \in \mathcal{I}$. Enfin, comme les ensembles A

et Ω sont finis, l'ensemble des vecteurs d'ensembles finis d'arbres de décisions possibles $\mathcal{Q}_{0:N} est fini, et de dimension : to$ $|Q_{0:N}| \leqslant \prod_{i \in \mathcal{I}} |A^i|^{\frac{|\Omega^i|^{N+1}-1}{|\Omega^i|-1}}$ (3.3)

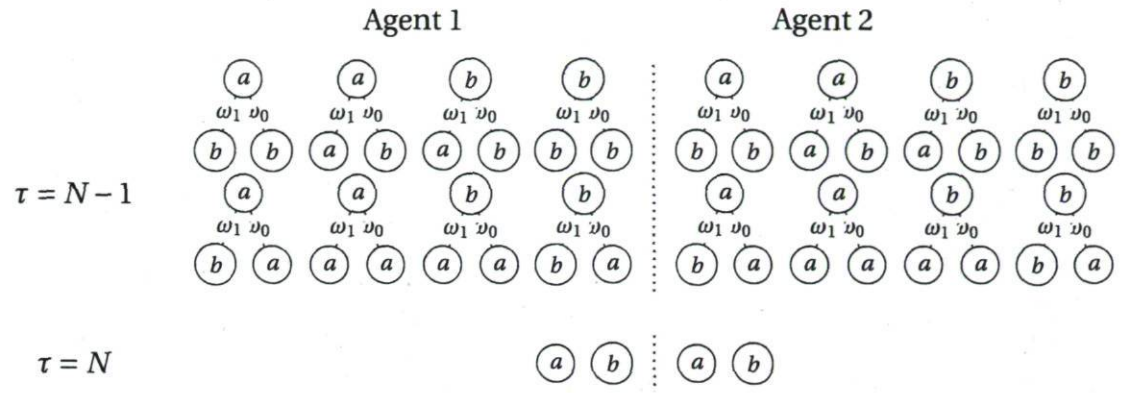


FIGURE 3.6 – Génération exhaustive des politiques.

Partant d'une information initiale, l'état initial ou de la distribution de probabilités sur les états initiaux, il est possible de sélectionner un seul arbre de décisions afin de représenter la politique individuelle d'un agent. Cette politique est alors optimisée pour la portion accessible des croyances individuelles de cet agent.

Lorsque nous ne disposons d'aucune information initiale, nous sélectionnons un arbre de décisions individuel maximal pour chaque état $s \in S$: soit un ensemble de $|S|$ arbres de décisions, en pire cas pour chaque agent $i \in \mathcal{I}$. Malheureusement, la nature finie de l'ensemble des vecteurs d'ensembles finis d'arbres de décisions possibles $\mathcal{Q}_{0:N}$ ne simplifie en rien la difficulté du problème. Pour s'en convaincre considérons un exemple simple et illustrons la croissance de l'espace de recherche avec l'horizon de planification.

Considérons un DEC-POMDP à deux agents, munis chacun de deux actions individuelles $|A^i| = 2$ et deux observations individuelles $|\Omega^i| = 2$. Le nombre de vecteurs d'arbres de décisions à générer pour obtenir une solution optimale à horizon $N = 1$ est de 4. À horizon $N = 2$, il est de 64 vecteurs. À horizon $N = 4$, il faudrait générer plus d'un milliard de vecteurs d'arbres de décisions. Le nombre d'arbres de décisions à générer croît de façon doublement exponentielle avec l'horizon de planification N, et cela même pour un DEC-POMDP insignifiant. Ce constat limite grandement l'applicabilité de la méthode de génération exhaustive puis de sélection de la politique conjointe optimale.

3.2.2 Programmation dynamique

La programmation dynamique est une méthode générale de résolution des problèmes combinatoires pour lesquels le principe de Bellman est valide : « toute solution optimale est construite à partir de sous solutions optimales ». Ainsi, il est important de s'assurer que le processus de sélection de la politique conjointe optimale d'un DEC-POMDP vérifie le principe de Bellman. Il est facile de se convaincre que toute politique optimale est constituée de sous politiques optimales en observant la fonction de valeurs. En effet, l'évaluation d'une trace d'exécution d'une politique optimale doit être telle que toute partie de cette trace soit de valeurs maximales. Cet argument intuitif ne constitue certes pas une preuve formelle, mais nous nous en contenterons pour l'instant. Encore une fois, la preuve formelle sera détaillée au chapitre 5.

Programmation dynamique exhaustive

Etant donné le principe d'optimalité de Bellman, Hansen et al. [2004] ont proposé le premier algorithme de programmation dynamique capable de déterminer la politique conjointe optimale d'un DEC-POMDP. Cette méthode consiste en la construction incrémentale de l'ensemble des politiques conjointes pour tout horizon $\tau = 0, 1, \cdots, N$. En particulier, à l'horizon $\tau = N$, à l'étape de la dernière décision à prendre, chaque agent doit exécuter une seule action individuelle. Cette action individuelle peut être représentée comme un arbre de décisions de profondeur 1. Ainsi, l'ensemble des politiques conjointes $\mathcal{Q}_{N:N}$ à l'horizon $\tau = N$ correspond à l'ensemble des actions conjointes A. Comme nous ne souhaitons garder que les meilleures sous politiques, nous évaluons les différents vecteurs d'arbres de décisions $\delta \in \mathcal{Q}_{N:N}$, pour tout état $s \in S$, et ne gardons que les politiques conjointes non dominées. Un vecteur d'arbres de décisions $\delta \equiv (\delta^1, \delta^2, \cdots, \delta^{|\mathcal{I}|})$ est non dominé, s'il existe une croyance individuelle $b \in \mathcal{P}(S \times \mathcal{Q}_{N:N}^{\neq i})$, pour tout agent $b \in \mathcal{P}(S \times \mathcal{Q}_{N:N}^{\neq i})$, pour laquelle aucun autre vecteur d'arbres de décisions $\bar{\delta} \equiv (\bar{\delta}^1, \bar{\delta}^2, \cdots, \bar{\delta}^{|\mathcal{I}|})$ n'obtient une plus grande valeur. L'algorithme décrit à la Figure 15 est un programme linéaire permettant de supprimer les arbres de décisions dominés pour tout agent $i \in \mathcal{I}$, et tout horizon $\tau = 0, 1, \cdots, N$. Un arbre de décisions peut ainsi être supprimé si la variable ϵ se révèle négative. Cet algorithme est appliqué à l'ensemble des agents $i \in \mathcal{I}$.

L'étape suivant de l'algorithme de programmation dynamique est de générer l'ensemble $\mathcal{Q}_{N-1:N}$ des politiques conjointes. Comme lorsque l'horizon $\tau = N - 1$, il ne reste que deux décisions à sélectionner, l'ensemble $\mathcal{Q}_{N-1:N}$ correspond à l'ensemble des vecteurs d'arbres de décisions de profondeur 2 construit à partir de l'ensemble $\mathcal{Q}_{N:N}$. C'est à dire que pour chaque action individuelle $a^i \in A^i$ et chaque observation individuelle $\omega^i \in \Omega^i$,

Algorithme 15 Algorithme d'élimination d'arbres de décisions dominés (DEC-POMP).

procedure ÉLIMINATION D'ARBRES DE DÉCISIONS DOMINÉS($\mathcal{Q}_{\tau:N}$)
  Soit ϵ un réel, $\tau = 0, 1, \cdots, N$, et $b \in \mathcal{P}(S \times \mathcal{Q}^{\neq i}_{\tau:N})$.
  pour tout $i \in \mathcal{I}$ **faire**
    pour tout $\delta^i \in \mathcal{Q}^i_{\tau:N}$ **faire**
      Maximiser ϵ, tel que : $\forall \hat{\delta}^i \in \mathcal{Q}^i_{\tau:N} \backslash \{\delta^i\}$,

$$\sum_{s \in S, \delta^{\neq i} \in \mathcal{Q}^{\neq i}_{\tau:N}} b(s, \delta^{\neq i}) v_{\tau:N}(s, \hat{\delta}^i \delta^{\neq i}) + \epsilon \leqslant \sum_{s \in S, \delta^{\neq i} \in \mathcal{Q}^{\neq i}_{\tau:N}} b(s, \delta^{\neq i}) v_{\tau:N}(s, \delta^i \delta^{\neq i}) \tag{3.4}$$

où $\sum_{s \in S, \delta^{\neq i} \in \mathcal{Q}^{\neq i}_{\tau:N}} b(s, \delta^{\neq i}) = 1$ et $\forall (\delta^{\neq i}, s) \in S \times \mathcal{Q}^{\neq i}_{N:N}\ b(s, \delta^{\neq i}) \geqslant 0$.
      si $\epsilon < 0$ **alors**
        Supprimer δ^i de $\mathcal{Q}^i_{\tau:N}$.
      fin si
    fin pour
  fin pour
fin procedure

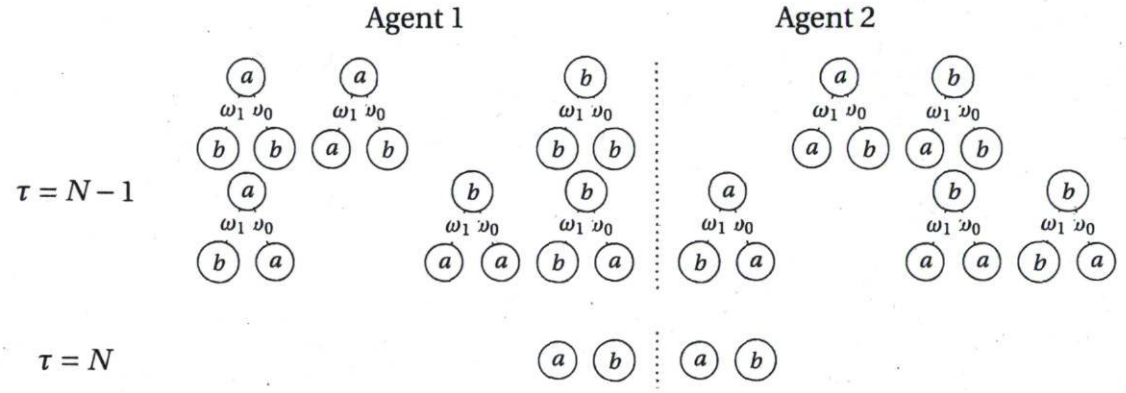


FIGURE 3.7 – Génération exhaustive et élagage des arbres de décisions dominés.

un arbre de décisions est choisi dans l'ensemble $\mathcal{Q}^i_{N:N}$, pour tout agent $i \in \mathcal{I}$. Ainsi, si l'agent dispose de $|\mathcal{Q}^i_{N:N}|$ arbres de décisions de profondeur 1, $|A^i|$ actions individuelles, et $|\Omega^i|$ observations individuelles, il y aura $|A^i||\mathcal{Q}^i_{N:N}|^{|\Omega^i|}$ arbres de décisions de profondeur 2. À l'issue de cette seconde génération exhaustive des arbres de décisions pour chaque agent $i \in \mathcal{I}$, nous élaguons à nouveau l'ensemble des vecteurs d'arbres de décisions afin de réduire le nombre de vecteurs conservés. Ces générations exhaustives et élagages continuent jusqu'à l'horizon $\tau = 0$, comme illustré à la Figure 3.7. L'algorithme de programmation dynamique exhaustif qui en résulte est illustré Algorithme 16.

De façon identique à l'approche de génération exhaustive, cet algorithme de programmation dynamique nécessite la construction de vecteurs d'arbres de politiques en pire

cas.

$$\mathcal{O}(\prod_{i\in\mathcal{I}} |A^i|^{\frac{|\Omega^i|^{N+1}-1}{|\Omega^i|-1}}) \tag{3.5}$$

L'élagage des arbres de décisions dominés est extrêmement coûteuse, car elle requiert le calcul de l'ensemble des fonctions de valeurs associées à chaque vecteur d'arbres de politiques, soit une complexité en pire cas de

$$\mathcal{O}(|S|^2 \prod_{i\in\mathcal{I}} |A^i|^{\frac{|\Omega^i|^{N+1}-1}{|\Omega^i|-1}}). \tag{3.6}$$

Néanmoins, cette étape d'élagage est bénéfique car il est utile de réduire les arbres de décisions possibles à chaque horizon $\tau = 0, 1, \cdots, N$, pour tout agent $i \in \mathcal{I}$.

Algorithme 16 Algorithme de programmation dynamique (DEC-POMDP).

procedure DP
  Soit $\tau = N$.
  répéter
    pour tout $i \in \mathcal{I}$ **faire**
      Générer exhaustivement l'ensemble $\mathcal{Q}^i_{\tau:N}$.
    fin pour
    pour tout $i \in \mathcal{I}$ **faire**
      Élimination d'arbres de décisions dominés($\mathcal{Q}_{\tau:N}$)
    fin pour
    Décrémenter $\tau \leftarrow (\tau - 1)$, puis retourner à l'étape 2.
  jusqu'à $\tau = 0$
  Retourner $\mathcal{Q}_{0:N}$.
fin procedure

3.2.3 Exploitation des croyances accessibles

Nous avons souligné l'importance qu'il y avait à réduire le nombre de d'arbres de décisions $\mathcal{Q}^i_{\tau:N}$ mémoriser pour chacun des agents $i \in \mathcal{I}$, pour tout horizon $\tau = 0, 1, \cdots, N$. Cela est d'une importance cruciale afin de simplifier la complexité de la construction des arbres de décisions pour les horizons suivants. Dans cette direction, de nombreuses tentatives ont vu le jour. Dans le cadre de la programmation dynamique : d'une part, des approches procèdent aux mises à jour de la fonction de valeurs exclusivement sur les croyances individuelles accessibles ; d'autre part, certaines méthodes proposent des encodages plus compactes des politique, en particulier sur la base des croyances individuelles accessibles.

Programmation dynamique à base de croyances

En particulier, Szer and Charpillet [2006] a proposé la première méthode de planification pour les DEC-POMDPs basée sur l'exploitation de l'accessibilité des croyances individuelles de chaque agent. L'idée est détendre au cadre des DEC-POMDPs l'algorithme à base de croyances introduit dans le cadre des POMDPs [Lovejoy, 1991, Pineau, 2004, Smith and Simmons, 2005, Spaan and Vlassis, 2005]. À la différence des approches exactes précédentes, celle-ci peut effectuer l'élagage des arbres de décisions dominés en utilisant exclusivement les croyances individuelles accessibles par opposition à l'ensemble de toutes les croyances individuelles possibles. La difficulté de cette méthode réside dans l'extraction d'un échantillon représentatif de l'ensemble des croyances individuelles accessibles.

Les croyances individuelles d'un agent $i \in \mathscr{I}$ sont représentées par un vecteur de plusieurs croyances $b_\tau^i \equiv (b_\tau^{S,i}, b_\tau^{1,i}, b_\tau^{2,i}, \cdots, b_\tau^{i-1,i}, b_\tau^{i+1,i}, \cdots, b_\tau^{|\mathscr{I}|,i})$, où $b_\tau^{S,i}$ est une distribution de probabilités sur l'espace des états $b_\tau^{S,i} \in \mathscr{P}(S)$ représentant la croyance de l'agent i sur la position de l'ensemble des agents dans l'espace des états à l'horizon $\tau = 0, 1, \cdots, N$; $b_\tau^{j,i}$ est une distribution de probabilités sur l'ensemble des arbres de décisions de l'agent j c'est à dire $b_\tau^{j,i} \in \mathscr{P}(\mathscr{Q}_{\tau:N}^j)$ pour tout agent $j \in \mathscr{I}\backslash\{i\}$ représentant la croyance de l'agent$i$ quant à l'arbre de décisions que l'agent j exécutera à l'horizon $\tau = 0, 1, \cdots, N$.

La difficulté de génération des croyances individuelles b_τ^i pour tout agent $i \in \mathscr{I}$, vient de la nécessité de connaître les politiques conjointes potentielles $\mathscr{Q}_{0:N-\tau}$ de l'ensemble des agents $\mathscr{I}.Eneffet, toutarbrededécisions\delta_{0:N}$ peut se décomposer en deux parties. La première englobe $(N-\tau)$ premiers horizons, et elle correspond à un unique arbre de décisions $\delta_{0:N-\tau}$. La seconde partie correspond aux arbres de décisions qui pourraient être exécutés pour les τ derniers horizons : ces arbres de décisions sont notés $\delta_{\tau:N} \in \mathscr{Q}_{\tau:N}$. Ainsi, afin d'élaguer ces derniers, il faut avoir une connaissance des premiers. La Figure 3.8 illustre cet état de fait. Supposons qu'il s'agit d'un arbre de décisions de profondeur $N = 3$. Afin de définir les meilleurs sous arbres de profondeur $\tau = 2$ (ellipse), il nous a fallu définir un arbre de décisions et de profondeur $N - \tau = 1$ (cercle).

La connaissance d'un vecteur d'arbres de décisions $\delta_{0:N-\tau}$, permet d'extraire les croyances individuelles b_τ^i utiles afin de mener à bien l'élagage des arbres de décisions dominées dans l'ensemble $\mathscr{Q}_{\tau:N}$. Les équations suivantes donnent le moyen d'y parvenir : pour tout état $s \in S$,

$$b_\tau^{S,i}(s) \quad = \quad \mathbb{P}(s|h_{0:N-\tau}, s_0)\mathbb{P}(h_{0:N-\tau}|\delta_{0:N-\tau}) \tag{3.7}$$

où $h_{0:N-\tau}$ est un historique d'observations de l'ensemble des agents $\mathscr{I}$ pour les $(N-\tau)$ premières observations, et extrait du vecteur d'arbres de décisions $\delta_{0:N-\tau}$; Pour tout $\delta_{\tau:N}^j \in$

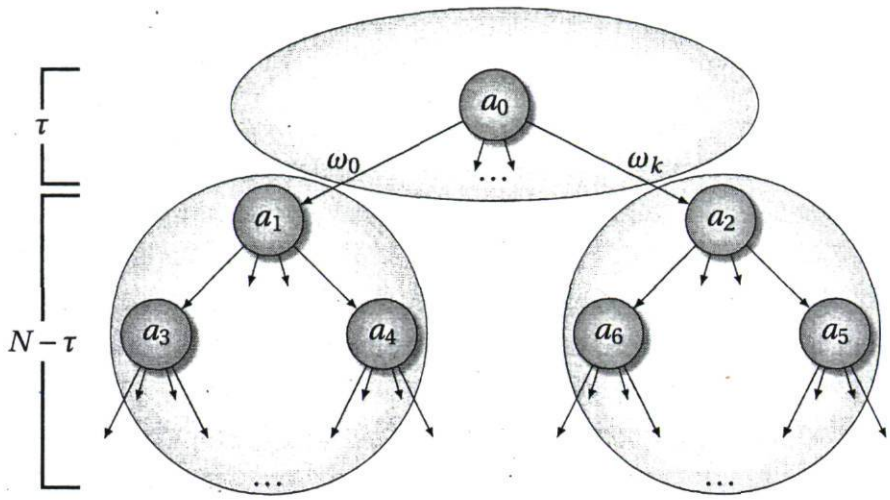


FIGURE 3.8 – Déterminisme des arbres de décisions utiles (cercles) à partir d'arbres de décisions prédéfinis à priori (ellipse).

$\mathcal{Q}^j_{\tau:N}$, et pour tout agent $j \in \mathcal{I}\backslash\{i\}$, on a :

$$b^{j,i}_{\tau}(\delta^j_{\tau:N}) = \sum_{h^i_{0:N-\tau}} \mathbb{P}(\delta^j_{\tau:N}|h^i_{0:N-\tau},\delta_{\tau:N}) \cdot \mathbb{P}(h^i_{0:N-\tau}|\delta_{\tau:N}) \tag{3.8}$$

où $h^i_{0:N-\tau}$ est un historique d'observations locales de l'agent i pour les $(N-\tau)$ premières observations, et extrait de l'arbre de décisions $\delta^i_{\tau:N}$ lui-même extrait du vecteur d'arbres de décisions $\delta_{\tau:N}$;

L'algorithme 17 résume l'ensemble des étapes de résolution d'un DEC-POMDP via la méthode à base de croyances individuelles. Une fois l'ensemble des croyances individuelles $\{B^i_\tau\}_{i\in\mathcal{I}}$ généré, le problème de l'élagage des arbres de décisions dominés est alors équivalent à celui rencontré dans le cadre des POMDPs.

Algorithme 17 Algorithme à base de croyances individuelles (PBDP).

procedure PBDP
  Poser $\tau \leftarrow N$.
  répéter
    Génération exhaustive de $\mathcal{Q}^i_{\tau:N}$, pour tout $i \in \mathcal{I}$.
    Générer l'ensemble des croyances individuelles accessibles $\{B^i_\tau\}_{i\in\mathcal{I}}$.
    Élaguer les arbres de décisions dominés dans $\mathcal{Q}^i_{\tau:N}$, pour $i \in \mathcal{I}$.
    Décrémenter $\tau \leftarrow (\tau - 1)$.
  jusqu'à $\tau = 0$
  Retourner $\mathcal{Q}_{0:N}$.
fin procedure

Le principal inconvénient de cette approche réside dans l'étape de génération des historiques. Afin de bien comprendre cela, notons tout d'abord que si nous disposons pour tout horizon $\tau = 0, 1, \cdots, N$ de la politique conjointe optimale $\delta^*_{0:N-\tau}$, alors les politiques conjointes préservées dans l'ensemble $\mathcal{Q}_{\tau:N}$ seront utiles pour la politique conjointe optimale $\delta^*_{0:N}$. Malheureusement, cette méthode ne donne aucune garantie sur la qualité des politiques conjointes potentielles utilisées dans la génération des historiques. On pourrait légitimement se demander quelle serait la qualité de la politique conjointe $\delta_{0:N}$ ainsi construite si la politique conjointe optimale $\delta^*_{0:N-\tau}$ n'est jamais inclus dans les ensembles de politiques $\mathcal{Q}_{0:N-\tau}$ prédéfinis à priori ; ou encore peut-on toujours construire une politique conjointe optimale sur des historiques extraits de politiques conjointes sous-optimales ? Les auteurs n'abordent guère ces questions.

Cette dépendance mutuelle entre les données nécessaires à la construction de la solution et la solution elle même affaiblit considérablement l'applicabilité de cette méthode. En somme, cette méthode exacte produit une politique conjointe optimale sur l'ensemble des historiques générés. Szer and Charpillet [2006] a proposé également une version approximative de cette méthode. Cette dernière procède à l'échantillonnage des politiques conjointes potentielles utiles dans la génération des croyances individuelles. De façon identique au cas précédant, aucune garantie n'existe sur la qualité de la politique conjointe ainsi construite – autre que celle liée à l'ε-optimalité sur l'ensemble des historiques générés.

Regroupement des croyances équivalentes

L'approche proposé par Oliehoek et al. [2009] suggère de réduire le nombre de croyances individuelles considérées pour chaque agent $i \in \mathcal{I}$ en regroupant celles qui sont dites *equivalentes.*

L'approche proposé par Oliehoek et al. [2009] suggère de réduire le nombre de croyances individuelles considérées pour chaque agent $i \in \mathcal{I}$ en regroupant celles qui sont dites équivalentes.

Définition 7. *Deux croyances individuelles h^i et $\bar{h}^i$ sont dites* equivalentes *si :*

$$\forall_{h^{\neq i}}, \forall_s \ \mathbb{P}(s, h^{\neq i} \mid h^i) \quad = \quad \mathbb{P}(s, h^{\neq i} \mid \bar{h}^i) \tag{3.9}$$

En réduisant le nombre de croyances individuelles pour chaque agent $i \in \mathcal{I}$, l'on parvient à réduire le coût consacré à l'élagage des arbres de décisions dominées. Les auteurs surmontent le problème des garanties dont nous discutions précédemment en générant

l'ensemble des croyances individuelles possibles. Bien que la réduction du nombre des croyances individuelles soit une idée intéressante en théorie, en pratique elle n'a que peu d'impact.

En effet, la complexité de la résolution d'un DEC-POMDP dépend des deux opérations précédemment énoncées. Malheureusement, cette méthode n'a pas ou presque pas d'influence sur ces deux opérations. D'abord, la génération des politiques individuelles de chacun des agents à chaque horizon n'est en aucun cas influencée par la réduction du nombre de croyances individuelles. On pourrait éventuellement penser que la réduction du nombre de croyances individuelles aurait un impact sur le nombre de politiques individuelles préservées. Malheureusement, il n'en est rien. En effet, deux croyances individuelles équivalentes ont la même valeur quelle que soit la politique individuelle sélectionnée. Ainsi, éliminer des croyances individuelles ne peut en rien réduire le nombre de politiques individuelles préservées. En outre, le temps d'identification des politiques individuelles non dominées n'est pas significativement réduit. Ceci parce que le temps requis afin de regrouper les croyances individuelles équivalentes est non négligeable et peut s'avérer prohibitif pour le coût total de résolution du problème original. Les résultats expérimentaux compilés par les auteurs montrent cependant une amélioration du temps de calcul, d'une part, et des horizons de planification, d'autre part.

Compression de l'espace des politiques conjointes

L'encodage compacte des politiques conjointes est une autre alternative possible afin de combattre la croissance exponentielle de l'espace mémoire nécessaire pour la résolution des DEC-POMDPs. Boularias and Chaib-draa [2008] ont suggéré d'étendre une méthode de compression de l'espace des croyances introduite dans le cadre des POMDPs afin de compresser l'espace des politiques dans le cadre des DEC-POMDPs. Pour ce faire, les auteurs notent tout d'abord qu'une politique représentée sous forme d'un arbre de décisions peut de façon équivalente être représentée sous forme d'un ensemble d'historiques composés de paires d'actions et d'observations. Comme le montre la Figure 3.9, un arbre de décisions est composé d'un nombre fini de branches, chacune d'elles représente un historique.

Fort de cette observation, les auteurs définissent une matrice de correspondance entre un ensemble fini d'arbres de décisions et un ensemble fini d'historiques de longueurs finies et identiques. Ils montrent par la suite qu'il existe un sous-ensemble de ces historiques, suffisant pour encoder l'ensemble des arbres de décisions. C'est de la que la méthode tire son avantage par rapport à la représentation d'une politique sous la forme d'arbres de décisions. En effet, l'ensemble des évaluations des politiques se fait sur l'enco-

dage compact de ces politiques. Ainsi, si cet encodage est plus compact, les étapes d'évaluation et d'énumération des politiques sont significativement améliorées. Cela est en particulier vrai pour des horizons de planification relativement petits $\tau < 10$.

Néanmoins, il est nécessaire de garder en mémoire l'encodage sous forme d'arbres de décisions. C'est le point qui affaiblit cette méthode. En effet, lorsque l'horizon croît seules les coûts d'évaluation de la valeurs des politiques restent relativement réduits. À la différence, les coûts de d'énumération exhaustif des politiques sous la forme d'historiques vont toujours croître de façon doublement exponentielle avec l'horizon de planification. Or, en DEC-POMDPs, la complexité d'une mise à jour est largement dominée par l'étape d'énumération exhaustive.

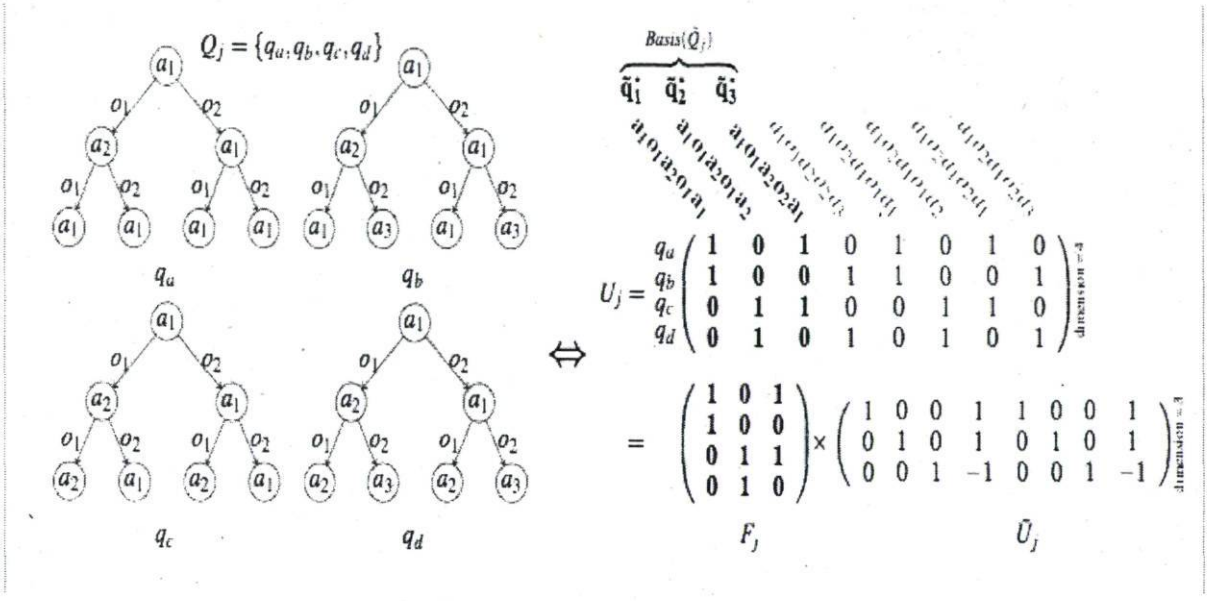


FIGURE 3.9 – Politiques sous-forme d'historiques et de matrices – extrait de [Boularias and Chaib-draa, 2008].

L'algorithme de programmation dynamique par compression de politiques selon l'acronyme anglo-saxon « dynamic programming with lossless policy compression » (DPLPC) est décrit Algorithme 18. Il s'agit essentiellement d'un algorithme de programmation dynamique, auquel les routines de d'évaluation et d'énumération sur les arbres de décisions ont été substituées à des routines d'évaluation et d'énumération sur des ensembles d'historiques.

Cette méthode souffre d'un certain nombre de points faibles. D'abord, le fait qu'il faille maintenir deux encodages équivalents, à savoir celui utilisant les arbres de décisions $\mathcal{Q}^i_{\tau:N}$ et celui faisant recours aux historiques de base $\mathbf{Basis}(H^i_{\tau:N})$ pour chaque agent $i \in \mathcal{I}$. C'est d'ailleurs cette remarque qui limite considérablement un usage systématique de cet encodage plus compacte. Les résultats expérimentaux compilés pour cet algorithme

Algorithme 18 Algorithme DPLPC – extrait de Boularias and Chaib-draa [2008].

procedure DPLPC
  Poser $\tau \leftarrow N$.
  répéter
    Générer de façon exhasutive $\mathcal{Q}^i_{\tau:N}$, pour tout $i \in \mathcal{I}$.
    Générer de façon exhaustive les bases de l'ensemble des historiques $H^i_{\tau:N}$:

$$\textbf{Basis}(H^i_{\tau:N}) \leftarrow A^i \times \Omega^i \times \textbf{Basis}(H^i_{\tau-1:N}) \tag{3.10}$$

    Évaluer les fonctions de valeurs sur les historiques de base $\textbf{Basis}(H^i_{\tau:N})$, pour tout $i \in \mathcal{I}$.
    Élaguer les arbres de décisions dominés des ensembles $\mathcal{Q}^i_{\tau:N}$, pour tout $i \in \mathcal{I}$.
    Supprimer les historiques hors base dans les ensembles $\textbf{Basis}(H^i_{\tau:N})$, pour tout $i \in \mathcal{I}$.
    Décrémenter $\tau \leftarrow (\tau - 1)$.
  jusqu'à $\tau = 0$
  Retourner $\mathcal{Q}_{0:N}$.
fin procedure

montrent une sensible amélioration des performances en comparaison des algorithmes de programmation dynamique exhaustive et celle de programmation dynamique à base de croyances.

3.2.4 Recherche heuristique

La recherche heuristique offre une méthode générale et alternative de résolution des problèmes combinatoires. L'une des heuristiques des plus utilisée en Intelligence Artificielle est l'heuristique nommée A*. Elle permet d'éviter l'énumération exhaustive de l'ensemble des solutions possibles d'un problème combinatoire. Pour ce faire, elle utilise une mesure approximative sur la qualité de la meilleure solution afin de guider la recherche vers celle-ci tout en éliminant définitivement certains sous-espaces entiers de solutions potentielles. En outre, l'heuristique A* offre parfois des garanties quant à la qualité de la solution retournée si la recherche s'achève. La solution retournée est optimale si la mesure approximative utilisée est optimiste par rapport à la qualité réelle de la solution optimale. De plus, il a été prouvé que cette heuristique évalue un nombre minimum de nœuds en comparaison à tout autre heuristique muni de la même mesure de qualité [Dechter and Pearl, 1985].

Szer et al. [2005] ont proposé le premier algorithme de recherche heuristique (MAA*) pour la résolution des DEC-POMDPs. Cet algorithme consiste à l'usage de l'heuristique A* dans l'espace des politiques conjointes muni d'une mesure optimiste sur la fonction de valeurs optimales. L'identification des mesures optimistes sur la fonction de valeurs optimales est assez facile. En effet, il suffit de considérer les fonctions de valeurs optimales des relaxations possibles des DEC-POMDPs, à savoir : POMDPs multi-agents (MPOMDPs) ou MDPs multi-agents (MMDPs). Les auteurs supposent que l'information sur l'état initial du système est disponible. Dans ce cas, la politique conjointe optimale peut se restreindre en un vecteur d'arbres de décisions, comme discuté ci-dessus. Afin de faciliter l'implémentation de cet algorithme, les auteurs utilisent la représentation sous forme de vecteurs d'arbres de décisions $\delta_{0:\tau} \equiv (\delta^1_{0:\tau}, \delta^2_{0:\tau}, \cdots, \delta^{|\mathscr{I}|}_{0:\tau})$, un arbre de décisions par agent.

Algorithme MAA*

L'algorithme MAA* développé par Szer et al. [2005] génère un arbre de recherche où les nœuds de profondeur $\tau = 0, 1, \cdots, N$ sont étiquetés par un vecteur d'arbres de décisions $\delta_{0:\tau}$. Contrairement à un algorithme de recherche exhaustive, MAA* ne développe qu'une partie des nœuds possibles. En effet, MAA* évalue chaque nœud feuille de l'arbre de recherche via une mesure optimiste, puis développe uniquement le nœud feuille dont la mesure est la meilleure selon les critères choisis. La Figure 3.10 illustre une portion de l'arbre de recherche développé par MAA*. Le calcul de la mesure optimiste $\hat{v}(s, \delta_{0:\tau})$ d'un nœud feuille $\delta_{0:\tau}$ considère deux estimés : d'une part, l'estimé exact $v(s, \delta_{0:\tau})$ du nœud $\delta_{0:\tau}$; d'autre part, l'estimé optimiste $\hat{v}(s, \Delta_{\tau+1:N})$ des compléments $\Delta_{\tau+1:N}$ possibles du nœud $\delta_{0:\tau}$ – de sorte que $\{\delta_{0:\tau}, \Delta_{\tau+1:N}\}$ constitue un vecteur d'arbres de décisions $\delta_{0:N}$ et tel que :

$$\hat{v}(s, \delta_{0:\tau}) \quad = \quad v(s, \delta_{0:\tau}) + \sum_{\bar{s}} \mathbb{P}(\bar{s}|s, \delta_{0:\tau}) \hat{v}(s, \Delta_{\tau+1:N}) \tag{3.11}$$

La Figure 3.11 illustre un arbre de décisions $\delta_{0:N}$ et ces différentes composantes $\delta_{0:\tau}$ (ellipse) et $\Delta_{\tau+1:N}$ encerclé.

Il est crucial que la mesure optimiste $\hat{v}(s, \Delta_{\tau+1:N})$ soit non seulement facilement calculable mais aussi proche que possible de la vrai mesure. La vrai mesure $v(s, \Delta_{\tau+1:N})$ du complément $\Delta_{\tau+1:N}$ du nœud $\delta_{0:\tau}$ correspond à une fonction de valeurs $v_{\tau+1:N}$ du DEC-POMDP à résoudre. Or, la fonction de valeurs optimales $\hat{v}_{\tau+1:N}$ du MMDP ou du MPOMDP sous-jacent à ce DEC-POMDP, est une mesure optimiste telle que $\hat{v}_{\tau+1:N}(s) \geqslant v(s, \Delta_{\tau+1:N})$, pour tout complément $\Delta_{\tau+1:N}$ et pour tout état $s \in S$. Ainsi, une mesure optimiste du nœud $\delta_{0:\tau}$ est donnée par :

$$\hat{v}(s, \delta_{0:\tau}) \quad = \quad v(s, \delta_{0:\tau}) + \sum_{\bar{s}} \mathbb{P}(\bar{s}|s, \delta_{0:\tau}) \hat{v}_{\tau+1:N}(\bar{s}) \tag{3.12}$$

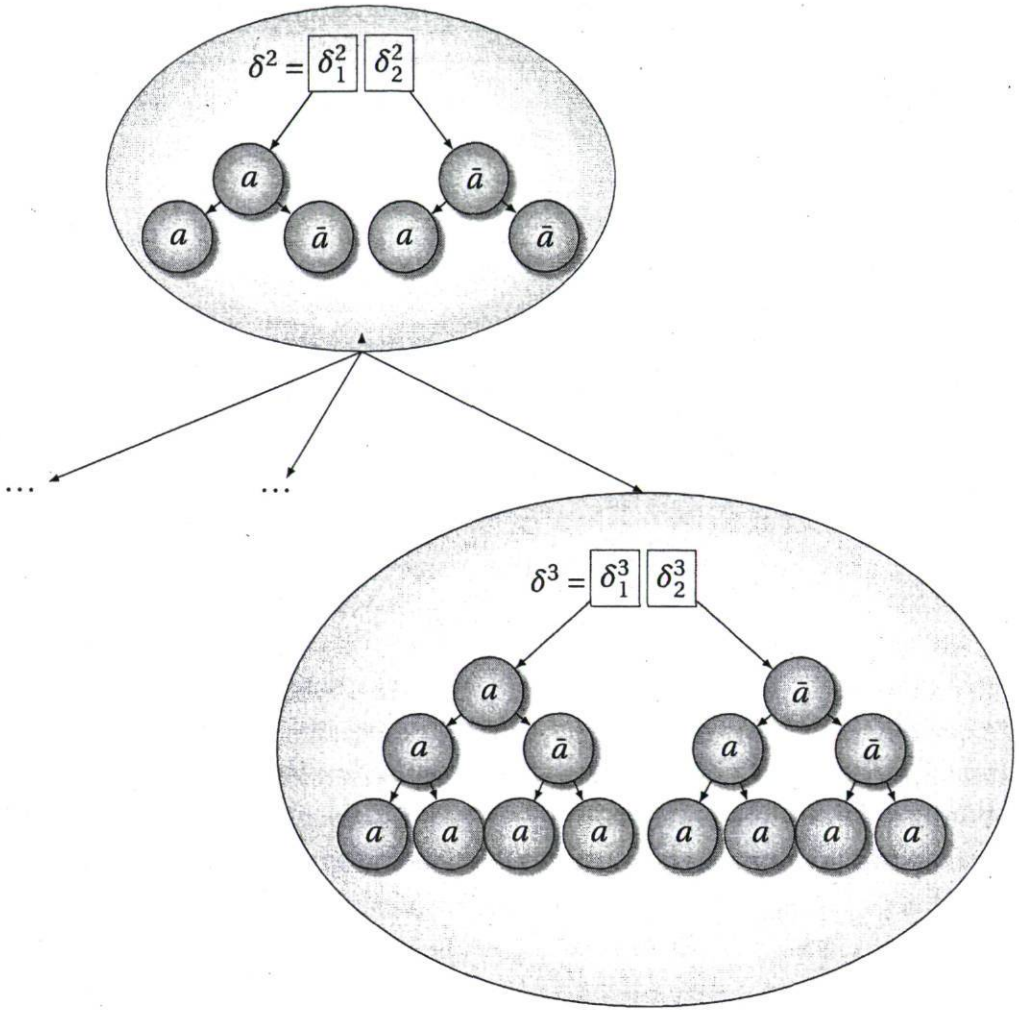


FIGURE 3.10 – Portion de l'arbre de recherche développé par l'algorithme MAA* montrant un vecteur de deux arbres de décisions. Deux agents contrôlent le processus de Markov, muni chacun de 2 observations individuelles $(\omega, \bar{\omega})$ et 2 actions individuelles $(a, \bar{a})$.

où $\hat{v}_{\tau+1:N}$ N est la fonction de valeurs optimales du MMDP ou MPOMDP sous-jacent au DEC-POMDP à résoudre.

De façon similaire aux méthodes exactes précédemment citée, MAA* est exponentielle en pire cas. En effet, bien qu'il ne développe qu'un seul nœud pour chaque profondeur τ, il doit évaluer l'ensemble des nœuds de profondeur τ afin de sélectionner le nœud à développer. Les performances de l'algorithme MAA* sont intimement liées à la proximité de la mesure optimiste utilisée par rapport à la fonction de valeurs optimales du DEC-POMDP à résoudre.

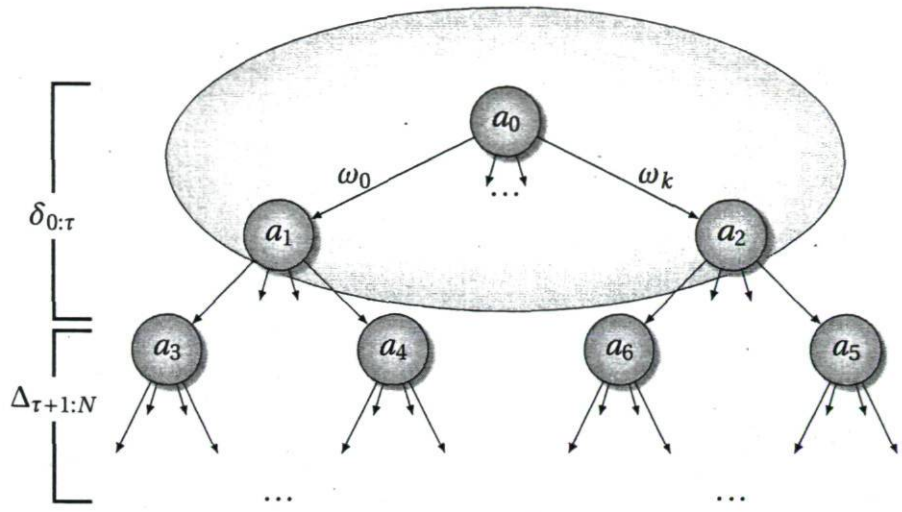


FIGURE 3.11 – Représentation d'un arbre de décisions du point de vue d'un algorithme de recherche heuristique de type MAA* : la portion entourée d'une ellipse correspond à l'arbre de décisions courant $\delta_{0:\tau}$ tandis que le reste correspond à son complément $\Delta_{\tau+1:N}$.

Algorithme MAA* généralisé (GMAA*)

Différentes mesures optimistes de la fonction de valeurs optimales d'un DEC-POMDP ont été proposées. Oliehoek et al. [2008] ont proposé un cadre différent d'utilisation de ces mesures. Comme nous l'avons noté plus haut ces mesures sont extraites des fonctions de valeurs optimales de l'ensemble des relaxations possibles des DEC-POMDPs. La plus simple des relaxations d'un DEC-POMDP est le MMDP sous-jacent, c'est à dire un DEC-POMDP privé de sa fonction d'observations et pour lequel le contrôle est centralisé. Dans la littérature des POMDPs [Littman et al., 1995], cette mesure optimiste est appelée Q-MDP et est donnée par :

$$\hat{v}^{\mathbf{MDP}}_{\tau+1:N}(a,b) = \sum_{s\in S} b(s)\cdot v^{\mathbf{MDP}}_{\tau+1:N}(s,a) \tag{3.13}$$

où $v^{\mathbf{MDP}}_{\tau+1:N}(s,a)$ est la fonction de valeurs optimales du MMDP sous-jacent au DEC-POMDP, lorsqu'à l'état s, l'action a est choisie. L'avantage de cette mesure optimiste est son coût extrêmement faible, et cela même pour des horizons très larges. Malheureusement, la distance entre cette mesure et la fonction de valeurs optimales du DEC-POMDP à résoudre est très large en général.

Afin de réduire la distance entre la mesure optimiste $\hat{v}$ et la fonction de valeurs optimales, il faut relaxer le moins possible le DEC-POMDP. Szer et al. [2005] ont proposé le MPOMDP sous-jacent au DEC-POMDP à résoudre comme relaxation raisonnable. Ils introduisent ainsi la mesure optimiste basée sur la fonction de valeurs optimales du MPOMDP

sous-jacent au DEC-POMDP à résoudre et donné par :

$$\hat{v}^{\mathbf{POMDP}}_{\tau+1:N}(a,b) \;=\; v^{\mathbf{POMDP}}_{\tau+1:N}(a,b) \tag{3.14}$$

où $v^{\mathbf{POMDP}}_{\tau+1:N}$ correspond à la fonction de valeurs optimales d'un MPOMDP à l'horizon$(\tau+1)$, et $v^{\mathbf{POMDP}}_{\tau+1:N}(a,b)$ représente la valeur de la croyance $b \in \mathscr{P}(S)$ lorsque l'on choisie l'action a. Encore une fois la relaxation d'un DEC-POMDP sous forme d'un MPOMDP consiste essentiellement à réduire le contrôle distribué en un contrôle centralisé. Cette mesure optimiste est bien entendue plus proche de la fonction de valeurs optimales du DEC-POMDP à résoudre, mais son coût est excessif en générale car la résolution d'un MPOMDP est extrêmement difficile.

La dernière mesure optimiste proposée, afin de guider au mieux la recherche dans l'espace des politiques conjointes d'un DEC-POMDP, est basée sur la reformulation du problème de résolution d'un DEC-POMDP comme celui de la résolution d'une série de jeux bayésiens. Dans ce contexte, Oliehoek et al. [2008] ont proposé un mesure optimiste notée $\hat{v}^{\mathbf{BG}}_{\tau+1:N}$ où le contrôle du DEC-POMDP est centralisé mais différé d'une étape. Ils établissent également une hiérarchie des mesures optimistes :

$$v^{*}_{\tau+1:N} \leqslant \hat{v}^{\mathbf{BG}}_{\tau+1:N} \leqslant \hat{v}^{\mathbf{POMDP}}_{\tau+1:N} \leqslant \hat{v}^{\mathbf{MDP}}_{\tau+1:N} \tag{3.15}$$

Le lecteur doit néanmoins garder à l'esprit que l'intérêt d'une mesure optimiste dans la recherche d'un solution d'un problème combinatoire dépend de deux facteurs antagonistes : d'une part, la mesure optimiste doit être aussi proche que possible de l'estimé exacte de la solution recherchée ; d'autre part, le coût de construction de cette mesure ne doit pas être prohibitive au point qu'au total le gain escompté ne soit pas au rendez-vous. Pour ces raisons, en pratique la mesure la plus utilisée pour la résolution DEC-POMDP reste encore celle issue du MMDP c'est à dire $\hat{v}^{\mathbf{MDP}}_{\tau+1:N}$.

Les algorithmes de recherche heuristique basés sur l'algorithme A* offrent un moyen efficace de construction de politiques conjointes d'un DEC-POMDP. À l'image de l'algorithme A*, l'algorithme de séparation et évaluation peut également être utilisé afin de construire des politiques conjointes d'un DEC-POMDP. La programmation mathématique offre un cadre de formulation du problème de construction de telles politiques et l'algorithme de séparation et évaluation y est utilisé afin de résoudre ce problème.

3.2.5 Programmation mathématique

Aras et al. [2007b] ont introduit un programme linéaire mixte afin de résoudre un DEC-POMDP. Cette approche est basée sur une représentation différente des politiques individuelles. En effet, contrairement au représentations précédemment introduites, les auteurs

se proposent de représenter une politique individuelle d'un agent sous forme séquentielle [Aras et al., 2007b]. Il s'agit, en d'autres termes, de voir une politique individuelle comme un ensemble d'historiques « d'actions et d'observations » pour chaque agent. L'objectif est ainsi d'identifier les séquences individuelles non dominées au travers d'un programme linéaire mixte en nombres entiers (MILP).

Bien que cette approche fasse appel à des algorithmes mathématiques longuement éprouvés de type séparation et évaluation, les résultats bien que sensiblement meilleurs que ceux produits par la programmation dynamique exhaustive ou à base de croyances, ne prouvent pas la supériorité de l'approche. L'une des raisons de ces résultats mitigés provient du fait que dans ce contexte, de nombreuses questions fondamentales restes sans réponses. Ces réponses seraient utiles afin de concevoir nos propres algorithmes de recherche heuristiques permettant d'exploiter pleinement les spécificités des DEC-POMDPs, y compris éventuellement au travers de cette représentation. À ce jour, le fait de changer l'encodage des politiques individuelles de la forme arborescente à la forme séquentielle n'a pas encore montré tout son intérêt. Il serait intéressant de réaliser une étude comparative plus exhaustive des deux encodages et cela indépendamment des méthodes qui les exploitent. Par exemple, quelles seraient les performances d'un programme dynamique utilisant un encodage séquentielle ? comment ce programme se différencie t-il par rapport à l'approche de programmation dynamique à base de croyances ? Autant de questions dont les réponses permettraient de clarifier les zones d'ombres quant à la supériorité de l'encodage séquentiel sur l'encodage arborescent.

3.3 Approximation des DEC-POMDPs à horizon fini

Compte tenu des difficultés liées soit à la complexité intrinsèque des DEC-POMDPs soit à la méconnaissance actuelle du problème, il semble plus prometteur de concentrer les efforts de résolution des DEC-POMDPs dans la direction des approches approximatives. Malheureusement, la complexité du problème de calcul d'une solution ε-optimale pour un DEC-POMDP reste NEXP-difficile, comme l'ont prouvé [Rabinovich et al., 2003]. Ce constat suggère que le seul moyen de résoudre des DEC-POMDPs de tailles raisonnables est de sacrifier la qualité de la solution afin de réduire considérablement la complexité du problème.

Rappelons qu'à la différence du sens strict donné au terme « approximation » en Informatique théorique, où il s'agit essentiellement d'algorithmes dont le résultat est muni d'une borne sur l'erreur. En Intelligence Artificielle, ce sens est légèrement affaibli, car par algorithmes approximatifs nous entendons tous les algorithmes pour lesquels la solution

retournée n'est ni optimale ni ε-optimale. Il s'agira dans certain cas d'heuristiques sans aucunes garanties, mais dont les résultats empiriques sont satisfaisants en comparaison aux algorithmes exacts ou à ε de l'optimum. Ces résultats empiriques extraits sur la base d'un ensemble d'exemples de problèmes de la littérature des DEC-POMDPs, permettent notamment de comparer ces algorithmes approximatifs les uns aux autres.

3.3.1 Algorithme de maximisation alternative

Nair et al. [2003] ont présenté une classe d'algorithmes, appelée « Recherche de politiques à base d'équilibre conjoint » (JESP), qui ne recherche pas une solution globalement optimale d'un DEC-POMDP, mais une solution localement optimale. La meilleure implémentation de cette famille d'algorithmes, DP-JESP, incorpore trois idées différentes détaillées Figure 3.12.

Algorithme JESP.

1. D'abord, la modification de la politique individuelle d'un agent se fait en considérant que les politiques individuelles des autres agents soient fixées.
2. Ensuite, la programmation dynamique est utilisée afin de générer l'ensemble des politiques individuelles candidates pour chaque agent comme dans les cas exacts. ♣
3. Enfin, seules les croyances individuelles accessibles sont considérées lors de l'élagage des politiques individuelles dominées.

FIGURE 3.12 – Algorithme JESP

Cette méthode permet d'accroître l'horizon de planification des problèmes à résoudre en comparaison aux approches exactes. Notamment parce que, l'élagage se fait indépendamment pour chaque agent sur un ensemble de croyances individuelles certes exponentielle mais largement inférieur à l'ensemble doublement exponentielle des croyances conjointes utiles si l'élagage est réalisé conjointement.

3.3.2 Programmation dynamique à base de croyances

Une approche de calcul d'une solution approximative d'un DEC-POMDP serait détendre les algorithmes utilisés dans le cadre des POMDPs au cadre des DEC-POMDPs, en particulier les algorithmes à base de croyances. Malheureusement, la majorité des approches à base de croyances introduites dans le cadre des POMDPs font appel aux croyances conjointes. Or, le problème du calcul d'une politique conjointe pour un DEC-POMDP en utilisant les croyances conjointes reste encore un problème ouvert. Ceci essentiellement parce qu'à l'exécution, un agent n'a pas accès aux croyances des autres agents. Ainsi, généraliser les algorithmes existant dans le cadre des POMDPs n'est pas trivial. Cette question est au centre de l'analyse formelle que nous proposons aux chapitres 5 êt 6.

Néanmoins, de nombreuses tentatives d'extension des approches à base de croyances ont vu le jour dans le cadre des DEC-POMDPs. Parmi d'autres, nous citerons l'approche introduite par Szer and Charpillet [2006] et utilisant les croyances individuelles plutôt que les croyances conjointes. Néanmoins, les méthodes parmi les plus efficaces suggèrent d'utiliser les croyances conjointes de l'ensemble des agents.

3.3.3 Programmation dynamique à mémoire limitée

Cette méthode est une extension de la programmation dynamique à base de croyances conjointes, comme dans le cadre des POMDPs [Lovejoy, 1991, Pineau, 2004, Smith and Simmons, 2005, Spaan and Vlassis, 2005]. À la différence qu'ici les croyances utilisées sont conjointes à l'ensemble des agents. Seuken and Zilberstein [2007b] argumentent à juste titre ce choix de la façon suivante : « Si lors du contrôle, les agents n'ont accès qu'à leurs observations individuelles durant la phase de planification, un contrôleur central peut autoriser les agents à partager leurs observations individuelles et obtenir ainsi une croyance conjointe à l'ensemble des agents ». Il reste cependant à résoudre la question de savoir comment cette croyance conjointe doit-elle être utilisée dans le cadre de la planification pour la résolution des DEC-POMDPs. En annexe à cette question, se pose celles encore ouvertes de savoir s'il est possible de planifier de façon optimale en utilisant les croyances conjointes ? et si oui comment ? Les auteurs n'apportent pas de réponses à ces questions. Ils se focalisent sur la résolution du problème de la complexité en mémoire des DEC-POMDPs en bornant le nombre maximum d'arbres de décisions (conjoints) à garder à chaque horizon $\tau = 0, 1, \cdots, N$.

La Figure 3.13 décrit symboliquement l'approche de programmation dynamique à mémoire limitée. Cette méthode à l'instar de l'ensemble des méthodes présentées jusqu'ici

procède de la façon suivante : d'abord elle génère de façon exhaustive l'ensemble des arbres de décisions $\mathcal{Q}^i_{\tau-1:N}$ possibles pour chaque agent $i \in \mathcal{I}$ en partant des arbres de décisions $\mathcal{Q}^i_{\tau:N}$ issue de l'horizon $\tau = 0, 1, \cdots, N$; ensuite, un nombre fini et borné K d'entre eux sont retenus comme les arbres de décisions possible pour l'horizon de planification courant.

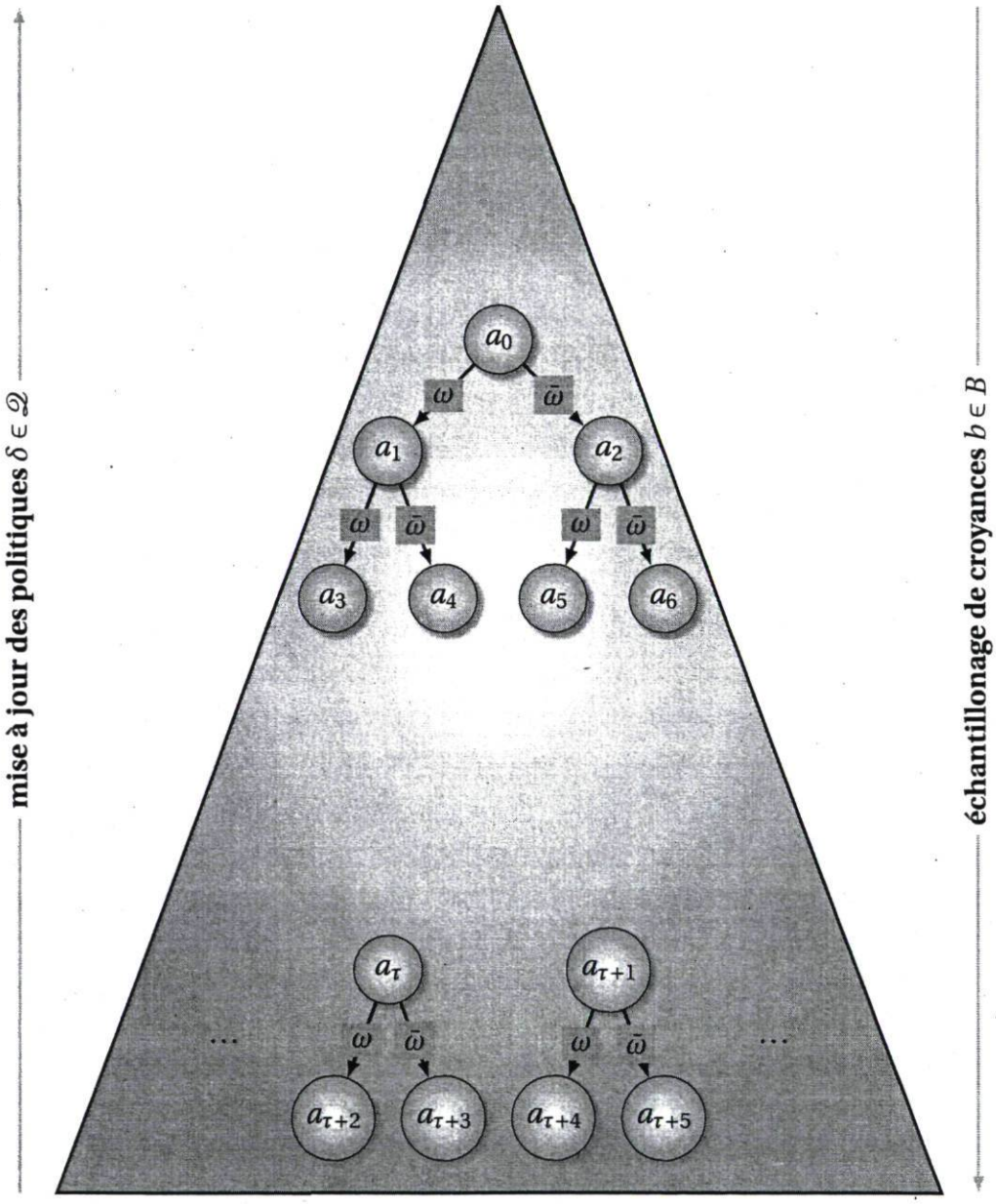


FIGURE 3.13 – Construction d'arbres de décisions – adapté de [Seuken and Zilberstein, 2007b].

La sélection de K arbres de décisions $\delta^i_{\tau-1:N}$ par agent est faite en identifiant les meilleures vecteurs d'arbres de décisions $\delta_{\tau-1:N}$ pour chaque croyance conjointe $b \in \mathcal{P}(S)$. C'est à dire les vecteurs d'arbres de décisions dont les valeurs sont maximales pour au moins

une croyance conjointe. Ces croyances conjointes sont sélectionnées comme l'indique la Figure 3.13. C'est à dire en collectant les traces d'exécution d'une politique conjointe quelconque du MPOMDP sous-jacent au DEC-POMDP à résoudre. En particulier les auteurs font appel à différentes politiques conjointes afin de sélectionner les traces utilisées comme croyances conjointes. Ainsi, l'approche utilise en quelque sorte une programmation dynamique à mémoire limitée.

L'algorithme de programmation dynamique à mémoire limitée selon l'acronyme anglais « memory bounded dynamic programming » (MBDP) est décrit Algorithme 19. Il se décompose en six étapes : la première correspond à l'initialisation c'est à dire la définition des actions individuelles $\mathcal{Q}^i_{N:N}$ qu'un agent $i \in \mathcal{I}$ pourrait exécuter à la dernière étape de prise de décision $\tau = N$. La second étape consiste à collecter l'ensemble des K croyances conjointes B_τ pour chaque horizon de planification $\tau = 0, 1, \cdots, N$. Vient par la suite, l'étape de génération exhaustive des arbres de décisions $\mathcal{Q}^i_{\tau-1:N}$ à partir des arbres de décisions $\mathcal{Q}^i_{\tau:N}$, pour chaque agent $i \in \mathcal{I}$. Afin de ne conserver que K arbres de décisions par agent, l'on considère l'ensemble des vecteurs d'arbres de décisions $\mathcal{Q}_{\tau-1:N}$ donné par le produit croisé $\prod_{i\in\mathcal{I}} \mathcal{Q}^i_{\tau-1:N}$. Chaque vecteur d'arbres de décisions $\delta_{\tau-1:N} \in \mathcal{Q}_{\tau-1:N}$ est alors évalué pour chaque croyance conjointe $b \in B_{\tau-1}$ comme suit :

$$v(b, \delta_{\tau-1:N}) = \sum_s b(s) v(s, \delta_{\tau-1:N}) \tag{3.16}$$

Nous sélectionnons ensuite par agent les K arbres de décisions individuelles qui composent les meilleurs vecteurs d'arbres de décisions. L'étape suivante permet de vérifier si nous disposons d'une solution complète ($\tau = 0$), si oui elle est retournée. Le cas échéant, l'horizon de planification est décrémenté ($\tau \leftarrow \tau - 1$) puis l'algorithme retourne à la troisième étape, comme décrit Algorithme 19.

Algorithme 19 Algorithme MBDP – adapté de Seuken and Zilberstein [2007b].

1: **procedure** MBDP
2: Poser $\tau \leftarrow N$, $\mathcal{Q}^i_{N:N} \leftarrow A^i$, pour tout agent $i \in \mathcal{I}$.
3: **répéter**
4: Générer l'ensemble des croyances conjointes $\{B_\tau\}_{\tau=0,1,\cdots,N}$.
5: Générer l'ensemble des arbres de décisions $\mathcal{Q}^i_{\tau-1:N}$, pour chaque agent $i \in \mathcal{I}$.
6: Élaguer les arbres de décisions dominés dans $\mathcal{Q}_{\tau+1:N}$, suivant $B_{\tau+1}$.
7: Décrémenter $\tau \leftarrow (\tau - 1)$.
8: **jusqu'à** $\tau = 0$
9: Retourner $\mathcal{Q}_{0:N}$;
10: **fin procedure**

Cette approche ne dispose malheureusement pas de garanties théoriques. En particulier, il est difficile de comparer la qualité de la solution ainsi construite avec une solution

optimale. En outre, la méthode d'échantillonnage des croyances conjointes ne prend pas en compte la nature du contrôle, à savoir le fait que le contrôle soit distribué. L'intérêt principal de cette approche réside dans sa capacité à passer à l'échelle du moins en comparaison aux autres approches approximatives. En effet, sa capacité à passer à l'échelle reste en réalité limité par le nombre d'actions ; d'observations ; et de politiques individuelles conservées. Lorsque par exemple le nombre de politiques individuelles à conserver dépasse cinq, l'algorithme est incapable de résoudre même les problèmes les plus insignifiants. Ce constat a conduit les auteurs à envisager une version approximative de cette approche elle même déjà approximative.

Les limites de passage à l'échelle dont peut souffrir l'algorithme approximatif MBDP peuvent trouver certaines de leurs origines dans le nombre d'observations individuelles par agent dont le problème dispose. En effet, MBDP procédant à l'énumération exhaustive des politiques individuelles de chacun de ses agents, et cela à chaque horizon. Ainsi, la complexité de cette opération est d'autant plus grande que le nombre d'observations individuelles est élevé. L'approximation de MBDP nommée IMBDP selon l'acronyme anglais « improved memory bounded dynamic programming » consiste donc à ne considérer qu'un sous-ensemble de ses observations individuelles. C'est à dire à réduire, du moins artificiellement, le nombre d'observations individuelles considérées par agent.

Se pose alors la question de savoir comment choisir les observations individuelles qui feront partie du sous-ensemble d'observations individuelles considérées lors de la phase de planification. Pour les sélectionner, les auteurs préconisent de procéder à un échantillonnage. Cette technique permet notamment de se concentrer sur les observations individuelles les plus probablement accessibles lors de la phase de contrôle. L'échantillonnage se fait de façon globale, c'est à dire que l'on échantillonne les observations conjointes, car la fonction d'observations est définie conjointement pour l'ensemble des agents. Puis, l'on extrait les observations individuelles sélectionnées pour chaque agent. Ces observations individuelles seront les seuls à intervenir dans le processus de génération exhaustive des politiques individuelles par agent, pour l'horizon suivant. La Figure 3.14 donne un aperçu de la méthode générale.

La spécificité de cette méthode est sa phase de génération des politiques individuelles pour l'horizon suivant. À la différence de l'algorithme MBDP, l'algorithme IMBDP construit chaque politique individuelle comme illustré Algorithme 20. D'abord une action individuelle est sélectionnée ; puis pour chacune des observations individuelles choisies précédemment, on affecte une politique individuelle de l'horizon courant ; pour les autres observations individuelles une politique par défaut leur est assignée. On parle alors de mise à jour partielle des politiques individuelles d'un agent pour l'horizon suivant. Comme la génération des politiques pour l'horizon suivant n'est exhaustive que sur un sous-ensemble

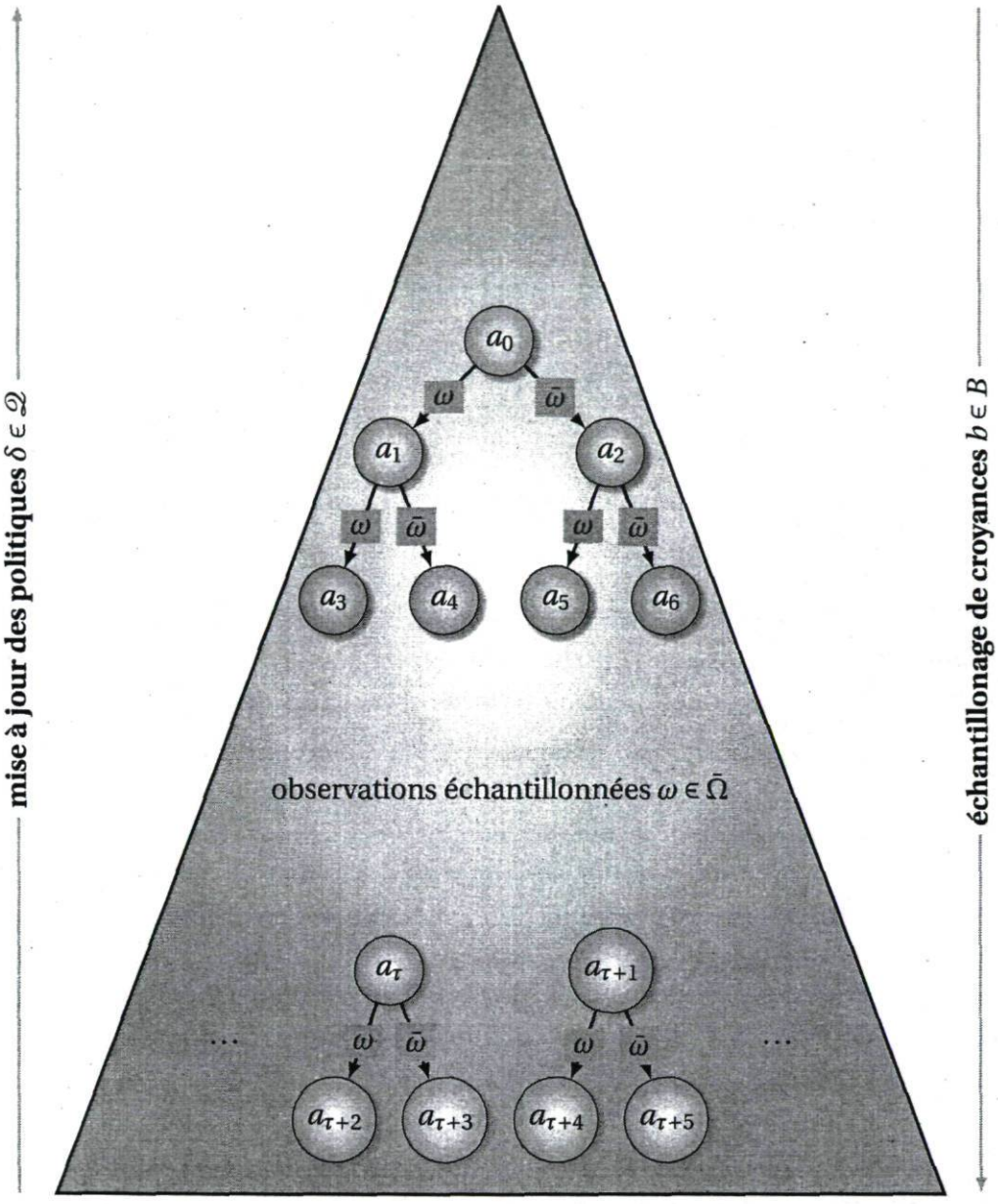


FIGURE 3.14 – Construction d'arbres de décisions partiels – adapté de [Seuken and Zilberstein, 2007a].

des observations conjointes, la complexité générale de cette opération est considérablement réduite en comparaison à MBDP. En contre partie, la qualité de la solution est en général réduite. Cette contre partie peut être bornée, comme le discutent les auteurs.

Les meilleurs algorithmes approximatifs à ce jour, MBDP, IMBDP et MBDP-OC tentent tant bien que mal de réduire la complexité de la programmation dynamique de deux manières : (1) en sélectionnant un nombre limité de politiques individuelles à conserver à chaque horizon de planification ; (2) en réduisant le rôle exponentiel des observations par

Algorithme 20 Algorithme IMBDP – adpaté de Seuken and Zilberstein [2007a].

procedure IMBDP
  Poser $\tau \leftarrow N$, $\mathcal{Q}^i_{N:N} \leftarrow A^i$, pour tout agent $i \in \mathcal{I}$.
  répéter
    Générer l'ensemble des croyances conjointes $\{B_\tau\}_{\tau=0,1,\cdots,N}$.
    Échantillonner les observations conjointes $\bar{\Omega} \subset \Omega$.
    Générer l'ensemble des arbres de décisions $\mathcal{Q}^i_{\tau-1:N}$, pour chaque agent $i \in \mathcal{I}$ suivant $\bar{\Omega}$.
    Élaguer les arbres de décisions dominés dans $\mathcal{Q}_{\tau+1:N}$, suivant $B_{\tau+1}$.
    Décrémenter $\tau \leftarrow (\tau - 1)$.
  jusqu'à $\tau = 0$
  Retourner $\mathcal{Q}_{0:N}$;
fin procedure

la sélection des plus probablement accessibles à l'exécution. Cependant, le choix du bon nombre K de politiques individuelles à conserver pour un DEC-POMDP quelconque n'est pas trivial. En outre, tenter d'ajuster ce nombre au cours de la planification peut accroître les coûts de planification. De plus, l'utilité de la sélection d'un sous-ensemble des observations conjointes n'est perceptible que pour certains problèmes. En effet, certaines applications peuvent voir la qualité de la solution chutée de façon significative, et cela simplement parce que certaines des observations individuelles n'ont pas été intégrées lors du processus de planification. Enfin, la complexité en pire cas de ces algorithmes reste exponentielle à chaque horizon de planification. Les solutions sont construites en temps exponentielle en fonction du nombre K de politiques individuelles conservées ; du nombre $maxObs$ d'observations conjointes considérées ; du nombre $|\mathcal{I}|$ d'agents, c'est à dire $\mathcal{O}(|S^2||A||\Omega| maxTrees^{maxObs|\mathcal{I}|+1})$.

La principale limitation de l'ensemble des algorithmes de programmation dynamique y compris les approches approximatives MBDP, IMBDP, et MBDP-OC, est l'opération de génération exhaustive. Cette opération consiste à générer de façon exhaustive l'ensemble des politiques conjointes possibles sachant un certain nombre de paramètres. Et cela bien qu'à l'issue de cette opération, la grande majorité des politiques individuelles générées soit trivialement dominées. À ce jour aucune approche n'a tenté d'exploiter cette observation, nous proposons le premier algorithme approximatif permettant de surmonter cela au Chapitre 5.

3.4 Résolution de DEC-POMDPs à horizon infini

Dans cette section, nous présentons les algorithmes de résolution exacte des problèmes de contrôle distribué des processus de Markov dans le cas où le nombre de décisions séquentielles à prendre est infini – on parle alors du cas à horizon infini $N = \infty$. De façon similaire aux algorithmes développés dans le cas fini, tout algorithme exacte de résolution d'un DEC- POMDP excèdera très vite les limites disponibles en mémoire et en temps. Encore une fois, l'intérêt principal de ces approches est de donner un cadre propice au développement d'algorithmes approximatifs efficaces.

Avant toute chose, signalons que les algorithmes exacts dont nous discuterons ont tous recours à la représentation des politiques sous la forme des machines à états finis – déterministes ou stochastiques. Ainsi, les algorithmes que nous exposerons sont plus précisément des approches ε-optimales que des approches exactes. Comme de plus les DEC-POMDPs à horizon infini sont indécidables, il est peut probable de trouver un algorithme optimal.

3.4.1 Itération de politiques

À l'instar des approches de programmation dynamique introduites dans le cadre fini, l'algorithme d'itération de politiques (qui est un algorithme de programmation dynamique) procède à chaque itération en deux étapes : d'une part, à la génération exhaustive des politiques conjointes possibles ; et d'autre part, à l'élagage des politiques individuelles dominées.

Bernstein et al. [2005] ont présenté le premier algorithme de résolution des DEC-POMDPs à horizon infini convergeant vers une solution ε-optimale. Cette technique fait appel aux machines à états finis stochastiques. Compte tenu des limites en mémoire dont souffrent les approches de résolution des DEC-POMDPs, plus la représentation de la politique est compacte plus performant est l'algorithme. Afin de réduire la taille des politiques tout en conservant leur qualité, les auteurs ont proposé d'ajouter à la politique individuelle de chaque agent un outil commun de corrélation.

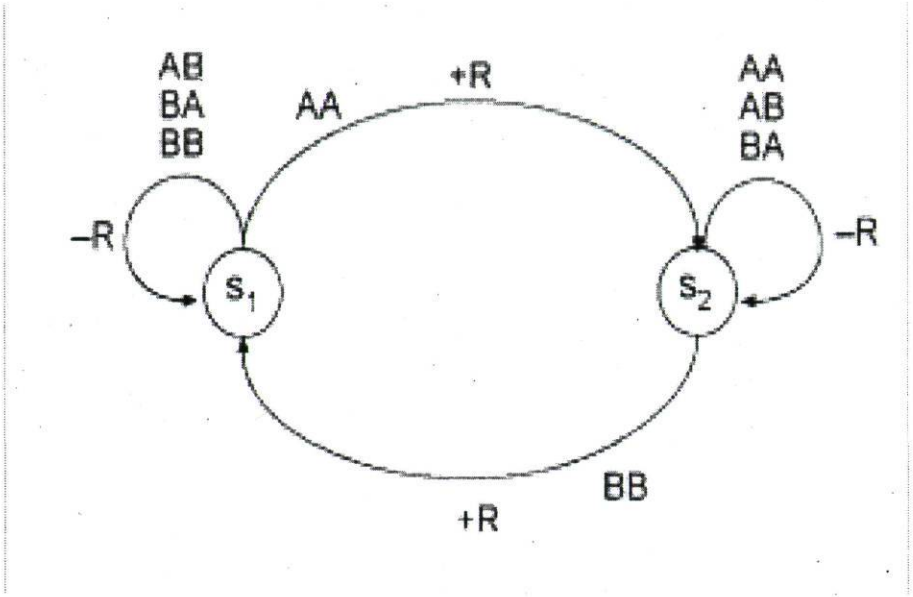


FIGURE 3.15 – Un DEC-POMDP dont la politique conjointe optimale à mémoire limitée requiert une corrélation – extrait de Seuken and Zilberstein [2008].

Utilité de la corrélation

La Figure 3.15 illustre un problème de contrôle distribué des processus décisionnels de Markov pour lequel la corrélation est importante. Dans cet exemple, on considère deux états, deux agents, deux actions individuelles (A et B) par agent et une observation conjointe. Parce qu'il n'y a qu'une observation conjointe, les agents ne peuvent distinguer les états du système. Ci-dessous nous comparons deux représentations à mémoire limitée des politiques conjointes :

1. Comme les agents ne connaissent pas l'état dans lequel ils sont, une politique déterministe sera arbitrairement mauvaise à long terme. Si les deux agents peuvent introduire de l'aléa dans leur stratégie, le mieux qu'ils puissent faire est de sélectionner soit l'action A soit l'action B suivant une distribution uniforme à chaque étape. Avec une récompense de $-R/2$ par étape cela conduit à une récompense à long terme de $-R/2(1-\gamma)$.
2. Si les agents corrèlent leurs politiques individuelles en utilisant une source commune d'aléa, la meilleure politique consistera à exécuter l'action conjointe AA avec une probabilité de $1/2$ et l'action conjointe BB avec une probabilité de $1/2$. Soit un gain espéré à long terme de 0. Ainsi, la différence entre l'usage ou non de la corrélation peut conduire à des divergences considérables de la qualité de la solution retournée. Ces divergences peuvent être arbitrairement grandes selon la valeur donnée au paramètre R.

Cet exemple souligne l'utilité de la corrélation lorsque la représentation de la politique est à mémoire limitée. Dans ce cadre spécifique de la planification pour le contrôle distribué, les auteurs introduisent un outil de corrélation sous la forme d'une machine à états finis additionnelle. L'outil de corrélation est un processus de création d'aléa qui n'a pas accès aux observations individuelles que perçoivent les agents durant la planification. Tout se déroule comme si tous les agents avaient accès à un signal de plus en provenance de l'outil de corrélation à chaque étape de la phase d'exécution, mais les agents ne peuvent échanger d'information à travers cet outil. Par définition un outil de corrélation est tuple (C, η), où :

- C est un ensemble d'états finis.
- η: $C \to \Delta C$ est une fonction de transitions.

La définition d'une politique individuelle doit être étendue afin d'y inclure l'outil de corrélation. L'action individuelle sélectionnée par un agent et la fonction de transitions de sa machine dépendra désormais du signal c en provenance de l'outil de corrélation. À chaque étape, l'outil de corrélation transite vers un nouvel état et tous les agents peuvent percevoir cet état. La fonction de valeur d'une politique conjointe représentée sous forme d'une machine à états finis et muni d'un outil de corrélation est donnée par : pour tout état $s \in S$, toute politique conjointe $\delta \in \mathcal{Q}$, et un état de l'outil de corrélation $c \in C$,

$$v(s,\delta,c) = \sum_{a} \mathbb{P}(a|\delta,c)\left[R(s,a) + \gamma \sum_{\bar{s}\omega\delta c'} \mathbb{P}(\bar{s},\omega|s,a)\mathbb{P}(\delta'|\delta,a,\omega,c)\mathbb{P}(c|c')v(\bar{s},\delta,c')\right]$$

où :

- $\mathbb{P}(a|\delta,c) := \alpha(x_0,a,c)$, rappelons que x_0 est l'état initial de la machine à états finis δ et $\alpha(x_0,a,c)$ est sa fonction de transition incorporant le signal c.
- $\mathbb{P}(\bar{s},\omega|s,a) := T(\bar{s}|s,a)O(\omega|s,a)$ est la fonction de transition du système définie à partir des fonctions de transitions T et d'observations O.
- $\mathbb{P}(\delta'|\delta,a,\omega,c) := \eta(x,\omega,c,x')$ est la probabilité de transiter de l'état x représentant δ à l'état x' représentant δ' après perception conjointe de l'observation conjointe ω.
- $\mathbb{P}(c|c') := \eta(c,c')$ illustre la fonction de transition de l'outil de corrélation.

Algorithme d'itération de politiques

Comme nous l'avons souligné l'algorithme d'itération de politiques (DEC-PI) est avant tout un algorithme de programmation dynamique. Ainsi, à chaque itération de cet algorithme il procède d'une part à la génération exhaustive des politiques individuelles de chaque agent. Dans la cadre spécifique de l'algorithme d'itération de politiques, il s'agit d'ajouter $|A^i||X_t^i|^{|\Omega^i|}$ états à la machine à états fini composée des états X_t^i, pour tout agent

$i \in \mathscr{I}$, et toute étape τ. De plus, l'algorithme procède à une phase d'élimination des états individuels inutiles – on parlera de transformation de la machine à états finis. Cette phase est accomplie par l'intermédiaire d'un programme linéaire garantissant soit que la machine à états finis résultant de la transformation soit de taille réduite et de qualité identique soit de meilleure qualité et de taille identique. Le programme mathématique à résoudre consiste à rechercher la meilleure machine à états finis telle que : sachant deux machines D et C à états finis muni chacune d'un outil de corrélation avec pour états respectifs Q et R, nous disons qu'une machine à états finis transformée de C à D préserve la qualité s'il existe deux fonctions $f_i : Q_i \to \triangle R_i$ pour chaque agent i et $f_c : Q_c \to \triangle R_c$ telles que :

Comme nous l'avons souligné, l'algorithme d'itération de politiques (DEC-PI) est avant tout un algorithme de programmation dynamique. Ainsi, à chaque itération de cet algorithme, on procède d'abord à la génération exhaustive des politiques individuelles de chaque agent. Dans la cadre spécifique de l'algorithme d'itération de politiques, il s'agit d'ajouter $|A^i||X_\tau^i|^{|\Omega^i|}$ états à la machine à états fini composée des états X_τ^i, pour tout agent $i \in \mathscr{I}$, et toute étape τ. Ensuite, l'algorithme procède à une phase d'élimination des états individuels inutiles – on parlera de transformation de la machine à états finis. Cette phase est accomplie par l'intermédiaire d'un programme linéaire garantissant que la machine à états finis résultant de la transformation soit de taille réduite et de qualité identique ; ou de meilleure qualité et de taille identique. Le programme mathématique à résoudre consiste à rechercher la meilleure machine à états finis telle que : sachant deux machines D et C à états finis, toutes deux munis d'un outil de corrélation avec pour états respectifs Q et R, nous disons qu'une machine à états finis transformée de C à D préserve la qualité s'il existe deux fonctions $f_i : Q_i \to \triangle R_i$ pour chaque agent i et $f_c : Q_c \to \triangle R_c$ telles que :

$$v(s,\delta) = \sum_r \mathbb{P}(r|\delta) v(s,r), \quad \forall s \in S,\ \delta \in Q \tag{3.17}$$

L'algorithme d'itération de politiques est décrit Algorithme 21. L'algorithme termine à l'itération τ si $\gamma^{\tau+1} \cdot \frac{|R_{\max}|}{1-\gamma} > \varepsilon$, où $R_{\max}$ représente la récompense immédiate maximale. Intuitivement, l'algorithme exploite la décroissance résultant du facteur de décompte γ. En effet, à une certaine itération τ les récompenses cumulées au-delà sont négligeables.

Cet algorithme jette le trouble sur certain nombre de questions. L'une des motivations de l'introduction de l'outil de corrélation est sa supériorité apparente sur l'exemple proposé. Néanmoins, les auteurs n'apportent aucune information sur l'existence ou non d'une politique déterministe de même qualité. L'approche suivante est en ce sens rassurante, car elle se propose de construire une politique ε-optimale dans l'espace des politiques conjointes déterministes. Malheureusement, ces deux approches à horizon infini sont extrêmement difficiles à comparer. D'abord parce qu'elles n'utilisent pas la même re-

Algorithme 21 Algorithme DEC-PI – adapté de Bernstein et al. [2009].

1: **procedure** DEC-PI
2: Poser $\tau \leftarrow 0$, initialiser δ_0 et C.
3: **répéter**
4: Evaluer la machine à états finis δ_τ muni de l'outil de correlation C.
5: Générer de façon exhaustive les nouveaux états à ajouter à δ_τ de sorte à créer $\delta_{\tau+1}$.
6: Incrémenter $\tau \leftarrow (\tau + 1)$.
7: **jusqu'à** $\gamma^{\tau+1} \cdot \frac{|r_\infty|}{1-\gamma} > \varepsilon$
8: Retourner δ_τ.
9: **fin procedure**

présentation des politiques. Ensuite parce que contrairement au cas fini où il est possible de mesurer la qualité de la solution à chaque horizon, dans le cas infini la qualité de la solution par horizon n'a pas de sens. Enfin, même en supposant que les deux méthodes utilisent des machines à états finis, et en basant la comparaison sur le nombre d'états des machines construites, il est fort probable de biaiser la comparaison. En effet, si l'une des méthodes utilise des machines stochastiques et l'autre des machines déterministes, le nombre d'états ne constitue pas un point de comparaison, car pour un état extrait d'une machine stochastique, il faut bien souvent y associer plusieurs états d'une machine déterministe. Le seul critère reste donc la solution dont la valeur est la plus grande.

3.4.2 Recherche heuristique

Szer and Charpillet [2005] ont proposé d'appliquer la méthode générale de recherche heuristique afin de résoudre des DEC-POMDPs à horizon infini. Cette méthode s'inspire de l'algorithme MAA* introduit dans le cas fini. À la différence de MAA*, les auteurs font usage des machines à états finis déterministes afin de représenter les politiques individuelles des agents. Cet algorithme recherche la meilleure politique conjointe pour un nombre prédéterminé d'états. Il s'agit alors d'un algorithme à mémoire limitée, comme illustré à la Figure 3.16.

L'idée de cette approche est de rechercher les assignations d'actions aux états et d'observations aux arcs permettant de construire une machine de valeur maximale. Cette recherche se fait de façon incrémentale, en rajoutant un élément à la fois comme l'illustre la Figure 3.17.

Plus formellement, il s'agit de définir progressivement les fonctions qui caractérisent

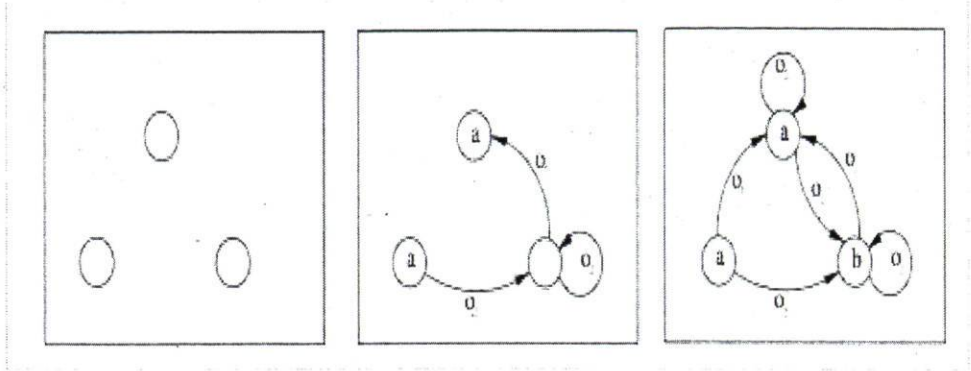


FIGURE 3.16 – Trois étapes de la construction d'une machine à trois états. à gauche : machine vide ; Au milieu : la machine partiellement complétée ; à droite : la machine complètement définie.

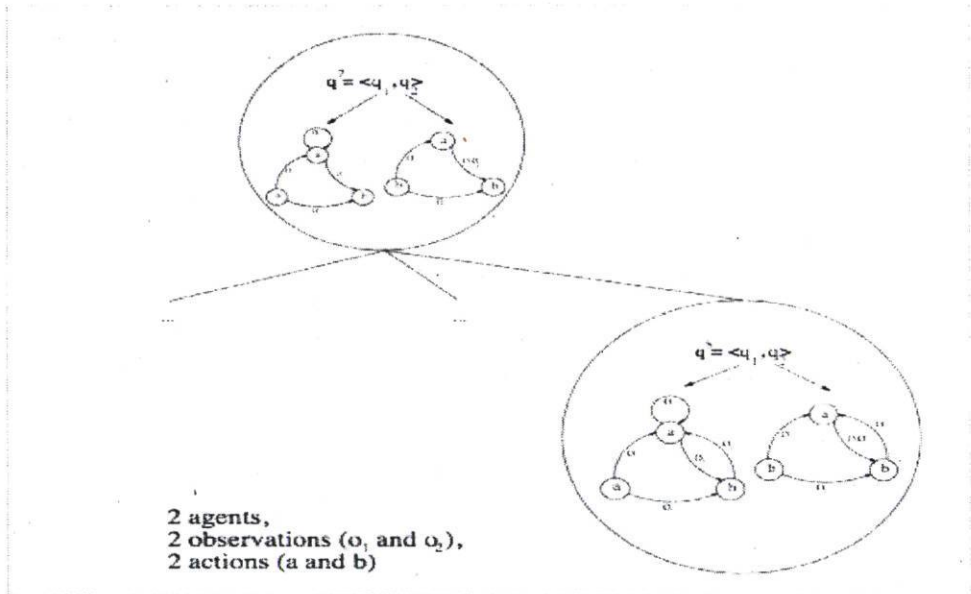


FIGURE 3.17 – Une section de l'arbre de recherche montrant les politiques conjointes et partielles de deux agents, l'une ayant une contrainte en plus que l'autre.

les machines $\delta^i \equiv (X^i, \alpha, \textbf{successeurs}, x_0^i)$ à états finis de chaque agent $i \in \mathscr{I}$. L'astuce est de supposer qu'initialement ces fonctions peuvent prendre toutes les valeurs possibles ; puis de progressivement contraindre les valeurs admissibles ; jusqu'à ce qu'il n'y ait plus qu'une seule assignation possible. Considérons pour ce faire les deux fonctions suivantes :

– la fonction des actions :

$$\Lambda(x) = \begin{cases} \{\alpha(x)\}, & \text{si } \alpha \text{ est définie sur } x \\ A^i, & \text{sinon.} \end{cases}$$

– la fonction de transitions :

$$\Pi(x,\omega) = \begin{cases} \{\eta(x,\omega)\}, & \text{si } \eta \text{ est définie sur } (x,\omega) \\ \Omega, & \text{sinon.} \end{cases}$$

Ainsi, il est possible d'évaluer la politique conjointe partielle de l'ensemble des agents comme suit :

$$v(s,x,\delta) = \max_{a\in\Lambda(x)} \left\{ R(s,a) + \gamma \left[\sum_{\bar{s}\omega} \mathbb{P}(\bar{s},\omega|s,a) \max_{x'\in\Pi(x,\omega)} v(\bar{s},x',\delta) \right] \right\} \tag{3.18}$$

Cette équation correspond à l'équation d'évaluation d'une politique dans le cas où l'ensemble des agents a accès à l'ensemble des observations à l'exécution. En ce sens, il s'agit d'une borne supérieure sur la valeur exacte de la politique conjointe optimale d'un DEC-POMDP.

Il est dès lors possible de définir un MDP équivalent à partir d'un DEC-POMDP et d'une politique sous forme d'une machine à états finis déterministe. Pour y parvenir, il suffit de définir le MDP augmenté $(\hat{S}, \hat{A}, \hat{T}, \hat{R})$ donné par :

1. $\hat{S} := \prod_{i\in\mathcal{I}} X^i \times S$,
2. $\hat{A} := \prod_{i\in\mathcal{I}} A^i \times \prod_{\omega^i\in\Omega^i} X^i$,
3. $\hat{T}(\langle s,x\rangle, \langle a, \textbf{successeurs}\rangle, \langle \bar{s},x'\rangle) := T(\bar{s}|s,a) \cdot \sum_{\omega\in\Omega:\ \eta(x,\omega,x')=1} O(\omega,\bar{s}|s,a)$,
4. $\hat{R}(\langle s,x\rangle, \langle a, \textbf{successeurs}\rangle) := R(s,a)$.

La résolution d'un tel MDP peut se faire selon les techniques classiques telles que la programmation dynamique, aboutissant à une fonction de valeurs sur l'espace augmenté des états $\hat{S}$, et l'équation de Bellman correspondante suivante :

$$\hat{v}(s,x,\delta) = \max_{a\in\Lambda(x)} \left\{ R(s,a) + \gamma \left[\sum_{\bar{s}\omega} \mathbb{P}(\bar{s},\omega|s,a) \max_{x'\in\Pi(x,\omega)} \hat{v}(\bar{s},x',\delta) \right] \right\} \tag{3.19}$$

$$\hat{v}(s_0,\delta) = \max_{x\in X} \hat{v}(s_0,x,\delta) \tag{3.20}$$

Cette fonction de valeurs peut être utilisée comme heuristique admissible dans le cadre de la résolution des DEC-POMDPs à horizon infini, pour une politique sous forme d'une machine à états finis et à mémoire limitée. Il apparaît en effet qu'au fur et à mesure que les fonctions $\Lambda(\cdot)$ et $\Pi(\cdot,\cdot)$ sont contraintes, la fonction de valeurs $\hat{v}$ décroît jusqu'à atteindre la fonction de valeurs exactes de le machine à états finis de valeur maximale.

Cette approche est très originale et à montrer que pour des machines à états finis de même dimension (nombre d'états), une machine déterministe pouvait obtenir une solution de qualité aussi bonne voire meilleure que les machines à états finis stochastiques préconisées par l'approche précédente. Néanmoins, il n'existe toujours pas de preuve théorique qui prouve l'existence systématique d'une machine déterministe à états finis au

moins d'aussi bonne qualité que toute machine stochastique à états finis. Nous notons un seul inconvénient de taille à cette approche : la nécessité de prédéfinir la dimension de la politique à priori. Cette remarque n'est pas propre à cette méthode, c'est un problème récurrent des méthodes à mémoire limité. Comment déterminer une dimension pas trop grande afin que nous puissions calculer la solution en temps limité et la mémoriser aisément. Et cela tout en garantissant qu'elle soit pas inutile en pratique. Par exemple, on pourrait considérer des machines munis d'un grand nombre d'états. Pour y parvenir, de nombreuses approches approximatives ont été introduites.

3.5 Approximation de DEC-POMDPs à horizon infini

Les méthodes ε-optimales discutées ci-dessus souffrent du fait qu'elles sont très vite limitées en mémoire, comme c'est le cas pour les approches ε-optimales dans le cas fini. À la différence de la stratégie de recherche opérée par les chercheurs en POMDPs, et consistant essentiellement à transférer les connaissances acquises dans le cas fini afin de résoudre le cas infini, en DEC-POMDPs, la démarche est différente. Le cas fini et infini sont très largement considérés – à mon avis à tord – comme deux cas significativement différents. De sorte que les travaux qu'ils soient exacts ou approximatifs produits pour le cas fini ne sont généralement pas étendus au cas infini. L'argument serait, bien qu'il en soit de même en POMDPs, que ces deux cas font appel à des représentations différentes des politiques ; ou encore, que toute tentative d'extension serait non triviale. En somme les méthodes approximatives introduites pour la résolution des DEC-POMDPs à horizon infini sont assez éloignées de celles proposées à horizon fini.

3.5.1 Itération de politiques à mémoire limitée

La première approche proposée pour la résolution approximative des DEC-POMDPs à horizon infini, s'inspire de l'approche exacte d'itération de politiques. Il s'agit à l'origine d'une méthode introduite dans le cadre des POMDPs et étendue au cadre des DEC-POMDPs. Elle permet de construire une politique sous contrainte de mémoire limitée [Bernstein et al., 2005]. Similaire à l'approche précédente, elle consiste en l'optimisation des valeurs d'un certain nombre de paramètres d'une machine stochastique à états finis et de dimension fixée. La différence essentielle avec l'algorithme d'itération de politiques précédemment discuté réside dans la dimension prédéfinie de la politique. Seules les distributions de probabilités, les actions associées aux états et leurs distributions de probabilités, peuvent changer.

L'algorithme dit d'itération de politiques à mémoire limitée commence ainsi avec une distribution arbitraire sur l'ensemble des K états de la machine stochastique à états finis. Cette dernière est initialement de valeur quelconque. La machine conjointe munie de l'outil de corrélation est alors améliorée par une opération de mise à jour dite à mémoire limitée, jusqu'à ce qu'elle atteigne un optimum local. À chaque itération de l'algorithme, un état x^i de la machine de l'agent $i \in \mathcal{I}$ est choisi. L'amélioration d'une machine à individuelle se fait en modifiant les paramètres de la distribution de probabilités conditionnelles $\mathbb{P}(a^i, \delta'^i | \delta^i, \omega^i, c)$ pour tout observation individuelle $\omega^i \in \Omega^i$. Cela est réalisé grâce à un programme linéaire assurant que les nouveaux paramètres améliorent la qualité de la machine conjointe pour chaque état du système ; chaque état de chaque machine individuelle ; et chaque état de l'outil de corrélation. Une fois les nouveaux paramètres identifiés, une nouvelle itération peut débuter. L'amélioration de l'outil de corrélation est réalisée de façon similaire.

Bernstein et al. [2005] ont prouvé que la mise à jour à mémoire limitée, appliquée à une machine individuelle ou un outil de corrélation, produit une machine conjointe de qualité au moins aussi bonne que la précédente, pour toute distribution de probabilités sur l'état initial. Ce résultat induit une propriété de croissance monotone de la qualité de la politique construite par la mise à jour à mémoire limitée. Malheureusement, l'algorithme retourne un optimal local, car l'optimisation se fait indépendamment pour chaque machine individuelle.

3.5.2 Programmation non-linéaire

L'approche de programmation non linéaire vise à pallier la principale limite de l'algorithme d'itération de politiques à mémoire limitée à savoir la qualité des solutions retournées. Amato et al. [2007] ont introduit un programme non linéaire capable de trouver une solution optimale pour une dimension prédéfinie. Afin d'y parvenir, les auteurs proposent d'améliorer conjointement les politiques individuelles des agents. En plus du programme linéaire initial, ils font également usage de la valeur de chaque nœud de la machine à améliorer dans chacun des états $s \in S$, $v(x^1, x^2, \cdots, x^{|\mathcal{I}|}, s)$. Pour s'assurer des la validité des valeurs, des contraintes non linéaires représentant les équations de Bellman sont ajoutées au programme. Des contraintes additives sont nécessaires afin de garantir que les politiques individuelles soient indépendantes. C'est à dire que les actions aux nœuds et les transitions locales ne dépendent que des informations disponibles localement, pour chaque agent. Après réduction de la complexité de la représentation, la nouvelle formulation correspond à un programme linéaire sous contraintes quadratiques noté QCLP. Malheureusement, les QCLPs sont bien plus difficiles à résoudre que des programmes linéaires. La description complète du QCLP est donné Algorithme 3.18.

Programme non-linéaire.

- Soient les variables $\zeta(\bar{x}, a, x, \omega)$ et $\xi(x, s)$
- Maximiser $\sum_s \mathbb{P}(s|h_0)\xi(x, s)$, sachant les contraintes,

1. $\forall x, s, \xi(x, s)$

$$\sum_a \left[\left(\sum_{\bar{x}} \zeta(\bar{x}, a, x, \omega) \right) R(s, a) \right.$$
$$\left. + \gamma \sum_{\bar{s}} T(\bar{s}|s, a) \sum_{\omega} O(\omega|\bar{s}, a) \sum_{\bar{x}} \zeta(\bar{x}, a, x, \omega) \xi(\bar{x}, \bar{s}) \right]$$

2. $\forall x, \omega,$

$$\sum_{\bar{x}, a} \xi(\bar{x}, a, x, \omega) = 1 \tag{3.21}$$

3. $\forall x, \omega, a,$

$$\sum_{\bar{x}} \zeta(\bar{x}, a, x, \omega) = \sum_{\bar{x}} \zeta(\bar{x}, a, x, \omega^i) \tag{3.22}$$

4. $\forall \bar{x}, a, x, \omega,$

$$\zeta(\bar{x}, a, x, \omega) \geqslant 0 \tag{3.23}$$

♣

FIGURE 3.18 – Programme non-linéaire.

Il est à noté que cet algorithme est similaire dans son objectif (déterminisme d'une solution optimale mais à mémoire limitée) que l'approche heuristique proposée par Szer and Charpillet [2005]. La différence entre ces deux approches réside dans l'espace de recherche. L'approche heuristique effectue sa recherche dans l'espace des politiques déterministes, bien qu'encore une fois il n'y a aucune preuve théorique de l'optimalité de ses politiques en générale. L'approche de programmation non linéaire quant à elle effectue sa recherche dans l'espace des politiques stochastiques. Bien que nous soyons sur d'y trouver la politique optimale, l'espace des politiques stochastiques est beaucoup trop vaste pour assurer le passage à l'échelle de cette approche. Les auteurs argumentent que cela est un avantage lorsque la politique conjointe recherchée est à mémoire limitée. Quand bien même cela serait vrai, il faut remarquer que la dimension réelle d'une politique stochastique à n nœuds n'est en rien comparable à celle d'une politique déterministe à n nœuds.

3.6 Sous classes des DEC-POMDPs

Cette section aborde des sous-classes des problèmes de contrôle distribué des processus décisionnels de Markov partiellement observables. L'une des principales sources de la complexité de ces problèmes est l'impossibilité des agents à communiquer aux autres agents leurs informations individuelles. De nombreux modèles permettant de prendre en compte différentes conditions ou protocoles de communication ont vu le jour ainsi que quelques algorithmes de résolution de ces modèles. Nous discuterons essentiellement d'une sous-classe qui s'il y avait communication permettrait d'avoir connaissance conjointe de l'état sous-jacent de système.

3.6.1 Processus décisionnels de Markov distribués

Si à chaque étape de prise de décisions l'ensemble des agents a accès à l'ensemble des perceptions de l'ensemble des agents, et que cela permet d'avoir connaissance de l'état sous-jacent du système, nous disons qu'il y a observabilité conjointe totale. Il existe de nombreuses situations où cela peut se produire. Par exemple, les satellites de surveillance aérien devant couvrir conjointement une région, dispose localement d'une connaissance partielle sur l'état de la région mais d'une connaissance conjointe complète de l'état de la région. Nous appelons un problème de contrôle distribué des processus décisionnels de Markov partiellement observables muni de la propriété d'observabilité conjointe totale, une problème de contrôle distribué des processus décisionnels de Markov conjointement complètement observable noté DEC-MDP :

$$O(\omega|\bar{s},a)>0 \quad \Rightarrow \quad \mathbb{P}(\bar{s}|\omega)=1 \tag{3.24}$$

Cadre formel

Un DEC-MDP est un formalisme multi-agents du contrôle distribué des processus décisionnels de Markov conjointement complètement observables. De façon similaire aux DEC-POMDPs, chaque agent dispose de son propre ensemble d'actions et d'observations, à la différence que bien qu'inaccessible à l'exécution, l'information conjointe des l'ensemble des agents fournie une connaissance sur l'état du système. à noter qu'il ne s'agit pas explicitement de communication des observations individuelles des uns aux autres, car cela remettrait en cause le contrôle distribué. Non, les agents n'ont toujours pas accès aux informations locales des autres agents du moins à l'exécution.

Définition 8. *Un DEC-MDP est un tuple* $(S, A, \Omega, T, R, O, \mathscr{I})$ *donné par :*

1. *L'ensemble fini S des états du système.*
2. *L'ensemble fini* $A \equiv A^1 \times A^2 \times \cdots \times A^{|\mathscr{I}|}$ *des actions conjointes des agents, chaque agent* $i \in \mathscr{I}$ *dispose de son propre ensemble d'actions individuelles* A^i.
3. *L'ensemble fini* $\Omega \equiv \Omega^1 \times \Omega^2 \times \cdots \times \Omega^{|\mathscr{I}|}$ *d'observations conjointes de l'ensemble des agents, chaque agent* $i \in \mathscr{I}$ *dispose de son propre ensemble fini d'observations individuelles* Ω^i.
4. *La fonction de transitions* $T(\bar{s}|s,a)$ *représentant la probabilité de transiter vers l'état* $\bar{s}$ *après avoir exécuter l'action a à l'état s.*
5. *La fonction de récompenses ou de coûts* $R(s,a)$ *définit un modèle de récompenses (resp. coûts) du système perçus lorsque dans l'état conjoint s l'ensemble des agents exécutant conjointement l'action a et perçoivent conjointement un signal réel.*
6. *La fonction d'observations* $O(\omega|\bar{s},a)$ *décrivant la probabilité que l'ensemble des agents perçoive l'observation conjointe ? lorsque dans l'état s les agents exécutent conjointement l'action a et transitent dans l'état* s^-. *Cette fonction est telle que si as* $O(\omega|\bar{s},a) > 0$ *alors il existe une corrélation entre l'observation* ω *et l'état* $\bar{s}$ *c'est à dire* $\mathbb{P}(\bar{s}|\omega) = 1$.

Prise de décisions optimales

À ce jour, il n'existe pas de théorie de la planification pour les problèmes de contrôle distribué des processus décisionnels de Markov conjointement complètement observables. En particulier, il est communément admis à tord que la propriété de d'observabilité conjointe totale n'apporte aucune simplification significative à cette sous-classe en comparaison aux DEC-POMDPs. Cette opinion est soutenue par la complexité NEXP-difficile des DEC-MDPs à horizon fini. En somme la théorie de la planification se résume à la connaissance que nous avons des DEC-POMDPs. En d'autres termes, la théorie se résumerait à la recherche d'une politique $\pi \in \bar{\Pi}^{\mathbf{HA}}$ telle que :

$$\pi^* = \arg\max_\pi v^\pi(s), \quad \forall s \in S \tag{3.25}$$

Nous argumentons au Chapitre 7 que l'exploitation de la propriété d'observabilité conjointe totale rend ce modèle général l'un des plus simples à résoudre. Malheureusement, aucun des travaux actuels n'exploite cette propriété. En conséquence, il n'existe aucun algorithme de résolution générale des DEC-MDPs à ce jour. Effrayer par la complexité apparente des DEC-MDPs, tous les travaux en DEC-MDPs concernent exclusivement des sous-classes des DEC-MDPs. Nous n'en ferrons pas un état de l'art complet car il y a presqu'autant de sous-classes que d'applications réelles.

3.6.2 Hypothèses d'indépendances

L'ensemble des applications réelles que le modèle des problèmes de contrôle distribués des processus décisionnels de Markov ambitionne de formaliser et résoudre, ne requiert pas l'ensemble des contraintes du modèle général. Au contraire, certaines de ces applications réelles sont telles que les effets des actions, ou les perceptions des agents, sont plus ou moins corrélées. Il existe en effet un grand nombre de cas pratiques où le problème correspond en réalité à plusieurs sous problèmes distribués plus ou moins faiblement corrélées. Il est donc important d'accorder une place à l'étude des hypothèses de dépendances de tout ou partie des composantes d'un problème de contrôle distribué. Ces indépendances peuvent transparaître à travers les fonctions de transitions, observations ou récompenses. Becker et al. [2004b] proposa la première exposition des hypothèses d'indépendances ainsi que quelques unes des propriétés associées.

L'analyse de ces propriétés repose bien souvent sur un modèle factorisé des DEC-POMDP. Nous appelons DEC-POMDP factorisé un DEC-POMDP $\mathcal{M}$ décomposable en k sous-DEC-POMDPs $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_k$, où chaque sous DEC-POMDP est défini par un tuple $(S_j, A_j, \Omega_j, T_j, R_j, O_j, \mathcal{I})$ pour tout $j = 1, 2, \cdots, k$.

DEC-POMDP à transitions indépendantes

Un DEC-POMDP factorisé $\mathcal{M}$ définit par $(S, A, \Omega, T, R, O, \mathcal{I})$ est dit à transitions indépendantes si et seulement s'il peut être décomposé en plusieurs composantes $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_k$ telle que :

$$T(\langle s_1, s_2, \cdots, s_k \rangle, \langle a_1, a_2, \cdots, a_k \rangle, \langle \bar{s}_1, \bar{s}_2, \cdots, \bar{s}_k \rangle) = \prod_{j=1}^{k} T_j(s_j, a_j, \bar{s}_j)$$

DEC-POMDP à observations indépendantes

De façon similaire au cas précédent, il est possible de définir un DEC-MDP à observations indépendantes. Un DEC-POMDP factorisé $\mathcal{M}$ définit par $(S, A, \Omega, T, R, O, \mathcal{I})$ est dit à observations indépendantes si et seulement s'il peut être décomposé en plusieurs composantes $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_k$ telle que :

$$O(\langle s_1, s_2, \cdots, s_k\rangle, \langle a_1, a_2, \cdots, a_k\rangle, \langle \omega_1, \omega_2, \cdots, \omega_k\rangle, \langle \bar{s}_1, \bar{s}_2, \cdots, \bar{s}_k\rangle) = \prod_{j=1}^{k} O_j(s_j, a_j, \omega_j, \bar{s}_j)$$

DEC-POMDP à récompenses indépendantes

Enfin, il est également possible de définir l'hypothèse d'indépendances des récompenses. Un DEC-POMDP factorisé $\mathscr{M}$ définit par $(S, A, \Omega, T, R, O, \mathscr{F})$ est dit à récompenses indépendantes si et seulement s'il peut être décomposé en plusieurs composantes $\mathscr{M}_1, \mathscr{M}_2, \cdots, \mathscr{M}_k$ telle que :

$$R(\langle s_1, s_2, \cdots, s_k\rangle, \langle a_1, a_2, \cdots, a_k\rangle, \langle \bar{s}_1, \bar{s}_2, \cdots, \bar{s}_k\rangle) = \varphi(R_1, R_2, \cdots, R_k)$$

où φ est une fonction d'agrégation, en particulier $\varphi(R_1, R_2, \cdots, R_k) \equiv \sum_j R_j$.

3.6.3 DEC-POMDPs à base de tâches

Les problèmes de prise de décisions séquentielles peuvent modéliser un multitude d'applications réelles, y compris des applications dont l'objectif principale est d'accomplir un ensemble fini de tâches prédéfinies. Nous parlerons de DEC-POMDPs à base de tâches. Ce modèle permet d'encoder des applications allant des problèmes de planification d'une stratégie commune pour un jeux de football en robotique ; ou encore de l'ordonnance de la satisfaction d'une série de tâches organisées selon une structure de précédences temporelles illustrant les célèbres missions spatiales de la NASA [Zilberstein and Mouaddib, 2000, Zilberstein et al., 2001, **?**]. En encodant une application suivant ces modèles, il est possible de réduire de façon exponentielle la complexité du DEC-POMDP jusqu'à une complexité polynomiale dans les meilleures cas. Cela a pour conséquence, d'augmenter la capacité à résoudre des problèmes de dimensions bien plus larges.

DEC-MDPs à base de buts

L'une des premières sous-classes de DEC-MDPs à base de tâches, consiste en un modèle de DEC-MDPs disposant d'un ensemble fini de sous buts à satisfaire (GO-DEC-MDP). Il s'agit pour l'essentiel d'un DEC-MDP composé d'une unique tâche qui consiste à atteindre à un certain moment τ, un ou plusieurs des états buts prédéfinis. Une récompense

est perçue en cas de succès. Becker et al. [2003] ont définit un DEC-MDP à base de buts et à horizon fini comme un DEC-MDP satisfaisant les contraintes suivantes :

1. Il existe un ensemble d'états buts $\mathcal{G} \subset S$.
2. La fonction conjointe de récompenses est indépendante et définie comme suit : $R(s,a) = \sum_i R(s,a^i)$, pour tout agent $i \in \mathcal{I}$.
3. Une récompense conjointe additive est définie et donnée pour tout état but $s \in \mathcal{G}$ atteint avant le temps $\tau' < \tau$ par, $R(s)$.

Cette classe de DEC-MDP peut être simplifiée en considérant certaines propriétés telles que les hypothèses d'indépendances. C'est ainsi que Becker et al. [2004a], Seuken and Zilberstein [2008] prouve qu'un GO-DEC-MDP avec observations et transitions indépendantes est NP-difficile. De plus, lorsque cette dernière classe est munie d'un seul état but et d'une fonction de coûts uniformes, le problème est alors P-difficile

DEC-MDP à base de coût occationné

La seconde approche de DEC-MDPs à base de tâches considère un ensemble fini de $|\mathcal{I}|$ MDPs complètement distribués $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_{|\mathcal{I}|}$. Ce modèle fût proposé initialement par Beynier and Mouaddib [2005, 2006] et nommé selon l'acronyme anglo-saxon opportunity cost DEC-MDP (OC-DEC-MDP).

Définition 9. *Un DEC-MDP à base de coût occationné $\mathcal{M}$ est définit par un ensemble fini de $|\mathcal{I}|$ MDPs $\mathcal{M}_i$ et donné par le tuple (S^i, A^i, T^i, R^i), pour chaque agent $i = 1,2,\cdots,|\mathcal{I}|$ où :*

1. *S^i représente l'ensemble des états individuels de l'agent i.*
2. *A^i est l'ensemble fini des actions individuelles de l'agent i.*
3. *$T^i(\bar{s}^i | s^i, a^i)$ est la fonction de transitions de l'agent i.*
4. *$R^i(s^i, a^i)$ est la fonction de récompenses ou de coûts de l'agent i.*

Dans ce modèle chaque agent est modélisé indépendamment via un MDP lui-même encodant une tâche ou plusieurs tâches. La principale difficulté réside dans la corrélation existante entre les tâches individuelles à un agents et entre les tâches de l'ensemble des agents. En effet, les auteurs supposent l'existence de dépendances causales entre les tâches. Ces dépendances peuvent se traduire de différentes manières. En particulier, elles peuvent se traduire à travers des relations de précédences temporelles. L'encodage de ces dépendances se fait à travers un graphe dit de mission. En particulier, les auteurs ne considèrent que des graphes de missions acycliques. En d'autres termes, il existe toujours un ordre topologique $<$ sur l'ensemble des tâches assignées aux agents. Ainsi en résolvant

indépendamment chacune des tâches incluses dans les MDPs des agents $\mathscr{I}$ suivant l'inverse de l'ordre topologique, les auteurs parviennent à construire une politique individuelle $\pi^i \in \Pi^{\mathbf{MD}}$ pour chaque agent $i \in \mathscr{I}$. Le lecteur doit faire attention à ne pas confondre une résolution indépendante d'une exécution indépendante. S'il est possible de résoudre ce modèle indépendamment pour chacun des agents, il ne reste pas moins vrai que l'exécution peut se faire non seulement de façon distribuée mais aussi simultanément pour toute ou partie des agents.

Notons cependant que les politiques individuelles ainsi construites sont définies dans l'espace des politiques markoviennes et déterministes $\Pi^{\mathbf{MD}}$. Cette restriction de l'espace de définition des politiques suggère de deux choses l'une : soit que le cadre environnementale considéré est localement complètement observable pour chaque agent (cas 1) ; soit que l'hypothèse d'observabilité locale complète restreint la complexité du problème. Dans le premier cas, l'approche peut produire une solution conjointe optimale, mais le problème résolu n'est pas un DEC-MDP.

3.7 Autres modèles associés

3.7.1 Le modèle MTDP

Le problème du contrôle distribué des processus décisionnels de Markov a également été formalisé via le modèle MTDP nommé selon l'acronyme anglo-saxon « multi-agents team decision problem » et introduit par Pynadath and Tambe [2002]. Ce formalisme, alternatif et prouvé équivalent au modèle DEC-POMDP [Seuken and Zilberstein, 2008], est une extension d'un modèle qui avait été proposé afin de modéliser la planification multi-agents [Pynadath and Tambe, 2002].

Définition 10. *Un MTDP est définit par un tuple* $(S, A, \Omega, P, R, O, B, \mathscr{I})$*, où :*

1. *S est l'ensemble fini des états conjoints $s \in S$,*
2. *$A \equiv A^1 \times A^2 \times \cdots \times A^{|\mathscr{I}|}$ est un ensemble fini d'actions conjointes $a \in A$, et A^i pour tout agent $i \in \mathscr{I}$ un ensemble fini d'actions individuelles de l'agent i.*
3. *$\Omega \equiv \Omega^1 \times \Omega^2 \times \cdots \times \Omega^{|\mathscr{I}|}$ est un ensemble fini d'observations conjointes $\omega \in \Omega$, et Ω^i est un ensemble fini d'observations individuelles pour tout agent $i \in \mathscr{I}$,*
4. *$P(\bar{s}|s, a)$ est la probabilité de transiter de l'état conjoint s à l'état conjoint $\bar{s}$ après l'exécution de l'action conjointe a,*
5. *$R(s, a)$ est la récompense (coût) perçu par l'ensemble des agent suite à l'exécution de l'action conjointe a dans l'état conjoint s,*

6. *$O(\vec{s}, \omega | s, a)$ est la probabilité de percevoir l'observation conjointe ω lorsque l'ensemble des agent transite de l'état conjoint s à l'état conjoint $\vec{s}$ après avoir exécuté l'action a,*
7. *$B \equiv B^1 \times B^2 \times \cdots \times B^{|\mathcal{I}|}$ est l'ensemble des croyances conjointes, et B^i est l'ensemble des croyances individuelles pour tout agent $i \in \mathcal{I}$.*

L'une des différences fondamentales avec le modèle DEC-POMDP réside dans l'ajout de l'ensemble des croyances conjointes B. Pour cette raison, il est possible que les solutions retournées par les deux formalismes soient différentes. En effet, il est possible que l'ensemble des croyances conjointes considérées par MTDP ne soit pas suffisant pour le déterminisme d'une stratégie conjointe optimale pour l'ensemble des agents. Néanmoins, il est toujours possible des déterminer une stratégie conjointe optimale pour l'ensemble des agents sur l'ensemble des croyances conjointes considérées.

3.7.2 POMDPs Interactifs

L'étude des interactions des agents comme moyen de formaliser des problèmes de planification multi-agents est largement répandue dans le cadre du contrôle distribué. Nous citerons par exemple, l'approche Interact-DEC-MDP inspirée des modèles biologiques et proposée par Thomas et al. [2004] ; ou encore celle de MOMAP se focalisant sur les interactions locales entre agents, afin de définir des stratégies heuristiques de construction de politiques conjointes approximatives [Mouaddib et al., 2007]. La principale critique formulée à l'égard des ces approches est celle de savoir à quelle distance de l'optimum ces solutions se trouvent-elles ?

L'une des raisons de la complexité des modèles MTDPs et DEC-POMDPs s'explique par la nécessité de prendre en compte l'ensemble des perceptions des agents au cours du contrôle lors de la planification. Doshi [2004] se propose de n'étudier un tel système qu'à travers le regard d'un seul agent, laissant les autres agents comme partie intégrante de l'environnement avec lequel l'agent en question est en interaction. Il s'agit du modèle I-POMDP selon l'acronyme anglais interactive POMDP. Le principe fondamental de l'I-POMDP est de capturer les informations locales perçues par un agent et inaccessibles des autres agents, en particulier les croyances et les règles de décisions.

3.7.3 ND-POMDP

Il existe cependant un modèle permettant de traduire les interactions entre agents dans le cadre du contrôle distribué [Marecki et al., 2008, Nair et al., 2005, Varakantham et al., 2007]. Le modèle ND-POMDP selon l'acronyme anglo-saxon network distributed POMDP est un DEC-POMDP factorisé.

Définition 11. *Un ND-POMDP est donné par un tuple* (S, A, Ω, P, R, O) *où :*

1. $S \equiv \prod_{i \in \mathscr{I}} S^i \times S_u$ *est l'ensemble fini des états conjoint* $s \in S$. S^i *correspond à l'ensemble des états individuels* $s^i \in S^i$ *pour tout agent* $i \in \mathscr{I}$. S_u *correspond à un ensemble d'états non contrôlables par l'ensemble des agents.*
2. $A \equiv \prod_{i \in \mathscr{I}} A^i$ *est l'ensemble fini des actions conjointes* $a \in A$, *où* A^i *est l'ensemble fini des actions individuelles* $a^i \in A^i$ *pour tout groupe d'agents* $i \in \mathscr{I}$.
3. $\Omega \equiv \prod_{i \in \mathscr{I}} \Omega^i$ *est l'ensemble fini des observations conjointes, où* Ω^i *est l'ensemble fini des observations individuelles* $\omega^i \in \Omega^i$ *pour tout groupe d'agents* $i \in \mathscr{I}$.
4. $P(\vec{s}|s, a) = \mathbb{P}(\vec{s}_u|s_u) \cdot \prod_{i \in \mathscr{I}} \mathbb{P}(\vec{s}^i|s^i, a^i)$ *est la fonction de transitions.*
5. $O(\omega|s, a) = \prod_{i \in \mathscr{I}} \mathbb{P}(\omega^i|s^i, s_u, a^i)$ *est la fonction d'observations.*
6. $R(s, a) = \sum_{i \in \mathscr{I}} R(s^i, s_u, a^i)$ *est la fonction de récompenses.*

où l'ensemble $\mathscr{I}$ *est un ensemble d'ensemble d'agents, en d'autres termes* $i \in \mathscr{I}$ *représente un indice d'un groupe d'agents.*

Le lecteur notera que le modèle ND-POMDP est muni de propriétés intéressantes, à savoir l'indépendance des observations, des transitions et la séparabilité des récompenses (ce problème est tout au plus NP-difficile si $\mathscr{I}$ est un ensemble d'agents). En un sens, un ND-POMDP peut être perçu comme un graphe de DEC-POMDPs. Cette dernière remarque est importante. En effet, bien qu'il soit naturellement possible d'encoder des ND-POMDPs allant jusqu'à une dizaine d'agents, les cliques d'agents ne contiennent jamais plus de deux agents. Cela nous ramène malheureusement aux dimensions des problèmes de la littérature des DEC-POMDPs. Ces problèmes ne surpassent pas non plus deux agents. En d'autres termes en parvenant à résoudre des DEC-POMDPs modélisant plus de deux agents, nous augmenterons également notre capacité à passer à l'échelle dans l'ensemble des sous-classes et modèles similaires.

3.8 Résumé

Nous avons proposé un état de l'art des recherches en planification pour le contrôle distribué dans le cadre des processus décisionnels de Markov. Nous avons volontairement

Questions ouvertes en DEC-POMDPs.

- **Planification en DEC-POMDP :**
  1. Existe t-il une statistique suffisante pour la planification pour le contrôle distribué plus compacte que la croyance multi-agents ?
  2. Quelle est la classe de politiques conjointes la plus compacte contenant toujours au moins une politique conjointe optimale pour tout problème de contrôle distribué ?
  3. Est-il possible de procéder à une construction incrémentale des politiques conjointes évitant ainsi l'opération d'énumération exhaustive à chaque horizon ?
  4. Peut-on déterminer une politique conjointe d'un problème de contrôle distribué en utilisant les croyances conjointes ? Si oui, comment ?
  5. Est-il possible de construire une politique conjointe optimale pour un problème de contrôle distribué en utilisant des trajectoires d'historiques ? Si oui, comment ?
- **Planification en DEC-MDP :**
  1. Existe t-il une statistique suffisante plus compacte pour la planification des DEC-MDPs en comparaison à celle des DEC-POMDPs ?
  2. Existe t-il un algorithme général de résolution des DEC-MDPs qui soit moins complexe que ceux existant dans le cadre des DEC-POMDPs ?
  3. Comment l'hypothèse d'observabilité conjointe complète simplifie t-elle la théorie du contrôle distribué.

FIGURE 3.19 – Questions ouvertes en DEC-POMDPs.

mis l'emphase sur les aspects algorithmiques, en accord avec l'objectif premier de cette thèse. Bien qu'il existe de nombreux modèles (MTDP ou DEC-POMDP) pouvant formaliser ces problèmes de contrôle distribué et ainsi de permettre leur résolution, nous avons fait le choix de présenter essentiellement le modèle qui au fil des années à fait le consensus dans le littérature du contrôle distribué, à savoir le modèle des DEC-POMDPs. Ce modèle est le plus général et permet de prendre en compte l'ensemble des spécificités des problèmes du contrôle distribué. Nous avons tout au long de notre exposé souligner la jeunesse du champs de recherche expliquant un nombre considérables de questions encore ouvertes et résumées à la Figure 3.19. En particulier, le fait qu'il n'existe aucune étude rigoureuse de la théorie de la planification pour le contrôle distribué. La quantité et la

nature des questions encore ouvertes expliquent la divergence des innombrables orientations des recherches parfois antagonistes. Contrairement aux recherches en POMDPs, « il n'existe malheureusement pas en DEC-POMDP un consensus sur les principaux enjeux ou axes de recherches sur lesquelles la communauté des chercheurs dans ce domaine peut se pencher afin d'améliorer à coup sûr les connaissances ». Nous proposons dans les chapitres 5 et 6 suivants des réponses à certaines des questions jusqu'ici ouvertes, avec l'espoir qu'elles permettront de réduire l'écart théorique existant entre la théorie de la planification et l'apprentissage dans le cadre du contrôle distribué (DEC-POMDPs) et celle antérieure du contrôle centralisé (MDPs, POMDPs).

Nous avons également donné un bref aperçu des modèles des sous-classes des DEC-POMDPs. Ces modèles tentent d'apporter des réponses à la complexité générale des DEC-POMDPs en exploitant diverses sources de réduction de cette complexité. Bien qu'utiles en pratique, ni ces modèles ni leurs méthodes associées n'apportent de connaissances nouvelles sur le problème général. Bien au contraire, ces classes s'appliquent à exploiter des aspects communs à l'ensemble des systèmes multi-agents indépendamment du type de contrôle auquel le système est soumis. Plutôt que d'explorer l'ensemble infini des sous-classes possibles des DEC-POMDPs, dont l'étude n'apporte aucune information sur le problème général, nous soutenons qu'il est préférable de concentrer notre attention sur l'analyse formelle du problème général. À l'issue de cette analyse, nous pourrons fusionner les connaissances acquises aux connaissances génériques sur les systèmes multi-agents afin d'implémenter des systèmes multi-agents réels et soumis au contrôle distribué.

Afin de soutenir la recherche de nouveaux modèles de contrôle distribué, de nombreux chercheurs argumentent que le modèle DEC-POMDP est inapproprié pour le contrôle réel des systèmes multi-agents compte tenu de sa complexité en pire cas. Ils suggèrent ainsi qu'il existerait un tout autre modèle capable de formaliser les problèmes de contrôle distribué à échelle réelle et de permettre leur résolution en pratique. S'il est vrai qu'il existe des problèmes de contrôle distribué pouvant être formalisés de sorte à exploiter bien plus d'hypothèses simplificatrices que celles considérées dans le cadre des DEC-POMDPs, il est impossible que ces modèles parviennent à formaliser tous les problèmes de contrôle distribué que formalise un DEC-POMDP, tout en garantissant une complexité en pire cas moindre. La complexité d'un DEC-POMDP est donnée par la complexité des problèmes les plus difficiles qu'elles peut formaliser. L'intérêt de cet outil est de traduire les propriétés mathématiques observées dans les applications réelles. Plus le modèle est général plus grande est l'impact des propriétés observées, et vice-versa.

Deuxième partie

Contributions

« Knowledge is limited ; but imagination encircles the world. »

Albert Einstein

Chapitre 4

Résolution topologique pour le contrôle centralisé

Une partie de ce chapitre est apparu précédemment dans les papiers : « Topological Order Planner for POMDPs » [Dibangoye et al., 2008b,c, 2009c] rédigés avec Guy Shani, Brahim Chaib-draa et Abdel-Illah Mouaddib ; « A Novel Prioritization Technique for Solving Markov Decision Processes » [Dibangoye et al., 2008a] avec Brahim Chaib-draa et Abdel-Illah Mouaddib ; et « Periodic real-time resource allocation for teams of progressive processing agents » [Dibangoye et al., 2007] avec Abdel-Illah Mouaddib et Brahim Chaib-draa.

LES modèles mono agent de contrôle des processus décisionnels de Markov offrent des cadres expressifs pour la modélisation et la résolution des problèmes de contrôle centralisé, comme discuté au Chapitre 2. Cependant, de nombreux facteurs limitent l'utilisation des processus de Markov dans la modélisation et résolution d'applications de dimensions réelles. Ces facteurs incluent, d'une part, la malédiction de la dimension (où la dimension d'un problème de planification est directement reliée aux nombre d'états) [Bellman, 1957], et d'autre part, la malédiction de l'historique (où le nombre de plans contingentés croît exponentiellement avec l'horizon de planification) [Pineau, 2004].

Dans ce chapitre, nous présentons l'ensemble de nos contributions dans le cadre de la résolution centralisée des modèles mono agent de contrôle des processus de Markov, qu'ils soient partiellement ou complètement observables. L'ensemble des algorithmes conçus à cette fin exploite les dépendances causales du système afin de réduire la complexité des mises à jour de la fonction de valeurs. Cela a pour conséquence d'accélérer le calcul de la politique optimale ou approximative. Nous proposons plusieurs algorithmes dits de programmation dynamique topologique (TDP), qui étendent et généralisent l'approche

introduite par [Dai and Goldsmith, 2007] et discutée au Chapitre 2. Ce chapitre se décompose en trois grandes parties. Les deux premières sont consacrées respectivement à la résolution topologique des processus de Markov complètement observables (MDPs) et à la résolution topologique des processus de Markov partiellement observables (POMDPs). La dernière partie est consacrée à l'étude et la résolution d'un problème de défense navale nommé NEREUS [1]. Ce problème réel exhibe les propriétés structurelles exploitées par la programmation dynamique topologique.

La Section 4.2 explique les propriétés structurelles prérequises à l'application d'un algorithme de programmation dynamique topologique dans la résolution des MDPs. Nous montrons ensuite comment exploiter ces propriétés afin de trouver un juste équilibre entre deux facteurs de performances : d'une part, le temps consacré à l'organisation des mises à jour de la fonction de valeurs en vue d'améliorer la convergence de celle-ci ; et d'autre part, le temps total nécessaire afin de résoudre le problème original. La Section 4.3 étend non sans mal cette idée au cadre des POMDPs. Nous y décrivons les conditions et les algorithmes de programmation dynamique topologique exacts et approximatifs de résolution d'un POMDP. Enfin, la Section 4.5 illustre les performances de la programmation dynamique topologique en l'appliquant à un problème réel nommé NEREUS.

4.1 Motivation

Soit un environnement où un agent dispose d'un ensemble d'actions lui permettant de se mouvoir d'un état à un autre. Supposons en outre que les effets de certaines actions prises dans certains états sont définitifs. Par exemple, dans une fabrique de jouets, les robots peuvent coller deux composants ; une fois les composants collés, il est impossible de les décoller. De façon similaire, un robot d'exploration sur Mars peut échantillonner une roche. Mais le succès de cette action, modifie définitivement l'intérêt du robot pour la roche échantillonnée. Ainsi, les transitions vers les états où les composants ont été collés ; ou la roche échantillonnée, ne peuvent être traversées dans le sens inverse. Dans ces cas, nous pouvons décomposer les états de l'environnement en couches. C'est à dire en groupe d'états. De sorte que l'agent puisse se déplacer d'un état à un autre à l'intérieur d'une même couche. Mais une fois sorti d'une couche, il ne peut y revenir. Ce type de structure est appelé structure topologique. Cette propriété structurelle est incluse dans de nombreuses applications réelles, y compris : les problèmes de chaînes approvisionnement ; de chaînes de montage ; ou encore de réseaux routier et ferroviaire.

1. Naval Environnement for Resource Engagement in Unpredictable Situations.

4.1.1 Exemples d'applications

station 0 station 1 station 2 station 3

FIGURE 4.1 – Chaîne de montage de véhicules.

Considérons par exemple, le problème d'une chaîne de montage de véhicules illustrée dans la Figure 4.1. Pour simplifier, nous ne considérons que quatre stations $k = 0,1,2,3$ - dans lesquelles le véhicule peut transiter dans un des trois états suivants :

1. à l'état s_{3k}, le véhicule arrive dans la station courante et a été correctement assemblé dans les stations précédentes (s'il y en a) – c'est l'état initial de la station k ;
2. en s_{3k+1}, le véhicule a été assemblé à la station courante – les pièces disponibles à cette station lui ont été installées. Cependant, les robots de montage munis de leurs capteurs perçoivent des dysfonctionnements ;
3. en s_{3k+2}, le véhicule a été correctement assemblé à la station courante – il peut être envoyé à la station suivante.

Les opérations ou actions disponibles dans chacune des stations sont les suivantes :

1. l'action a_0 correspond au montage des pièces disponibles à cette station sur le véhicule s'y trouvant ;
2. l'action a_1 démonte les pièces montées à la station courante sur le véhicule s'y trouvant ;
3. et l'action a_2 envoie le véhicule de la station courante à la station suivante.

Lorsque le véhicule est effectivement envoyé à la station suivante il n'est plus possible qu'il revienne à la station courante. On en déduit alors que le problème de gestion d'une chaîne de montage de véhicules est bien muni d'une structure topologique.

Le graphe illustré à la Figure 4.2 correspond au graphe des transitions des états de la chaîne de montage de véhicules. Il est facile de voir que les groupes d'états $\{s_{3k}, s_{3k+1}\}$ et $\{s_{3k+2}\}$ constituent des couches d'états, pour toute station $k = 0,1,2,3$. Par exemple, le groupe d'états $\{s_6, s_7\}$ est une couche d'états pour notre problème de chaîne de montage de véhicules.

Vérifier qu'un problème donné est ou non muni d'une structure topologique peut se faire à l'aide d'un simple test. Il s'agit du déterminisme de l'existence ou non de plusieurs couches telles que définies ci-dessus.

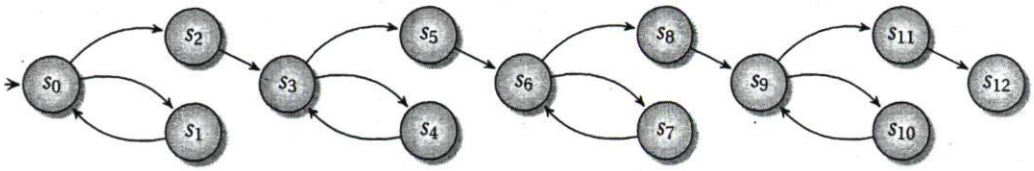


FIGURE 4.2 – Représentation graphique d'une chaîne de montage de véhicules.

Plus formellement, les dépendances causales d'un problème de prise de décisions séquentielles $\mathscr{M}$ peuvent être représentées au travers d'un graphe orienté $G_{\mathscr{M}} = (V, E)$, où l'ensemble des états du problème correspond à l'ensemble des nœuds E du graphe, et l'ensemble des transitions d'un état à un autre correspond à l'ensemble des arcs V du graphe.

Définition 12. *On appelle couche d'états d'un problème de prise de décisions séquentielles $\mathscr{M}$, toute composante fortement connexe du graphe $G_{\mathscr{M}}$ – ce graphe est aussi appelé graphe de transitions.*

Dès lors, déterminer si un problème $\mathscr{M}$ possède une structure topologique revient au calcul des couches d'états du problème. Il s'agit du calcul des composantes fortement connexes du graphe de transitions $G_{\mathscr{M}}$ – par exemple le graphe illustré en Figure 4.2 – associé au problème de gestion d'une chaîne de montage de véhicules.

En théorie des graphes orientés, une composante fortement connexe est un sous-ensemble des nœuds, où toute paire de nœuds $s, s' \in V$ est mutuellement accessible. C'est à dire $s \rightsquigarrow s'$ et $s' \rightsquigarrow s$. Il existe deux principaux algorithmes de complexité linéaire permettant de calculer les composantes fortement connexes d'un graphe : l'algorithme de Tarjan et celui de Kosaraju [Cormen et al., 2001, Tarjan, 1972].

Par exemple, dans le cas du problème de gestion d'une chaîne de montage de véhicules, les états s_0 et s_1 sont mutuellement accessibles. En effet, il existe un arc (s_0, s_1) et un autre arc (s_1, s_0) dans le graphe de transitions $G_{\mathscr{M}}$ illustré à la Figure 4.2. Pour cette raison, les états $\{s_0, s_1\}$ font partie d'une même composante fortement connexe. De plus, comme il n'existe pas d'autres états s mutuellement accessibles avec $\{s_0, s_1\}$, l'ensemble $\{s_0, s_1\}$est une composante fortement connexe. La Figure 4.3 identifie le graphe des composantes fortement connexes associées au graphe des transitions du problème de gestion d'une chaîne de montage de véhicules.

L'ensemble des couches d'états d'un problème de prise de décisions séquentielles est noté $\{S_\ell\}_{\ell=0}^{L}$. Le graphe orienté $\bar{G}_{\mathscr{M}}$ appelé graphe réduit, dont les nœuds sont les couches d'états et les arcs sont les transitions d'une couche à une autre, est par construction acyclique. Un tel graphe est illustré à la Figure 4.3, cette figure correspond au graphe réduit

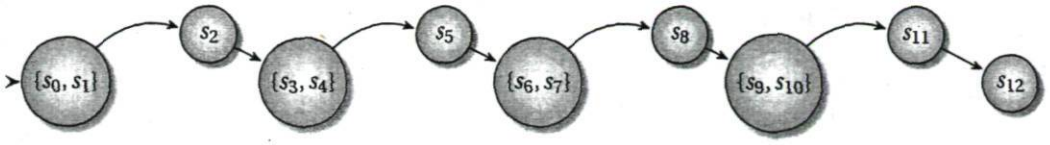


FIGURE 4.3 – Graphe réduit du problème de gestion d'une chaîne de montage de véhicules.

du problème de gestion d'une chaîne de montage de véhicules. Cette propriété facilite le calcul d'une politique optimale pour un problème de prise de décisions séquentielles.

En effet, soit $\mathcal{M}$ un processus décisionnel de Markov complètement observable. Considérons par exemple le graphe réduit représenté à la Figure 4.3. Il apparaît que la valeur $v^*(s_{12})$ de l'état s_{12} est indépendante de toute autre valeur de tout autre état sauf s_{12}. Par conséquent, la seule connaissance de la récompense immédiate $R(s_{12}, a)$ est suffisante pour déterminer la meilleure action à prendre dans cet état : $d^*(s_{12}) = \text{argmax}_{a \in A} R(s_{12}, a)$ et $v^*(s_{12}) = R(s_{12}, d^*(s_{12}))$. Nous avons pu déterminer la valeur optimale pour l'état s_{12} sans pour autant avoir connaissance ne fusse qu'une valeur approximative de tout ou partie des autres états.

Considérons maintenant l'état s_{11} , il est facile d'observer que sa valeur $v^*(s_{11})$ ne dépend d'aucun autre état sauf de la valeur $v^*(s_{12})$. Cela peut se vérifier en écrivant l'équation de Bellman pour l'état s_{11} :

$$v^*(s_{11}) = \text{argmax}_{a \in A} \left\{ R(s_{11}, a) + \lambda v^*(s_{12}) \right\}. \tag{4.1}$$

On y voit clairement que seul l'état successeur de s_{11}, c'est à dire l'état s_{12}, peut influencer la valeur $v^*(s_{11})$. Par conséquent, comme nous avons connaissance de la valeur exacte $v^*(s_{12})$, le calcul de $v^*(s_{11})$ est immédiat. Le processus continue jusqu'à ce que l'ensemble des valeurs optimales de chacune des couches d'états soit déterminé.

Ce principe algorithmique peut être perçu comme la décomposition d'un problème $\mathcal{M}$ de prise de décisions séquentielles quelconque en plusieurs sous problèmes $\{\mathcal{M}_\ell\}_{\ell=0}^{L}$ Chaque sous problème $\mathcal{M}_\ell$ est une restriction de $\mathcal{M}$ à une seule couche d'états S_ℓ. De plus, il existe un ordre de priorité dans la résolution de ces sous problèmes. Ils sont ordonnés suivant l'inverse de l'ordre topologique sur le graphe réduit $\bar{G}_{\mathcal{M}}$. En outre, la résolution du problème initial $\mathcal{M}$ peut se faire en utilisant l'algorithme d'induction rétrograde suivant l'inverse de l'ordre topologique. Dans le cadre de la prise de décisions séquentielles, ce principe n'est donc ni plus ni moins que le principe de Bellman en application. De nombreux chercheurs l'ont utilisé avec succès pour la résolution des processus de Markov complètement observables [Abbad and Boustique, 2003, Bonet and Geffner, 2003a, Dai and Goldsmith, 2007].

4.1.2 Exemples de la littérature

Bon nombre d'applications réelles consistent à satisfaire un ensemble de sous tâches. Dans ces cas, la dimension du problème croît au moins linéairement par rapport au nombre de sous tâches à effectuer. Afin de jeter un coup d'œil sur ces applications, nous considérons deux exemples de la littérature des processus décisionnels de Markov complètement et partiellement observables.

Exemple 3. *Soit $\mathcal{M}$ un processus de Markov complètement observable où toute paire d'états de ce MDP est mutuellement accessible. Soit $C \subseteq S$ un ensemble d'états identifiant chacun une état but à visiter, et $s_0 \in C$ l'état initial. Dès lors le problème du commis de voyage stochastique (STSP) [Tang and Miller-Hooks, 2007] définit par $(\mathcal{M}, C, s_0)$ consiste à trouver une politique (plus probablement non stationnaire et non markovienne $\pi \in \Pi^{\mathbf{HD}}$) qui minimise le total espéré du temps nécessaire afin de visiter au moins une fois chacun des états buts $s \in C$, en partant de l'état initial s_0 et en y revenant qu'après avoir visiter l'ensemble des états buts $s \in C$. Bien que STSP soit un problème non stationnaire, afin de coller au cadre des processus décisionnels de Markov complètement observables, nous créons un MDP augmenté $\mathcal{M}'$. Ce dernier augmente l'ensemble des états de $\mathcal{M}$ avec $|C|$ variables booléennes indiquant quels états dans C doivent être visités. Ainsi, l'état but dans le MDP augmenté $\mathcal{M}'$ consiste en l'état augmenté où toutes les variables booléennes sont « vrai » lorsque le commis retourne à l'état s_0. Ceci, de sorte qu'il soit possible de restreindre la recherche des politiques de $\mathcal{M}'$ à l'ensemble des politiques pures.*

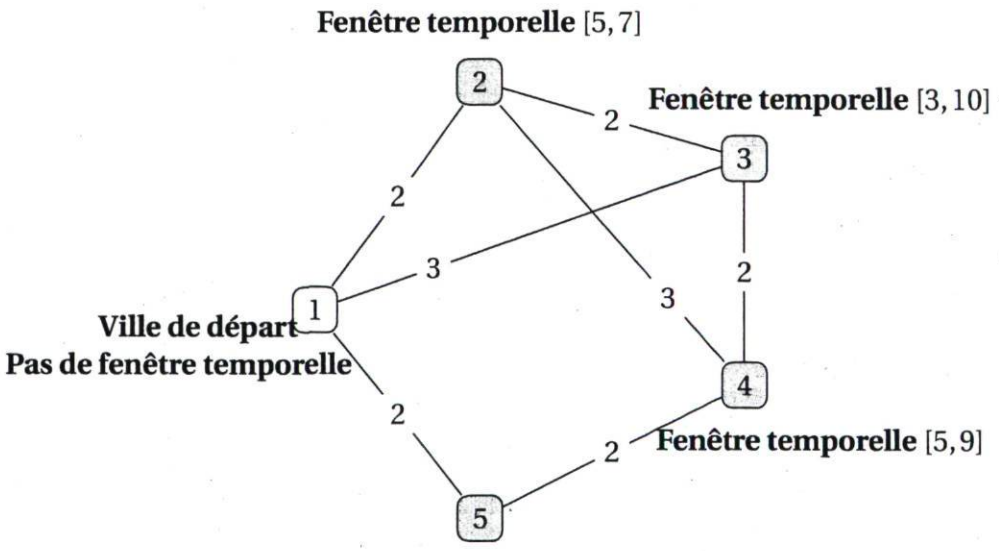


FIGURE 4.4 – Illustration du problème du commis de voyage stochastique avec fenêtres temporelles.

Des problèmes similaires ont été proposés dans le cadre des processus décisionnels de Markov partiellement observables. En particulier, [Smith and Simmons, 2004] proposent un problème d'échantillonnage de roches, problème de dimension non négligeable, nommé « Rocksample » selon l'acronyme anglais. Ce problème modélise un véhicule d'exploration sur Mars.

Exemple 4. *Considérons le problème d'échantillonnage de roches Figure 4.5, illustrant une des missions spatiales de la NASA. Dans ce problème, un véhicule peut cumuler des récompenses en échantillonnant les roches dans son environnement immédiat. En continuant son exploration, s'il parvient à retourner à la base, il récolte également des récompenses. Dans sa version de simulation, les positions initiales du véhicule et des roches sont connues, mais seule une partie des roches est « bonne ». Cela indique que le véhicule peut récolter des récompenses en les échantillonnant. L'opération d'échantillonnage est coûteuse. Ainsi, le véhicule est équipé d'un senseur de grande portée mais dont la précision n'est pas très bonne. Ce senseur peut être utilisé afin de l'aider à déterminer si la roche est bonne avant de s'en approcher et de l'échantillonner.*

FIGURE 4.5 – Véhicule utilisé dans le cadre d'une mission exploration spatiale.

Une instance de scénario de simulation prend en entrée la taille de l'espace d'exploration, c'est à dire une grille $c \times c$ et k roches, dénotée RockSample$[c, k]$ [Smith and Simmons, 2004]. Le problème de prise de décisions séquentielles est décrit comme suit : l'ensemble des états est le produit croisé de $(c+1)$ positions possibles du véhicule Position$=\{(1,1),(1,2),\cdots,(c,c)\}$ et k variables binaires $\text{TypeRoche}_i = \{\texttt{Bonne},\texttt{Mauvaise}\}$ pour tout $i = 1,2,\cdots,k$ indiquant lesquelles des roches sont bonnes. On y ajoute un état terminal additionnel, accessible lorsque le véhicule se dirige à gauche de la grille. Le véhicule

dispose en outre de $(k+5)$ actions comprenant Nord, Sud, Est, Ouest, ´Echantillonner, Vérifier$_1$,$\cdots$,Vérifier$_k$. Les quatre premières actions sont des actions de déplacement du véhicule. L'action ´Echantillonner échantillonne la roche correspondant à la position courante du véhicule. Si la roche est bonne, le véhicule perçoit une récompense de (+10) et la roche devient mauvaise. Si au contraire la roche est mauvaise, le véhicule reçoit une pénalité (?10). En retournant à sa base, le véhicule perçoit également une récompense de (+10). De tout autre action, il ne résulte aucune récompense ou coût. Chacune des actions Vérifier$_i$ correspond à l'application du senseur de longue portée sur la roche i , pour tout $i = 1,2,\cdots,k$. Cette action retourne une réponse floue dans l'ensemble Bonne, Mauvaise. Le bruit du senseur de longue portée est déterminé par un facteur d'efficacité ζ. Ce dernier décroit de façon exponentielle suivant la distance Euclidienne du véhicule à la roche. Lorsque le facteur $\zeta = 1$, le senseur renvoie systématiquement l'état réel de la roche. Lorsque $\zeta = 0$, il y a 50% de chance qu'il renvoie l'état réel de la roche. Initialement, une roche à autant de chance d'être bonne que mauvaise.

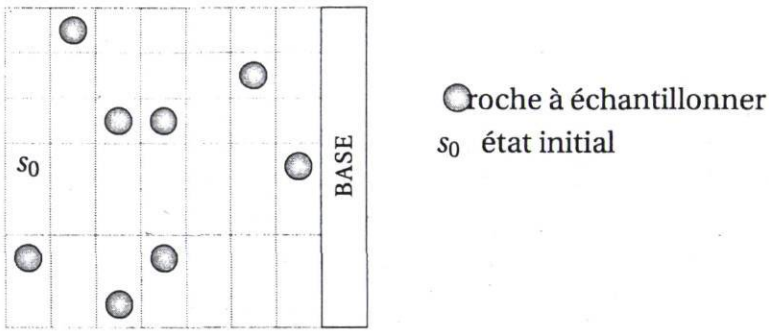


FIGURE 4.6 – Une instance du problème d'échantillonnage de roches.

Ces deux exemples sont munis de structures topologiques que les approches de programmation dynamique topologique tentent d'exploiter. Dans le cas du problème d'échantillonnage de roches par exemple, lorsqu'une roche « bonne » a été échantillonnée, elle devient « mauvaise ». Et par conséquent, elle ne pourra plus jamais retrouver son état « bonne » précédant son échantillonnage. Ces propriétés topologiques induisent un ordre partielle sur l'ensemble des états, comme discuté à la section suivante.

La programmation dynamique topologique est principalement motivée par la recherche de bonnes séquences de mises à jour de la fonction de valeurs afin d'accélérer sa vitesse de convergence vers une fonction de valeurs ε-optimales. Cela explique pourquoi nous nous intéressons maintenant à une classe spécifique de problèmes de prises de décisions séquentielles. En effet, la recherche de bonnes séquences de mises à jour de la fonction de valeurs peut conduire à des coûts prohibitifs. Ainsi, seules certaines classes de problèmes offrent les structures permettant l'extraction de séquences de bonnes qualités en temps raisonnable. À noter que l'objectif n'est pas de trouver les meilleures séquences à tout prix mais de trouver les meilleures séquences dans un laps de temps relativement restreint.

Ceci de sorte que le temps total de calcul de la fonction de valeurs soit réduit en comparaison au temps requis par une approche classique. Cette contrainte explique la simplicité des opérations d'ordonnancement des mises à jour que nous allons proposer.

4.2 Programmation dynamique topologique pour les MDPs

Nous nous intéressons ici aux problèmes de prises de décisions séquentielles munis d'un ordre partiel sur l'ensemble des partitions. Cet ordre permet d'organiser les mises à jour de la fonction de valeurs. Dans un premier temps, nous introduisons des partitions d'un processus décisionnel de Markov complètement observable ayant la forme d'un ensemble orienté $(\mathcal{M}, \prec_1)$.

Définition 13. *Un ensemble orienté $(\mathcal{M}, \prec_1)$ est un ensemble $\mathcal{M} \equiv \{\mathcal{M}_\ell\}_{\ell=0}^{L}$ de sous MDPs, où l'ensemble des sous MDPs est ordonné selon l'ordre total $\prec_1$ comme suit : $\mathcal{M}_0 \prec_1 \mathcal{M}_1 \prec_1 \cdots \prec_1 \mathcal{M}_L$.*

Cet ordre total indique la dépendance causale d'un sous problème par rapport aux autres sous problèmes. Par exemple, $\mathcal{M}_0 \prec_1 \mathcal{M}_1$ signifie que la résolution du sous problème $\mathcal{M}_1$ est un prérequis à la résolution du sous problème $\mathcal{M}_0$. Ainsi tout ensemble orienté $(\mathcal{M}, \prec_1)$ possède la propriété qu'en résolvant les sous problèmes $\mathcal{M}_\ell$ suivant l'ordre total $\prec_1$, il est possible de résoudre de façon exacte le problème composite $\mathcal{M}$. Notons néanmoins que bien qu'il soit possible de résoudre chacun des sous problèmes MI indépendamment et suivant l'ordre $\prec_1$, il est cependant nécessaire d'utiliser la fonction de valeurs composite v incorporant les fonctions de valeurs v_ℓ des sous problèmes $\mathcal{M}_\ell$ déjà résolus.

Un processus décisionnel de Markov complètement observable $\mathcal{M}$ équivaut à un ensemble ordonné $(\mathcal{M}, \prec_1)$ si et seulement si $\mathcal{M}$ est muni d'une structure topologique. Soit $\mathcal{M}$ un MDP muni d'une structure topologique. Soit $\{\mathcal{M}_\ell\}_{\ell=0}^{L}$ l'ensemble des sous MDPs associé chacun à la restriction de $\mathcal{M}$ à une couche d'états S_ℓ, pour tout $\ell = 0, 1, \cdots, L$. L'inverse de l'ordre topologique sur le graphe réduit $\bar{G}_{\mathcal{M}}$ définit un ordre partiel $\prec_1$ sur les couches d'états $\{S_\ell\}_{\ell=0}^{L}$. Et par conséquent, il définit un ordre partiel sur les sous MDPs $\{\mathcal{M}_\ell\}_{\ell=0}^{L}$. Cet ordre partiel peut être étendu en un ordre total en résolvant les égalités par l'usage de l'ordre lexicographique sur les ensembles S_ℓ, pour tout $\ell = 0, 1, \cdots, L$. Tout autre ordre de préférence peut être utilisé afin de résoudre les égalités. Dès lors, $(\mathcal{M}, \prec_1)$ est bien un ensemble orienté muni de la propriété qu'en résolvant les sous MDPs $\mathcal{M}_\ell$ suivant l'ordre total $\prec_1$, en utilisant soit un algorithme d'itération de valeurs ou un algorithme d'itération de politiques, il est garanti de déterminer une politique ε-optimale.

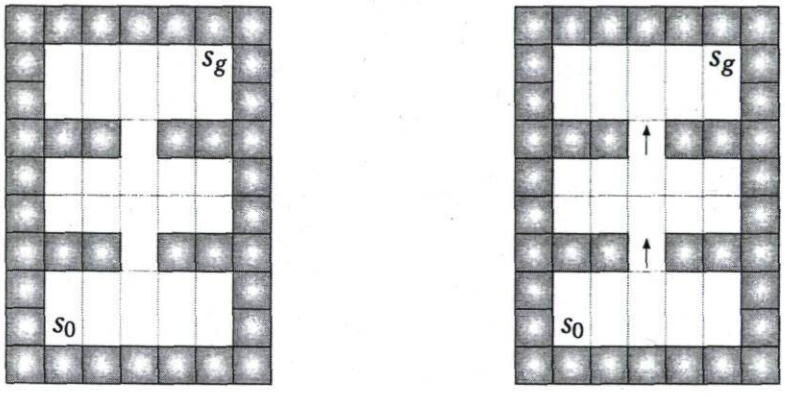
Si l'approche précédente paraît naturelle, elle souffre de certaines limites. La plus difficile à surmonter est celle où l'ensemble des états du problème constitue une seule couche d'états : le problème n'est pas muni d'une structure topologique. On pourrait également se demander qu'en est-il des performances de cette approche lorsque la dimension d'une seule couche d'états est extrêmement élevée. Notre contribution dans la résolution des processus de Markov complètement observables est motivée par ces observations. En effet, l'approche décrite ci-dessus n'est rien de plus qu'un algorithme classique tel que l'algorithme d'itération de valeurs. Pire, dans certains cas, le temps total de calcul de la politique optimale serait en faveur de l'algorithme classique car il faudrait considérer le temps inutile consacré à la vérification de l'existence d'une structure topologique.

4.2.1 L'ordre interne aux couches d'états

Nous nous intéressons maintenant à l'organisation des mises à jour à l'intérieur d'un sous MDP $\mathcal{M}_\ell$, pour tout $\ell = 0, 1, \cdots, L$. En particulier, nous recherchons un ensemble ordonné $(\mathcal{M}_\ell, \prec_2)$. On appelle ensemble ordonné $(\mathcal{M}_\ell, \prec_2)$, tout problème $\mathcal{M}_\ell \equiv \{\mathcal{M}_\ell^\kappa\}_{\kappa=0}^K$, composé de partitions de sous MDPs $\mathcal{M}_\ell$, tel que l'on puisse y associer une séquence $\mathcal{M}_\ell^0 \prec_2 \mathcal{M}_\ell^1 \prec_2 \cdots \prec_2 \mathcal{M}_\ell^{K-1} \prec_2 \mathcal{M}_\ell^K$. À la différence d'un ensemble orienté $(\mathcal{M}, \prec_1)$, un ensemble ordonné $(\mathcal{M}_\ell, \prec_2)$ ne peut être résolu en résolvant indépendamment chacune des partitions suivant l'ordre $\prec_2$. Afin de garantir la convergence vers une fonction de valeurs ε-optimale, il est en effet nécessaire de résoudre suffisamment souvent l'ensemble des sous problèmes $\{\mathcal{M}_\ell^\kappa\}_{\kappa=0}^K$, jusqu'à ce que la fonction de valeurs v_ℓ converge. Ainsi, l'ordre $\prec_2$ n'indique qu'une préférence quant à l'ordre de résolution des sous problèmes. Une fois de plus un ensemble ordonné peut être extrait d'un sous MDP $\mathcal{M}_\ell$. Cela se fait en décomposant la couche d'états S en plusieurs sous-couches $\{S_\ell^\kappa\}_{\kappa=0}^K$. Il existe de nombreux moyens de déterminer de bonnes partitions des couches S_ℓ, mais nous ne nous intéressons qu'aux approches garantissant que la phase de décomposition et d'organisation ne soit pas de coût prohibitif. En effet, certaines approches issues de la théorie des graphes permettent d'obtenir de bonnes partitions mais à coût trop exorbitant. En particulier, en utilisant le graphe de transitions sur l'ensemble des états d'une couche S_ℓ, il est possible d'en déduire les composantes biconnexes, ou triconnexes ou encore k-connexes [Hopcroft and Tarjan, 1973, Tarjan, 1972]. Seulement ces approches sont très coûteuses en temps de calcul et n'offrent pas nécessairement de bonnes séquences de mises à jour. C'est pour cette raison que nous nous sommes tournés vers une approche simple et peu coûteuse. Néanmoins, les performances de l'ordre ainsi déterminé sont encourageantes.

4.2.2 Algorithme topologique interne aux couches

Considérons un véhicule automatisé de navigation dans un environnement donné. Pour simplifier, les états de l'environnement sont identifiés par un ensemble de cellules : les cellules blanches correspondent aux états accessibles et les cellules colorées correspondent aux états inaccessibles ou aux murs, voir Figure 4.7. La mission pour ce véhicule consiste à se déplacer d'un état initial s_0 à un état final s_g. Le véhicule dispose de quatre actions motrices à savoir gauche, droite, haut et bas. Les effets de ces actions sont incertains et parfois impossibles, par exemple le véhicule ne peut transiter vers un état inaccessible. Cependant, il peut transiter vers tout autre état de son voisinage immédiat. Compte tenu de cette dynamique du système, le graphe de transitions d'un tel problème de prise de décisions séquentielles est constitué de deux seules couches d'états accessibles. L'une d'entre elles ne comprend qu'un seul état, l'état final s_g. Ainsi le problème est essentiellement sans structure topologique exploitable pour le calcul efficace d'une politique optimale.



Hallway Hallway-layered

FIGURE 4.7 – Deux exemples du problème Hallway (sans et avec couches). Lorsque l'agent atteint un état étiqueté par " → ", il ne peut retourner en arrière.

Afin de définir un ordre interne à une couche d'états S_ℓ, considérons les relations de dépendances entre les états de cette couche. En particulier, considérons les cellules dénotées s_{g-1} et s_{g-2} précédant l'état final s_g. Un ordre approximatif de mises à jour des valeurs des états s_{g-1}, s_{g-2} et s_g serait de mettre à jour tout d'abord l'état final s_g puis ses états prédécesseurs c'est à dire s_{g-1} et s_{g-2}. En effet, cet ordre favorise la propagation de la valeur $v(s_g)$ aux valeurs des états s_{g-1} et s_{g-2}. Ainsi les états s_{g-1}, s_{g-2} et s_g seraient ordonnés comme suit $s_{g-1} \prec_2 s_{g-2} \prec_2 s_g$. Puis, répéter ce processus en considérant désormais les états s_{g-1} et s_{g-2} comme états finaux, et ainsi de suite jusqu'à l'état initial s_0. L'ordre résultant est bien entendu très loin d'être optimal dans la majorité des cas, mais il est très efficace en pratique comme nous le verrons. Malheureusement, toutes les applications qui nous intéressent ne disposent pas nécessairement d'un état final s_g. Nous

devons être capable d'extraire cet ordre indépendamment de l'état final. Afin d'y parvenir, nous notons que cet ordre met à jour en priorité les états les plus éloignés de l'état initial s_0. Nous allons exploiter cette observation afin d'extraire un ordre de mises à jour interne aux couches d'états.

Le calcul d'un tel ordre d'ordonnancement des mises à jour revient au calcul de la distance minimale distance$^*(s)$ de tout état $s \in S$ à l'état initial s_0 – l'ordre $\prec_2$ est alors défini par la distance distance$^*(s)$. Nous considérons une fois de plus le graphe de transitions $G_{\mathcal{M}} = (V, E)$ associé au MDP $\mathcal{M}$ à résoudre. De plus chaque arc V est muni d'un poids de (+1). Et le nœud initial est s_0. L'idée est d'ordonner les mises à jour dans le sens décroissant des valeurs distance$^*(s)$. Intuitivement la mesure distance$^*(s)$ dénote le nombre de mises à jour exhaustives qu'il faudrait réaliser avant que l'état s ne bénéficie de la valeur de l'état sg si ce dernier existait. Pour ce faire, il suffit de déterminer le plus court chemin dans le graphe pondéré $G_{\mathcal{M}}$. Pour y parvenir, il est possible d'utiliser l'algorithme de Dijkstra nécessitant un temps de calcul de $\mathcal{O}(|E|\log|V|)$.

Lorsqu'on dispose de l'état final s_g, nous optons pour un algorithme qui permet de procéder entre autres à l'élagation des états inaccessibles. Cet algorithme se décompose en deux étapes : (1) Nous effectuons tout d'abord un parcours en profondeur du graphe $G_{\mathcal{M}}$ et collectons l'ensemble des états accessibles en partant de l'état initial s_0 ; (2) Nous considérons ensuite la transposé du graphe $G_{\mathcal{M}}$ réduit exclusivement aux états accessibles et noté $G^T_{\mathcal{M}}$. Nous procédons par la suite au parcours en largeur du graphe $G^T_{\mathcal{M}}$, en partant de l'état final s_g. L'ordre FIFO (selon acronyme anglais first-in-first-out) suivant lequel le parcours en largeur traite les nœuds est l'ordre de mises à jour que nous recherchons. Lors du traitement d'un état s, l'algorithme lui assigne sa valeur distance$^*(s)$ égale au nombre d'états précédemment traités. Il est facile de montrer qu'un algorithme d'itération de valeurs utilisant l'ordre de mises à jour décrit converge vers une politique ε-optimale – car il ne s'agit ni plus ni moins qu'un algorithme d'itération de valeurs. En particulier, cet algorithme peut être utilisé dans une seule couche d'états. L'algorithme, nommé i TVI et décrit en Figure 22, synthétise l'approche d'ordonnancement des mises à jour des couches internes.

4.2.3 Évaluation Empirique

Afin de valider notre algorithme, nous l'avons testé sur une variété de problèmes de la littérature des MDPs. Nous avons comparé au cours de ces expérimentations *i*TVI aux algorithmes classiques tels que VI [Bellman, 1957], aussi bien qu'aux algorithmes prioritisés à savoir TVI [Dai and Goldsmith, 2007] et PS [Wingate and Seppi, 2005] (resp. PI [Howard, 1960], et IPS [McMahan et al., 2005]). Nous avons également considéré une version

Algorithme 22 Algorithme *i*TVI.

procedure *i*TVI(S, A, T, R, λ)
  pour tout S_ℓ dans l'ordre $\prec_1$ **faire**
    Calculer l'ordre $\prec_2$ de mises à jour à l'intérieur de chaque couche S_ℓ.
    répéter
      pour tout S_ℓ^κ dans l'ordre $\prec_2$ **faire**
        Mettre à jour la sous-couche d'états S_ℓ^κ.
      fin pour
    jusqu'à critère d'arrêt est atteint.
  fin pour
fin procedure

de *i*TVI munie d'une heuristique et notée *i*TVI$^+$. C'est le cas de certains des algorithmes auxquels nous nous comparons. Comme les algorithmes topologiques tendent à éviter les mises à jour inutiles, nous nous intéressons plus particulièrement à l'analyse de la vitesse de convergence vers une politique ε-optimale. Nous prouvons empiriquement que *i*TVI converge bien plus vite sur les problèmes testés en comparaison aux autres approches.

L'ensemble des algorithmes a été exécuté sur la même machine disposant d'un processeur Intel Core Duo 1.83GHz avec une mémoire de 1Giga. Le temps total d'exécution ainsi que le nombre de mises à jour effectuées ont été collectés minutieusement.

4.2.4 Discussion et méthodes associées

Dans cette section, nous discutons des principales limites de la programmation dynamique topologique face aux POMDPs. En particulier, nous situons la programmation dynamique topologique par rapport aux autres approches susceptibles d'apporter des solutions à ces mêmes problèmes.

La malédiction de la dimension

L'une des principales sources de complexité des processus décisionnels de Markov complètement observables est la malédiction de la dimension. Bien que la complexité de résolution des MDPs soit classée P-complet, il n'est pas toujours très facile de déterminer la politique optimale ou même ε-optimale. En effet, le problème peut devenir très riche et subtile lorsque l'on introduit l'incertitude liée aux effets des actions. Mais le principal challenge apparaît lorsque le nombre d'états |S |, le nombre d'actions $|A|$ et les va-

algorithme	temps (sec.)			# mises à jour
	en ligne	hors ligne	total	
Statistiques pour les modèles de MDPs de type M1	large (20424 states, 4 actions)			
Gauß-Seidel VI	119	0	119	10^8
PS [Wingate and Seppi, 2005]	97.8	0	97.8	141880
TVI [Dai and Goldsmith, 2007]	0.05	6.18	6.23	18864
i-TVI	0.23	0	**0.23**	107800
i-TVI$^+$	0.05	6.18	6.23	**18840**
Statistiques pour les modèles de MDPs de type M1	large (41856 states, 4 actions)			
Gauß-Seidel VI	371	0	371	10^9
TVI [Dai and Goldsmith, 2007]	29.8	104	133.8	**1858100**
i-TVI	31.1	0	**31.1**	6425272
i-TVI$^+$	31.3	104	135	**1858100**
Statistiques pour les modèles de MDPs de type M2	large-bi (20720 states, 4 actions)			
Gauß-Seidel VI	138	0	138	10^8
PS [Wingate and Seppi, 2005]	102	0	102	145212
TVI [Dai and Goldsmith, 2007]	0.39	7.96	8.35	99892
i-TVI	0.25	0	**0.25**	110192
i-TVI$^+$	0.05	7.96	8.01	**18864**
Statistiques pour les modèles de MDPs de type M2	larger-bi (41856 states, 4 actions)			
Gauß-Seidel VI	381	0	381	10^{10}
TVI [Dai and Goldsmith, 2007]	30.1	105	135.1	**5687312**
i-TVI	32.2	0	**32.2**	5704068
i-TVI$^+$	30.5	105	135.5	**5687312**

TABLE 4.1 – Statistics for different MDP models.

riables d'encodage des états croient. En d'autres termes, lorsque nous n'avons plus à faire à quelque dizaines d'états mais à des millions d'états ; lorsque les encodages utilisés pour ces états ne sont plus des entiers mais des vecteurs à plusieurs composantes. Considérons par exemple la version multi-agents du problème de contrôle de véhicules sur une piste de rallye. Les états d'un véhicule sont encodés en utilisant des vecteurs de réels de la forme $(x, y, \dot{x}, \dot{y})$ représentant la position et la vitesse du véhicule dans une grille 2D . Comme il y a plusieurs véhicules, ces variables seront requises pour chacun des véhicules. Ainsi, pour un tel problème, les tailles de l'espace des actions et l'espace des états sont exponentielles. On a donc à faire à une malédiction de la dimension. Faire face à ce phénomène est un problème fondamental de la théorie des processus décisionnels de Markov complètement observables.

La première classe de techniques qui a fait face à ce problème suppose que l'on dis-

algorithme	temps (sec.)			# mi
	en ligne	hors ligne	total	
Statistiques pour les modèles de MDPs de type M1 and A1-A2	c-large (20424 states, 4 actio			
Gauß-Seidel VI	22.7	0	22.7	
HDP ($h = 0$)	310	0	310	
LRTDP ($h = 0$)	2.70	0	2.70	1
LAO* ($h = 0$)	6.50	0	6.50	9
HDP ($h = h_{\min}$)	0	21.0	21.0	
LRTDP ($h = h_{\min}$)	2.14	21.0	23.14	1
LAO* ($h = h_{\min}$)	4.20	21.0	25.2	6
TVI (Dai & Goldsmith 2007)	1.43	0.43	1.86	2
i-TVI	0.60	0	**0.60**	2
i-TVI$^+$	1.23	0.43	1.66	**1**
Statistiques pour les modèles de MDPs de type M1 and A1-A2	c-larger (41856 states, 4 actio			
Gauß-Seidel VI	66.4	0	66.4	
HDP ($h = h_{\min}$)	0	66.5	66.5	
TVI (Dai & Goldsmith 2007)	52.9	91.3	144	1.
i-TVI	50.8	0	**50.8**	1.
i-TVI$^+$	17.7	91.3	109	**8**
Statistiques pour les modèles de MDPs de type M2 and A1-A2	c-large-bi (20720 states, 4 acti			
Gauß-Seidel VI	26.8	0	26.8	
HDP ($h = h_{\min}$)	0	26.9	26.9	
LRTDP ($h = h_{\min}$)	370	26.9	397	
TVI (Dai & Goldsmith 2007)	16.0	11.2	27.2	4
i-TVI	15.7	0	15.7	4
i-TVI$^+$	3.33	11.2	**14.5**	**2**
Statistiques pour les modèles de MDPs de type M2 and A1-A2	c-larger-bi (41856 states, 4 act			
Gauß-Seidel VI	65.3	0	65.3	
HDP ($h = h_{\min}$)	0	65.6	65.6	
TVI (Dai & Goldsmith 2007)	55	90.5	145	1.
i-TVI	50.3	0	**50.3**	1.
i-TVI$^+$	3.4	90.5	93.9	**8**

TABLE 4.2 – Statistiques pour différents modèles de MDPs.

pose d'un modèle de simulation du MDP. C'est à dire que sachant un état s et une action a, l'algorithme peut accéder à un modèle (boîte noire) qui lui retourne un état successeur s'. Les algorithmes de recherche de politiques supposent une telle représentation de la dynamique du système [Baxter and Bartlett, 1999, Ng and Jordan, 2000, Ng et al., 1999, Sutton et al., 1999]. Ils utilisent cette représentation afin de simuler des trajectoires d'exé-

cutions et déterminer de bonnes politiques d'une classe de politiques paramétriques. Ces méthodes sont pertinentes lorsque le MDP dispose d'espaces d'actions et d'états continus.

La seconde classes de techniques suppose que la dynamique du système peut être encodée sous une forme de représentation compacte comme par exemple : celle des réseaux bayésiens dynamiques (DBNs) [Boutilier et al., 1996] des arbres de décisions ; des diagrammes algébriques de décisions (ADDs) [Hoey et al., 1999] ; ou encore des langages de planification probabiliste de type STRIPS [Dearden and Boutilier, 1997]. Malheureusement, la majorité des algorithmes de résolution des MDPs utilisent une représentation explicite des états et de la dynamique du système. Certaines approches tentent de pallier à cet état de fait. Par exemple, [Boutilier et al., 2000] résolvent un MDP qui utilise un DBN afin de définir la dynamique du système et un arbre de décisions afin de représenter le modèle de récompenses. Pour ce faire, il généralise les méthodes de programmation dynamique classique afin de construire des fonctions de valeurs et des politiques encodées via des arbres de décisions. [Guestrin et al., 2002] calcule une approximation de la fonction de valeurs en ayant recours à un modèle factorisé du MDP muni d'une propriété additive sur les fonctions de récompenses. Les techniques de réduction du modèle prennent en entrée un MDP factorisé et construisent un MDP bien plus petit, avec une représentation explicite de la dynamique construite en agrégeant les états dits équivalents. Cette approche a initialement été proposée par [Dean and Givan, 1997, Givan et al., 2003], puis elle a connu de nombreux raffinements liés à la notion d'équivalence. En particulier, [Ferns et al., 2004, Kim and Dean, 2003] ont proposé de n'agréger les états que selon un critère d'équivalence dit faible. Le lecteur peut percevoir la programmation dynamique topologique comme une des ces approches, à l'exception que notre approche n'utilise pas la dynamique du système afin de déterminer les partitions du MDPs – seul le graphe de transitions $G_{\mathcal{M}}$ est utile. Cette différence est de taille car elle nous permet d'obtenir des partitions plus facilement même si les classes d'applications sont plus restreintes en comparaison aux techniques d'équivalence ou de bissimulation [Dean and Givan, 1997, Givan et al., 2003]. Ces dernières souffrent néanmoins du coût exorbitant nécessaire afin de réduire le problème initial en un problème de taille plus petite.

Plusieurs chercheurs ont suggéré différentes méthodes de décompositions hiérarchiques permettant de représenter les macro actions (ou options) et les macro états (ou couches d'états) dans des MDPs de très grandes tailles [Dietterich, 1998]. [Dean and Lin, 1995] ont défini les fondements des techniques de décomposition des MDPs en décrivant en cadre général pour la décomposition de MDPs en sous MDPs de plus petites tailles, de sorte que ces derniers puissent être résolus indépendamment les uns des autres. Des raffinements de cette idées ont notamment conduit à l'idée des macro actions comme un moyen de résoudre efficacement les MDPs [Laroche et al., 1999, Meuleau et al., 1998, Parr, 1998, Precup et al., 1998, Singh and Cohn, 1998]. Sachant une partition de l'espace des

états, un ensemble de politiques appelées macro actions, est calculé pour chaque partition d'états. Une fois la dynamique du système transformées de sorte à être en accord avec les macro actions, un MDP abstrait (contenant uniquement des états périphériques aux partitions) est résolu en utilisant ces macro actions. Le principal inconvénient avec ces techniques est que la qualité de la solution calculée dépend grandement du nombre et de la qualité des macro actions pré-calculées. Afin de garantir, une presqu'optimalité de la fonction de valeurs recomposée, il faudrait calculer un nombre exponentiel de macro actions, à moins que le problème soit faiblement couplé [Meuleau et al., 1998, Mouaddib and Zilberstein, 1998, Parr, 1998]. En outre, ces techniques présupposent souvent l'existence d'une partition du problème. Récemment, des extensions de ces travaux ont été suggérés afin de faire face à des classes particulières des MDPs, y compris les MDPs multi objectifs [Lane and Kaelbling, 2002] et les MDPs sous contraintes de ressources [Dolgov and Durfee, 2004b]. En particulier, [Dolgov and Durfee, 2004b, 2006b] utilise la programmation mathématique (programmation linéaire mixte) en y incorporant les propriétés liées aux interactions faibles entre les membres d'une équipe d'agents ayant à réaliser une tâche en commun. Cette approche a montré des performances intéressantes car elle permet d'éviter l'explicite énumération de l'ensemble des états du système. De façon, similaire notre approche procède en décomposant le MDP à résoudre en plusieurs partitions, puis effectue les mises à jour de la fonction de valeurs suivant un ordre sur les partitions. À la différence de méthodes de décomposition discutées, nous résolvons nos partitions suivant un ordre prédéfini et parvenons à garantir l'optimalité de la politique retournée. Encore plus important, nous ne résolvons que les partitions accessibles à partir d'un état initial s_0.

Il existe de nombreuses tentatives visant à faire usage de l'accessibilité des états afin de contraindre l'espace des états sur lesquels il est utile de planifier [Barto et al., 1993, Bonet and Geffner, 2003a,b, Hansen and Zilberstein, 2001, McMahan et al., 2005, Smith and Simmons, 2006]. Il s'agit essentiellement d'algorithme de mises à jour de la fonction de valeurs à travers des trajectoires simulées d'états. En mettant à jour la fonction de valeurs uniquement au travers de trajectoires, ces approches garantissent de ne mettre à jour que des états accessibles et cela proportionnellement à la probabilité d'y accéder. La majorité de ces approches sont des algorithmes de recherche heuristique permettant d'explorer uniquement un sous-ensemble accessible des états d'un MDP disposant d'un très large espace d'états. Ces techniques consistent souvent à l'alliance entre une borne inférieure et une borne supérieure sur la fonction de valeurs optimales, et une connaissance de l'état initial du système s_0. À la différence des algorithmes classiques de programmation dynamique, à savoir l'algorithme d'itération de valeurs ou l'algorithme d'itération de politiques, ces algorithmes ne calculent la politique optimale que sur le sous-ensemble des états accessibles de l'état initial. Cela permet notamment de consacrer les efforts de mises à jour uniquement sur le sous-ensemble des états accessibles de l'état initial s_0. Malheureusement, dans certains cas ces approches heuristiques peuvent montrer des per-

formances très décevantes. Dans certaines applications en effet, les trajectoires simulées peuvent se trouver coincées dans un sous-espace des états accessibles et ne pas pouvoir en sortir. Cela a pour conséquence d'accroître considérablement le temps de planification. La programmation dynamique topologique parvient à surmonter ce problème en résolvant séparément chaque couche d'états – ceci est essentiellement dû au fait que les algorithme de programmation topologique pour MDPs que nous avons introduit ne procèdent pas aux mises à jour de la fonction de valeurs via des trajectoires.

La lenteur du taux de convergence

Pour converger correctement dans le cas d'un MDP général, les algorithmes de programmation dynamique doivent de façon itérative mettre à jour les valeurs de tous les états jusqu'à ce que cette mise à jour converge. Cependant, lorsque la dimension du MDP est considérable, même une seule mise à jour de la fonction de valeurs peut s'avérer extrêmement coûteuse. Cela explique pourquoi le taux de convergence des algorithmes de programmation dynamique classique est souvent lent lorsqu'ils font face aux MDPs de très grande taille. Cette observation souligne la nécessité d'améliorer le taux de convergence des algorithmes de résolution des MDPs. Le taux de convergence peut être considérablement amélioré en évitant les mises à jour inutiles ou redondantes de la fonction de valeurs. Les méthodes dites asynchrones parviennent en effet à améliorer le taux de convergence des algorithmes classiques, à la différence des approches synchrones. Le Chapitre 2, Section 2.2.2, donne un état de l'art des approches asynchrones. Malheureusement, ces méthodes procèdent au calcul dynamique de séquences de mises à jour de la fonction de valeurs. Ces calculs dynamiques contrastent avec les ordres statiques suggérés par la programmation dynamique topologique. Or, la nature dynamique du calcul des ordres de mises à jour de la fonction de valeurs augmente considérablement le coût total de la planification.

4.3 Résoudre des POMDPs

Les processus décisionnels de Markov partiellement observables (POMDPs) offrent un puissant cadre de formalisation et résolution des problèmes de prise de décisions séquentielles dans des environnements pour lesquels les agents ont des perceptions imparfaites et dont les effets de leurs actions sont incertaines. Bien que les algorithmes exactes existent, ils ne peuvent résoudre que des problèmes de tailles insignifiantes, à l'opposé les méthodes approximatives, telles que les méthodes à base de croyances (Chapitre 2, Sec-

tion 2.5.1), ont montré de bien belles aptitudes à résoudre des problèmes de tailles plus conséquentes. Dans de nombreux cas de figures, les environnements réels sont munis de structures. De façon similaire aux cas des processus décisionnels de Markov complètement observables, un certain nombre de structures ont été exploitées dans le cadre de processus décisionnels de Markov partiellement observables [Pineau et al., 2003a, Shani et al., 2008]. Les algorithmes exploitant ces structures ont bien sûr montré de bien meilleurs performances que les approches génériques.

Dans cette section, nous montrons que lorsqu'un environnement est doté d'une structure topologique, il est possible de spécialiser les algorithmes à base de croyances afin de focaliser leur attention sur un seul sous-ensemble de croyances à la fois. Il s'agit en théorie du sous-ensemble de croyances où la fonction de valeurs peut être significativement améliorée tout en réduisant les mises à jour redondantes ou inutiles. En effet, lors de l'applications des algorithmes génériques, bon nombre de mises à jour exécutées correspondent à des mises à jour dont les effets sur la fonction de valeurs sont soit nuls soit insignifiants [Pineau et al., 2003b, Shani et al., 2007, Smith and Simmons, 2005]. Comme les opérations de mises à jour à base de croyances sont au cœur des algorithmes à base de croyances, réduire le nombre de mises à jour exécutées revient à accélérer considérablement la vitesse de convergence d'un algorithme.

Dans ce contexte, nous proposons l'algorithme TOP (selon l'acronyme anglais « Topological Order Planner ») qui fait usage de l'ordre topologique du MDP sous-jacent au POMDP à résoudre afin de trouver les bonnes trajectoires de croyances à mettre à jour. L'algorithme TOP groupe ensemble des états mutuellement dépendants – il s'agit des couches d'états. TOP crée ainsi un graphe acyclique de couches d'états. Les couches d'états comme dans le cas des MDPs sont résolus dans le sens inverse de l'ordre topologique induit par le graphe acyclique des couches d'états. Les trajectoires de croyances sont orientées dans le sens des couches d'états résolvables. Une fois qu'une couche est résolue, les trajectoires sont arrêtées lorsqu'elles atteignent cette couche – il en résulte que les trajectoires se raccourcissent au fur et à mesure des mises à jour et par conséquent il y a de moins en moins de mises à jour effectuées. Cependant, avant de choisir TOP comme algorithme pour la résolution d'un POMDP, nous devons nous assurer que le MDP sous-jacent au POMDP à résoudre soit muni d'une structure topologique. Dans ce qui suit, nous décrivons une famille d'algorithmes topologiques et nous expliquons quand est-ce que cette famille d'algorithmes est susceptible de déterminer une politique optimale. Nous présentons ensuite deux instances de cette famille – tout en expliquant les difficultés d'implémentation des différentes parties de ces algorithmes dans le cadre des POMDPs. Nous proposons enfin des tests expérimentaux intensifs, sur divers environnements de la littérature des POMDPs, montrant combien ces algorithmes gagnent en efficacité plus le nombre de couches est grand.

4.3.1 Structure topologique dans les POMDPs

Une structure topologique dans l'espace des états d'un MDP sous-jacent du POMDP à résoudre peut induire une structure topologique dans l'espace des croyances du dit POMDP. Considérons par exemple une croyance b tel que : d'une part, la probabilité $b(s)$ soit non nulle pour certains états s d'une couche S_ℓ ; et d'autres part, toutes les probabilités b(si) soient nulles pour tout état si appartenant aux couches parents de S_ℓ. En d'autres termes, si un agent observe la croyance b, il n'y a aucune chance qu'il se retrouve à moyen ou à long terme dans une croyance b', telle que la probabilité $b'(s')$ soit non nulle pour tout état si appartenant à une couche parent de S_ℓ. Considérons par la suite une croyance b' successeur (pas nécessairement immédiat) de la croyance b. En outre, supposons que la croyance b ait une probabilité nulle sur l'ensemble des états dans la couche S_ℓ. Alors, bien que l'agent puisse observer la croyance b puis la croyance b', il est impossible qu'il puisse observer la croyance bi puis la croyance b. Cependant, comme l'espace des croyances est infini, il est difficile en général d'identifier ce type de relations. En conséquence, il est difficile d'identifier les couches de croyances c'est à dire les composantes fortement connexes au sein de l'espace des croyances. De plus, il n'y a pas de lien direct entre les couches d'états du MDP sous-jacent et les couches de croyances du POMDP associé. En effet, il est possible qu'une croyance ait une probabilité non nulle sur des états appartenant à diverses couches d'états. Néanmoins, dans certaines applications, il est possible d'identifier assez facilement les couches de croyances.

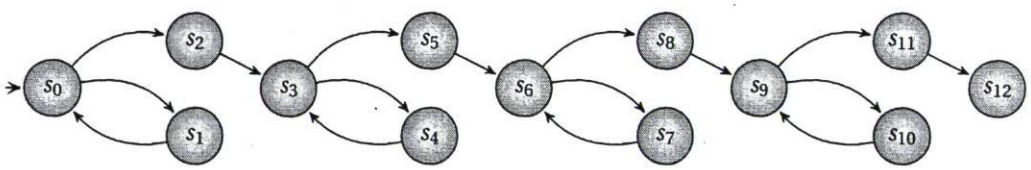


FIGURE 4.8 – Représentation graphique d'une chaîne de montage de véhicules.

Considérons à nouveau notre chaîne de montage de véhicules dont le graphe de transitions est illustré en Figure 4.8. En plus de disposer d'une structure topologique sur l'espace des états, cette application est munie d'une autre caractéristique intéressante. En effet, dans chacune des stations de montage des véhicules, les robots chargés du montage des pièces, sont munis de capteurs. Ces derniers servent à identifier l'état dans lequel se trouve le véhicule au sein d'une station. Ainsi, lorsque le véhicule transite d'une station à une autre, il est impossible que le système perçoive les mêmes observations dans les deux stations. Les observations sont alors caractéristiques de la station dans laquelle le véhicule se trouve. Dans ce contexte, l'incertitude de l'agent quant à l'état du système est restreinte aux observations au sein d'une seule station, c'est à dire de sous-ensembles d'états à la fois. Si de plus ces sous-ensembles d'états sont organisés suivant une structure topologique, alors les croyances de l'agent peuvent également être organisée suivant une

structure topologique. Le théorème suivant énonce des conditions suffisantes à l'identification de couches de croyances dans un POMDP.

Théorème 1. *Soit un POMDP $\mathcal{M}$, supposons qu'il existe :*

1. *d'une part, une fonction bijective $f : \{\Omega_\tau\}_{\tau=0}^N \to \{S_\tau\}_{\tau=0}^N$ qui associe à une partition Ω_τ sur l'espace des observations une unique partition S_τ sur l'espace des états – tel que, pour tout état $s' \in S_\tau$ et pour toute action $a \in A$, $\sum_{\omega\in\Omega_\tau} O(\omega|s', a) = 1$.*
2. *d'autre part, une dépendance causale entre les ensembles d'états S_τ telle que le graphe de transition sur ces ensembles soit acyclique.*

Alors $\mathcal{M}$ est muni d'une structure topologique, et les couches B_τ de croyances sont données par :

$$B_\tau \equiv \mathcal{P}(f(\Omega_\tau)) = \mathcal{P}(S_\tau), \quad \textit{pour tout } \tau = 0, 1, \cdots, N \tag{4.2}$$

Démonstration. La preuve du théorème ci-dessus est construite par l'absurde. Supposons qu'il existe deux croyances b et b' appartenant à deux ensembles de croyances B_τ et $B_{\tau+1}$, construits comme décrit ci-dessus – tels qu'il existe une séquence de transitions menant de b à b' et vice versa. En particulier, cela revient à supposer qu'il existe un chemin menant d'un état $s \in S_\tau$ à un état $s' \in S_{\tau+1}$ et vice versa. Or, si un tel chemin existe alors les états s et s' appartiennent à la même couche d'états. Cela est impossible car par hypothèse il existe une dépendance causale stricte entre S_τ et $S_{\tau+1}$ (le graphe de transitions des ensembles S_τ et $S_{\tau+1}$ est acyclique). Il s'ensuit alors qu'il n'existe pas de séquence de transitions menant de b à b' et vice versa : les ensembles $\{B_\tau\}_{\tau=0}^N$ sont bien de couches d'états croyance pour le POMDP $\mathcal{M}$. □

Algorithme 23 Algorithme de couplage.

1: **procedure** COUPLAGE(S_ℓ)
2: Identifier l'ensemble Ω_τ des observations ω telles que : $O(s, \cdot, \omega) > 0$, pour tout $s \in S_\tau$
3: S'il existe un couplage entre Ω_τ et S_τ retourner la paire (Ω_τ, S_τ).
4: Sinon, ajouter dans l'ensemble S_τ toutes les couches d'états S_ℓ comprenant les états s, tels que : $O(s, \cdot, \omega) > 0$, pour tout $\omega \in \Omega_\tau$.
5: **fin procedure**

Le lecteur notera que les ensembles $f(\Omega_\tau)$ identifiés dans le théorème ci-dessus ne sont pas nécessairement des couche d'états $\{S_\ell\}_{\ell=0}^L$. Il peut s'agir d'un ensemble de plusieurs couches d'états. L'algorithme du couplage « ensemble d'observations – ensemble d'états » est décrit à la Figure 23. L'algorithme de couplage construit de façon incrémentale la paire (Ω_τ, S_τ). Tout d'abord, il sélectionne une composante fortement connexe S_τ non couplée jusqu'ici. Puis, il identifie l'ensemble des observations Ω_τ perçu lorsque l'agent se trouve dans un des états $s \in S_\tau$. L'algorithme vérifie ensuite le couplage (Ω_τ, S_τ). S'il est

valide, ce couple est retourné. Et l'algorithme reprend avec un nouvel ensemble S_τ initialement égale à une nouvelle couche d'états S_ℓ. Si certains des états s où l'agent peut percevoir une observation $\omega \in \Omega_\tau$ ne sont pas inclut dans S_τ, l'algorithme identifie leurs couches d'états et les rajoute à l'ensemble S_τ, puis reprend la vérification du couplage.

Algorithme 24 Algorithme de génération des couches de croyances.

procedure GÉNÉRATION DES COUCHES DE CROYANCES
  Poser $\tau \leftarrow 0$, $\ell \leftarrow L$
  pour tout S_ℓ **faire**
    si S_ℓ non couplée **alors**
      $S_\tau \leftarrow S_\ell$
      $(\Omega_\tau, S_\tau) \leftarrow$ COUPLAGE(S_τ)
      Poser $f(\Omega_\tau) \leftarrow S_\tau$.
      Calculer la couche de croyances B_τ donnée par $B_\tau \leftarrow \mathscr{P}(f(\Omega_\tau))$.
    fin si
  fin pour
fin procedure

Une fois l'ensemble des couples identifié il est facile de construire l'ensemble des couches de croyances comme l'illustre la Figure 24. Bien que cet algorithme identifie des couches de croyances, il n'est pas vrai que deux croyances quelconques d'une même couche soient mutuellement accessibles. En effet, il se pourrait qu'à l'intérieure de cette couche il existe d'autres couches de croyances, en particulier les couches constituées des croyances inaccessibles. Néanmoins le critère de dépendance retenu ici est celui lié à l'impact sur la fonction de valeurs. En effet, toute modification de la valeur d'une croyance dans une couche peut en principe influencer la valeur de tout autre croyance dans la même couche – qu'ils soient ou non mutuellement accessibles.

4.3.2 Planification topologique exacte

Lorsque le POMDP $\mathscr{M}$ à résoudre est muni d'une structure topologique, un algorithme exact du calcul de la fonction de valeurs ε-optimales peut être défini. Cet algorithme générique décrit en Figure 4.9, démontre comment exploiter la structure topologique afin de surmonter à la fois la complexité liée à la malédiction de la dimension du POMDP à résoudre mais aussi la complexité liée à la malédiction de l'historique c'est à dire l'espace des croyances.

La malédiction de la dimension est surmontée en effectuant exclusivement des mises à jour partielles. On appelle une mise à jour partielle, toute modification de la fonction de

Algorithme topologique d'itération de valeurs pour les POMDPs

1. Sélectionner B_τ suivant l'ordre inverse de l'ordre topologique sur $\{B_\tau\}_\tau$.
2. Mettre à jour la fonction de valeurs v_τ exclusivement sur les états S_τ.
3. S'il n'y a plus de couches de croyances, retourner v donnée par :

$$v(b) = v_\tau(b), \quad \text{si } b \in B_\tau, \text{ pour tout } \tau = 0, 1, \cdots, N. \tag{4.3}$$

♣

FIGURE 4.9 – Algorithme d'itération de valeurs topologique pour POMDPs.

valeurs exclusivement sur un sous-ensemble des états $S_\tau \subseteq S$, pour tout $\tau = 0, 1, \cdots, N$.

La malédiction de l'historique est surmontée : d'une part, en effectuant un échantillonnage des croyances accessibles comme expliqué par [Pineau et al., 2003b] ; d'autre part, en associant à chaque croyance b un ensemble B_τ. Ce dernier point a pour but d'organiser les mises à jour des croyances suivant l'ordre de priorité des ensembles B_τ. L'implémentation des mises à jour partielles quant à elle mérite cependant un certain nombre de détails.

La processus de calcul de la fonction de valeurs ε-optimales v_ε^* se déroule de façon inductive. On commence tout d'abord par calculer la fonction de valeurs v_N^* ε-optimale sur l'ensemble des croyances B_N comme suit : pour tout $b \in B_N$,

$$v_N^*(b) = \max_{a \in A} R(b, a), \quad \text{pour tout } b \in B_N. \tag{4.4}$$

En connaissance de v_N^* et en vertu de la dépendance causale entre les ensembles B_{N-1} et B_N, il est par la suite possible de calculer la fonction de valeurs v_{N-1}^* comme suit :

$$v_{N-1}^*(b) = \max_{a \in A} \left\{ R(b, a) + \lambda \sum_{\omega \in \Omega_{N-1}} \mathbb{P}(b' | b, a, \omega) v(b') \right\} \tag{4.5}$$

$$v(b') = \begin{cases} v_N^*(b') & \text{si } b' \in B_N, \\ v(b') = v_{N-1}^*(b') & \text{sinon.} \end{cases} \tag{4.6}$$

L'équation 4.5 s'explique par le fait que toute croyance $b \in B_{N-1}$ ne peut avoir comme croyance successeur b', qu'une croyance appartenant exclusivement soit à B_{N-1} soit à B_N, par construction des couches de croyances $\{B_\tau\}_{\tau=0,1,\cdots,N}$. Ainsi, la fonction de valeurs v_τ^* pour tout $\tau = 0, 1 \cdots, N$ est donnée par l'équation : pour tout $b \in B_\tau$,

$$v_\tau^*(b) = \max_{a \in A} \left\{ R(b, a) + \lambda \sum_{\omega \in \Omega_\tau} \mathbb{P}(b' | b, a, \omega) v(b') \right\} \tag{4.7}$$

où $v(b') = v_\ell^*(b')$ si $b' \in B_\ell$, pour tout $\ell = \tau, \tau+1, \cdots, N$. Bien que les équations 4.4, 4.5, et 4.7 soient valides, la nature continue des couches $\{B_\tau\}_{\tau=0,1,\cdots,N}$ rend difficile la résolution directe de ses équations. Afin de pallier à cette difficulté, nous proposons un opérateur $\mathbb{H}$ dont l'application itérative sur une fonction de valeurs v_τ pour tout $\tau = 0, 1, \cdots, N$, permet progressivement de déterminer une fonction de valeurs ε-optimales v_τ^*.

Opérateur de mises à jour de v_τ.

1. Nous commençons tout d'abord par calculer les ensembles de fonctions de valeurs intermédiaires $\Lambda_\tau^{a,*}$ et $\Lambda_\tau^{a,\omega}$: pour toute action $a \in A$, toute observation $\omega \in \Omega_\tau$, pour tout état $s \in S_\tau$, et la fonction de valeurs courantes $v \equiv \Lambda$:

$$\Lambda_\tau^{a,*} \leftarrow v_\tau^{a,*}(s) = R(a,s) \tag{4.8}$$

$$\Lambda_\tau^{a,\omega} \leftarrow v_\tau^{a,\omega}(s) = \lambda \sum_{s' \in S_{\tau:N}} \mathbb{P}(s'|s,a,\omega) u(s'), \quad u \in \Lambda \tag{4.9}$$

où $S_{\tau:N} = \cup_{\ell=\tau}^{N} S_\ell$ et $u \in \Lambda$ est une fonction de valeurs définie sur un sous-ensemble S_ℓ avec $\ell \in \{\tau+1, \tau+2, \cdots, N\}$.

2. Nous créons par la suite un ensemble de fonctions de valeurs Λ_τ^a pour toute action $a \in A$, en effectuant une somme croisée sur l'ensemble des observations $\omega \in \Omega_\tau$:

$$\Lambda_\tau^a \equiv \Lambda_\tau^{a,*} \oplus \Lambda_\tau^{a,\omega_1} \oplus \Lambda_\tau^{a,\omega_2} \oplus \cdots \oplus \Lambda_\tau^{a,\omega_{|\Omega_\tau|}} \tag{4.10}$$

3. Par la suite, nous prenons l'union des ensembles Λ_τ^a :

$$\Lambda_\tau \equiv \cup_{a \in A} \Lambda_\tau^a \tag{4.11}$$

Si deux ensembles successifs $\mathbb{H}\Lambda_\tau$ et Λ_τ ne diffère que d'un petit réel ε, c'est à dire $\|\mathbb{H}\Lambda_\tau - \Lambda_\tau\|_\infty \leqslant \varepsilon$ alors la fonction de valeurs v_τ^* est donnée par $v_\tau^*(b) = \max_{v_i \in \Lambda_\tau} v_i \cdot b$, pour toute croyance $b \in B_\tau$. Sinon retourner à l'étape 1.

4. Enfin, l'ensemble des fonctions de valeurs Λ_τ est inclus dans l'ensemble des fonctions de valeurs Λ. Lorsque $\tau = 0$, on peut alors extraire la fonction de valeurs ε-optimales v_ε^* comme suit : $v_\varepsilon^*(b) = v_\tau^*(b)$ si $b \in B_\tau$, pour tout $\tau = 0, 1, \cdots, N$.

FIGURE 4.10 – Opérateur de mises à jour de la fonction de valeurs v_τ.

L'opérateur $\mathbb{H}$ détermine la fonction de valeurs ε-optimales v_ε^* en procédant aux cal-

culs des fonctions de valeurs ε-optimales v_τ^* pour tout $\tau = 0, 1, \cdots, N$ suivant l'inverse de l'ordre topologique sur les couches de croyances $\{B_\tau\}_{\tau=0}^N$, comme illustrer à la Figure 4.10. Cet opérateur peut être vu comme une adaptation de l'opérateur classique de mises à jour exacte de la fonction de valeurs d'un POMDP, décrite au Chapitre 2 à la Figure 2.9. De même que pour l'opérateur classique, il est également possible d'élaguer au fur et à mesure les fonctions de valeurs dominées dans les ensembles $\Lambda_\tau^{a,*}$, $\Lambda_\tau^{a,\omega}$, Λ_τ^a, et Λ_τ en utilisant le programme linéaire décrit au Chapitre 2. Le Théorème suivant garantit que toute application successive de l'opérateur $\mathbb{H}$ sur l'ensemble Λ comme décrit à la Figure 4.10, conduit non seulement au calcul des fonctions de valeurs ε-optimales v_τ^* pour tout $\tau = 0, 1, \cdots, N$ mais et aussi à la fonction de valeurs ε-optimales v_ε^*.

Théorème 2. *L'opérateur $\mathbb{H}$ est un opérateur de contraction pour toute fonction de valeurs v_τ définie sur une couche B_τ, pour tout $\tau = 0, 1, \cdots, N$, et pour la fonction de valeurs v définie sur l'espace des croyances $\triangle S$.*

Démonstration. La preuve de ce théorème est construite par induction. Considérons l'hypothèse suivante : soient deux fonctions de valeurs v_τ et v'_τ définies sur la couche B_τ, pour tout $\tau = 0, 1, \cdots, N$, il vient alors que :

$$\|\mathbb{H}v_\tau - \mathbb{H}v'_\tau\|_\infty \leqslant \lambda\|v_\tau - v'_\tau\| \tag{4.12}$$

Tout d'abord, notons que la fonction de valeurs ε-optimales v_N^* définie sur la couche B_N est donnée par

$$v_N^*(b) = \max_{a\in A} R(b,a) \tag{4.13}$$

pour toute croyance$b \in B_N$. Ainsi $\mathbb{H}v_N = v_N$, l'hypothèse d'induction est vraie pour $\tau = N$. Supposons que l'hypothèse d'induction soit vraie pour $\tau \geqslant \ell$, montrons qu'il en est de même pour $\tau = (\ell - 1)$. Soient deux fonctions de valeurs $v_{\ell+1}$ et $v'_{\ell-1}$ définies sur la couche $B_{\ell-1}$. On a successivement : pour toute croyance $b \in B_{\ell-1}$,

$$\begin{aligned} \mathbb{H}v_{\ell-1}(b) - \mathbb{H}v'_{\ell-1}(b) \leqslant \quad & R(b,a) + \lambda\sum \mathbb{P}(b'|b,a,\omega)v(b') \\ & - \big(R(b,a) + \lambda\sum \mathbb{P}(b'|b,a,\omega)v'(b')\big) \end{aligned} \tag{4.14}$$

$$\begin{aligned} \mathbb{H}v_{\ell-1}(b) - \mathbb{H}v'_{\ell-1}(b) &= \lambda\sum \mathbb{P}(b'|b,a,\omega)\big(v_{\ell-1:N}(b') - v'_{\ell-1:N}(b')\big) && (4.15)\\ &= \lambda\sum \mathbb{P}(b'|b,a,\omega)\big(v_{\ell-1}(b') - v'_{\ell-1}(b')\big) && (4.16)\\ &\leqslant \lambda\sum \mathbb{P}(b'|b,a,\omega)\|v_{\ell-1} - v'_{\ell-1}\|_\infty && (4.17)\\ &= \lambda\|v_{\ell-1} - v'_{\ell-1}\|_\infty && (4.18) \end{aligned}$$

où les fonctions de valeurs v et v' sont les fonctions de valeurs globales associées aux fonctions de valeurs locales $v_{\ell+1}$ et $v'_{\ell-1}$; les fonctions de valeurs $v_{\ell-1:N}$ et $v'_{\ell-1:N}$ correspondent aux fonctions de valeurs v_τ pour tout $\tau = \ell - 1, \ell, \cdots, N$. La passage de l'équation 4.14 à l'équation 4.15, s'explique par le fait que les fonctions de valeurs v_τ pour tout $\tau = \ell, \ell + 1, \cdots, N$ sont identiques pour les deux fonctions de valeurs $v_{\ell-1:N}$ et $v'_{\ell-1:N}$.

On en déduit alors que l'opérateur $\mathbb{H}$ est un opérateur de contraction pour toute fonction de valeurs v_τ pour tout $\tau = 0, 1, \cdots, N$, et de la fonction de valeurs v. Il vient alors que l'application successive de l'opérateur $\mathbb{H}$ conduit à la fonction de valeurs ε-optimales v_ε^*. □

L'algorithme topologique exacte, dont l'opérateur de mises à jour est présenté en Figure 4.10, pour la résolution des POMDPs munis d'une structure topologique est bien plus efficace que tout autre approche exacte. Pour s'en convaincre, nous proposons une brève analyse de sa complexité. Cette dernière est intimement liée à la complexité de l'opérateur $\mathbb{H}$. Observons tout d'abord que l'étape (1) produit $|A||\Omega_\tau||\Lambda|$ fonctions de valeurs. La seconde étape procède à $|A||\Lambda|^{\Omega_\tau}$ sommes croisées. Ainsi dans le pire des cas, n mises à jour d'une couche B_τ produisent $\mathcal{O}(|A|^{\frac{|\Omega_\tau|^{n+1}-1}{|\Omega_\tau|-1}})$ fonctions de valeurs et cela en temps $\mathcal{O}(|S_{\tau:N}|^2|A||\Lambda|^{|\Omega_\tau|})$ – où $S_{\tau:N} = \cup_{\ell=\tau}^{N} S_\ell$ Le Tableau 4.3 donne un aperçu de la complexité de l'opérateur de mises à jour topologique en comparaison à l'opérateur classique. On constate que la complexité est réduite de façon exponentielle, et cela d'autant plus que le nombre de couches est grand.

	Types d'opérateurs $\mathbb{H}$	
Complexité	Topologique	Classique
Mémoire	$N\|A\|^{\frac{\|\Omega^*\|^{n+1}-1}{\|\Omega^*\|-1}}$	$\|A\|^{\frac{\|\Omega\|^{n+1}-1}{\|\Omega\|-1}}$
Temps	$N\|S_{\tau:N}\|^2\|A\|\|\Lambda\|^{\|\Omega^*\|}$	$\|S\|^2\|A\|\|\Lambda\|^{\|\Omega\|}$

TABLE 4.3 – Comparaisons des opérateurs, où $\Omega^* = \max_\tau \Omega_\tau$.

Comme nous l'avons souligné plus haut, résoudre un processus décisionnel de Markov partiellement observable de façon exacte est extrêmement difficile et cela quand bien même le problème serait de dimension insignifiante – par exemple $|S| = 2$. Pour cette raison l'ensemble des efforts de la communauté des chercheurs dans ce domaine est orienté vers la recherche d'approches approximatives capables de résoudre des problèmes de tailles plus réalistes. Compte tenu de la complexité que nous avons mise en évidence, une approche topologique et approximative est très souhaitable. À l'instar des autres approches approximatives, une approche topologique et approximative consiste en la construction de la fonction de valeurs v exclusivement sur des ensembles finis B_τ pour tout $\tau = 0, 1, \cdots, N$. Ces ensembles peuvent être générés par échantillonnage. Une telle approche vise non seulement à approximer un POMDP quelconque en un POMDP à structure topologique, mais aussi à approximer la fonction de valeurs exactes du POMDP à structure topologique.

	Types d'opérateurs $\bar{\mathbb{H}}$	
Complexité	Topologique	Classique
Mémoire	$N\lvert B_\tau\rvert\lvert A\rvert\lvert\Omega^*\rvert\lvert S_{\tau:N}\rvert$	$\lvert B\rvert\lvert A\rvert\lvert\Omega\rvert\lvert S\rvert$
Temps	$N\lvert B_\tau\rvert\lvert A\rvert\lvert\Omega^*\rvert\lvert S_{\tau:N}\rvert^2$	$\lvert B\rvert\lvert A\rvert\lvert\Omega\rvert\lvert S\rvert^2$

TABLE 4.4 – Comparaisons des opérateurs à base de croyances, où $\Omega^* = \max_\tau \Omega_\tau$.

Comme les ensembles B_τ pour tout $\tau = 0, 1, \cdots, N$ sont finis, il n'est pas nécessaire d'appliquer l'opérateur $\mathbb{H}$, il suffit au contraire d'appliquer l'opérateur de mises à jour à base de croyances $\bar{\mathbb{H}}$ défini comme suit : pour tout $s \in S_\tau$,

$$v^{a,\omega}(s) = \sum_{s \in S_{\tau:N}} O(\omega|a,s')T(s'|s,a)v(s'), \quad \text{pour tout } \omega \in \Omega_\tau \tag{4.19}$$

$$v^a(s) = R(a,s) + \lambda \sum_{\omega \in \Omega_\tau} \text{argmax}_{v^{a,\omega}:\, v \in \Lambda} \left(v^{a,\omega} \cdot b\right) \tag{4.20}$$

$$v^b(s) = R(a,s) + \lambda \sum_{\omega \in \Omega_\tau} \text{argmax}_{v^a:\, a \in A} \left(v^a \cdot b\right) \tag{4.21}$$

où Λ est un ensemble composite de fonctions de valeurs pour les différentes couches de croyances. Le Tableau 4.4 récapitule les complexités des opérateurs approximatifs qu'ils soient topologiques ou classiques. La différence essentielle se situe dans le temps consacré à la mises à jour d'états $s \in S$. Soit parce que cela est redondant soit parce que cela est prématuré. En organisant l'ordre des mises à jour, l'approche topologique permet de ne mettre à jour qu'un sous-ensemble S_τ d'états à la fois permettant ainsi non seulement un gain en temps de calcul de la fonction de valeurs v mais aussi un gain en mémoire car les fonctions de valeurs sont restreintes aux ensembles d'états S_τ, pour tout $\tau = 0, 1, \cdots, N$.

Malheureusement, dans la grande majorité des applications réelles, les processus de décisions de Markov partiellement observables ne sont pas muni d'une structure topologique. Il faut alors trouver un moyen de lui associer une structure topologique. Une solution consiste à exploiter la structure topologique (exacte ou non) de son MDP sous-jacent. Afin d'exploiter cette idée, nous avons recours à une approche approximative d'identification des couches de croyances du POMDP associé. Lorsqu'un POMDP est tel que le MDP sous-jacent est muni d'une structure topologique, la section suivant offre un moyen de déterminer approximativement les couches de croyances.

4.3.3 Planification approximative à base d'ordres topologiques

Etant donné des couches d'états (ou de croyances), un algorithme peut exploiter cette information afin de mettre à jour ces états (ou croyances) de façon intelligente. Les cycles dans l'espace des états forcent les algorithmes d'itération de valeurs ou de politiques à mettre à jour de façon répétitive la valeur du même état, jusqu'à convergence. Dans notre cas, les cycles n'existent qu'à l'intérieur d'une couche. En effet, le graphe des couches est par construction acyclique. Pour cette raison, il est possible de résoudre chaque couche une seule fois. Afin de définir l'ordre correcte suivant lequel les couches devrait être mises à jour, nous introduisons les notions de couches résolues et couches résolvables. Une couche S_ℓ est résolue si pour tout état $s \in S_\ell$, $v(s)$ est à ε de sa valeur optimale. Ainsi, la valeur de tout état dans une couche résolue ne peut être améliorée de façon significative. Une couche S_ℓ est dite résolvable si elle ne dispose d'aucune couche successeur, ou alors si toutes les couches successeurs de S_ℓ sont résolues. Un algorithme de mises à jour topologique réalise les mises à jour uniquement sur les états des couches résolvables, car les mises à jour sur les états des couches résolues n'améliorent pas la fonction de valeurs v. Lorsque l'environnement dispose d'états finaux ou terminaux c'est à dire des états absorbants, ces états forment chacun une couche. Ces dernières sont toujours résolvables.

Théorème 3. *Un algorithme de mises à jour topologique qui procède aux mises à jour uniquement sur les états des couches résolvables, converge vers une fonction de valeurs ε-optimales si :*

1. *Chaque couche devient résolvable au cours de l'exécution de l'algorithme.*
2. *Chaque couche résolvable est résolue éventuellement.*

Démonstration. La preuve découle directement de la preuve de convergence de l'algorithme topologique applicable au cadre des processus décisionnels de Markov complètement observables. En particulier, un processus décisionnels de Markov partiellement observable est équivalent à un processus de Markov complètement observable où l'ensemble des états n'est ni plus ni moins que l'ensemble des croyances. Ainsi si l'on parvient à grouper ces croyances en couches, le théorème est valide. □

4.3.4 Planification à base d'ordres topologiques pour les POMDPs

En POMDPs, parce que nous ne pouvons pas toujours déterminer avec faciliter la structure topologique, l'implémentation d'un algorithme topologique requiert quelques approximations. Un algorithme topologique pour POMDP requiert un certain nombre de composantes clés :

1. Identifier qu'une couche est résolue.
2. Identifier qu'une couche est résolvable.
3. Déterminer les croyances appartenant à une couche résolvable.

Ci-dessous, nous expliquons les difficultés et nous suggérons des solutions à toutes ces composantes. L'approche est motivée par la famille des algorithmes de résolution des POMDPs à base de trajectoires. Nous exécutons les trajectoires de croyances en direction des couches résolvables, et mettons à jour les croyances qui appartiennent aux couches résolvables uniquement.

Identifier les couches de croyances

Comme nous l'avons expliqué ci-dessus, il est difficile d'identifier les couches de croyances dans le cas général des POMDPs. Même si nous parvenions à le faire, le temps requis serait prohibitif et ainsi le gain escompté n'apparaîtrai pas dans le temps total de résolution du POMDP. Nous allons donc procéder à une association entre les couches d'états du MDP sous-jacent au POMDP à résoudre et les couches de croyances du dit POMDP. Pour y parvenir, nous exécutons des trajectoires à la fois dans l'espace des états et dans l'espace des croyances simultanément, comme dans le cadre de l'algorithme FSVI voir Figure 4.11. Nous générons ainsi des trajectoires de paires (s, b) état - croyance. On dit qu'une paire (s, b) appartient à une couche S_ℓ si $s \in S_\ell$. Cette approche n'est qu'une approximation des véritables couches de croyances. En effet, il se pourrait qu'une croyance puisse être incluse dans plusieurs couches d'états du MDP sous-jacent. De plus, cette méthode associe différentes couches de croyances du POMDP à la même couche d'états du MDP sous-jacent. Néanmoins, comme nous le verrons plus tard, cette approximation est très utile en pratique, et elle nous permet d'estimer si une croyance appartient ou non à une couche résolvable.

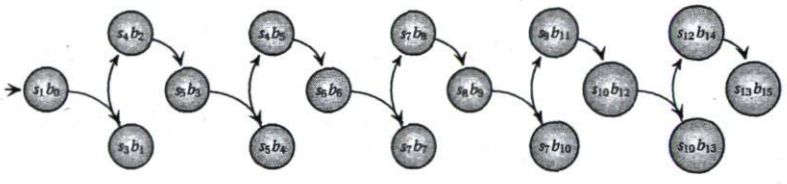


FIGURE 4.11 – Simulation d'une trajectoire de paires état-croyance.

Plus formellement, une couche S_ℓ est résolue si pour tout état $s \in S_\ell$, et toute croyance b tel que $b(s) > 0$, la valeur issue de s et influençant la valeur de b, ne peut plus être améliorée. En d'autres termes, si v est la fonction de valeurs courante, et v^* la fonction de valeurs optimales, soit $v(b) = v^*(b)$, alors $v(s) = v^*(s)$. Bien que la propriété précédente soit valide, il est difficile de vérifier l'ensemble des croyances b dont la probabilité en s

est non nulle car le nombre des croyances possédant cette propriété est probablement infini. Nous surmontons cela en maintenant un estimé sur le fait que les croyances associées à un état peuvent ou non voir leurs valeurs être améliorées. Pour chaque état s, nous maintenons une valeur $\hat{v}(s)$ qui est un estimé du potentiel d'amélioration des croyances associées à l'état s. Nous initialisons $\hat{v}(s) = \max_a R(s,a)$. À chaque fois qu'une croyance associée à s est mise à jour, nous posons $\hat{v}(s) = 0$. Lorsque la mise à jour d'une croyance b associé à s' produit une nouvelle fonction de valeurs v_b améliorant la valeur de l'état s' de δ, nous parcourons les prédécesseurs de s'. Pour chaque prédécesseurs s de s', nous posons $\hat{v}(s) = \max\{\delta \cdot T(s'|s,\cdot), \hat{v}(s)\}$. Ainsi, un état et toutes ses croyances associées sont résolus lorsque tous les états successeurs et leurs croyances sont résolus.

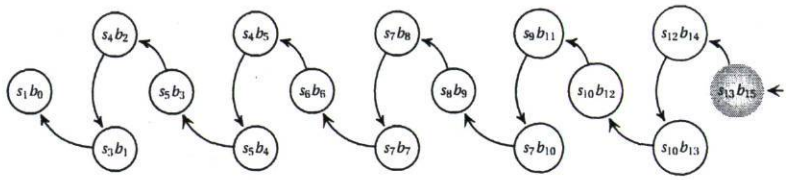


FIGURE 4.12 – Phase de mises à jour : $(s_{13}b_{15})$ appartient à une couche résolvable.

Une fois la trajectoire d'états et de croyances générée, se pose alors la question de la mise à jour des croyances. Fidèle au paradigme des algorithmes de mises à jour à base de trajectoires, TOP procède au parcours de la trajectoire générée par chaînage arrière, comme illustré à la Figure 4.12. Pour chaque paire d'état et de croyance rencontrée, TOP vérifie le statut de la couche à laquelle cette paire appartient. En particulier, la paire $(s_{13}b_{15})$ appartient à la couche d'états incluant l'état s_{13}. C'est à dire une couche résolvable. TOP peut donc mettre à jour la valeur de la croyance b_{15} et de même que la fonction $\hat{v}$ pour les états s_{13} et ses prédécesseurs. Comme $\hat{v}(s_{13}) = 0$, on en déduit que la couche $\{s_{13}\}$ est résolue et par conséquent la couche $\{s_{12}\}$ est résolvable.

Traversée des couches

TOP est un algorithme à base de trajectoires c'est à dire qu'il exécute des trajectoires dans l'espace des croyances, puis procède aux mises à jour des croyances qui ont été produites dans l'ordre inverse de la génération des trajectoires. Comme nous ne nous intéressons qu'aux couches résolvables, nous avons besoin de construire des trajectoires au sein de ces couches. Pour y parvenir, nous parcourons à la fois les états du MDP et les croyances du POMDP, comme suggéré par l'algorithme FSVI. Avant la génération d'une trajectoire, nous commençons par la sélection aléatoire d'un état initial s_0 et d'un état but s_g accessible de s_0. Lorsqu'il n'y a pas de couche résolue, l'état but s_g peut être ou un état terminal de l'environnement si un tel état existe, ou tout état dans une couche résolvable. Lorsque les couches ont été résolues, nous choisissons un état but. Ce dernier est l'un des états

des couches dernièrement résolues et accessible de l'état initial. Les trajectoires partent de l'état initial vers l'état but en suivant le chemin le plus probable dans l'espace des états. Nous utilisons l'algorithme de Floyd-Warshall, afin de calculer ces chemins. Comme nous ne sélectionnons pas arbitrairement nos actions, toutes les trajectoires générées sont assurées d'aboutir à l'état but. Lors de la construction d'une trajectoire, nous maintenons les paires d'états et de croyances. Une fois la construction de la trajectoire terminée, nous exécutons les mises à jour dans l'ordre inverse. Cependant, nous limitons les mises à jour de la fonction de valeurs aux paires d'états et de croyances appartenant à une couche résolvable. Après la mises à jour, nous vérifions si les couches résolvables qui été mises à jour sont désormais résolues. Si tous les états dans une couche ont un potentiel d'amélioration $\hat{v}(s) \leqslant \varepsilon$, nous concluons que la couche est résolue. Autrement, la couche reste résolvable. L'algorithme termine lorsque toutes les couches sont résolues.

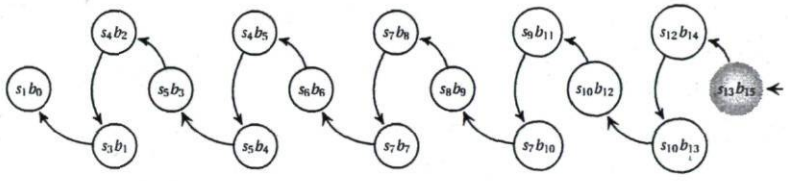


FIGURE 4.13 – Phase de mises à jour : b_{15} peut être mis à jour.

Comme nous l'avons observé précédemment la couche $\{s_{12}\}$ devient résolvable à l'issue de la mise à jour de b_{15}. Lorsque TOP traite la paire $(s_{12}b_{14})$ voir Figure 4.14, cette dernière appartient à une couche résolvable donc TOP peut mettre à jour b_{14}. Une fois de plus les valeurs $\hat{v}(s_{12})$, $\hat{v}(s_{10})$, et $\hat{v}(s_9)$ sont mises à jour. On en déduit que la couche $\{s_{12}\}$ est résolue et la couche $\{s_{10}, s_9\}$ devient résolvable.

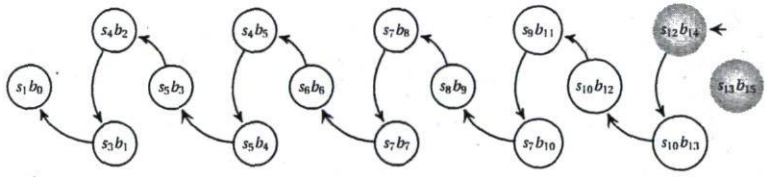


FIGURE 4.14 – Phase de mises à jour : b_{14} peut être mis à jour.

Comme $\{s_{10}, s_9\}$ est désormais résolvable, lorsque TOP traite les paires $(s_{10}b_{12})$ et $(s_{10}b_{13})$, il peut mettre à jour les croyances b_{12} et b_{13}. Seulement, la mise à jour de $\hat{v}(s_{10})$ et $\hat{v}(s_9)$ $neper met pas$ est résolue. Il faut alors générer une nouvelle trajectoire d'états et de croyances mais cette fois-ci elles doivent s'arrêter à l'état s_{12}.

Un fois toutes les couches résolues, l'algorithme est terminé et retourne la fonction de valeurs ainsi calculée.

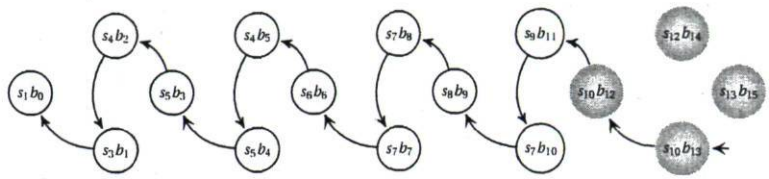


FIGURE 4.15 – Phase de mises à jour : $\{b_{12}, b_{13}\}$ peuvent-être mis à jour.

FIGURE 4.16 – Phase de mises à jour : terminer lorsque toutes les couches sont résolues.

Algorithme TOP

1. Choisir l'état initial s_0.
2. Choisir l'état terminal s_g.
3. Exécuter TOPTRIAL(s_0, s_g, b_0).
4. Si v n'à pas convergé retourner à l'étape 1.
5. Sinon retourner v.

♣

FIGURE 4.17 – Méthode TOP(b_0).

Sous-routine TOPTRIAL

– Si l'état s n'est pas résolu, faire :
  1. $s' \leftarrow$ PICKNEXTSTATE(s, s_g).
  2. $a \leftarrow$ PICKACTION(s, s').
  3. $o \leftarrow$ PICKNEXTOBSERVATION(s', a).
  4. Exécuter TOPTRIAL$(s', s_g, \tau(b, a, o))$.
– Si la paire (s, b) appartient à une couche résolvable exécuter UPDATE(b, v).

♣

FIGURE 4.18 – Méthode TOPTRIAL(s, s_g, b).

4.4 Évaluation expérimentales

Nous évaluons les performances de l'algorithme TOP en comparaison à celles des algorithmes approximatifs les plus compétitifs tels que HSVI [Smith and Simmons, 2005], FSVI, PVI un algorithme prioritisé pour les POMDPs [Shani et al., 2006], PBVI [Pineau et al., 2003b], et SCVI [Virin et al., 2007]. Les performances ont été collectées sur une même machine muni d'un processeur Intel Core Duo 1.83GHz CPU avec 1Gb de mémoire principale.

4.4.1 Performances

Nous commençons par souligner l'avantage de TOP au fur et à mesure que le nombre de couches croît. Afin d'y parvenir, nous modifions un problème de navigation classique dans la littérature des POMDPs – Hallway et Hallway2. Nous y introduisons des entrées à sens uniques comme illustrer Figure 4.7. Ces entrées sont dirigés dans le sens des états terminaux. Comme nous pouvons le constater dans la Table **??**, tandis que les performances de HSVI et FSVI restent sensiblement identiques bien que le nombre de couches croît, TOP au contraire ne cesse d'améliorer ces performances.

Comme nous l'avons expliquer plus haut, TOP réduit le nombre de mises à jour redondantes. De telles mises à jour sont de diverses sources – certaines proviennent du fait que des croyances ont déjà atteint leur valeur maximale ; d'autres proviennent du fait que les croyances mises à jour possèdent des croyances successeurs qui n'ont pas encore été mis à jour ; ou encore des mises à jour qui améliorent la fonction de valeurs mais sont effectuées bien trop tôt. Dans le dernier cas, des mises à jour additifs seront nécessaires sur la même croyance. Afin d'estimer les deux premières sources de mises à jour redondantes, nous comptons le nombre de mises à jour qui n'améliore pas la fonction de valeurs v, on les appelle les mises à jour inutiles. Afin d'estimer le dernier type de mises à jour redondantes, nous pouvons comparer le nombre de mises à jour utiles, en soustrayant le nombre de mises à jour inutiles du total des mises à jour.

Nous continuons la comparaison des performances de TOP aux autres algorithmes sur des problèmes bien connus dans la littérature des POMDPs. Sur des problèmes sans structures topologiques, il n'y a aucune raison que TOP surpasse FSVI, car les deux algorithmes utilisent le même principe de sélection des trajectoires. Nous modifions alors des problèmes sans structures topologiques en y introduisant des issues à sens unique orienté vers les états buts. Le problème d'échantillonnage de roches (selon le terme anglo-saxon

Méthode	ADR	$V(b_0)$	$\|V\|$	Temps (sec)	#Mises à jour	#Mises à jour inutiles	Croissance
Hallway-layered $\#layers$: 10							
HSVI	0.62	1.46	143	27	199	35	×6.75
FSVI	0.62	2.08	337	57	612	143	×14.2
PBVI	0.60	2.18	249	155	3659	1289	×38.7
SCVI	0.62	1.88	693	58	1420	104	×14.5
PVI	0.50	0.36	64	288	120	12	×72
TOP	**0.62**	**1.43**	**25**	**4**	**127**	**0**	**×1**
Hallway-layered $\#layers$: 15							
HSVI	0.62	1.54	57	11	123	10	×11
FSVI	0.62	2.21	266	49	612	139	×49
PBVI	0.62	2.24	127	87	2581	1045	×87
SCVI	0.62	2.0	609	56	1430	109	×56
PVI	0.50	0.36	64	288	120	11	×288
TOP	**0.62**	**0.47**	**12**	**1**	**34**	**0**	**×1**
Hallway2-layered $\#layers$: 10							
HSVI	0.48	0.60	357	200	473	64	×10.5
FSVI	0.48	0.60	450	127	408	293	×6.68
PBVI	0.47	0.64	32	94	795	23	×4.94
SCVI	0.48	0.65	1448	435	2230	148	×22.9
TOP	**0.48**	**0.34**	**63**	**19**	**135**	**6**	**×1**
Hallway2-layered $\#layers$: 15							
HSVI	0.57	2.42	368	244	1114	258	×10.2
FSVI	0.57	1.51	290	71	357	60	×2.95
SCVI	0.57	1.61	482	120	1280	127	×5
TOP	**0.57**	**0.95**	**48**	**24**	**164**	**0**	**×1**
Hallway2-layered $\#layers$: 20							
HSVI	0.57	2.73	296	201	1102	133	×14.4
FSVI	0.57	1.55	218	65	357	38	×4.64
SCVI	0.57	1.58	388	48	1240	109	×3.43
TOP	**0.57**	**0.77**	**38**	**14**	**104**	**8**	**×1**

TABLE 4.5 – Mesures des performances des problèmes de navigation Hallway à structures topologiques.

« RockSample ») Figure 4.5 est le seul qui dispose déjà d'une structure topologique, et dans les différentes instances de ce domaine TOP surpasse tous les autres algorithmes. Nous pouvons constater que cette amélioration est principalement due au fait que TOP exécute très peu de mises à jour en comparaison aux autres méthodes. En outre, il ne procède presque pas à l'exécution de mises à jour redondantes ou prématurées et cela sur

Méthode	ADR	$V(b_0)$	$\|V\|$	Temps (sec)	#Mises à jour	#Mises à jour inutiles	Croissance
cit-layered $\#layers$: 12							
HSVI	0.83	0.66	492	2170	873	370	×50
FSVI	0.83	0.68	403	1278	1122	718	×29.7
SCVI	0.83	0.72	535	224	550	5	× 5
TOP	**0.83**	**0.16**	**36**	**43**	**134**	**0**	×**1**
Mit-layered $\#layers$: 11							
HSVI	0.90	0.85	731	1515	1076	214	×39
FSVI	0.90	0.80	164	494	1275	248	×13
SCVI	0.90	0.63	496	101	520	4	×2.65
TOP	**0.90**	**0.53**	**47**	**38**	**196**	**0**	×**1**
Rock Sample (4,4) $\#layers$: 17							
HSVI	18.04	17.92	173	11	239	33	×5
FSVI	18.04	13.63	79	7	98	12	×3
SCVI	16.80	15.93	405	520	16350	797	×260
TOP	**18.04**	**13.60**	**37**	**2**	**63**	**0**	×**1**
Rock Sample (5,7) $\#layers$: 129							
HSVI	24.2	22.96	208	2754	461	124	×5
FSVI	22.2	14.97	353	4057	384	12	×8
SCVI	21.3	20.22	432	535	800	147	×1.09
TOP	**24.3**	**14.51**	**68**	**488**	**95**	**0**	×**1**
Rock Sample (7,8) $\#layers$: 257							
HSVI	20.1	19.69	178	7641	226	31	×6.87
FSVI	19.7	18.43	105	4503	185	16	×4.04
TOP	**20.1**	**17.68**	**73**	**1112**	**72**	**0**	×**1**

TABLE 4.6 – Mesure de performances dans les domaines à strucutres topologiques.

l'ensemble des problèmes

Discussion

D'autres chercheurs ont suggéré d'exploiter différents types de structures afin d'accélérer le taux de convergence vers une politique approximative. Par exemple, [Shani et al., 2008] explique comment résoudre des POMDPs factorisés en utilisant des diagrammes de décisions algébriques et [Pineau et al., 2003a] propose de créer une hiérarchies de POMDPs afin de résoudre des problèmes de plus grandes tailles. Exploitation des structures topologiques est en ce sens une approche orthogonale à ces approches. Il est néan-

moins possible qu'un POMDP factorisé possède également une structure topologique significative, de telle sorte que notre approche puisse être à juste titre être perçue comme une approche complémentaire à la majorité des autres approches exploitant diverses types de structures différentes. De plus notre approche peut assez facilement s'intégrer dans ces autres techniques structurées. Cependant, comme l'algorithme TOP se focalise essentiellement sur les structures topologique sur les états, il serait nécessaire d'apporter quelques modifications afin de l'intégrer dans d'autres techniques. Nous travaillerons à l'étude des conditions propices à de telles intégrations.

L'idée d'utiliser la structure topologique des états a été étudiée à plusieurs reprise dans le contexte des MDPs [Bonet and Geffner, 2003a, Dai and Goldsmith, 2007]. Dans les MDPs, une fois les couches (composantes fortement connexes) ont été identifiées, nous pouvons exécuter l'itération de valeurs uniquement sur les états dans une seule couche résolvable, jusqu'à ce que cette dernière soit résolue.

Comme nous l'avons signalé plus haut, une application directe de cette idée au cadre des POMDPs est difficile. Tout d'abord, le MDP où les états correspondent aux croyances et équivalent à un POMDP dispose d'un espace d'états infini – ceci rend l'identification des composantes fortement connexes difficile.

Notre algorithme possède également des similarités avec les algorithmes prioritisés. [Wingate and Seppi, 2005] fait un état de l'art des approches d'ordonnancement des mises à jour dans le cadre des MDPs, en utilisant l'erreur de Bellman. Dans ce contexte, la priorité d'un état est le potentiel d'amélioration de sa valeur dans la fonction de valeurs courante s'il est mis à jour. [Shani et al., 2006] a étendu ces idées au cadre des POMDPs. Ces derniers utilisent également l'erreur de Bellman, mais appliqué désormais sur les croyances. Signalons, cependant, qu'une erreur de Bellman élevée n'indique pas qu'une croyance appartient à une couche résolvable. Une croyance b, tel que $b(s) > 0$ et $R(s,a) > 0$ peut initialement avoir une erreur de Bellman élevée, même si s appartient à une couche éloignée des couches résolvables. Mais néanmoins, TOP mettra à jour b tardivement dans le processus de résolution du problème.

Une autre technique d'ordonnancement des mises à jour a été suggérée par [Virin et al., 2007]. L'algorithme SCVI groupe les états en clusters d'états selon leurs valeurs extraites de la fonction de valeurs du MDP sous-jacent. Les auteurs collectent un ensemble fini de croyances et assignent des poids de clusters des croyances basés sur les probabilités des états dans ces clusters. Puis, ils itèrent sur les clusters en réduisant leurs valeurs et en mettant à jour les croyances qui y sont associées. Il est possible d'interpréter SCVI comme s'il induisait des couches dans des domaines apriori sans structures topologiques. SCVI n'est pas un algorithme à base de trajectoires, mais il se pourrait que ses idées puissent

être importées dans l'algorithme TOP afin d'introduire des couches artificielles dans une POMDP quelconque.

En ce qui concerne les algorithmes à base de trajectoires, il est essentielle de trouver un équilibre entre l'identification de bonnes trajectoires, et le temps requis pour leur génération. HSVI utilise une borne supérieure et une borne inférieure sur la fonction de valeurs recherchée afin d'identifier les croyances pour lesquelles la différence entre ces deux bornes peut être réduite. Les trajectoires ne sont alors pas nécessairement directement dirigées vers les régions où la borne inférieure, qui contrôle la politique, peut être améliorée. De plus, la maintenance de la borne supérieure est extrêmement coûteuse, ainsi HSVI passe énormément de temps à calculer ses trajectoires.

FSVI, d'autre part, est très rapide dans la génération de ses trajectoires. Comme il utilise essentiellement une fonction de valeurs statique du MDP sous-jacent, calculer l'état suivant est très facile. Cependant, FSVI ne modifie pas sa stratégie de génération des trajectoires au fur et à mesure que la fonction de valeurs est mises à jour. Ainsi, FSVI pourrait visiter encore et encore des régions de l'espace des croyances qui ont déjà été résolues.

Tous les deux algorithmes FSVI et HSVI exécutent de nombreuses mises à jour redondantes de la fonction de valeurs, comme ils ne disposent d'aucun mécanisme de détection de bénéfice d'une mise à jour, ou encore s'il faudra des mises à jour supplémentaires plus tard. Comme nous l'avons démontré plus haut, TOP réduit le nombre de mises à jour redondants car il ne considère que les couches résolvables.

Conclusion intermédiaire

Nous avons introduit une nouvelle méthode de résolution des POMDPs – nommée TOP (de l'acronyme anglais « topological order-based planning »), qui utilise les propriétés structurelles des POMDPs afin de réduire le nombre extrêmement coûteux de mises à jour de la fonction de valeurs. Notre méthode utilise les couches des états du MDP sous-jacent afin d'identifier les couches des croyances. TOP met à jour exclusivement les croyances appartenant aux couches résolvables et par conséquent réduit considérablement le nombre de fois qu'une croyance est mis à jour. Nous avons démontré comment TOP surpasse les autres algorithmes de résolution de POMDPs dans des domaines contenant une structure topologique significative.

Nous avons dans ce chapitre identifié les conditions nécessaires pour l'application de l'algorithme TOP, et nous avons suggérer une implémentation possible pour les POMDPs. Dans le future, nous nous intéresserons à l'intégration des autres méthodes telles que

les méthodes d'ordonnancement basées sur l'erreur de Bellman ou encore les méthodes de regroupement basée sur la fonction de valeurs du MDP sous-jacent. Nous nous intéressons également à l'application de TOP dans des domaines factorisés, en particulier étendre TOP de sorte à formaliser les couches d'états sous forme de couches de variables d'états.

4.5 Exemple applicatif : le projet « NEREUS »

Dans cette section, nous nous intéressons à un exemple applicatif réel extrait d'un problème de défense anti attaques d'une frégate de l'armée canadienne [Dibangoye et al., 2007]. Nous formulons cette application comme un problème de recherche d'une stratégie d'allocation périodique de ressources limitées à plusieurs cibles en vue de définir une stratégie de défense la plus efficace possible. À cette fin, nous proposons un algorithme de programmation dynamique nommé RTDA*, exploitant deux propriétés du problème afin d'éviter l'explosion combinatoire due aux multiples combinaisons d'allocations de ressources à considérer. D'une part, l'allocation des ressources à période de décisions suit l'ordre décroissant des priorités sur les cibles. Nous exploitons l'indivisibilité des ressources, toute ressource assignée à une cible ne peut être simultanément assignée à une autre cible. D'autre part, les protocoles d'engagement des différentes ressources sont structurées, de sorte que certaines ressources ne peuvent être utilisées qu'à la suite de l'usage de certaines autres ressources. En somme, il existe une relation de dépendance causale entre les différentes ressources. Ces contraintes inscrivent naturellement ce problème comme un problème topologique. Cela explique l'usage d'une architecture de représentation des dépendances causale entre les ressources et de structuration de la stratégie efficace de calcul d'une solution en temps réel. Cette architecture nommée « cyclic progressive reasoning unit » (CPRU) étend l'architecture acyclique proposée par [Mouaddib and Zilberstein, 1995], afin de gérer des structures plus complexes, en particulier les structures topologiques. Grâce à l'identification de cette structure, RTDA* peut, dans une phase hors ligne, déterminer une politique pour chaque cible apparaissant indépendamment les unes des autres ; dans une seconde phase en ligne, RTDA* réussi à combiner les stratégies individuelles en une stratégie globale et presqu'optimale.

Le problème de défense anti attaques d'une frégate que nous étudions ici est une version simpliste d'une application réelle de l'armée canadienne nommée NEREUS (Naval Environment for Resource Engagement in Unpredictable Situations). NEREUS est une application de contrôle de plates-formes militaires munies d'armes plongées dans de vastes environnements maritimes. NEREUS a pour objectif de rechercher des stratégies de défense de ces plates-formes afin d'éviter qu'elles soient détruites par d'éventuels missiles

ennemis tout en contrôlant l'usage des ressources. Afin d'y parvenir, NEREUS doit parvenir à détruire les missiles ennemis ou parvenir à les détourner des plates-formes.

Compte tenu des contraintes liées aux mécanismes de commandes et contrôles automatiques des armes, les ressources à disposition sur les plates-formes doivent être utilisées dans un certain ordre afin de garantir le bon déroulement de l'engagement d'une arme par exemple. Ces contraintes définissent la structure d'un engagement, définie au travers d'une famille de sous tâches organisées dans un certain ordre suivant lequel ils doivent être déclenchés. Comme illustrer à la Figure 4.19, la tâche d'engagement d'un missile ennemi est définie comme suit :

1. La première sous tâche consiste à l'illumination de la cible faisant appel aux radars (STIR et CIWS).
2. Une fois l'illumination de la cible réussie, la seconde sous tâche est déclenchée. Elle consiste à l'interception de la cible en usant d'armes (SAM, GUN, et CIWS). Notons que s'il est possible d'utiliser les armes SAM et GUN après une illumination via le radar STIR, l'arme CIWS ne peut être utiliser qu'après avoir illuminé la cible avec son propre radar.
3. Finalement, le système de commande et contrôle peut effectuer l'étape de vérification de la destruction de la cible.

Les cibles peuvent également être détournées ou encore leurs radars brouillés par deux armes : CHAFF et JAMMER. Ces dernières peuvent être utilisées à tout moment sans conditions préliminaires aucunes, et même simultanément.

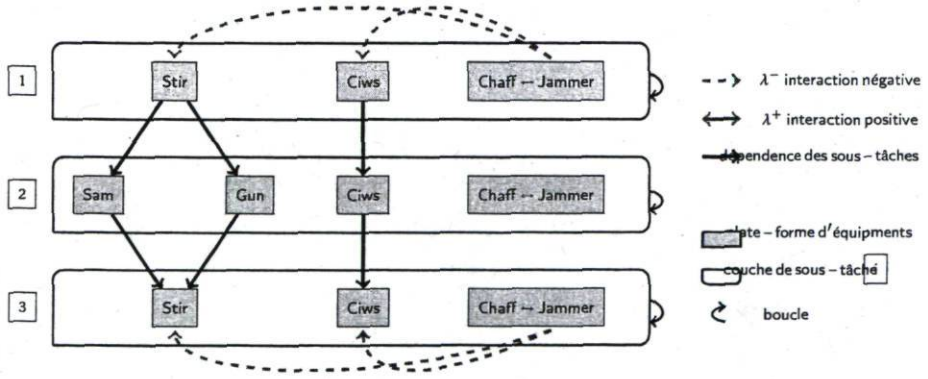


FIGURE 4.19 – Tâche d'engagement d'une cible.

L'application que nous venons de décrire implique de nombreuses variables aléatoires influençant l'état du système. Ces variables aléatoires capturent l'incertitude de diverses sources telles que les cibles apparaissant dans le temps aléatoirement, les échecs des engagements effectués, les échecs des interactions entres les différentes ressources dans le sys-

tème de commande et contrôle, etc. Par exemple, les interactions entre les différentes ressources peuvent aboutir à des effets négatifs. Comme illustrer à la Figure 4.19, si le nuage produit par l'arme CHAFF, afin de détourner un missile ennemi, est libéré, il pourrait affaiblir la qualité des radars d'illumination. D'un autre côté, des effets positifs peuvent résulter de l'usage simultané du CHAFF et JAMMER, améliorant ainsi l'effet brouillard du CHAFF.

Ce problème de gestion de ressources peut être vu comme un problème d'allocation stochastique de ressources aux différentes cibles apparaissant au cours du temps. Ces cibles sont alors considérées comme des agents auxquels il faut assigner des ressources. Ces agents sont représentées par l'architecture CPRU comme décrit à la Figure 4.19. Les actions des agents (cibles) correspondent à l'assignation de ressources suivant l'ordre de la CPRU, en vu de leur destruction ou leur détournement. L'architecture CPRU est une simple tâche structurée, à la différence de l'architecture PRU « progressive reasoning unit » [Mouaddib and Zilberstein, 1995], où les échecs d'exécution mènent l'architecture à boucler au même niveau, à la même sous tâche. Malheureusement, l'architecture PRU ne permet pas de prendre en compte ce type de domaine. Expliquant ainsi l'introduction de l'architecture appelé « cyclic progressive reasoning unit » (CPRU) permettant de capturer les caractéristiques énoncés précédemment. Cette dernière correspond à un MDP cyclique, mais topologique. Ce constat est facile à prouver en observant les différentes couches de l'architecture où les transitions cycliques n'apparaissent qu'au sein d'une même couches. Afin de résoudre ce problème, nous avons eu recours à un algorithme topologique exécuté en temps réel. Ces exécutions son entrelacées par des exécutions des politiques partielles construites durant la phase d'exécution de l'algorithme.

4.6 Conclusion partielle

Ce chapitre a décrit une sous-classe de problèmes décisionnels de Markov (complètement ou partiellement observables) ainsi que les algorithmes de résolution associés à ces mêmes problèmes. Les principales contributions concernant ce travail sont résumées dans cette section.

Exploitation de structures topologiques. Les algorithmes topologique *i*TVI et TOP supposent l'existence d'une structure topologique sous-jacent au MDP ou POMDP. Après extraction de la structure topologique du problème, ils remplacent les opérateurs de mises à jour classiques par d'avantageux opérateurs topologiques. Ces derniers sont capables d'organiser les mises à jour par couche d'états ou de croyances, de sorte à éviter ou les mises à jour redondantes ou les mises à jour prématurées.

Heuristique d'exploration. La structure topologique exploitée dans ce chapitre peut éga-

lement être perçue comme une heuristique informelle. En effet, elle permet de guider la sélection des croyances à mettre à jour et surtout elle permet d'ordonner les mises à jour de ces croyances de sorte que le moins souvent une croyance soit mise à jour avant l'heure. Lorsque le problème est complètement observable et acyclique, cette structure permet d'effectuer une seule mise à jour par état, réduisant ainsi la complexité de la résolution d'un MDP de polynomiale à linéaire.

Amélioration des performances empiriques. Que ce soit *i*TVI ou TOP, tous deux ont démontré des performances plus que satisfaisantes. Sur la totalité des problèmes testé dans le cadre des MDPs, *i*TVI surpasse l'ensemble des approches de l'état de l'art. TOP a démontré sa capacité à réduire le temps de planification sur un large nombre de problèmes de la littérature des POMDPs, à savoir RockSample, les versions topologiques des domaines Hallway, CIT et MIT. En ne mettant à jour les croyances que lorsqu'elles sont estimées solvables, TOP parvient à réduire considérablement le nombre de mises à jour superflus. En orientant, les trajectoires vers les régions de croyances les plus nécessiteuses, il parvient à construire une fonction de valeurs uniforme (empiriquement) sur l'espace des croyances accessibles.

Chapitre 5

DEC-POMDPs à Horizon Fini – « *la malédiction de la distributivité* »

« Le désert est la seule chose qui ne puisse être détruite que par la construction. »

Boris Vian

Une partie de ce chapitre est apparu précédemment dans les papiers : « Incremental Policy Generation for Finite-Horizon DEC-POMDPs » [Amato et al., 2009] co-rédigé avec Christopher Amato et Shlomo Zilberstein ; et « Point-based Incremental Pruning Heuristic for Solving Finite-Horizon DEC-POMDPs » [Dibangoye et al., 2008d,e, 2009b] avec Abdel-Illah Mouaddib et Brahim Chaib-draa.

Les modèles généraux de contrôle distribué des problèmes de prise de décisions séquentielles ont été proposés il y a bientôt dix ans. L'étude de ces modèles, d'une très grande expressivité, fait état de problèmes de complexité soit NEXP-difficiles soit indécidables. La principale stratégie de résolution de ces problèmes distribués est leur conversion en plusieurs sous problèmes centralisés, un par agent. Il s'agit alors d'une approche essentiellement distribuée. Ce chapitre introduit une théorie de la planification centrale pour le contrôle distribué des problèmes de prise de décisions séquentielles. Cette théorie offre la première analyse formelle de la résolution centrale des problèmes de contrôle distribué, en s'inspirant du formalisme introduit par Putterman [1994].

Dans ce chapitre, nous exposons nos contributions aux problèmes du contrôle distribué des processus décisionnels de Markov à temps discret et horizon fini. Nous offrons ainsi la première analyse formelle [1] de la planification centrale pour la résolution des modèles de contrôle distribué des problèmes de prise de décisions séquentielles.

Dans ce cadre, nous introduisons et démontrons tous les concepts de base, y compris le critère d'optimalité en Section 5.2, les équations d'optimalité, le principe d'optimalité en Section 5.3, et propriétés des politiques en Section 5.4. Ces résultats sont soit des extensions des résultats existants dans le cadre des MDPs et POMDPs soit des résultats spécifiques au contrôle distribué, c'est à dire DEC-MDPs, ND-POMDPs et DEC-POMDPs. La Section 5.5 décrit un nouvel algorithme non trivial pour le déterminisme d'une politique optimale ou ε-optimale et des illustrations de son usage à travers des exemples simples. La Section 5.6 propose plusieurs variantes de cet algorithme, y compris des algorithmes approximatifs dont l'erreur est bornée. On y trouve également l'évaluation de tous ces algorithmes sur des exemples issus de la littérature en Section 5.7.

Hypothèses 3. *Nous supposons tout au long de ce chapitre que :*

1. *S, A, et Ω sont discrets,*
2. *l'horizon de planification est fini $N < \infty$,*
3. *les récompenses sont bornées, $|r_t(s,a)| \leqslant M < \infty$, pour tout $(s,a) \in S \times A$.*

1. *« Nous exhortons très fortement le lecteur à revoir les Chapitres 2 et 3 pour une révision des définitions de base et des modèles de contrôle des processus décisionnels de Markov, avant de plonger dans ce chapitre. Votre effort sera largement récompensé car vous pourrez envoyer tout l'état de l'art aux oubliettes. »*

5.1 Motivation

Considérons un système multi-agents et coopératif dont l'objectif est de contrôler de façon distribuée et conjointe un processus décisionnels de Markov. Un tel problème peut se formaliser comme un DEC-POMDP.

À l'issue du choix puis de l'exécution d'une politique, un agent ne peut percevoir les récompenses r_τ à chaque horizon $\tau = 1,2,\cdots,N-1$ que conjointement avec le reste de ses coéquipiers. Comme ces récompenses ne peuvent être perçues que par l'ensemble des agents, un décideur central peut être requis afin de collecter les récompenses. Bien qu'elles ne soient observables que lors du contrôle, le décideur peut simuler le contrôle distribué afin de percevoir les récompenses r_τ à chaque horizon $\tau = 1,2,\cdots,N-1$ d'une simulation. Par simulation du contrôle distribué, nous entendons l'exécution d'une politique distributive $\pi \in \bar{\Pi}^{\mathbf{HA}} \subset \Pi^{\mathbf{HA}}$. Nous appelons politique distributive toute politique correspondant à un ensemble de politiques, une politique pour chaque agent. Ce concept est plus amplement abordé en Section 5.4.2. Tous les résultats que nous énoncerons ci-dessous, dans un cadre plus général, sont applicables au cadre du contrôle distribué en restreignant l'espace des politiques à l'ensemble des politiques distributives non Markoviennes et aléatoires $\bar{\Pi}^{\mathbf{HA}}$.

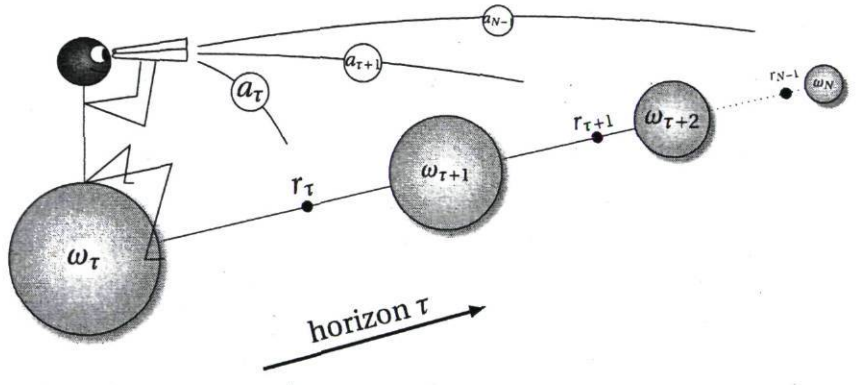


FIGURE 5.1 – Simulation d'un contrôle distribué des processus décisionnels de Markov.

La Figure 6.1 illustre l'exécution par un décideur d'une politique pour la simulation du contrôle distribué d'un processus de Markov partiellement observable. Comme ces récompenses ne peuvent être perçues avant le choix puis la simulation d'une politique, le décideur doit voir la séquence des récompenses comme aléatoire. Dans ce contexte, l'objectif du décideur est de choisir une politique telle que la séquence aléatoire des récompenses associée soit aussi grande que possible. Il est nécessaire alors de se prémunir de méthodes permettant de choisir la politique optimale selon les séquences aléatoires de récompenses observables lors de l'exécution de celle-ci. Nous rappelons en Section 5.2.2 que l'espérance des récompenses cumulées est le critère permettant de distinguer les po-

litiques puis d'en choisir une parmi toutes les politiques possibles.

De façon similaire aux récompenses, le décideur ne perçoit les observations ω_τ sur l'état du système, qu'après avoir choisi puis exécuté une politique. De ce fait, il doit également voir une séquence d'observations comme aléatoire. Il se doit dès lors de maintenir une information complète sur ses croyances quant à l'état courant du système afin d'estimer les récompenses. Bien qu'il ne puisse déterminer avec certitude l'état courant du système, le décideur peut extraire, des séquences $(s_0, \omega_1, \cdots, \omega_\tau)$ aléatoires d'observations, l'information sur l'état s_τ du système, pour tout horizon $\tau = 0, 1, \cdots, N$. Nous démontrons, Section 5.2.2, qu'à toute séquence aléatoire d'observations $(s_0, \omega_1, \cdots, \omega_\tau)$, correspond une distribution sur les historiques de l'ensemble H_τ, c'est à dire l'ensemble des historiques h_τ collectés à chaque horizon $\tau = 0, 1, \cdots, N$.

Les résultats établis dans les sections 5.2 et 5.3 peuvent être vus comme des généralisations des résultats similaires existants dans le cadre du contrôle centralisé des processus décisionnels de Markov complètement Putterman [1994] ou partiellement observables, à temps discrets et horizon fini. De tels résultats n'étaient pas établis pour le contrôle distribué des processus de Markov, et en encore moins dans le cadre de la planification centrale. Nous nous restreindrons par la suite, sections 5.4.2, 5.5 et 5.6, au cas des propriétés spécifiques du contrôle distribué des processus de Markov partiellement observables, à temps discrets et horizon fini.

5.2 Critères d'optimalité

Cette section introduit les outils de base pour l'analyse des problèmes de contrôle distribué dans le cadre de la planification centralisée. Après un court rappel des notations utilisées, nous spécifions les critères de différenciations des traces de simulation de politiques en Section 5.2.2. Puis, nous décrivons les critères de différenciation des politiques en Section 5.2.3. Et enfin, nous rappelons l'algorithme d'évaluation d'une politique en Section 5.2.4.

5.2.1 Rappels

Par convention, nous noterons, sauf contre indication, les horizons de définition d'une fonction en index. Soit par exemple $\pi_{\ell:\ell+K}$ une politique pour les décisions sur les horizons $\tau = \ell, \ell+1, \cdots, \ell+K$, avec deux entiers naturels $\ell \geqslant 0$ et $K \leqslant N-1$. Pour alléger les

notations, nous désignons π_τ la politique $\pi_{0:\tau}$ définie sur les horizons allant de 0 à τ, et en particulier nous dénotons π la politique $\pi_{0:N-1}$. Une politique $\pi = (d_0, d_1, \cdots, d_{N-1})$ désigne une politique non Markovienne et aléatoire. C'est à dire qu'à chaque horizon $\tau = 0, 1, \cdots, N-1$, le décideur en possession de π choisira l'action à simuler selon la règle de décisions $d_\tau : H_\tau \to \mathscr{P}(A_\tau)$, une règle de décision non Markovienne et aléatoire qui associe à tout historique h_τ une action a_τ suivant une distribution de probabilités $\mathbb{P}_{A_\tau}^{d_\tau}(\cdot|h_\tau)$. Rappelons que $H_\tau = \Omega_0 \times A_0 \times \cdots \times A_{\tau-1} \times \Omega_\tau$ est l'ensemble des historiques d'actions et d'observations précédents l'horizon $(\tau+1)$. Comme décrit au Chapitre 2, chaque politique $\pi \in \Pi^{\mathbf{HA}}$ génère une distribution de probabilités $\mathbb{P}_{H_N}^{\pi}(\cdot)$ sur l'ensemble H_N. Nous dénotons $D_\tau^{\mathbf{HA}}$ l'ensemble des règles de décisions d_τ non Markoviennes et aléatoires, pour tout horizon $\tau = 0, 1, \cdots, N-1$.

5.2.2 États et récompenses

À chaque réalisation d'un historique $h = (s_0, a_0, \omega_1, \cdots, a_{N-1}, \omega_N)$ correspond une séquence de récompenses $\{r_0(h_0, a_0), r_1(h_1, a_1), \cdots, r_{N-1}(h_{N-1}, a_N), r_N(h_N)\}$, où l'historique h_τ est défini récursivement comme suit $h_\tau = (h_{\tau-1}, a_\tau, \omega_\tau)$, pour tout horizon $\tau = 1, 2, \cdots, N$ et pour $\tau = 0$. En outre, un historique h_0 est donné. Dans certain cas le décideur à connaissance de l'état initial s_0 du système et dans ce cas $h_0 = (s_0)$. La récompense immédiate est alors donnée par :

$$r_\tau(h_\tau, a_\tau) = \mathbb{P}(s_\tau|h_{\tau-1}, a_{\tau-1}, \omega_\tau) R(s_\tau, a_\tau), \tag{5.1}$$

pour tout $\tau = 0, 1, \cdots, N-1$; et par

$$r_N(h_N) = \mathbb{P}(s_N|h_{N-1}, a_{N-1}, \omega_N) R(s_N), \tag{5.2}$$

pour $\tau = N$.

Soit $\rho_\tau = r_\tau(h_\tau, a_\tau)$ la récompense aléatoire perçue pour tout horizon $\tau = 0, 1, \cdots, N-1$, et pour $\tau = N$, $\rho_N = r_N(h_N)$. Nous dénotons $\rho = (\rho_0, \rho_1, \cdots, \rho_N)$ la séquence aléatoire des récompenses perçues pour tout horizon $\tau = 0, 1, \cdots, N$, et Φ l'ensemble de toutes les séquences aléatoires de récompenses ρ. Une politique π induit alors une distribution de probabilités $\mathbb{P}_\Phi^\pi(\cdot)$ sur l'ensemble Φ :

$$\mathbb{P}_\Phi^\pi(\rho) = \mathbb{P}^\pi \left\{(s_0, a_0, \omega_1, \cdots, a_{N-1}, \omega_N)|\rho = (r_0(h_0, a_0), \cdots, r_{N-1}(h_{N-1}, a_{N-1}), r_N(h_N))\right\}$$

En d'autres termes, la probabilité $\mathbb{P}_\Phi^\pi(\rho)$ de la séquence aléatoire de récompenses $\rho = (r_0(h_0, a_0), \cdots, r_{N-1}(h_{N-1}, a_{N-1}), r_N(h_N))$ est conditionnée par l'historique $h_N = (s_0, a_0, \omega_1, \cdots, a_{N-1}, \omega$ correspondant, c'est à dire $\mathbb{P}_\Phi^\pi(\rho) = \mathbb{P}_{H_N}^\pi(h_N)$.

Par un raisonnement parallèle au précédent, nous démontrons qu'une politique π induit une distribution de probabilités $\mathbb{P}_{H_\tau}^{\pi_{\tau-1}}(\cdot)$ sur l'ensemble H_τ, pour tout horizon $\tau = 0, 1, \cdots, N$. En effet, à une réalisation d'une séquence aléatoire d'observations $(s_0, \omega_1, \omega_2, \cdots, \omega_\tau)$ pour tout horizon $\tau = 1, \cdots, N$, correspond un historique $h_\tau = (s_0, a_0, \omega_1, a_1, \cdots, a_{\tau-1}, \omega_\tau)$, avec $a_\tau = d_\tau(h_{\tau-1}, a_{\tau-1}, \omega_\tau)$ pour tout horizon $\tau = 1, \cdots, N$, et pour $\tau = 0$, $a_0 = d_0(h_0)$. Nous dénotons $r_\tau(H_\tau, d_\tau)$ la récompense perçue par le décideur et donnée par :

$$\begin{aligned} r_\tau(H_\tau, d_\tau) &\equiv \mathbb{E}_\pi\{r_\tau(h_\tau, a_\tau)\} && (5.3)\\ &= \sum_{h_\tau \in H_\tau} \mathbb{P}_{H_\tau}^{\pi_{\tau-1}}(h_\tau) \sum_{a_\tau \in A_\tau} \mathbb{P}_{A_\tau}^{d_\tau}(a_\tau | h_\tau) r_\tau(h_\tau, a_{\tau-1}), && (5.4)\end{aligned}$$

pour tout horizon $\tau = 0, 1, \cdots, N-1$; et par :

$$\begin{aligned} r_N(H_N) &= \mathbb{E}_\pi\{r_N(h_N)\} && (5.5)\\ &= \sum_{h_N \in H_N} \mathbb{P}_{H_N}^{\pi_{N-1}}(h_N) r_N(h_N), && (5.6)\end{aligned}$$

pour $\tau = N$.

Soit $v_N^\pi(H_0)$ le total espéré des récompenses cumulées de l'horizon $\tau = 0$ à l'horizon $\tau = N$, si la politique π est utilisée et que le système est dans l'état $s_0 \in S$ initialement. Pour toute politique $\pi \in \Pi^{\mathbf{HA}}$, cette quantité est définie par :

$$v_N^\pi(H_0) \equiv \mathbb{E}_\pi\left\{\sum_{\tau=0}^{N-1} r_\tau(H_\tau, d_\tau) + r_N(H_N)\right\} \quad (5.7)$$

Par hypothèse les récompenses sont bornées, c'est à dire $|r_\tau(H_\tau, d_\tau)| \leqslant M < \infty$, et les ensembles S et A sont discrets. Alors, v_N^π existe et est bornée pour chaque politique $\pi \in \Pi^{\mathbf{HA}}$ et chaque horizon N.

5.2.3 Politique et valeur optimales

Pour un problème de contrôle des processus décisionnels de Markov, l'objectif est de déterminer une politique $\pi^* \in \Pi^{\mathbf{HA}}$ dont l'espérance des récompenses est la plus grande possible, et caractériser sa valeur. En d'autres termes, nous cherchons une politique $\pi^* \in \Pi^{\mathbf{HA}}$ telle que : $\forall H_0$,

$$v_N^{\pi^*}(H_0) \geqslant v_N^\pi(H_0), \quad (5.8)$$

pour toute politique $\pi \in \Pi^{\mathbf{HA}}$. On appelle politique optimale une telle politique. Dans certain cas, une politique optimale n'existe pas, on recherche alors des politiques dites ε-optimales. Pour tout $\varepsilon > 0$, une politique ε-optimale et notée π_ε^* est telle que : $\forall H_0$,

$$v_N^{\pi_\varepsilon^*}(H_0) + \varepsilon \geqslant v_N^\pi(H_0), \quad (5.9)$$

pour toute politique $\pi \in \Pi^{\mathbf{HA}}$.

Nous recherchons également à caractériser la valeur v_N^* associée à un problème de contrôle des processus décisionnels de Markov, définie par : $\forall H_0$,

$$v_N^*(H_0) \equiv \sup_{\pi \in \Pi^{\mathbf{HA}}} v_N^{\pi}(H_0), \tag{5.10}$$

et lorsque le supremum est atteint, nous la réécrivons comme suit : $\forall H_0$,

$$v_N^*(H_0) = \max_{\pi \in \Pi^{\mathbf{HA}}} v_N^{\pi}(H_0), \tag{5.11}$$

L'espérance des récompenses d'une politique optimale π^* est telle que : $\forall H_0$,

$$v_N^{\pi^*}(H_0) = v_N^*(H_0), \tag{5.12}$$

de même la valeur d'une politique ε-optimale π_ε^* est telle que : $\forall H_0$,

$$v_N^{\pi_\varepsilon^*}(H_0) + \varepsilon > v_N^*(H_0), \tag{5.13}$$

Par définition d'un supremum, une politique ε-optimale π_ε^* existe toujours pour tout $\varepsilon > 0$. Nous avons supposé que le décideur désirait sélectionner une politique optimale pour tous les états initiaux possibles du système. En pratique, il peut connaître l'état initial du système et ainsi n'avoir qu'à sélectionner la politique optimale pour ce seul état. Le cas échant, il cherchera une politique qui maximise $\sum_s \mathbb{P}(s = s_0) v_N^*(H_0)$, avec $\mathbb{P}(\cdot)$ la distribution de probabilités sur l'état initial du système. Clairement, il pourrait résoudre ce problème en déterminant une politique optimale pour tous les états s du système tels que $\mathbb{P}(s = s_0) > 0$.

Nous appelons problème de contrôle des processus décisionnels de Markov un processus de Markov centralisé ou distribué et complètement ou partiellement observable conjointement avec un critère d'optimalité. Cette classe de problèmes inclut les MDPs, POMDPs, DEC-MDPs et DEC-POMDPs. Les problèmes auxquels nous nous intéressons dans ce chapitre sont des problèmes de contrôle des processus décisionnels de Markov à temps discrets, à horizon fini. De plus, ils disposent comme critère d'optimisation l'espérance des récompenses. Nous nous intéressons plus particulièrement au contrôle distribué dans les dernières sections de ce chapitre.

5.2.4 Évaluation d'une politique quelconque

La théorie de la décision dans les processus de Markov est basée sur l'usage du raisonnement rétrograde (programmation dynamique) afin d'évaluer récursivement les récom-

penses espérées. Dans cette section, nous exposons une méthode d'évaluation de l'espérance des récompenses pour une politique donnée.

Soit $\pi = (d_1, d_2, \cdots, d_{N-1})$ une politique aléatoire et non Markovienne. Posons $u_\tau^\pi : H_\tau \to \Re$ l'espérance des récompenses obtenues en utilisant la politique π aux horizons $\tau, \tau+1, \cdots, N-1$. Si l'ensemble des historiques à l'horizon τ est H_τ, alors nous définissons u_τ^π pour tout $\tau < N$ par :

$$u_\tau^\pi(H_\tau) = \mathbb{E}_\pi\left\{\sum_{\ell=\tau}^{N-1} r_\ell(H_\ell, d_\ell) + r_N(H_N)\right\} \tag{5.14}$$

et soit $u_N^\pi(H_N) = r_N(H_N)$, pour $H_N = H_{N-1} \times A_{N-1} \times \Omega_N$. Il est possible de lier ces deux valeurs par $u_\tau^\pi(H_\tau) = v_{\tau:N}^\pi(H_\tau)$, pour tout $\tau = 0, 1, \cdots, N$.

Nous démontrons maintenant comment calculer v_N^π en évaluant par un raisonnement inductif u_τ^π. Nous appelons cet algorithme l'algorithme d'évaluation d'une politique à horizon fini. Cet algorithme peut être simplifié. Pour ce faire, l'on restreint la classe des politiques aux politiques $\pi \in \Pi^{\mathbf{HD}}$ non Markoviennes et déterministes. D'un autre côté, les résultats à venir requièrent ce niveau d'abstraction.

Algorithme d'évaluation d'une politique à horizon fini

1. Posons $\tau = N$ et $u_N^\pi(H_N) = r_N(H_N)$ pour tout H_N.
2. Si $\tau = 0$, arrêter, sinon aller à l'étape 3.
3. Posons $\tau \leftarrow (\tau - 1)$ et calculons $u_\tau^\pi(H_\tau)$ pour chaque H_τ par :

$$u_\tau^\pi(H_\tau) = r_\tau(H_\tau, d_\tau) + u_{\tau+1}^\pi(H_{\tau+1}) \tag{5.15}$$

   notons que $H_{\tau+1} = H_\tau \times A_\tau \times \Omega_{\tau+1}$.
4. Retourner à l'étape 2.

♣

Nous dérivons la valeur $u_\tau^\pi(H_\tau)$, équation (5.15), pour tout $\tau = 0, 1, \cdots, N$, comme suit : pour toute observation conjointe $o \equiv (\omega_1, \omega_2, \cdots, \omega_N)$,

$$u_\tau^\pi(H_\tau) = r_\tau(H_\tau, d_\tau) + \mathbb{E}_\pi\left\{u_{\tau+1}^\pi(H_\tau, A_\tau, \omega_{\tau+1})\right\} \tag{5.16}$$

$$= r_\tau(H_\tau, d_\tau) + u_{\tau+1}^\pi(H_{\tau+1}) \tag{5.17}$$

L'équation (5.16) tient par définition de la valeur u_τ^π, en effet l'espérance des récompenses à l'issue de l'exécution d'une politique, π sur les horizons $\tau, \tau+1, \cdots, N$, lorsque l'ensemble des historiques à l'horizon τ est H_τ, est égale à la récompense immédiate reçue en sélectionnant la règle de décisions d_τ plus l'espérance des récompenses sur le reste

des horizons. L'équation (5.17) s'explique par le fait qu'il n'existe qu'un seul successeur possible à l'ensemble H_τ à savoir l'ensemble $H_{\tau+1} = H_\tau \times A_\tau \times \Omega_{\tau+1}$, pour tout horizon $\tau = 0, 1, \cdots, N-1$. Ce raisonnement inductif réduit le problème du calcul de l'espérance des récompenses sur un horizon de N à une séquence de $(N-1)$ calculs similaires avec comme récompense immédiate r_τ et récompenses finales $u^\pi_{\tau+1}$. Le théorème suivant résumant cette discussion, établit formellement que l'algorithme ci-dessus trouve en effet u^π_τ. La preuve est construite par induction avec comme index d'induction τ.

Théorème 4. *Soit $\pi \in \Pi^{\mathbf{HA}}$, supposons que u^π_τ ait été généré par l'algorithme d'évaluation d'une politique à horizon fini. Il s'ensuit alors que pour tout $\tau \leqslant N$ l'équation (5.14) est vraie, et $v^\pi_N(H_0) = u^\pi_0(H_0)$ pour tout H_0.*

Démonstration. Le résultat est trivialement vrai pour $\tau = N$, par hypothèse $v^\pi_{N:N}(H_N) = u^\pi_N(H_N) = r_N(H_N)$ pour tout H_N. Supposons maintenant que l'équation (5.14) soit vrai pour tout horizon $\tau+1, \tau+2, \cdots, N$. En utilisant la définition de u^π_τ équation (5.16) et l'hypothèse d'induction, il vient alors que :

$$
\begin{aligned}
u^\pi_\tau(H_\tau) &= r_\tau(H_\tau, d_\tau) + \mathbb{E}_\pi\left[\mathbb{E}_\pi\left\{\sum_{\ell=\tau+1}^{N-1} r_\ell(H_\ell, d_\ell) + r_N(H_N)\right\}\right] && (5.18)\\
&= r_\tau(H_\tau, d_\tau) + \mathbb{E}_\pi\left\{\sum_{\ell=\tau+1}^{N-1} r_\ell(H_\ell, d_\ell) + r_N(H_N)\right\} && (5.19)\\
&= \mathbb{E}_\pi\left\{\sum_{\ell=\tau}^{N-1} r_\ell(H_\ell, d_\ell) + r_N(H_N)\right\} && (5.20)\\
&= \sum_{\ell=\tau}^{N-1} r_\ell(H_\ell, d_\ell) + r_N(H_N) && (5.21)
\end{aligned}
$$

Le terme entre crochets de l'égalité (5.18) est réduit au terme entre accolades dans l'équation (5.19) car la transition de l'ensemble d'historiques H_τ à l'ensemble d'historiques $H_{\tau+1} = H_\tau \times A_\tau \times \Omega_{\tau+1}$ est déterministe. De plus, comme H_τ est connu à l'horizon τ, le premier terme de l'égalité (5.19) peut être inclut dans l'espérance, cela établit le résultat équation (5.20). À horizon τ, il n'existe qu'un seul ensemble H_τ, ainsi se déduit le résultat souhaité, équation (5.21). □

5.3 Principe d'optimalité

Dans cette section, nous étendons les équations d'optimalité appelées équations de Bellman au cadre général du contrôle des processus décisionnels de Markov quels qu'ils soient, et nous étudions les propriétés de ces équations dans ce contexte. Nous montrons

par la suite que les solutions de ces équations correspondent aux fonctions de valeurs optimales et permettent d'extraire les politiques optimales.

5.3.1 Équations d'optimalité

Soit

$$v^*_{\tau:N}(H_\tau) = \sup_{\pi \in \Pi^{\mathbf{HA}}} v^\pi_{\tau:N}(H_\tau) \tag{5.22}$$

L'équation (5.22) dénote le supremum sur l'ensemble des politiques $\pi \in \Pi^{\mathbf{HA}}$ non Markoviennes et aléatoires, sur la somme espérée des récompenses cumulées aux horizons $\tau, \tau+1, \cdots, N-1$, pour l'ensemble d'historiques H_τ. Pour $\tau > 1$, nous n'avons pas à considérer toutes les politiques pour estimer le supremum ci-dessus. Sachant l'ensemble d'historiques H_τ, nous ne considérons que les politiques partielles $\pi_{\tau:N-1}$ partant de l'horizon τ. C'est à dire que nous ne devons calculer que le supremum sur les politiques $\pi_{\tau:N-1} = (d_\tau, d_{\tau+1}, \cdots, d_{N-1})$, telles que $\pi_{\tau:N-1} \in \Pi^{\mathbf{HA}}_{\tau:N-1}$. Nous rappelons que l'ensemble des politiques $\Pi^{\mathbf{HA}}_{\tau:N-1}$ est donné par $\Pi^{\mathbf{HA}}_{\tau:N-1} = D^{\mathbf{HA}}_\tau \times D^{\mathbf{HA}}_{\tau+1} \times \cdots \times D^{\mathbf{HA}}_{N-1}$.

Équations d'optimalité

Les équations d'optimalité sont alors définies comme suit :

$$v_{\tau:N}(H_\tau) = \sup_{d_\tau \in D^{\mathbf{HA}}_\tau} \{r_\tau(H_\tau, d_\tau) + v_{\tau+1:N}(H_{\tau+1})\} \tag{5.23}$$

pout tout horizon $\tau = 0, 1, \cdots, N-1$ et $H_{\tau+1} = H_\tau \times A_\tau \times \Omega_{\tau+1}$. Pour $\tau = N$, nous rajoutons la condition limite suivante : ♣

$$v_{N:N}(H_N) = r_N(H_N) \tag{5.24}$$

avec $H_N = H_{N-1} \times A_{N-1} \times \Omega_N$.

Le supremum de l'équation (5.23) est atteint car l'ensemble A est fini. Nous pouvons alors remplacer l'opérateur sup par l'opérateur max, l'équation (5.23) devient dès lors :

$$v_{\tau:N}(H_\tau) = \max_{d_\tau \in D^{\mathbf{HA}}_\tau} \{r_\tau(H_\tau, d_\tau) + v_{\tau+1:N}(H_{\tau+1})\} \tag{5.25}$$

Une solution au système d'équations (5.25) ou (5.23) et à la condition équation (5.24) est une séquence de fonctions $v_{\tau:N}$: $\{H_\tau\} \to \Re$, avec $\tau = 1, \cdots, N$. Elles ont la propriété que

$v_{N:N}$ satisfait l'équation (5.24), $v_{N:N-1}$ satisfait la (N -1)-ème équation où v_N est substitué au côté droit de la (N -1)-ème équation, et ainsi de suite. Nous signalons que l'ensemble $\{H_\tau\}$ des ensembles d'historiques correspond à tous les ensembles d'historiques accessibles pour tout historique initial $h_0 \in H_0$. Comme la transition d'un ensemble H_τ à un ensemble $H_{\tau+1}$ est déterministe, la cardinalité de $\{H_\tau\}$ est égale à celle de l'ensemble des historiques initiaux $|H_0|$. Bien que l'ensemble des historiques initiaux est un continuum, $H_0 \subset \mathscr{P}(S)$, c'est à dire l'ensemble des distributions de probabilité sur les états initiaux du système, le Lemme 1 ci-dessous établit qu'il suffit de déterminer les solutions des équations d'optimalité sur l'ensemble des états $s \in S$ afin de garantir l'optimalité pour tout historique de H_0, et plus généralement l'optimalité sur H_0. Ainsi, sans perte de généralité on peut mettre l'accent uniquement sur le cas où l'ensemble des historiques initiaux est restreint à un état $H_0 = \{(s_0)\}$, pour tout état $s_0 \in S$.

Lemme 1. *Soit w une fonction intégrable sur un ensemble de Borel W d'un espace mesurable $(\mathscr{F}, o, \mathbb{P})$. Il vient alors que,*

$$\sup_{u \in W} w(u) \geqslant \int_{u \in W} \mathbb{P}(u) w(u) \tag{5.26}$$

Démonstration. Soit $w^* = \sup_{u \in W} w(u)$. Si W est un continuum il vient alors que,

$$w^* = \int_{u \in W} \mathbb{P}(u) w^* \geqslant \int_{u \in W} \mathbb{P}(u) w(u). \tag{5.27}$$

□

Le Lemme 1 est en particulier vrai lorsque w est une fonction à valeurs réelles sur un ensemble discret W, dans ce cas simple l'intégrale est remplacé par l'opérateur somme.

5.3.2 Propriétés des solutions

Les équations d'optimalité sont des outils fondamentaux pour les problèmes décisionnels de Markov. Ils offrent de nombreuses et importantes propriétés, y compris les propriétés suivantes :

1. les solutions de ces équations d'optimalité sont les valeurs optimales pour les périodes allant de τ à N ;
2. ces équations offrent une méthode de vérification de l'optimalité d'une politique ou d'extraction d'une politique optimale ;

3. si l'espérance des récompenses d'une politique $\pi_{\tau:N}$ satisfait les équations d'optimalité alors cette politique est optimale sur les horizons $\tau, \tau+1, \cdots, N$;
4. ces équations sont également les bases pour la conception de méthodes efficaces afin de construire des politiques et fonctions de valeurs optimales;
5. enfin, elles peuvent être utiles pour l'identification des propriétés structurelles de la fonction de valeurs ou de la politique.

Le théorème suivant résume les propriétés des solutions des équations d'optimalité. La preuve est construite par induction et illustre bon nombre des principes de programmation dynamique. Elle se décompose en deux parties. Tout d'abord, nous établissons que les solutions des équations d'optimalité constituent des bornes supérieures par rapport à $v^*_{\tau:N}$ et par la suite nous démontons l'existence d'une politique π pour laquelle $v^{\pi}_{\tau:N}$ est arbitrairement proche de $v_{\tau:N}$.

Théorème 5. *Étant donné v_τ une solution à (5.23) pour $\tau = 1, \cdots, N-1$, et v_N satisfaisant la condition limite (5.24). Il s'ensuit alors que, pour tout ensemble d'historiques H_τ et tout horizon $\tau = 1, \cdots, N$,*

$$v_{\tau:N}(H_\tau) = v^*_{\tau:N}(H_\tau), \tag{5.28}$$

et pour $\tau = N$,

$$v_{N:N}(H_N) = r_N(H_N). \tag{5.29}$$

pour tout ensemble d'historiques H_N.

Démonstration. La preuve se décompose en deux parties. Premièrement, nous établissons par induction que pour tout H_ℓ et $\ell = 0, 1, \cdots, N$:

$$v_{\ell:N}(H_\ell) \geqslant v^*_{\ell:N}(H_\ell) \tag{5.30}$$

Comme aucune décision n'est prise à l'horizon $\ell = N$,

$$v_{N:N}(H_N) = r_N(H_N) = v^{\pi}_{N:N}(H_N), \tag{5.31}$$

pour tout H_N et $\pi \in \Pi^{\mathbf{HA}}$.

Supposons maintenant que $v_{\tau:N}(H_\tau) \geqslant v^*_{\tau:N}(H_\tau)$, pour tout H_τ à tout horizon $\tau = (\ell+1), \cdots, N$. Soit $\pi' = (d'_1, d'_2, \cdots, d'_{N-1})$ une politique non Markovienne et aléatoire $\pi \in \Pi^{\mathbf{HA}}$. Pour $\tau = \ell$, l'équation d'optimalité s'écrit :

$$v_{\ell:N}(H_\ell) = \sup_{d_\ell \in D^{\mathbf{HA}}_\ell} \{r_\ell(H_\ell, d_\ell) + v_{\ell+1:N}(H_{\ell+1})\} \tag{5.32}$$

Il s'ensuit par hypothèse d'induction :

$$\begin{aligned} v_{\ell:N}(H_\ell) &\geqslant \sup_{d_\ell \in D_\ell^{\mathbf{HA}}} \left\{ r_\ell(H_\ell, d_\ell) + v^*_{\ell+1:N}(H_{\ell+1}) \right\} && (5.33) \\ &\geqslant \sup_{d_\ell \in D_\ell^{\mathbf{HA}}} \left\{ r_\ell(H_\ell, d_\ell) + v^{\pi'}_{\ell+1:N}(H_{\ell+1}) \right\} && (5.34) \\ &\geqslant r_\ell(H_\ell, d'_\ell) + v^{\pi'}_{\ell+1:N}(H_{\ell+1}) && (5.35) \\ &= v^{\pi'}_{\ell:N}(H_\ell) && (5.36) \end{aligned}$$

L'inégalité (5.33) suit directement de l'hypothèse d'induction. L'inégalité (5.34) est déduite de la définition de $v^*_{\ell+1:N}$, en effet $v^*_{\ell+1:N}(H_{\ell+1}) \geqslant v^{\pi'}_{\ell+1:N}(H_{\ell+1})$ pour toute politique π' et pour tout horizon $\ell = 0, 1, \cdots, N$. L'inégalité (5.35) résulte par définition de l'opérateur sup. La dernière égalité (5.36) suit de l'équation (5.15) et du Théorème 4.

Comme π' est arbitraire, il s'ensuit que : pour toute politique $\pi \in \Pi^{\mathbf{HA}}$,

$$v_{\ell:N}(H_\ell) \geqslant v^{\pi}_{\ell:N}(H_\ell) \tag{5.37}$$

Par conséquent $v_{\ell:N}(H_\ell) \geqslant v^*_{\ell:N}(H_\ell)$ et l'hypothèse d'induction est vérifiée.

Maintenant, nous allons établir que pour tout $\varepsilon > 0$, il existe une politique $\pi' \in \Pi^{\mathbf{HA}}$ pour laquelle,

$$v^{\pi'}_{\ell:N}(H_\ell) + (N - \ell)\varepsilon \geqslant v_{\ell:N}(H_\ell) \tag{5.38}$$

pour tout H_ℓ et tout horizon $\ell = 0, 1, \cdots, N$. Afin d'y parvenir, nous construisons une politique $\pi' = (d_1, d_2, \cdots, d_{N-1})$ en choisissant d_n telle que : $d_\ell \in D_\ell^{\mathbf{HA}}$ et,

$$r_\ell(H_\ell, d_\ell) + v_{\ell+1:N}(H_{\ell+1}) + \varepsilon \geqslant v_{\ell:N}(H_\ell) \tag{5.39}$$

Nous établissons le résultat équation (5.38) par induction. Sachant $v^{\pi'}_{N:N}(H_N) = v_{N:N}(H_N)$, l'hypothèse d'induction tient pour $\tau = N$. Supposons maintenant que

$$v^{\pi'}_{\tau:N}(H_\tau) + (N - \tau)\varepsilon \geqslant v_{\tau:N}(H_\tau) \tag{5.40}$$

pour tout horizon $\tau = (\ell + 1), (\ell + 2), \cdots, N$.

Il vient alors d'après l'équation (5.15) et le Théorème 4 et l'inégalité (5.39) que :

$$\begin{aligned} v^{\pi'}_{\ell:N}(H_\ell) &= r_\ell(H_\ell, d_\ell) + v^{\pi'}_{\ell+1:N}(H_{\ell+1}) && (5.41) \\ &\geqslant r_\ell(H_\ell, d_\ell) + v_{\ell+1:N}(H_{\ell+1}) - (N - \ell - 1)\varepsilon && (5.42) \\ &\geqslant v_{\ell:N}(H_{\ell:N}) - (N - \ell)\varepsilon && (5.43) \end{aligned}$$

L'inégalité (5.41) suit directement l'évaluation d'une politique équation (5.15). L'hypothèse d'induction pour $\tau = \ell + 1$ permet de déduire (5.42) et enfin l'inégalité (5.43) est

donnée par la définition de la politique π' à l'inégalité (5.39). Ainsi l'hypothèse d'induction est satisfaite pour $\tau = \ell$ et l'inégalité (5.38) tient pour $\tau = 0, 1, \cdots, N$. Donc pour tout $\varepsilon > 0$, il existe une politique $\pi' \in \Pi^{\mathbf{HA}}$ pour laquelle :

$$\begin{aligned} v^*_{\ell:N}(H_\ell) + (N-\ell)\varepsilon &\geqslant v^{\pi'}_{\ell:N}(H_\ell) + (N-\ell)\varepsilon && (5.44) \\ &\geqslant v^*_{\ell:N}(H_\ell) && (5.45) \end{aligned}$$

□

Le théorème ci-dessus établit que les solutions des équations d'optimalité sont les fonctions de valeurs optimales pour les horizons τ et au-delà. Le résultat suivant montre comment utiliser les équations d'optimalité afin d'extraire les politiques optimales, et de vérifier qu'une politique est optimale.

Théorème 6. *Soient les fonctions $v^*_{\tau:N}$, pour $\tau = 0, 1, \cdots, N$, solutions des équations d'optimalité (5.25) et satisfaisant la condition (5.24). Soit également $\pi^* = (d^*_1, d^*_2, \cdots, d^*_{N-1}) \in \Pi^{\mathbf{HA}}$ une politique, telle que :*

$$r_\tau(H_\tau, d^*_\tau) + v^*_{\tau+1:N}(H_{\tau+1}) = \max_{d_\tau \in D^{\mathbf{HA}}_\tau} \left\{ r_\tau(H_\tau, d_\tau) + v^*_{\tau+1:N}(H_{\tau+1}) \right\} \quad (5.46)$$

pout tout $\tau = 0, 1, \cdots, N$. Il vient alors que, pour chaque $\tau = 0, 1, \cdots, N$, et tout ensemble d'historiques H_τ,

$$v^{\pi^*}_{\tau:N}(H_\tau) = v^*_{\tau:N}(H_\tau), \quad (5.47)$$

Démonstration. La preuve est construite par un raisonnement inductif. Il est facile de montrer que $v^{\pi^*}_{N:N}(H_N) = r_N(H_N) = v^*(H_N)$, pour tout ensemble d'historiques H_N à l'horizon N.

Supposons maintenant que le résultat équation (5.46) soit vérifié pour $\tau = (\ell+1), \cdots, N$. Par suite, pour tout $H_\ell = H_{\ell-1} \times A_{\ell-1} \times \Omega_\ell$,

$$\begin{aligned} v^*_{\ell:N}(H_\ell) &= \max_{d_\ell \in D^{\mathbf{HA}}_\ell} \left\{ r_\ell(H_\ell, d_\ell) + v^*_{\ell+1:N}(H_{\ell+1}) \right\} \\ &= r_\ell(H_\ell, d^*_\ell) + v^{\pi^*}_{\ell+1:N}(H_{\ell+1}) \\ &= v^{\pi^*}_{\ell:N}(H_\ell) \end{aligned}$$

La première égalité suit directement l'équation (5.46) et la seconde égalité est une conséquence de l'hypothèse d'induction, tandis que la dernière suit le Théorème 4. Donc l'hypothèse d'induction est satisfaite et le résultat est vérifié.

□

L'équation (5.46) peut être réécrite comme suit,

$$d_\tau^* = \underset{d_\tau \in D_\tau^{\mathbf{HA}}}{\arg\max} \{r_\tau(H_\tau, d_\tau)) + v_{\tau+1:N}^*(H_{\tau+1})\} \tag{5.48}$$

Le théorème 6 établit qu'une politique optimale est trouvée en résolvant tout d'abord les équations d'optimalité, puis en choisissant pour chaque ensemble d'historiques H_τ une règle de décision parmi l'ensemble des règles de décision non Markoviennes et aléatoires $D_\tau^{\mathbf{HA}}$. S'il existe plusieurs règles de décision maximales, alors il y a plus d'une politique optimale.

5.3.3 Enoncé du principe d'optimalité

Le Théorème 6 offre une présentation formelle du principe d'optimalité, résultat fondamental de la programmation dynamique. Une des toutes premières citations de ce principe remonte à Bellman :

Enoncé du principe d'optimalité

« An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision. » ♣

[Bellman, 1957].

Le principe d'optimalité tel qu'énoncé alors par Bellman n'est pas valide pour les processus de Markov partiellement observables, car l'état courant du système y est inconnu. Cependant si l'on considère l'état comme étant l'information exhaustive collectée durant tous les horizons de l'horizon 0 à l'horizon τ, c'est à dire les ensembles d'historiques H_τ, alors le principe d'optimalité reste valide et s'étend aisément au contrôle centralisé ou distribué des processus décisionnels de Markov partiellement observables, à temps discrets et horizon fini. Ainsi, toute politique π^* satisfaisant l'équation (5.46) possède ces propriétés, de plus une telle politique existe toujours car les maximums sont atteints.

Dans le cas où le supremum (5.23) n'est pas atteint, ou alors que les ressources disponibles ne permettent pas de calculer la politique optimale π^*, il est possible de calculer une politique arbitrairement proche de la politique optimale – c'est à dire une politique ε-optimale. Afin d'y parvenir, nous modifions le théorème précédent comme suit. La preuve est construite sur la base de celle du Théorème 5.

Théorème 7. *Soit $\varepsilon > 0$ un réel arbitraire. Supposons les fonctions $v^*_{\tau:N}$, pour $\tau = 0, 1, \cdots, N$, solutions des équations d'optimalité (5.23) et (5.24). Soit $\pi^\varepsilon = (d_1^\varepsilon, d_2^\varepsilon, \cdots, d_{N-1}^\varepsilon) \in \Pi^{\mathbf{HA}}$ une politique non Markovienne et aléatoire telle que :*

$$r_\tau(H_\tau, d_\tau^\varepsilon) + v^*_{\tau+1:N}(H_{\tau+1}) + \frac{\varepsilon}{N} \geqslant \sup_{d_\tau \in D_\tau^{\mathbf{HA}}} \left\{ r_\tau(H_\tau, d_\tau) + v^*_{\tau+1:N}(H_{\tau+1}) \right\} \tag{5.49}$$

avec $H_{\tau+1} = H_\tau \times A_\tau \times \Omega_{\tau+1}$ et $\tau = 0, 1, \cdots, N$. Il vient alors que,

1. *pour chaque $\tau = 1, 2, \cdots, N$,*

$$v^{\pi^\varepsilon}_{\tau:N}(H_\tau) + (N - \tau)\frac{\varepsilon}{N} \geqslant v^*_{\tau:N}(H_\tau) \tag{5.50}$$

2. *π^ε est une politique ε-optimale, c'est à dire pour tout H_0,*

$$v^{\pi^\varepsilon}_{0:N}(H_0) + \varepsilon \geqslant v^*_{0:N}(H_0) \tag{5.51}$$

Démonstration. Nous établissons ce résultat suivant un raisonnement par induction. Clairement, l'hypothèse d'induction tient pour $\tau = N$, car $v^{\pi^\varepsilon}_{N:N}(H_N) = r_N(H_N) = v^*_{N:N}(H_N)$. Supposons maintenant que :

$$v^{\pi^\varepsilon}_{\tau:N}(H_\tau) + (N - \tau)\frac{\varepsilon}{N} \geqslant v^*_{\tau:N}(H_\tau) \tag{5.52}$$

pour tout horizon $\tau = (\ell + 1), \cdots, N$. Il vient alors d'après l'équation (5.15), le Théorème 5 et l'inégalité (5.52) que :

$$\begin{aligned} v^{\pi^\varepsilon}_{\ell:N}(H_\ell) &= r_\ell(H_\ell, d_\ell) + v^{\pi^\varepsilon}_{\ell+1:N}(H_{\ell+1}) && (5.53)\\ &= r_\ell(H_\ell, d_\ell) + v^*_{\ell+1:N}(H_{\ell+1}) + (N - \ell - 1)\frac{\varepsilon}{N} && (5.54)\\ &= v^*_{\ell:N}(H_\ell) + (N - \ell)\frac{\varepsilon}{N} && (5.55) \end{aligned}$$

L'hypothèse de récurrence est ainsi vérifiée pour $\tau = \ell$, et l'inégalité (5.52) tient pour $\tau = 0, 1, \cdots, N$. Donc pour tout $\varepsilon > 0$, il existe une politique $\pi^\varepsilon \in \Pi^{\mathbf{HA}}$ pour laquelle :

$$\begin{aligned} v^{\pi^\varepsilon}_{\ell:N}(H_\ell) + (N - \ell)\frac{\varepsilon}{N} &\geqslant v^{\pi^\varepsilon}_{\ell:N}(H_\ell) + (N - \ell)\frac{\varepsilon}{N} && (5.56)\\ &\geqslant v^*_{\ell:N}(H_\ell) && (5.57) \end{aligned}$$

en particulier pour $\tau = 0$, $v^{\pi^\varepsilon}(H_0) + \varepsilon \geqslant v^*(H_0)$.

□

5.4 Existence de politiques optimales et pures

Dans cette section, nous étudions les propriétés structurelles des politiques optimales. Du point de vue des applications qui motivent cette recherche, plus la classe des politiques est restreinte plus facile est l'implémentation et l'évaluation des politiques correspondantes. Par exemple, il est bien plus facile d'implémenter et d'évaluer une politique markovienne déterministe $\pi \in \Pi^{\mathbf{MD}}$ qu'une politique pure $\pi \in \Pi^{\mathbf{HSD}}$ et encore moins une politique non Markovienne aléatoire $\pi \in \Pi^{\mathbf{HA}}$. Nous démontrons section 5.4.1 respectivement, qu'il existe toujours une politique déterministe et stationnaire, c'est à dire une politique pure, optimale pour tout problème de contrôle des processus décisionnels de Markov à temps discrets et horizon fini.

«Encore une fois ces résultats généralisent celui concernant l'existence d'une politique markovienne et déterministe optimale dans le cadre des MDPs [Putterman, 1994], et celui concernant l'existence d'une politique pure optimale dans le cadre des DEC-POMDPs [Oliehoek et al., 2008]. Le lecteur se rapportera Chapitre 2 Section 1.2.3 pour plus de détails sur les classes de politiques et leurs relations.»

5.4.1 Politiques déterministes

Nous démontrons dans le théorème ci-dessous qu'il existe toujours une politique optimale ou ε-optimale et non Markovienne déterministe, pour tout problème de contrôle distribué des processus décisionnels de Markov.

Théorème 8. *Les assertions suivantes sont vraies :*

1. *Pour tout $\varepsilon > 0$, il existe toujours une politique ε-optimale qui soit non Markovienne et déterministe $\pi^*_\varepsilon \in \Pi^{\mathbf{HD}}$.*
2. *Soit $v^*_{\tau:N}$ une solution des équations (5.23) et satisfaisant (5.24) et supposons que pour chaque horizon τ et ensemble d'historiques H_τ, il existe une politique non Markovienne et déterministe $\pi^* = (d^*_0, d^*_1, \cdots, d^*_{N-1})$, telle que :*

$$r_\tau(H_\tau, d^*_\tau) + v^{\pi^*}_{\tau+1}(H_\tau, d^*_\tau, \Omega_{\tau+1}) = \sup_{d_\tau \in D^{\mathbf{HA}}_\tau} \{r_\tau(H_\tau, d_\tau) + v^*_{\tau+1:H}(H_\tau, d_\tau, \Omega_{\tau+1})\} \quad (5.58)$$

Il existe alors une politique non Markovienne et déterministe qui soit optimale :

$$v^*_{\tau:N}(H_\tau) = v^{\pi^*}_{\tau:N}(H_\tau) \quad (5.59)$$

pour tout horizon $\tau = 0, 1, \cdots, N$.

Démonstration. Nous établissons tout d'abord que pour tout $\varepsilon > 0$, il existe une politique $\pi' \in \Pi^{\mathbf{HD}}$ déterministe et non Markovienne telle que :

$$v^{\pi'}_{\ell:N}(H_\ell) + (N-\ell)\varepsilon \;\geqslant\; v_{\ell:N}(H_\ell) \tag{5.60}$$

pour tout H_ℓ et $\ell = 0, 1, \cdots, N$. Afin d'y parvenir, nous construisons une politique $\pi' = (d_0, d_1, \cdots, d_{N-1}$ en choisissant une règle de décision déterministe d_ℓ telle que :

$$r_\ell(H_\ell, d_\ell) + v_{\ell+1:N}(H_{\ell+1}) + \varepsilon \;\geqslant\; v_{\ell:N}(H_\ell) \tag{5.61}$$

Établissons l'équation (5.60) par un raisonnement inductif. Comme $v_N(H_N) = r_N(H_N)$, l'hypothèse d'induction tient pour l'horizon $\tau = N$. Supposons que $v^{\pi'}_{\tau:N}(H_\tau) + (N-\tau)\varepsilon \geqslant v_{\tau:N}(H_\tau)$ pour tout $\tau = 0, 1, \cdots, N$. Il s'ensuit d'après le Théorème (4) et l'équation (5.57) que :

$$\begin{aligned} v^{\pi'}_{\ell:N}(H_\ell) &= r_\ell(H_\ell, d_\ell) + v^{\pi'}_{\ell+1:N}(H_{\ell+1}) && (5.62)\\ &\geqslant r_\ell(H_\ell, d_\ell) + v_{\ell+1:N}(H_{\ell+1}) + (N-\ell-1)\varepsilon && (5.63)\\ &\geqslant v_{\ell:N}(H_\ell) + (N-\ell)\varepsilon && (5.64) \end{aligned}$$

L'hypothèse d'induction est satisfaite et l'équation (5.60) est vérifiée pour tout horizon $\tau = 0, 1, \cdots, N$. On en déduit l'existence d'une politique ε-optimale non Markovienne et déterministe pour tout problème de contrôle des processus décisionnels de Markov.

Nous démontrons ensuite l'équation (5.61) par un raisonnement inductif. Clairement $v^{\pi^*}_{N:N}(H_N) = r_N(H_N) = v^*_{N:N}(H_N)$. Supposons maintenant que l'hypothèse d'induction soit vérifée pour tout horizon $\tau = (\ell+1), \cdots, N$.

$$\begin{aligned} v^*_{\ell:N}(H_\ell) &= \sup_{d_\ell \in D^{\mathbf{HA}}_\ell} \left\{ r_\ell(H_\ell, d_\ell) + v^*_{\ell+1:N}(H_{\ell+1}) \right\} && (5.65)\\ &= r_\ell(H_\ell, d^*_\ell) + v^{\pi^*}_{\ell+1:N}(H_{\ell+1}) && (5.66)\\ &= v^{\pi^*}_{\ell:N}(H_\ell) && (5.67) \end{aligned}$$

L'égalité (5.66) résulte de l'hypothèse équation (5.58) et la dernière égalité résulte de l'équation (5.15). L'hypothèse d'induction est vérifiée et l'équation (5.59) est vrai.

□

5.4.2 Malédiction de la distributivité

Nous établissons dans cette section les conditions sous lesquelles une politique déterministe calculée par un décideur centrale peut être distribuée à un ensemble $\mathscr{I}$ d'agents,

afin de permettre le contrôle distribué. Nous y établissons également un résultat aussi important que restrictif d'un point pratique, à savoir qu'en raison de la nécessité d'avoir des politiques distributives pour le contrôle distribué, la statistique exhaustive minimale est une croyance sur l'ensemble des historiques H_τ, pour tout horizon $\tau = 0, 1, \cdots, N$. C'est dans ce contexte que nous parlons de *malédiction de la distributivité.*

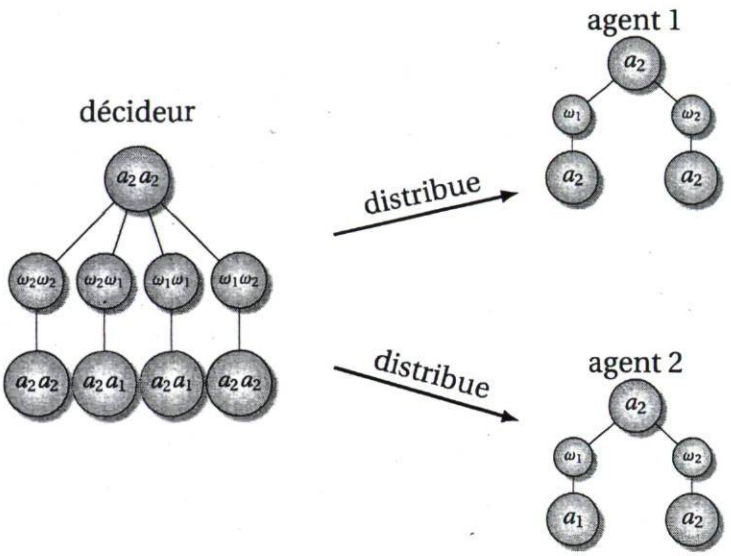


FIGURE 5.2 – Politique distributive et déterministe pour 2 agents.

Définition 14. *Nous appelons* politique déterministe et distributive, *toute politique déterministe* $\pi \in \Pi^{\mathbf{HD}}$ *séparable en plusieurs politiques individuelles et déterministe* $\pi^i \in \Pi^{\mathbf{HD}}$, *une pour chacun des agents* $i \in \mathscr{I}$.

Définition 15. *Nous appelons* règle de décisions déterministe et distributive, *toute règle de décisions déterministe* $d_\tau \in D_\tau^{\mathbf{HD}}$ *séparable en plusieurs règles de décisions déterministes individuelles* $d_\tau^i \in D^{\mathbf{HD}}$, *une pour chacun des agents, pour tout horizon* $\tau = 0, 1, \cdots, N$.

Soit $\bar{\Pi}^{\mathbf{HD}}$ l'ensemble des politiques déterministes et distributives. De même, nous dénotons $\bar{D}^{\mathbf{HD}}$ l'ensemble des règles de décisions déterministes et distributives, pour tout horizon $\tau = 0, 1, \cdots, N$. L'exemple Figure 5.2 illustre une politique déterministe et distributive représentée sous forme d'un arbre de décisions, pour un problème de contrôle distribué des processus décisionnels de Markov à 2 agents. On y observe qu'à la réalisation de toute historique h_τ du décideur équivaut un ensemble d'historiques individuels $\{h_\tau^i\}_{i \in \mathscr{I}}$ un pour chaque agent $i \in \mathscr{I}$, et cela pour tout horizon $\tau = 0, 1, \cdots, N$. Malheureusement, la connaissance d'un historique n'est pas suffisante pour la sélection d'une action contrairement au cas du contrôle centralisé des processus de Markov (voir Chapitre 2)). Plus généralement, nous démontrons que la prise de décisions centralisées dans le cadre du contrôle distribué ne peut se faire qu'en fonction de l'ensemble des historiques H_τ, pour tout horizon $\tau = 0, 1, \cdots, N$. Le théorème suivant résume cette discussion.

Théorème 9. *Soit $v^*_{\tau:N}$ une solution aux équations d'optimalité (5.23) et de la condition limite (5.24), pour tout horizon $\tau = 0, 1, \cdots, N$. La fonction de valeurs $v^*_{\tau:N}$ dépend de l'ensemble des historiques H_τ, pour chaque horizon $\tau = 0, 1, \cdots, N$.*

Démonstration. Dire que la fonction de valeurs $v^*_{\tau:N}$ dépend de l'ensemble des historiques H_τ, revient à dire que la sélection de la règle de décisions d_τ se fait conjointement pour l'ensemble des historiques H_τ. Afin de s'en assurer, raisonnons par l'absurde. Supposons qu'il existe un historique $h \in H_\tau$ dont la sélection de l'action associée $d_\tau(h)$ soit indépendante de l'ensemble des autres historiques $H_\tau \backslash \{h\}$. En outre, supposons que l'historique h équivaut à un ensemble d'historiques individuelles comprenant l'historique individuelle h^i de l'agent $i \in \mathscr{I}$.

Soit un historique $h' \in H_\tau \backslash \{h\}$ dont l'historique individuel de l'agent $i \in \mathscr{I}$ est également h^i.

Supposons que la sélection de action d ?(h) assignée à l'historique h se fasse indépendamment de la sélection de l'action $d_\tau(h')$ assignée à l'historique h'. Il est alors possible que l'action individuelle assignée l'historique individuelle h^i soit différente selon que l'on considère l'action conjointe $d_\tau(h)$ ou $d_\tau(h')$. En d'autres termes, l'agent i pourrait exécuter deux actions individuelles différentes pour le même historique, d'où la contradiction. Par conséquent la fonction de valeurs $v^*_{\tau:N}$ dépend conjointement des historiques H_τ, pour chaque horizon $\tau = 0, 1, \cdots, N$. □

Afin d'illustrer ce théorème, considérons à nouveau l'exemple Figure 5.2. Supposons que nous puissions sélectionner indépendamment des autres historiques, l'action à assigner à $h = (a_2 a_2, \omega_2 \omega_2)$. Soit $(a_2 a_2)$ l'action sélectionnée pour l'historique h indépendamment des autres historiques. Considérons maintenant l'historique $h' = (a_2 a_2, \omega_2 \omega_1)$ et $(a_1 a_2)$ l'action qui lui est assignée indépendamment de h. En extrayant l'historique individuel de l'agent 1, $h^1 = (a_2, \omega_2)$, nous constatons que l'action individuelle qui lui est assigné dépend de l'historique conjoint considéré. Si nous considérons l'historique conjoint h, alors l'action exécutée par l'agent 1 sera et a_2. Si nous considérons l'historique h', alors l'agent 1 exécutera l'action a_1. La sélection des actions individuelles dépend ainsi de l'historique conjoint considéré. Or par définition, lors du contrôle distribué les agents ne perçoivent que leurs propres observations impossible alors de construire un historique conjoint. D'où la dépendance mutuelle de l'ensemble des historiques H_τ dans le processus de sélection de leurs actions.

Nous établissons maintenant que la distribution de probabilités sur l'ensemble des historiques H_τ est la statistique exhaustive pour la planification dans le cadre du contrôle

distribué des problèmes décisionnels de Markov partiellement observables, à temps discrets et à horizon fini.

Théorème 10. *La distribution de probabilités sur l'ensemble des historiques H_τ est la statistique exhaustive pour la sélection des règles de décisions lors du contrôle distribué des processus de Markov, pour tout horizon $\tau = 0, 1, \cdots, N$.*

Démonstration. La preuve est construite à partir du Théorème 9 ci-dessus. Nous avons montré que la fonction de valeurs $v^*_{\tau:N}$ dépendait de la totalité des historiques H_τ, pour tout horizon $\tau = 0, 1, \cdots, N$. Elle est donnée par :

$$v^*_{\tau:N}(H_\tau) = \max_{d \in D^{\mathbf{HD}}} \sum_{h \in H_\tau} \mathbb{P}(h) \cdot r_\tau(h, d(h)) + v^*_{\tau+1:N}(H_{\tau+1}) \tag{5.68}$$

$$= \max_{d \in D^{\mathbf{HD}}} \sum_{h \in H_\tau} \mathbb{P}(h) \sum_{s \in S} \mathbb{P}(s|h) R(s, d(h)) + v^*_{\tau+1:N}(H_{\tau+1}) \tag{5.69}$$

où $H_{\tau+1} = H_\tau d_\tau \omega_\tau$.

□

5.5 Programmation dynamique

La programmation dynamique offre une méthode efficace de résolution des processus décisionnels de Markov à temps discrets et horizon fini. Dans cette section, nous introduisons de nouveaux algorithmes de la programmation dynamique centralisée et montre qu'ils peuvent être utilisés soit pour calculer des politiques ε-optimales soit des politiques approximatives dont l'erreur est bornée. Ces algorithmes sont basés sur les résultats théoriques énoncés aux sections précédentes.

5.5.1 L'algorithme optimal

Nous présentons l'algorithme pour des modèles dont le maximum est atteint pour l'équation d'optimalité (5.23), de sorte que nous soyons assuré d'obtenir une politique déterministe et optimale plutôt qu'une politique déterministe et ε-optimale. L'algorithme résout de façon optimale les équations d'optimalité (5.23) et la condition limite (5.24).

L'un des avantages majeurs de cet algorithme optimal est qu'il donne un cadre générique pour les versions approximatives. En effet, l'applicabilité d'un tel algorithme est limitée à la résolution de problèmes pour des horizons relativement limités $\tau \leqslant 10$.

Algorithme 25 Algorithme d'induction rétrograde (DEC-POMDPs).

1: **procedure** BI
2: Posons $\tau = N$ et $v^*_N(H_N) = r_N(H_N)$ pour tout H_N.
3: **répéter**
4: Posons $\tau \leftarrow (\tau - 1)$.
5: Calculons $v^*_{\tau:N}(H_\tau)$ pour chaque H_τ par :

$$v^*_{\tau:N}(H_\tau) \leftarrow \max_{d_\tau \in \bar{D}^{\mathbf{HD}}_\tau} \left\{ r_\tau(H_\tau, d_\tau) + v^*_{\tau+1:N}(H_{\tau+1}) \right\} \tag{5.70}$$

$$d^*_\tau \leftarrow \operatorname*{arg\,max}_{d_\tau \in \bar{D}^{\mathbf{HD}}_\tau} \left\{ r_\tau(H_\tau, d_\tau) + v^*_{\tau+1:N}(H_{\tau+1}) \right\} \tag{5.71}$$

notons que $H_{\tau+1} \leftarrow H_\tau \times A_\tau \times \Omega_{\tau+1}$.
6: **jusqu'à** $\tau = 0$
7: **fin procedure**

Nous distinguons deux méthodes de représentation de la solution retournée par deux approches différentes du calcul d'une politique distributive optimale (respectivement une politique jointe optimale) dans le cadre du contrôle distribué d'un processus décisionnel de Markov partiellement observable. L'une procède au calcule systématique de l'ensemble des politiques distributives possibles pour chaque horizon $\tau = N-1, N-2, \cdots, 0$. L'autre s'intéresse à l'ensemble des règles de décisions desquels elle extrait la meilleure pour chaque horizon $\tau = N-1, N-2, \cdots, 0$. Comme illustré, en cherchant dans l'ensemble des règles de décisions, nous devons avoir une certaine connaissance de l'ensemble $\bar{D}^{\mathbf{HD}}$ d'où est extrait ces règles de décisions. Pour ce faire, nous pouvons énumérer de façon exhaustive l'ensemble $\bar{D}^{\mathbf{HD}}$ des ces règles de décisions. Pour chaque horizon $\tau = 0, 1, \cdots, N$, cela correspond à un ensemble de dimension $|\bar{D}^{\mathbf{HD}}_\tau|$ doublement exponentielle :

$$|\bar{D}^{\mathbf{HD}}_\tau| = \prod_{i \in \mathcal{I}} |A^i|^{|H^i_\tau|} \tag{5.72}$$

$$= \prod_{i \in \mathcal{I}} |A^i|^{|\Omega^i|^\tau} \tag{5.73}$$

À la différence de l'approche de génération exhaustive de politiques conjointes, notre approche met énormément d'efforts lors de la résolution des derniers horizons $\tau = N-1, N-2, \cdots$. Or ces horizons sont censés être résolus en premier.

En effet, pour chaque règle de décisions à un horizon $\tau = \ell + 1$, toutes les méthodes de programmation dynamique à ce jour doivent considérer une action puis choisir un ensemble de sous politiques jointes de sorte à construire une politique jointe de profondeur τ, pour toute action $a \in A_\tau$ et pour tout horizon $\tau = N-1, N-2, \cdots, 0$. L'approche que nous proposons d'étendre au cadre général du contrôle distribué bien que nouvelle est néanmoins relativement exigeante. Le déterminisme d'une règle de décision d_τ requiert

la connaissance de l'ensemble d'historique H_τ, or ce dernier est à priori inconnu. En générant l'ensemble des ensembles d'historiques $\mathscr{H}_\tau \equiv \{H_\tau\}$, nous pouvons supprimer celles qui sont dominées parmi les règles de décisions $\bar{D}_\tau^{\mathbf{HD}}$ générées, en utilisant la fonction de valeurs $v_{\tau:N}^{d_\tau}(H_\tau)$ donnée par :

$$v_{\tau:N}^{d_\tau}(H_\tau) = r_\tau(H_\tau, d_\tau) + v_{\tau+1:N}^{*}(H_{\tau+1}) \tag{5.74}$$

$$v_{\tau+1:N}^{*}(H_{\tau+1}) = \max_{d_{\tau+1} \in \bar{D}^{\mathbf{HD}}} v_{\tau+1:N}^{d_{\tau+1}}(H_{\tau+1}) \tag{5.75}$$

Nous disons qu'une règle de décisions d_τ est dominée si et seulement si il n'existe aucun ensemble d'historiques H_τ pour lequel d_τ est la règle de décisions optimale : $\forall H_\tau \in \mathscr{H}_\tau$,

$$v_{\tau:N}^{d_\tau}(H_\tau) < v_{\tau:N}^{*}(H_\tau) \tag{5.76}$$

L'intérêt est de ne conserver qu'un nombre minime de règles de décisions, plutôt qu'un ensemble de dimension doublement exponentielle de valeurs pour chaque ensemble $\mathscr{H}_\tau$, pour tout horizon $\tau = 0, 1, \cdots, N$. Pour restreindre encore plus l'ensemble des règles de décisions conservées il est nécessaire de dégrader légèrement la qualité de la solution souhaitée. En recherchant une solution ε-optimale, nous pouvons réduire un peu plus le nombre de règles de décisions à conserver. En effet, une règle de décisions d_τ est ε-dominée si et seulement pour tout ensemble d'historique $H_\tau \in \mathscr{H}_\tau$ il existe une autre règle de décisions $\hat{d}_\tau$ telle que :

$$v_{\tau:N}^{d_\tau}(H_\tau) \leqslant v_{\tau:N}^{\hat{d}_\tau}(H_\tau) + \varepsilon \frac{N-\tau}{N} \tag{5.77}$$

Algorithme 26 Algorithme d'induction rétrograde à base de croyances.

1: **procedure** PBBI
2: Posons $\tau = N$.
3: Générons les ensembles $\mathscr{H}_\tau$, $\forall \tau = 0, 1, \cdots, N$.
4: **répéter**
5: Posons $\tau \leftarrow (\tau - 1)$.
6: Générons l'ensemble des règles de décisions $\bar{D}_\tau^{\mathbf{HD}}$.
7: Supprimons les règles de décisions ε-dominées dans $\bar{D}_\tau^{\mathbf{HD}}$.
8: **jusqu'à** $\tau = 0$
9: **fin procedure**

5.5.2 Convergence et bornes sur l'erreur

Pour tout ensemble d'historiques $H_\tau \in \mathscr{H}_\tau$ pour tout horizon $\tau = 0, 1, \cdots, N$, l'algorithme d'induction rétrograde produit une fonction de valeurs $v_{\tau:N}^{\mathscr{H}_\tau}$. Nous montrons maintenant que l'erreur entre cette fonction de valeurs et la fonction de valeurs optimales $v_{\tau:N}^{*}$ est bornée.

La borne est inspirée de celle introduite dans le cadre de l'algorithme à base de croyances PBVI [Pineau et al., 2003b]. La borne dépend de comment $\mathcal{H}_\tau$ est échantillonné par rapport à $\mathcal{H}_\tau^\infty$; plus l'ensemble $\mathcal{H}_\tau$ est proche de $\mathcal{H}_\tau^\infty$, plus $v_{\tau:N}^{\mathcal{H}_\tau}$ est proche de $v_{\tau:N}^*$. Soit $V_{\tau:N}$ (resp. $V'_{\tau:N}$) est la matrice $|S| \times |H_\tau|$ associée à la fonction de valeurs $v_{\tau:N}^{d'_\tau}$ représentée par un ensemble d'hyperplans, un pour chaque historique de H_τ (resp. H'_τ). Soit B_τ (resp. B'_τ) une matrice $|H_\tau| \times |S|$ représentant l'ensemble des distributions de probabilités associées à chaque historique $h \in H_\tau$ (resp. H'_τ). Afin d'établir l'erreur $\|V_{\tau:N}^{\mathcal{H}_\tau} - V_{\tau:N}^*\|_\infty$ produite par notre algorithme, pour tout horizon $\tau = 0, 1, \cdots, N$, nous commençons par définir la densité ζ_τ dans l'ensemble $\mathcal{H}_\tau$ comme la distance maximale de tout ensemble d'historiques $H'_\tau \in \mathcal{H}_\tau^\infty$ à tout ensemble d'historiques $H_\tau \in \mathcal{H}_\tau$. Plus précisément, nous avons,

$$\zeta_\tau = \max_{H'_\tau \in \mathcal{H}_\tau^\infty} \min_{H_\tau \in \mathcal{H}_\tau} \|H_\tau - H'_\tau\|_1 \tag{5.78}$$

où la norme $\mathbb{L}_1$ est donnée par $\|H - H'\|_1 = \max_{h \in H, h' \in H'} \|\mathbb{P}(h) - \mathbb{P}(h')\|_1$. Nous pouvons dès lors prouver le résultat suivant :

Théorème 11. *L'erreur introduite par l'algorithme d'induction rétrograde à base de croyances pour tout horizon $\tau = 0, 1, \cdots, N$, en planifiant sur $\mathcal{H}_\tau$ plutôt que sur $\mathcal{H}_\tau^\infty$, est bornée par :*

$$\eta_\tau \leqslant (N - \tau)\xi_\tau |r|_\infty \tag{5.79}$$

où $\mathcal{H}_\tau^\infty$ l'ensemble de la totalité des ensembles H_τ ; $\xi_\tau = \|(\mathcal{H}_\tau^\infty, \mathcal{H}_\tau\|$ est la norme-1 ; et $|r|_\infty = \max_{s,a} |R(s,a)|$.

Démonstration. Soit H'_τ l'ensemble des historiques dans $\mathcal{H}_\tau^\infty$ pour lequel l'algorithme d'induction rétrograde à base de croyances fait sa pire erreur et $H_\tau \in \mathcal{H}_\tau$ l'ensemble d'historiques le plus proche de H'_τ suivant la norme $\|\cdot\|_1$. Soit d_τ la règle de décisions maximale pour l'ensemble d'historiques H_τ et d'_τ la règle de décisions maximale pour l'ensemble d'historiques H'_τ. En ne parvenant pas à conserver d'_τ dans l'ensemble des règles de décisions non dominées à l'horizon τ, notre algorithme fait une erreur d'au plus $v_{\tau:N}^{d'_\tau}(H'_\tau) - v_{\tau:N}^{d_\tau}(H'_\tau)$. De plus, comme nous avons $v_{\tau:N}^{d'_\tau}(H_\tau) \leqslant v_{\tau:N}^{d_\tau}(H_\tau) + \varepsilon$ par définition d'une règle de décisions ε-dominée. Ainsi,

$$\begin{aligned}
\eta_\tau &\leqslant v_{\tau:N}^{d'_\tau}(H'_\tau) - v_{\tau:N}^{d_\tau}(H'_\tau) && (5.80)\\
&= v_{\tau:N}^{d'_\tau}(H'_\tau) - v_{\tau:N}^{d_\tau}(H'_\tau) + (v_{\tau:N}^{d'_\tau}(H_\tau) - v_{\tau:N}^{d_\tau}(H_\tau)) && (5.81)\\
&\leqslant v_{\tau:N}^{d'_\tau}(H'_\tau) - v_{\tau:N}^{d_\tau}(H'_\tau) + v_{\tau:N}^{d_\tau}(H_\tau) - v_{\tau:N}^{d_\tau}(H_\tau) && (5.82)\\
&= (V'_{\tau:N} - V_{\tau:N}) \cdot (B'_\tau - B_\tau) && (5.83)\\
&\leqslant \|V'_{\tau:N} - V_{\tau:N}\|_\infty \cdot \|B'_\tau - B_\tau\|_1 && (5.84)\\
&\leqslant \|V'_{\tau:N} - V_{\tau:N}\|_\infty \cdot \zeta_\tau && (5.85)\\
&\leqslant (N - \tau) \cdot |r|_\infty \cdot \zeta_\tau && (5.86)
\end{aligned}$$

□

Théorème 12. *Pour tout ensemble $\mathscr{H} = \{\mathscr{H}_\tau\}_\tau$, l'erreur produit par l'algorithme d'induction rétrograde à base de croyances est bornée par :*

$$\varepsilon \leqslant N \cdot \zeta \cdot |r|_\infty \tag{5.87}$$

où $\zeta = \max_\tau \zeta_\tau$.

La borne décrite dans cette section dépend du degré de densité de l'ensemble $\mathscr{H}_\tau$ par rapport à $\mathscr{H}_\tau^\infty$. L'ensemble $\mathscr{H}_\tau^\infty$ n'est constituée que des ensembles d'historiques H_τ accessibles par échantillonnage. Il est facile d'observer que la borne est valide pour $\mathscr{H}_\tau \equiv \mathscr{H}_\tau^\infty$. Néanmoins, il peut être assez difficile d'échantillonner assez densément l'ensemble $\mathscr{H}_\tau$ afin d'obtenir une borne de bonne qualité. Plutôt que de sélectionner à priori l'ensemble $\mathscr{H}_\tau$, il est possible d'en sélectionner les sous-ensembles H_τ au fur et à mesure de l'estimation de la fonction de valeurs. Cela permettrait de réduire l'erreur sur l'algorithme dans les régions de l'ensemble $\mathscr{H}_\tau^\infty$ où cela est nécessaire. Cette perspective sera envisagée dans des travaux à venir.

5.5.3 Génération des ensembles $\{\mathscr{H}_\tau\}_\tau$

La génération des ensembles $\mathscr{H}_\tau$ se fait par simulation de l'exécution d'une politique déterministe et distributive π à partir d'un historique initial h_0, pour tout horizon $\tau = 0, 1, \cdots, N$. Ces politiques peuvent être construites soit aléatoirement soit de façon heuristique. Une heuristique de génération des ensembles $\mathscr{H}_\tau$, consiste à générer plusieurs ensembles H_τ pour différentes politiques conjointes π. Pour y parvenir, sélectionnons une politique conjointe π et un historique initial h_0 (pouvant par exemple être un état $s \in S$). Puis simulons à chaque horizon τ l'action $d_\tau(h_\tau)$ ainsi qu'une observation $\omega_{\tau+1}$, de sorte que $(h_\tau, d_\tau(h_\tau), \omega_{\tau+1})$ constitue l'historique à l'horizon suivant (comme illustré à la Figure 5.3). Le processus se poursuit pour toute les politiques conjointes sélectionnées. Les ensembles d'historiques H_τ collectés sont alors stockés dans $\mathscr{H}_\tau$.

5.5.4 L'algorithme à base de croyances et à mémoire limitée

Dans cette section, nous présentons l'algorithme d'induction rétrograde à base de croyances et à mémoire limitée noté K-PBBI et présenté Algorithme 27.

La dimension des ensembles d'historiques H_τ est un véritable frein aux approches précédentes. Afin d'augmenter leur capacité à passer à l'échelle, nous préconisons de borner

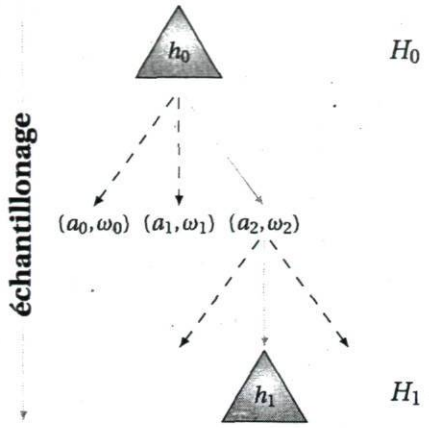


FIGURE 5.3 – Illustration des approches de mises à jour à base de trajectoires.

le nombre d'historiques à conserver dans les ensembles H_τ. Puis de ne planifier que sur ces historiques, en s'assurant néanmoins que la règle de décisions soit entièrement définie. Le choix des K historiques de l'ensemble H_τ à conserver est naturellement orienté par leur influence sur la fonction de valeurs. Nous choisissons de conserver les historiques $h \in H_\tau$ dont la probabilité $\mathbb{P}(h)$ est parmi les K plus élevés. Aux autres historiques il faudra assigner des actions garantissant la distributivité de la politique résultante.

Création de politiques déterministes et distributives heuristiques

1. Résoudre le problème de contrôle centralisé associé à celui du contrôle distribué.
2. Décomposer la politique déterministe π ainsi trouvée en plusieurs politiques individuelles pour chaque agent $i \in \mathscr{I}$.
3. Simuler l'ensemble des politiques distributives résultant du produit croisé des politiques individuelles des agents $\mathscr{I}$.

FIGURE 5.4 – Création de politiques déterministes et distributives heuristiques.

De même que pour l'algorithme précédent, nous pouvons établir une borne sur l'erreur produite par l'algorithme d'induction rétrograde à base de croyances et à mémoire limitée. Nous montrons que pour tout horizon $\tau = 0, 1, \cdots, N$, la fonction de valeurs définie sur les K plus probables historiques des ensembles $H_\tau \in \mathscr{H}_\tau$, et représentée par l'ensemble d'hyperplans $V^K_{\tau:N}$ est bornée. De même que dans le cas précédent, l'erreur est fonction du degré de densité de l'ensemble $\mathscr{H}_\tau$ mais aussi des K historiques sélectionnés pour chaque ensemble $H_\tau \in \mathscr{H}_\tau$. L'erreur ainsi produite, $\|V^K_{\tau:N} - V^*_{\tau:N}\|_\infty$, est bornée par :

$$\|V^K_{\tau:N} - V^*_{\tau:N}\|_\infty \leqslant \|V^K_{\tau:N} - V^{\mathscr{H}_\tau}_{\tau:N}\|_\infty + \|V^{\mathscr{H}_\tau}_{\tau:N} - V^*_{\tau:N}\|_\infty \tag{5.88}$$

Algorithme 27 Algorithme K-PBBI.

procedure K-PBBI
  Posons $\tau = N$ et K un entier.
  répéter
    Générons les ensembles $\mathscr{H}_\tau$ tel que $|H_\tau| \leqslant K^{|\mathscr{I}|}$, pour tout horizon τ.
    Posons $\tau \leftarrow (\tau - 1)$.
    Générons l'ensemble des règles de décisions $\prod_{i \in \mathscr{I}} |A^i|^K$.
    Supprimons les règles de décisions ε-dominées.
  jusqu'à $\tau = 0$
fin procedure

d'après l'inégalité du triangle. Le second terme est bornée par $N \cdot \zeta \cdot |r|_\infty$ d'après le Théorème 12. Le reste de cette sous-section établit et prouve la borne sur l'erreur du premier terme.

Théorème 13. *L'erreur introduite par l'algorithme d'induction rétrograde à base de croyances et à mémoire limitée pour tout horizon τ, en planifiant sur l'ensemble $\mathscr{H}_\tau$ plutôt que l'ensemble $\mathscr{H}_\tau^\infty$ est donné par :*

$$\eta_\tau \leqslant \left[\left(1 - \sum_{k=1}^{K} \mathbb{P}(h_k)\right) + \zeta_\tau\right] \cdot (N - \tau) \cdot |r|_\infty \tag{5.89}$$

où ζ_τ est la densité de l'ensemble $\mathscr{H}_\tau$; et h_k est un des K historiques sélectionnés pour chaque $H_\tau \in \mathscr{H}_\tau$, pour tout horizon $\tau = 0, 1, \cdots, N$.

Démonstration. La preuve de ce résultat est construite en considérant l'erreur , $(N - \tau) \cdot |r|_\infty$, en pire cas produite par la non prise en compte pour tout historique $h \in H_\tau$, pour tout horizon $\tau = 0, 1, \cdots, N$. En somme, l'erreur produite par la non utilisation de ces historiques est donnée par $\left(1 - \sum_{k=1}^{K} \mathbb{P}(h_k)\right) \cdot (N - \tau) \cdot |r|_\infty$.

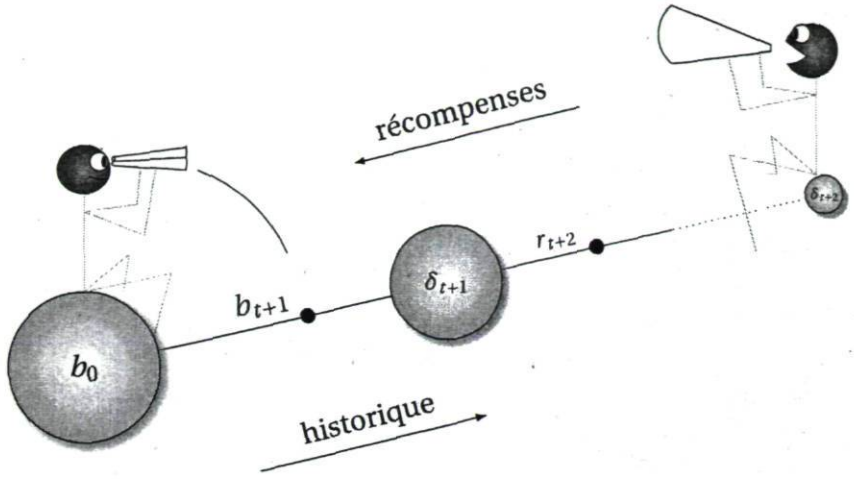
□

Théorème 14. *Pour tout ensemble $\mathscr{H} \equiv \{\mathscr{H}_\tau\}$, l'erreur produite par l'algorithme d'induction rétrograde à base de croyances et à mémoire limitée est donnée par :*

$$\varepsilon \leqslant \left[\rho + \zeta\right] \cdot N \cdot |r|_\infty \tag{5.90}$$

où $\zeta = \max_\tau \zeta_\tau$ et $\rho = \max_{\{h_k\}_k :\, H_\tau} \left(1 - \sum_{k=1}^{K} \mathbb{P}(h_k)\right)$.

La borne sur l'erreur produite par l'algorithme d'induction rétrograde à base de croyances et à mémoire limitée dépend de la densité de $\mathscr{H}_\tau$ mais aussi des K historiques sélectionnés



pour chaque ensemble $H_\tau \in \mathcal{H}_\tau$, pour tout horizon $\tau = 0, 1, \cdots, N$. Plus la somme des probabilités sur ces K historiques est proche de 1 plus l'erreur produite est proche de celle de l'algorithme d'induction rétrograde à base de croyances. Bien que sélectionner les K plus probables historiques semble à première vue une bonne heuristique, il faut néanmoins tenir compte de valeurs des différentes historiques. En effet, un historique de probabilité faible peut être associé à une valeur relativement élevée.

5.6 L'algorithme PBIP

Dans cette section, nous proposons l'algorithme induction rétrograde à base de croyances et à mémoire limitée, nommée « *point-based incremental pruning* » (PBIP). Cet algorithme vise à remplacer l'opérateur de mise à jour de tout algorithme de programmation dynamique communément nommé opérateur de « *génération exhaustive* ». En effet cet opérateur, permettant de générer l'ensemble des politiques $\mathcal{Q}_{\tau:N}$ à chaque horizon $\tau = 0, 1, \cdots, N$, requiert des efforts en temps et en mémoire. Nous proposons de le remplacer par un opérateur de recherche heuristique des K meilleures politiques individuelles par agent et horizon de planification.

L'algorithme PBIP est un algorithme itératif. C'est pour cette raison que nous limiterons notre exposé à un seul horizon $\tau = 0, 1, \cdots, N$ de planification. Pour chaque horizon τ, nous voulons sélectionner $K^{|\mathcal{S}|}$ politiques déterministes et distributives $\delta_{\tau:N} \in \mathcal{Q}_{\tau:N}$. En particulier, pour tout historique $h_\tau \in H_{0:\tau}$ et un ensemble de politique déterministes et distributives $\mathcal{Q}_{\tau+1:N}$, il s'agit de construire la politique déterministe et distributive sous forme d'un arbre de décisions $\delta_{\tau:N}$ dont la valeur est maximale pour l'historique h_τ. Un tel arbre de décisions est construit en sélectionnant d'une part une action conjointe $a(\delta_{\tau:N}) = a$ et d'autre par pour chaque observation une sous politique déterministe et distributive

$\delta_{\tau+1:N} \in \mathcal{Q}_{\tau+1:N}$ tel que $\eta(\delta_{\tau:N}, \omega) = \delta_{\tau+1:N}$. Cela de sorte que l'arbre de décisions ainsi construit soit distributif.

5.6.1 Espace de recherche

Hypothèses 4. *Tout au long de cette section, nous supposons que chacun des agents dispose d'un ensemble d'observations individuelles de même dimension $K = |o^i|$, pour tout agent $i \in \mathcal{I}$. Cette hypothèse peut être satisfaite en compensant les ensembles d'observations individuelles par des observations individuelles fictives et imperceptibles à l'exécution.*

La recherche d'une politique déterministe et distributive dont la qualité est la meilleure pour un historique prédéfini se fait par commodité dans l'espace des arbres de décisions conjoints correspondant chacun à un $|\mathcal{I}|$-tuple d'arbres de décisions individuelles, un pour chaque agent $i \in \mathcal{I}$. Compte tenu de la contrainte de distributivité de la politique recherchée, nous introduisons la propriété suivante des politiques déterministes et distributives. Ces propriétés permettent de simplifier la représentation des politiques distributive et par conséquent réduire l'espace de recherche et les efforts consacrés à son exploration.

Théorème 15. *Soit $\pi \in \tilde{\Pi}^{\mathbf{HD}}$ une politique déterministe et distributive. Il existe alors un $|\mathcal{I}|$-tuple $(\pi^1, \pi^2, \cdots, \pi^{|\mathcal{I}|})$ de politiques déterministes individuelles, une pour chaque agent $i \in \mathcal{I}$, telle que les contraintes $\mathcal{C}_1$ et $\mathcal{C}_2$ suivantes soient satisfaites :*

1. $\mathcal{C}_1 : \alpha(\pi) \equiv (\alpha(\pi^1), \alpha(\pi^2), \cdots, \alpha(\pi^{|\mathcal{I}|}))$
2. $\mathcal{C}_2 : \eta(\pi, h_\tau) \equiv (\eta(\pi^1, h_\tau^1), \cdots, \eta(\pi^{|\mathcal{I}|}, h_\tau^{|\mathcal{I}|}))$

où $h_\tau \equiv (h_\tau^1, h_\tau^2, \cdots, h_\tau^{|\mathcal{I}|})$, α est une application associant à tout politique π l'action en racine, et η une application qui associe à tout couple (π, h_τ) une sous politique.

Démonstration. La preuve est construite directement à partir de la définition d'une politique déterministe et distributive.

□

Cette propriété est en particulier vraie lorsque la politique déterministe et distributive est représentée sous forme d'un arbre de décisions $\delta \equiv (\delta^1, \delta^2, \cdots, \delta^{|\mathcal{I}|})$. Les contraintes $\mathcal{C}_1$ et $\mathcal{C}_2$ se déclinent alors comme suit :

1. $\mathcal{C}_1 : \alpha(\delta) \equiv (\alpha(\delta^1), \alpha(\delta^2), \cdots, \alpha(\delta^{|\mathcal{I}|}))$
2. $\mathcal{C}_2 : \eta(\delta, \omega) \equiv (\eta(\delta^1, \omega^1), \cdots, \eta(\delta^{|\mathcal{I}|}, \omega^{|\mathcal{I}|}))$

Ces contraintes permettent de construire de façon centralisée un arbre de décisions correspondant à une politique déterministe et distributive. Comme l'illustre la Figure 5.5, en construisant un arbre de décisions sur un sous-ensemble bien choisi des observations conjointes, il est possible de construire un $|\mathscr{I}|$-tuple d'arbres de décisions individuelles, un pour chaque agent $i \in \mathscr{I}$. En connaissance de ce $|\mathscr{I}|$-tuple d'arbres de décisions individuelles, on peut reconstruire l'arbre de décisions complet correspondant à une politique déterministe et distributive.

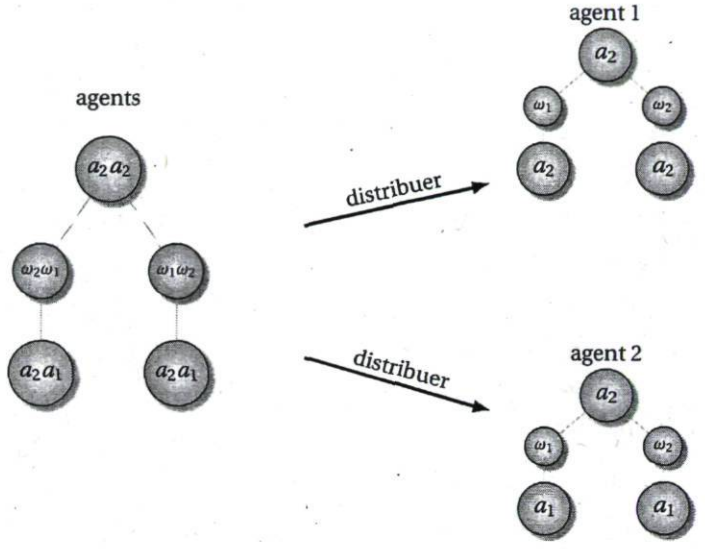


FIGURE 5.5 – Construction d'un arbre de décisions distributives de deux agents.

Afin d'énoncer les conditions pour lesquelles cela est possible, nous introduisons quelques définitions. Nous appelons *un ensemble d'observations conjointes de base*, tout sous-ensemble *minime* d'observations conjointes $\Omega^* \subset \Omega$, telles que chaque observation individuelle y est représentée au moins une fois. Considérons par exemple un problème de contrôle distribué d'un processus de Markov impliquant deux agents muni chacun d'un ensemble de deux observations individuelles $\omega^i \equiv \{\omega_1, \omega_2\}$, pour $i = 1,2$. L'ensemble des observations conjointes $\{\omega_2\omega_1, \omega_1\omega_2\}$ est un ensemble d'observations conjointes de base, car chacune des observations individuelles de chacun des agents y est représentée, de plus cette ensemble est de taille minime car la suppression d'une des observations conjointes rend caduque la satisfaction du premier critère.

Nous appelons *politique de base*, toute politique définie exclusivement sur des historiques $h_\tau = (a_0, \omega_1, a_1, \cdots, a_{\tau-1}, \omega_\tau)$ formées de séquences d'actions et d'observations pour lesquelles les observations $\omega_1, \omega_2, \cdots \omega_\tau$ appartiennent à un même ensemble d'observations conjointes de base $\dot{\Omega}$, pour tout horizon $\tau = 0, 1, \cdots, N$. Une fois de plus revenons sur l'exemple précédent, et considérons la politique de base illustrée à la Figure 5.5. Il apparaît alors que cette politique de base est suffisante pour construire une politique déterministe et distributive.

Théorème 16. *Toute politique déterministe de base π est une politique déterministe et distributive.*

Démonstration. Par construction une politique déterministe et distributive $\dot{\pi}$ équivaut un $|\mathscr{I}|$-tuple de politiques individuelles $(\pi^1, \pi^2, \cdots, \pi^{|\mathscr{I}|})$.

□

Ainsi à toute politique de base correspond une unique politique déterministe et distributive. Pour cette raison, l'algorithme PBIP focalise son attention à l'assignation d'une sous politique $\delta_{\tau+1:N}$ à chaque observation de base $\omega \in \dot{\Omega}$, une fois définie il est facile de déterminer les affectations de sous politiques aux observations $\omega' \in \Omega \backslash \dot{\Omega}$ dits hors base. C'est à ce niveau que les contraintes $\mathscr{C}_1$ et $\mathscr{C}_2$ sont utiles. En effet, connaissant une politique de base $\dot{\pi}$, nous pouvons extraire le $|\mathscr{I}|$-tuple de politiques individuelles $(\pi^1, \pi^2, \cdots, \pi^{|\mathscr{I}|})$ correspondant. Puis, d'après la contrainte $\mathscr{C}_2$, nous pouvons identifier les sous politiques assignées aux observations hors base $\omega' \in \Omega \backslash \dot{\Omega}$. Considérons encore notre exemple à deux agents illustrer à la Figure 5.5. Nous constatons que suite à l'assignation de sous politiques $(a_2 a_1)$ et $(a_2 a_1)$ respectivement aux observations de base $(\omega_2 \omega_1)$ et $(\omega_1 \omega_2)$, nous avons pu construire la paire de politiques individuelles à droite. Dès lors il nous est possible d'identifier les sous politiques à assigner aux observations hors base. Considérons par exemple l'observation $\omega_2 \omega_2$, d'après la contrainte $\mathscr{C}_2$, $\eta(\delta, \omega_2 \omega_2) = (\eta(\delta^1, \omega_2), \eta(\delta^2, \omega_2))$. Soit $\eta(\delta, \omega_2 \omega_2) = (a_2 a_1)$, en observant les politiques individuelles δ^1 et δ^2 des agents 1 et 2.

L'espace de recherche de l'algorithme d'élagation incrémentale à base de croyances est alors celui des politiques déterministe de base. Ces dernières seront représentées sous forme d'arbres de décisions. Cet espace de recherche est plus compacte que celui des politiques déterministes conjointes de dimension $|A|^{\frac{|\Omega|-1}{|\Omega|^{N+1}-1}}$, car sa dimension n'est que de $|A|^{\frac{|\dot{\Omega}|-1}{|\dot{\Omega}^{N+1}|-1}}$. Cet espace de recherche requiert bien moins d'efforts de calculs que l'espace des politiques déterministes et distributives de dimension $\prod_{i \in \mathscr{I}} |A^i|^{\frac{|\Omega^i|-1}{|\Omega^i|^{N+1}-1}}$. La différence réside dans la nécessité de vérifier la contrainte $\mathscr{C}_2$ autant de fois qu'il y a d'observations hors base. Malgré un réduction exponentielle de l'espace de recherche par rapport à l'espace des politiques déterministes, et de temps par rapport à l'espace des politiques déterministe et distributives, il ne reste pas moins vraie que cette espace est de dimension exponentielle. C'est dans ce contexte que s'inscrit notre algorithme de recherche heuristique.

5.6.2 Heuristiques d'exploration de l'espace de recherche

Cette section propose des stratégies d'exploration de l'espace de recherche constitué des politiques déterministes de base sous forme d'arbres de décisions. La première stratégie d'exploration de l'espace de recherche est de définir une estimation heuristique de la valeur d'une politique. Si cette valeur est optimiste par rapport à la valeur réelle de la politique recherchée, alors elle peut être utilisée afin de guider la recherche. La seconde stratégie d'exploration de l'espace de recherche est de définir un critère de différenciation de deux ensembles d'observations de base. En effet, il est fort probable que selon l'ensemble d'observations de base sélectionné les efforts de recherche requis soient différents. Enfin, en combinant les deux premières stratégies, la troisième stratégie consiste à concentrer le maximum d'efforts dans les régions des croyances qui sont susceptibles d'améliorer significativement la qualité de la politique en construction.

Estimations heuristiques

Afin de définir une estimation heuristique d'une politique sous forme d'un arbre de décisions, nous procédons comme suit : pour tout historique h_τ, nous sélectionnons une action conjointe $\alpha(\delta_{\tau:N}) = a_\tau$, et des sous politiques $\eta(\delta_{\tau:N}, \omega_\tau) = \delta_{\tau+1:N}$ une pour chaque observations de base $\omega_\tau \in \dot{\Omega}$, telle que :

1. la valeur exacte si la sous politique $\delta_{\tau+1:N} \in \mathcal{Q}_{\tau+1:N}$ est assignée à l'observation ω_τ est donnée par :

$$v^{\delta_{\tau+1:N}}(h_{\tau+1}) = \sum_{s' \in S} \mathbb{P}(s'|h_{\tau+1}) v^{\delta_{\tau+1:N}}(s') \tag{5.91}$$

   où $h_{\tau+1} = h_\tau a_\tau \omega_\tau$ et $\mathbb{P}(s|h_{\tau+1}) = \sum_{s \in S} \mathbb{P}(s|h_\tau) T(s'|s, a_\tau) O(\omega_\tau|s, a_\tau)$.

2. la valeur exacte d'une politique $\delta_{\tau:N}$ si nous avons connaissance de l'ensemble des assignations de sous politiques aux observations est donnée par :

$$v^{\delta_{\tau:N}}(h_\tau) = \sum_{s \in S} \mathbb{P}(s|h_\tau) R(s, a_\tau) + \sum_{\omega_\tau \in \Omega} v^{\eta(\delta_{\tau:N}, \omega_\tau)}(h_{\tau+1}) \tag{5.92}$$

3. la valeur optimiste d'une politique $\delta_{\tau:N}$ si seuls les observations ω_1 ont été associées à une sous politique chacune, et les autres observations ω_2 sont encore à assigner, est donnée par :

$$\hat{v}^{\delta_{\tau:N}}(h_\tau) = \underbrace{\sum_{s \in S} R(h_\tau, a_\tau) + \sum_{\omega_\tau \in \Omega_1} v^{\delta_{\tau+1:N}}(h_{\tau+1})}_{G(h_\tau, \delta_{\tau:N})} + \underbrace{\sum_{\omega_\tau \in \Omega_2} \hat{v}(h_{\tau+1})}_{H(h_\tau, \delta_{\tau:N})} \tag{5.93}$$

où $R(h_\tau, a_\tau) = \mathbb{P}(s|h_\tau)R(s, a_\tau)$, $h_{\tau+1} = (h_\tau, a_\tau, \omega_{\tau+1})$, $\delta_{\tau+1:N} = \eta(\delta_{\tau:N}, \omega)$, $\hat{v}(h_{\tau+1}) = \max_{\delta_{\tau+1:N}} v^{\delta_{\tau+1:N}}(h_{\tau+}$

Théorème 17. *Pour toute politique $\bar{\delta}_{\tau:N}$ obtenue à partir de la politique $\delta_{\tau:N}$ en y assignant des sous politiques aux observations non affectées, $\hat{v}^{\delta_{\tau:N}}(h_\tau) \geqslant \hat{v}^{\bar{\delta}_{\tau:N}}(h_\tau)$, pour tout historique h_τ et pour tout horizon $\tau = 0, 1, \cdots, N$.*

Démonstration. La preuve de ce théorème est triviale par construction de la fonction de valeurs $\hat{v}^{\delta_\tau}$. En effet, cette heuristique est basée sur la décomposition de la valeur d'une politique $\delta_{\tau:N}$ en deux estimés. Le premier estimé, $G(h_\tau, \delta_{\tau:N})$, est la valeur exacte issue de la récompense immédiate et des sous politiques sélectionnées en accord avec la contrainte $\mathscr{C}_2$. Le second estimé, $H(h_\tau, \delta_{\tau:N})$, est une borne supérieure sur la valeur exacte des sous politiques restant à assigner.

□

Ce théorème prouve que l'heuristique $\hat{v}$ est une heuristique admissible, et peut être ainsi utilisée afin de guider la recherche de la politique de valeur maximale pour un historique donné.

Sélection des observations de base

Comme nous l'avons énoncé d'entrée l'espace de recherche de l'algorithme d'élagation incrémentale de politiques à base de croyances est restreint à l'espace des politiques de base. Plus précisément, il s'agit en pire cas de l'ensemble $\mathscr{Q}_{\tau+1:N}^{|\dot{\Omega}|}$, pour tout horizon $\tau = 0, 1, \cdots, N-1$. Or à partir d'un historique h_τ, nous pouvons éliminer certaines politiques candidates à l'assignation pour chacune des observations de base $\omega_\tau \in \dot{\Omega}$. Soit $\mathscr{Q}_{\tau+1:N}^{\omega_\tau}$ l'ensemble des politiques non dominées pour une observation de base ω_τ, un historique h_τ et un ensemble $\mathscr{Q}_{\tau+1:N}$ donnés. Le meilleur ensemble d'observations de base correspond alors à un ensemble dont l'espace de recherche $\prod_{\omega_\tau \in \dot{\Omega}} \mathscr{Q}_{\tau+1:N}^{\omega_\tau}$ après élimination des politiques dominées est le plus petit, pour tout horizon $\tau = 0, 1, \cdots, N-1$.

L'idée est donc de remplacer l'espace de recherche en pire cas $\mathscr{Q}_{\tau+1:N}^{|\dot{\Omega}|}$ par un espace plus compact $\prod_{\omega_\tau \in \dot{\Omega}} \mathscr{Q}_{\tau+1:N}^{\omega_\tau}$, pour tout horizon $\tau = 0, 1, \cdots, N-1$. Si nous souhaitons construire la politique sous forme d'arbre de décisions dont l'action à la racine est $a \in A$, nous devons décider quelle sous politique $\delta_{\tau+1:N} \in \mathscr{Q}_{\tau+1:N}$ doit être exécutée après avoir observé une observation $\omega_\tau \in o$, pour tout horizon $\tau = 0, 1, \cdots, N-1$. Plutôt que de considérer toutes les possibilités d'assignations, nous maintenons uniquement les politiques de l'ensemble $\mathscr{Q}_{\tau+1:N}^{i}$ qui sont utiles pour tout historique h_τ, et les politiques des autres agents $\mathscr{Q}_{\tau+1:N}^{\neq i}$.

En d'autres termes, une politique $\delta^i_{\tau+1:N} \in \mathcal{Q}^i_{\tau+1:N}$ d'un agent $i \in \mathcal{I}$ est dite non dominée si la valeur de cette politique est plus élevée que celle de tout autre politique pour une distribution de probabilités sur les politiques des autres agents $\delta^{\neq i}_{\tau+1:N}$ et les états possibles et pour un historique h_τ donné par le programme Figure 5.6. Il s'agit d'un simple programme discret, de complexité $\mathcal{O}(|\mathcal{Q}||\Omega||S|)$ pour chaque politique individuelle à tester – soit une complexité globale en pire cas de $\mathcal{O}(|\mathcal{Q}^2||\Omega||S|)$.

Programme linéaire d'élimination de sous politiques dominées

1. Soit une action $a \in A$, un historique $h \in H$, une politique individuelle δ^i et une observation individuelle ω^i.
2. Posons $\omega = \omega^i\omega^{\neq i}$, $\bar{\delta} = \bar{\delta}^i\delta^{\neq i}$ et $\delta = \delta^i\delta^{\neq i}$.
3. Maximiser ξ, sachant : $\forall \bar{\delta}^i \in \mathcal{Q}^i \backslash \{\delta^i\}$, $\forall \omega^{\neq i} \in \omega^{\neq i}$,

$$\left(\sum_{\delta^{\neq i}} \zeta(\delta^{\neq i}) \cdot v^{\bar{\delta}}(\bar{h})\right) + \xi \leqslant \left(\sum_{\delta^{\neq i}} \zeta(\delta^{\neq i}) \cdot v^{\delta}(\bar{h})\right) \tag{5.94}$$

   où $\sum_{\delta^{\neq i}} \zeta(\delta^{\neq i}) = 1$, $\bar{h} = ha\omega$ et $\zeta(\delta^{\neq i}) \geqslant 0$.
4. Si la variable ξ est négative ($\xi \leqslant 0$) alors supprimer δ^i de $\mathcal{Q}^i$.

FIGURE 5.6 – Programme linéaire d'élimination de sous politiques dominées.

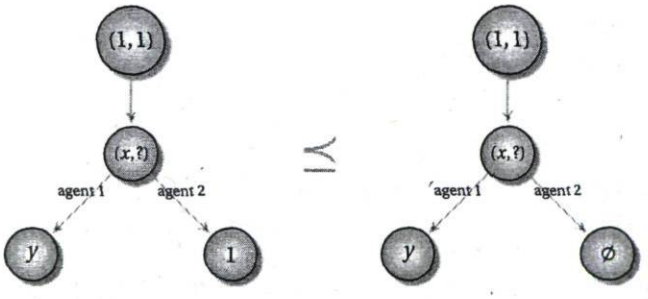


FIGURE 5.7 – Illustration du critère de domination de sous arbres de décisions.

Considérons par exemple un problème de contrôle distribué des processus de Markov $(S, A^1, A^2, T, R, \Omega^1, \Omega^2, O)$ défini comme suit : $S = \{0,1\}$, $A^1 = A^2 = \{0,1,?\}$ et $\Omega^1 = \Omega^2 = \{0,1,?\}$. Les ensembles des politiques individuelles à horizon $\tau = N$ sont restreints aux ensembles d'actions individuelles. Si nous souhaitons construire une politique conjointe sous forme d'arbre de décisions avec pour action initiale $(1,1)$, nous devons assigner une action conjointe pour chacune des observations conjointes. Seulement compte tenu du nombre exponentiel de possibilités d'assignation, nous voulons savoir à priori quelles politiques de l'horizon $\tau = N$ est inutile pour notre arbre de décisions en construction. De

plus, nous disposons d'une croyance, c'est à dire un historique h_0. Dès lors nous pouvons utiliser ces données afin d'appliquer la méthode ci-dessus d'élimination des politiques dominées. En particulier, nous aimerons savoir si la politique individuelle « 1 » domine la politique individuelle « ? » pour l'observation individuelle « ? » de l'agent 2 – voir Figure 5.7. Les variables x et y sont paramétriques et peuvent prendre n'importe quels valeurs dans $\{0,1,?\}$. La Figure 5.7 traduit le fait que si pour tout historique individuel $1xy$ de l'agent 1, les récompenses cumulées pour l'historique conjoint $(1xy, 1?1)$ sont supérieures à celles cumulées pour l'historique conjoint $(1xy, 1??)$, alors la politique individuelle 1 est dominée par la politique individuelle « ? » pour l'action conjointe $(1,1)$ et l'observation individuelle « ? » de l'agent 2.

Nous démontrons maintenant que l'algorithme d'élimination des politiques individuelles $\delta^i_{\tau+1:N}$ n'élimine pas des politiques individuelles qui aurait contribué dans la construction de la politique conjointe $\delta_{\tau:N}$ maximale pour un historique donné.

Lemme 2. *Tout politique individuelle $\delta^i_{\tau+1:N}$ supprimée par l'algorithme d'élimination de politiques individuelles dominées est une sous politique individuelle d'une politique conjointe $\delta_{\tau:N}$ dominée pour un historique h_τ donné.*

Démonstration. Nous montrons en particulier que si la politique individuelle δ^i de l'agent $i \in \mathscr{I}$ a une sous politique non maintenue par l'algorithme d'élimination de politiques individuelles dominées, alors δ^i doit être dominée par une distribution de probabilités sur les politiques individuelles non dominées $\hat{\delta}^i$. Cela est déterminé par l'équation suivante :

$$v^{\delta}(h) \leqslant \sum_{\hat{\delta}^i} \zeta(\hat{\delta}^i) \cdot v^{\hat{\delta}}(h), \quad \forall \delta^{\neq i} \tag{5.95}$$

où $\hat{\delta} = (\hat{\delta}^i, \delta^{\neq i})$ et $\delta = (\delta^i, \delta^{\neq i})$.

Considérons maintenant l'ensemble des politiques individuelles de l'agent $i \in \mathscr{I}$ commençant par l'action conjointe a. Si l'agent perçoit par la suite l'observation individuelle o^i_j et choisit la politique individuelle δ^i_j n'appartenant pas à l'ensemble des sous politiques utiles , alors nous pouvons montrer que δ^i doit être dominé par un ensemble de politiques identiques à la politique individuelle δ^i sauf qu'au lieu de choisir δ^i_j, une distribution de probabilités sur les politiques individuelles dans l'ensemble $\mathscr{Q}^i$ est choisie.

Comme la sous politique individuelle $\delta^i_j = \eta(\delta^i, \omega^i_j)$ n'a pas été maintenue dans l'ensemble $\mathscr{Q}^i_{a^i,\omega^i_j}$, nous savons ainsi qu'il existe une distribution de probabilités sur des sous arbres qui la domine pour toute action conjointe a et observation individuelle ω^i_j :

$$v^{\delta_j}(ha\omega_j) \leqslant \sum_{\hat{\delta}^i_j} \zeta(\hat{\delta}^i_j) v^{\hat{\delta}_j}(ha\omega_j), \quad \forall \delta^{\neq i}_j \tag{5.96}$$

où $\delta_j = \delta^i_j \delta^{\neq i}$ et $\hat{\delta}_j = \hat{\delta}^i_j \delta^{\neq i}$ de même $\omega_j = \omega^i_j \omega^{\neq i}$.

Sachant cette distribution $\zeta(\hat{\delta}^i_j)$ nous pouvons calculer une distribution de probabilités sur les sous politiques qui permettent de choisir les politiques de $\hat{\delta}^i_j$ lorsque l'observation individuelle ω^i_j est perçue, le cas échéant choisir les mêmes sous politiques que celle inclus dans δ^i. La valeur de la politique ainsi construite est donnée par :

$$v^{\delta_j}(ha\omega_j) \leqslant \sum_{\hat{\delta}^i_j} \zeta(\hat{\delta}^i_j) v^{\hat{\delta}_j}(ha\omega_j), \quad \forall \delta^{\neq i}_j \tag{5.97}$$

□

Exploitation du « *poids* » des croyances

Rappelons que nous voulons construire pour toute croyance h, une politique à horizon τ sous forme d'un arbre de décisions, dont l'action en racine est a et les sous arbres correspondent à des assignations à chacune des observations $\omega \in \Omega$ d'arbres de décisions δ construits pour l'horizon $(\tau + 1)$. Nous avons vu que cela pouvait être accomplit en assignant d'abord une sous politique à chacune des observations de base $\omega \in \dot{\Omega}$. La question est donc de savoir comment sélectionner au mieux ces observations de base. Dans ce sous section, nous nous intéressons en particulier aux poids de croyances comme autre critère de sélection des observations de base.

Bien que cela ne soit pas systématique, il arrive qu'une observation ω soit imperceptible à la suite d'une croyance h donnée, c'est à dire $\mathbb{P}(\omega|ha) = 0$. Dans ce cas précis l'observation doit être exclue des observations de base potentielles. Il arrive plus généralement que les récompenses cumulées à la suite d'une croyance suivi d'une observation donnée soit bornées par un petit réel positif ε, en d'autres termes,

$$\max_{\delta} \left| \sum_{s} \mathbb{P}(s|ha\omega) v^{\delta}(s) \right| \leqslant \varepsilon \tag{5.98}$$

Dans ce cas, en supprimant l'observation ω des observations de base potentielles, la qualité de la politique construite à partir des observations de base excluant ω est préservée à ε près. Cela signifie que n'importe quelle politique δ à horizon $(\tau + 1)$ peut être assigné à ces observations sans détériorer significativement la qualité de politique à horizon τ, tant que δ satisfait la contrainte $\mathscr{C}_2$. Nous appelons *critère d'élimination des observations de base potentielles* le critère symbolisé par l'équation 5.98 et l'ensemble des observations de base potentielles est noté $\hat{\dot{\Omega}}$.

Comme nous souhaitons déterminer l'espace de recherche le plus compacte possible, il nous faut sélectionner les observations de base $\omega \in \dot{\Omega}$ telles que la dimension $\prod_{\omega \in \dot{\Omega}} |\mathcal{Q}_{a\omega}|$ soit la plus petite possible. Ce problème est un simple problème combinatoire, et en ce sens il peut être résolu via un simple algorithme de recherche heuristique de type A* ou de façon équivalente l'algorithme de séparation et évaluation. La solution recherchée est bien évidemment un ensemble d'observations $\{\omega_1, \omega_2, \cdots, \omega_{|\dot{\Omega}|}\}$ constituant l'ensemble des observations de base. Plus formellement, il s'agit d'assigner à $|\dot{\Omega}|$ variables $v \in \mathcal{V}$ une valeur parmi l'ensemble des observations de base potentielles $\mathcal{D}_v \equiv \hat{\dot{\Omega}}$. L'idée de l'algorithme est de contraindre progressivement les domaines $\mathcal{D}_v$ des variables v de sorte qu'à terme il ne reste qu'une seule valeur par domaine, celle d'une observation de base. Ainsi la valeur heuristique d'une solution partielle $(v_1, v_2, \cdots, v_{|\dot{\Omega}|})$ est donnée par :

$$\underline{v}(v_1, v_2, \cdots, v_{|\dot{\Omega}|}) = \prod_{v} \min_{o \in D_v} |\mathcal{Q}_{a\omega}| \tag{5.99}$$

Afin de contraindre les domaines $\mathcal{D}_v$ associés aux variables v, de restreindre tout ou partie des variables v à un domaine ne contenant qu'une seule observation $D_v = \{\omega\}$ – on dira alors que v est assigné. En connaissance des variables assignées, il est possible de contraindre les domaines des autres variables en vertu de la contrainte $\mathcal{C}_2$. Le lecteur notera que lorsque l'ensemble des domaines $\mathcal{D}_v$ est restreint à une seule valeur, $\underline{v}(v_1, v_2, \cdots, v_{|\dot{\Omega}|}) = v(v_1, v_2, \cdots, v_{|\dot{\Omega}|})$. Le problème peut se formuler comme un problème d'optimisation sous la contrainte $(\mathcal{V}, \mathcal{C}, \mathcal{D}, \underline{v})$ définit comme suit :

1. $\mathcal{V} \equiv \{v_1, v_2, \cdots, v_{|\dot{\Omega}|}\}$ correspond à l'ensemble des variables, chacune représentant une observation de base.
2. $\mathcal{C} \equiv \mathcal{C}_2$ est la principale contrainte du problème, il s'agit d'une contrainte sur l'ensemble des variables $\mathcal{V}$.
3. $\mathcal{D} \equiv \{\mathcal{D}_v\}_{v \in \mathcal{V}}$ définit l'ensemble des domaines $\mathcal{D}_v \equiv \hat{\dot{\Omega}}$, incluant l'ensemble des observations de base potentielles, des variables $v \in \mathcal{V}$.
4. $\underline{v}$ est l'heuristique d'évaluation admissible des solutions candidates du problème.

De nombreux algorithmes de la littérature des COP s permettent de résoudre efficacement ce type de problèmes. La solution d'un tel algorithme correspond aux observations de base recherchées. Malheureusement, résoudre un COP est NP-difficile en générale or la sélection des observations de base n'est qu'une sous routine parmi d'autres. Nous pouvons réduire considérablement le temps consacré à cette tâche en utilisant un algorithme de recherche gloutonne. En particulier, nous pouvons affecter une variable v après l'autre en leur assignant l'observation ω dont l'espace des politiques $|\mathcal{Q}_{a\omega}|$ est le plus petit, tout en satisfaisant la contrainte $\mathcal{C}_2$. Nous pourrions ainsi construire en temps linéaire un ensemble d'observations de base de dimension réduite sans nécessairement être minime.

5.6.3 Application du branch-and-bound

Le problème de construction d'un arbre de décisions à horizon τ, sachant un ensemble d'arbres de décision $\mathscr{Q}$ à horizon $(\tau+1)$, une action a et un historique h, équivaut à un problème d'optimisation sous contraintes similaire à celui définit précédemment. Soit $\mathscr{M} \equiv (\mathscr{V}, \mathscr{C}, \mathscr{D}, \hat{v})$ un COP définit comme suit :

1. $\mathscr{V} \equiv \{v_1, v_2, \cdots, v_{|\dot{\Omega}|}\}$ correspond à l'ensemble des variables, une variable pour chaque observation de base.
2. $\mathscr{C} \equiv \mathscr{C}_2$ est la principale contrainte du problème, il s'agit d'une contrainte sur l'ensemble des variables $\mathscr{V}$ garantissant la distributivité de l'arbre à construire.
3. $\mathscr{D} \equiv \{\mathscr{D}_v\}_{v \in \mathscr{V}}$ définit l'ensemble des domaines $\mathscr{D}_v \equiv \mathscr{Q}_{av}$, incluant l'ensemble arbres de décisions non dominées pour l'observation o associée à la variable v.
4. $\hat{v}$ est l'heuristique d'évaluation admissible des solutions candidates du problème.

En assignant un arbre de décisions à chacune des observations de base représentées par les variables $v \in \mathscr{V}$, nous définissons une politique de base. En connaissance d'une politique de base et des observations de base, la contrainte $\mathscr{C}_2$ nous permet de construire la politique correspondante. L'heuristique $\hat{v}$ évalue une solution partielle ou complète de façon progressive, orientant ainsi la recherche dans un sens ou un autre. Une fois de plus il existe de nombreux algorithmes efficaces de résolution de ce type de problèmes dans la littérature des COPs.

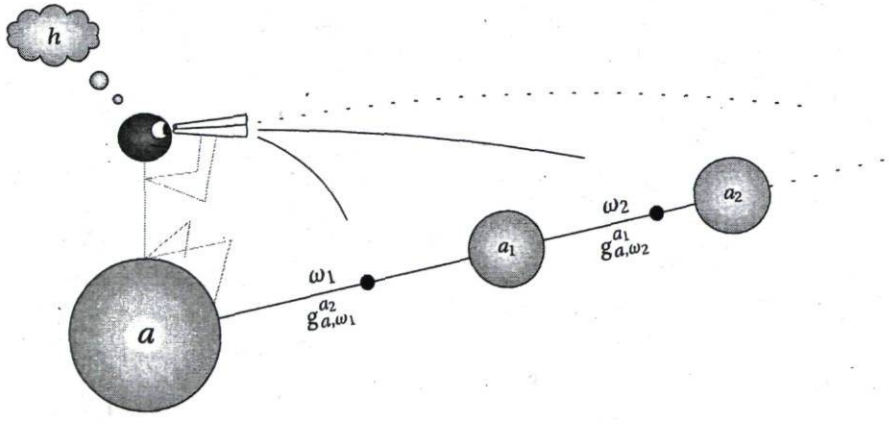


FIGURE 5.8 – Construction progressive d'une politique conjointe et distributive.

5.6.4 Illustration sur un exemple jouet

Considérons l'exemple jouet illustrant la construction d'un arbre de décisions maximal pour un historique donné. En particulier, nous considérons connu les arbres de décisions à horizon 0 correspondants à l'ensemble des actions conjointes $A = \{a_1 a_1, a_2 a_1, a_1 a_2, a_2 a_2\}$.

Algorithme 28 Algorithme PBIP.

procedure PBIP
  Initialiser $\tau = N$, $Selection_N \leftarrow A$.
  répéter
    Poser $\tau \leftarrow (\tau - 1)$.
    Poser $\mathcal{Q}_\tau \leftarrow Selection_\tau$
    Poser $Selection_{\tau-1} \leftarrow \emptyset$.
    Construire les arbres de décisions $Selection_{\tau-1}$ pour K historiques.
  jusqu'à $\tau = 1$
  Retourner $Selection_0$.
fin procedure

Le tableau Figure 5.9, illustre les valeurs exactes si à l'issue de l'exécution de l'action conjointe (a_2a_2 les agents venaient à percevoir conjointe une des observations (en ordonnée) et exécuter une des politiques (en abscisse). Par exemple si les agents perçoivent conjointement l'observation $\omega_2\omega_2$ et exécutent l'action conjointe a_2a_2 les récompenses cumulées seront de -1.00.

La Figure 5.10 représente l'état initial de l'algorithme, avec d'une part la politique partielle de base (dans le cercle orange) et une politique individuelle non contrainte permettant de visualiser l'impact des assignation de sous arbres de décisions aux observations de base ; et d'autre part son évaluation branche par branche. Initialement, c'est à dire avant tout assignation de sous arbre de décisions à une quelconque observation de base, la valeur optimiste de l'arbre de décisions maximale est de -3.48. Il est également à noter que les observations de base choisie sont $\{\omega_1\omega_1, \omega_2\omega_2\}$.

La Figure 5.11 illustre la seconde étape de l'algorithme PBIP, elle correspond à l'assignation du sous politique a_2a_2 à l'observation de base $\omega_2\omega_2$. Cette assignation contraint les choix possible de chaque politique individuelle de chaque agent. En particulier, nous pouvons constater que l'arbre de décisions d'un des agents est plus contraint pour l'observation ω_2. En outre, contenu de la propagation des effets de la contrainte $\mathscr{C}_2$ sur les observations hors base, l'heuristique $\hat{v}$ pour la politique courante est revue à la baisse, en effet les valeurs pour l'observation $\omega_1\omega_2$ est revue à la baisse. Au final, la politique partielle courante est de valeur -4.0.

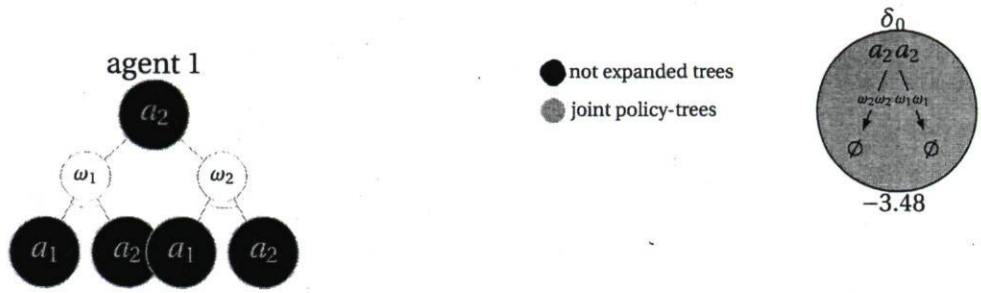
La Figure 5.12 illustre la troisième étape de l'algorithme PBIP appliqué à notre exemple jouet. Durant cette étape, l'algorithme assigne à la seconde observation de base $\omega_1\omega_1$ la sous politique a_2a_2. Cette assignation a pour effet de contraindre un peu plus la politique individuelle d'un des agents, elle est complètement définie. Noter que la définition complète de la politique conjointe de base coïncide avec celle des politiques individuelles

	a_1a_1	a_2a_1	a_1a_2	a_2a_2
$\omega_1\omega_1$	−1.40	−1.00	−6.02	−3.94
$\omega_2\omega_1$	−3.94	−0.60	−4.30	−6.02
$\omega_1\omega_2$	−5.16	−0.88	−11.3	−1.40
$\omega_2\omega_2$	−2.75	−7.48	−1.40	−1.00

FIGURE 5.9 – Tableau des estimations des sous politiques pour une politique dont l'action en racine est (a_2a_2).

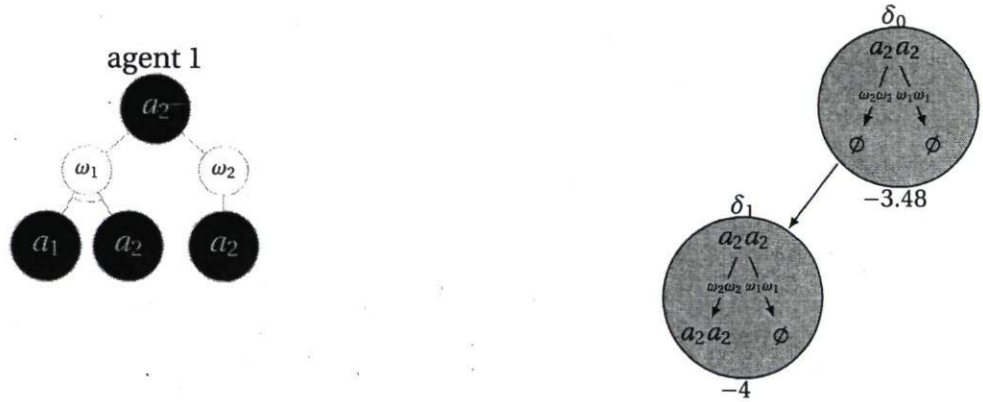
des agents. Cette contrainte additionnelle ne réduit pas la valeur de l'heuristique $\hat{v}$ qui se maintient à −4.0. Comme la politique de base est complètement défini, et par conséquent la politique conjointe l'est aussi, l'estimation heuristique est en fait la valeur réelle de cette politique conjointe. Cette politique conjointe ainsi que sa valeur vont désormais servir de témoin pour toutes les autres politiques construites par la suite.

La Figure 5.13 illustre la quatrième étape de l'algorithme PBIP dans le processus de résolution de notre exemple jouet. Comme l'étape précédente illustrait la définition complète d'une politique de base. L'algorithme tente désormais de vérifier si la dernière assignation ne peut être améliorée en changeant la sous politique assignée. Or comme la valeur heuristique $\hat{v}$ est la même avant et après l'assignation de la sous politique a_2a_2 à l'observation de base $\omega_1\omega_1$, quelque soit la sous politique remplaçant a_2a_2, la valeur ne pourrait être plus élevée que celle de la meilleure politique conjointe courante.



$g^{a_2 a_1}_{a_2 a_2, \omega_1 \omega_1}$	−1.00
$g^{a_2 a_1}_{a_2 a_2, \omega_2 \omega_1}$	−0.60
$g^{a_2 a_1}_{a_2 a_2, \omega_1 \omega_2}$	−0.88
$g^{a_2 a_2}_{a_2 a_2, \omega_2 \omega_2}$	−1.00
=	−3.48

FIGURE 5.10 – Illustration de l'algorithme PBIP : étape 1.



$g^{a_2 a_1}_{a_2 a_2, \omega_1 \omega_1}$	−1.00
$g^{a_2 a_1}_{a_2 a_2, \omega_2 \omega_1}$	−0.60
$g^{a_2 a_2}_{a_2 a_2, \omega_1 \omega_2}$	−1.40
$g^{a_2 a_2}_{a_2 a_2, \omega_2 \omega_2}$	−1.00
=	−4.00

FIGURE 5.11 – Illustration de l'algorithme PBIP : étape 2.

La Figure 5.14 illustre la cinquième étape de l'algorithme PBIP dans le processus de résolution de notre exemple jouet. Une fois de plus, l'algorithme procède à un « *backtrack* » vers la politique de base où aucune des observations de base n'est assignée. Puis il tente

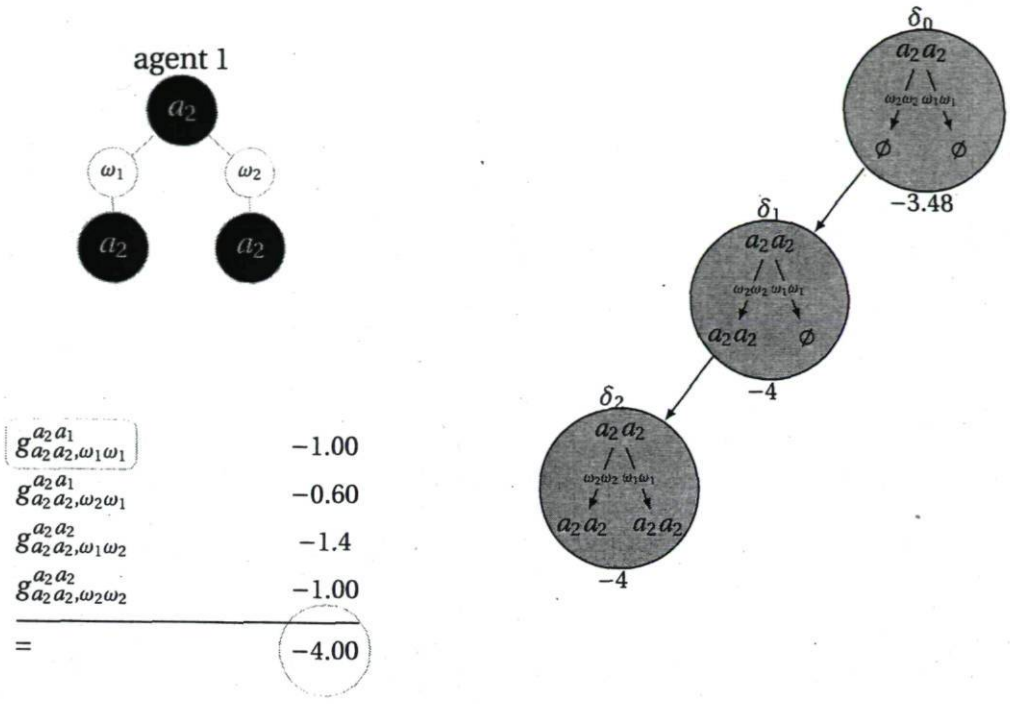


FIGURE 5.12 – Illustration de l'algorithme PBIP : étape 3.

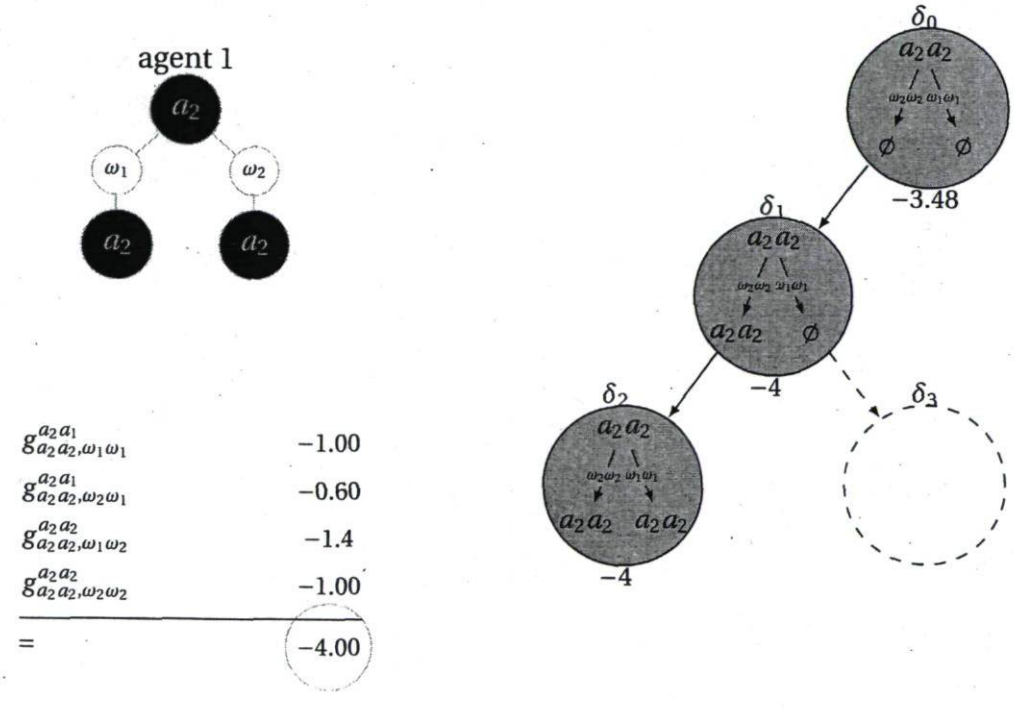
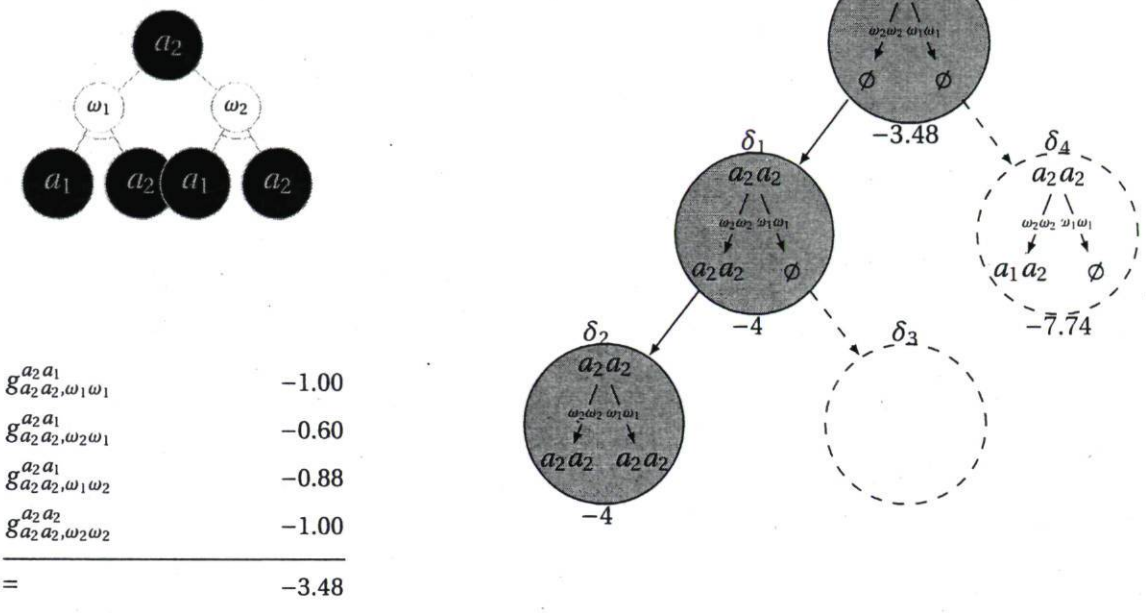


FIGURE 5.13 – Illustration de l'algorithme PBIP : étape 4.

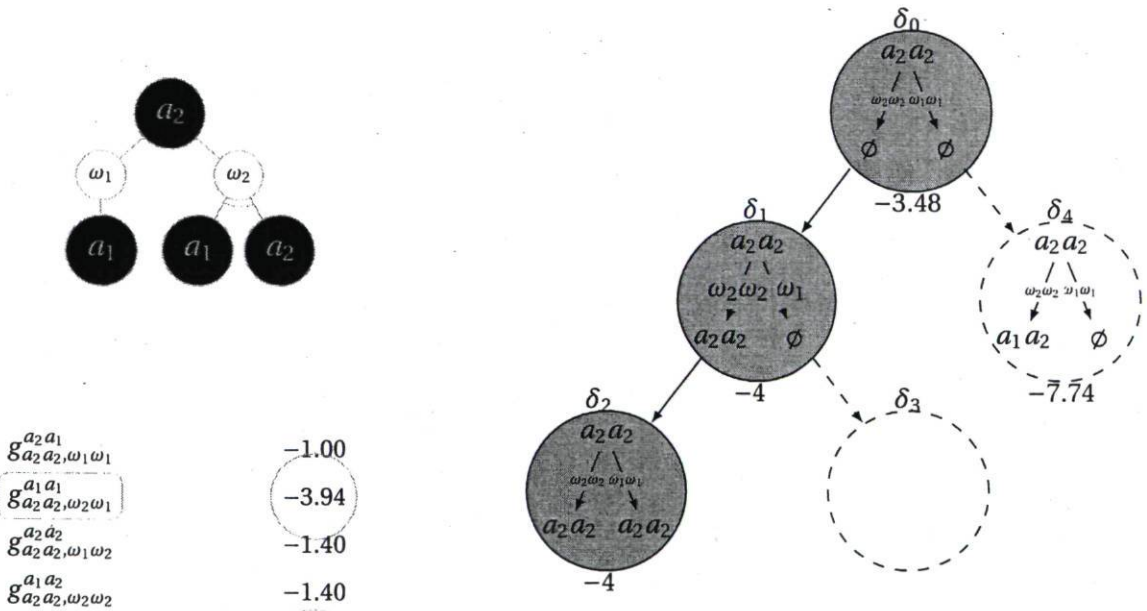
d'y assigner une sous politique (la meilleure) afin de vérifier s'il est possible d'améliorer la politique conjointe courant. Il s'avère que la meilleure valeur possible serait −7.74 soit une



$$
\begin{array}{ll}
g^{a_2a_1}_{a_2a_2,\omega_1\omega_1} & -1.00 \\
g^{a_2a_1}_{a_2a_2,\omega_2\omega_1} & -0.60 \\
g^{a_2a_1}_{a_2a_2,\omega_1\omega_2} & -0.88 \\
g^{a_2a_2}_{a_2a_2,\omega_2\omega_2} & -1.00 \\
\hline
= & -3.48
\end{array}
$$

FIGURE 5.14 – Illustration de l'algorithme PBIP : étape 5.

valeur inférieure à −4.0. Il vient alors que le développement de ce nœud est interrompu.



$$
\begin{array}{ll}
g^{a_2a_1}_{a_2a_2,\omega_1\omega_1} & -1.00 \\
g^{a_1a_1}_{a_2a_2,\omega_2\omega_1} & -3.94 \\
g^{a_2a_2}_{a_2a_2,\omega_1\omega_2} & -1.40 \\
g^{a_1a_2}_{a_2a_2,\omega_2\omega_2} & -1.40 \\
\hline
= & -7.74
\end{array}
$$

FIGURE 5.15 – Illustration de l'algorithme PBIP : étape 6.

La Figure 5.15 illustre la dernière étape de l'algorithme PBIP. Comme l'ensemble des al-

ternatives possibles a été exploré en vain, l'algorithme PBIP retourne la politique conjointe courante.

5.6.5 Propriétés théoriques

Dans cette section, nous introduisons quelques propriétés théoriques garantissant les performances de l'algorithme PBIP, y compris la complétude, l'*optimalité*, et la complexité.

Théorème 18. *La recherche heuristique de l'algorithme PBIP est complète.*

Démonstration. L'algorithme PBIP terminera éventuellement dans le pire des cas après énumération de toutes les arbres de décisions possibles. C'est à dire après avoir visité l'espace complet des politiques distributives. La meilleure solution pour un historique correspondra à celle dont la valeur exacte est la plus élevée.

□

Théorème 19. *L'algorithme PBIP retourne la politique maximale pour un historique et les sous politiques donnés.*

Démonstration. Il est possible de voir l'algorithme PBIP comme un algorithme de type A*. En effet, il visite les politiques complètes dans l'ordre décroissant de leurs valeurs heuristiques $\hat{v}$. Puis, si l'algorithme PBIP termine et retourne une politique conjointe δ, la propriété de convergence de A* et l'admissibilité de l'heuristique $\hat{v}$ garantissent l'optimalité de la solution sachant l'historique h et les sous politiques en entrées.

□

Notons que dans le pire des cas, l'algorithme PBIP reste exponentiel en temps suivant le nombre d'agents et la dimension $|\dot{\Omega}|$ de l'ensemble des observations de base. Dans le meilleur des cas, cependant, sa complexité est de $\mathcal{O}(|\dot{\Omega}|)$. La preuve de ce résultat est liée aux observations suivantes :

Premièrement, comme l'algorithme PBIP énumère éventuellement l'ensemble des politiques possibles, dans le pire des cas sa complexité en temps est exponentielle ; mais dans le meilleur des cas, l'algorithme PBIP ne requiert que $|\dot{\Omega}|$ étapes afin de construire une politique conjointe complète. Néanmoins, le temps total nécessaire peut être influencer par les étapes de calculs préliminaires qui requièrent en somme $\mathcal{O}(|S^2||A||\Omega|K^{|\mathcal{I}|})$ opérations.

Bien que l'algorithme PBIP soit basé sur l'algorithme de recherche heuristique A*, il ne souffre pas des mêmes limites en mémoire. En effet, la sous-routine de construction des politiques maximales, une pour les K historiques choisis à chaque horizon, ne requiert qu'une complexité linéaire $\mathcal{O}(|\dot{\Omega}|)$, c'est à dire la longueur de chemin de la racine de l'arbre de décisions à l'assignation d'une sous politique à la dernière observation de base. Cela est possible car l'ensemble des valeurs heuristiques des sous politiques est pré-calculé à l'avance. Finalement, la complexité spatiale de l'algorithme PBIP est de $\mathcal{O}(|A||\Omega|K)$.

5.7 Expérimentations

Dans cette section, nous proposons une série de tests expérimentaux visant à mesurer les performances des différentes versions de l'algorithme PBIP. En effet, nous utiliserons les acronymes PBIP et PBIP-IPG afin de désigner deux implémentations différentes de l'algorithme PBIP. L'acronyme PBIP désignera l'algorithme PBIP sans aucune optimisations : l'ensemble des observations de base est quelconque ; les ensembles de sous politiques pour chaque observations correspondent chacune à l'ensemble des politiques à l'horizon suivant sans élagation aucune. Tandis que l'acronyme PBIP-IPG désigne l'algorithme PBIP pour lequel il y a eu élagation de sous politiques dominées à priori pour chacune des observations. Le lecteur notera que de nombreuses optimisations restent à tester, par exemple celle incluant la sélection du meilleur ensemble d'observations de base, qui lui aussi est un problème difficile.

Afin de mesurer les performances de nos algorithmes, nous les avons comparés aux meilleurs algorithmes approximatifs à ce jour. En particulier, l'algorithme MBDP et ses extensions IMBDP et MBDP-OC ayant montré les meilleures performances sur l'ensemble des exemples issues de la littérature des DEC-POMDPs. Ces exemples de la littérature des DEC-POMDPs incluent : L'ensemble des algorithmes a été testé suivant les mêmes paramètres de contrôle de sélection des heuristiques du portefeuille, à savoir K représentant le nombre d'historiques par horizon ; L représentant le nombre maximum d'observations sur lesquelles la planification « *exacte* » est effectué. Nous rappelons que seul les algorithmes IMBDP et MBDP-OC font usage du paramètre L.

Le Tableau 5.1, les Figures 5.16, 5.17, 5.18 et 5.19 présentent les résultats expérimentaux pour PBIP. Parmi les critères de performances retenus, nous comptons :

1. La *valeur escomptée moyenne* (VEM) – afin d'atténuer la nature aléatoire l'ensemble des algorithmes testés car selon les K historiques sélectionnés la valeur de la solution retournée peut être sensiblement différente ;
2. Le *temps d'exécution de l'algorithme* (CPU) en secondes est reporté. Comme l'en-

semble des algorithmes est codé en JAVA, proprement optimisé et exécuté sur une même machine, cette mesure a du sens.

3. Le *nombre de nœuds* en pourcentage $100 \times \frac{N_{\text{PBIP}}}{N_{\infty}}$ développé par l'algorithme PBIP, il s'agit là d'une mesure de la performance de la sous-routine de recherche heuristique.

Algorithm	MBDP		IMBDP		MBDP-OC		PBIP		
N	AEV	CPU	AEV	CPU	AEV	CPU	AEV	CPU	%
MABC problem $maxTrees = 3$									
100	90.29	0.190	N.A.		N.A.		90.29	0.060	39
1000	900.29	2.270	N.A.		N.A.		900.29	0.940	38
10000	9000.29	51.93	N.A.		N.A.		9000.29	38.48	38
MA-TIGER problem $maxTrees = 20$									
10	12.5	67.1	N.A.		N.A.		13.6	5.55	7.83
20	25.8	159	N.A.		N.A.		26.8	15.1	12.4
50	73.9	438	N.A.		N.A.		74.2	45.3	15.3
100	149	901	N.A.		N.A.		147	94.2	12.9
COOPERATIVE BOX-PUSHING problem $maxTrees = 3$ $maxObs = 3$									
10	N.A.		99.59	7994.6	89.94	7994.4	102.5	11.8	10.9
20	N.A.		102.6	17031	135.0	17194	198.0	25.0	10.3
50	N.A.		82.99	44708	272.70	44210	422.2	61.6	9.07
100	N.A.		73.88	89471	440.08	90914	786.4	113	9.62

AEV = valeur escomptée moyenne N.A. = not applicable % = pourcentage of expanded nodes

TABLE 5.1 – Performances de PBIP.

5.7.1 Discussion

Le Tableau 5.1 présente les performances de l'algorithme PBIP sur l'ensemble des trois exemples choisis. Comme nous l'avons signalé les algorithmes PBIP et MBDP retournent des solutions identiques s'ils disposent des entrées identiques, car ils ne diffèrent qu'à travers la manière qu'ils ont de construire la solution maximale pour chaque historique et chaque horizon. Les résultats montrent clairement que l'algorithme PBIP surpasse en temps d'exécution les algorithmes MBDP, IMBDP ou encore MBDP-OC et parvient même à résoudre des instances de dimensions supérieures à celles solvables par MBDP.

Pour les instances disposant de très petit nombre d'observations conjointes, les algorithmes PBIP et MBDP retournent tous deux la même solution approximative (pour des entrées identiques), cependant l'algorithme PBIP est toujours bien plus rapide que l'algorithme MBDP. Par exemple, la résolution du problème MABC à horizon 1000 est résolution en moyenne en 560 secondes par l'algorithme PBIP tandis que l'algorithme MBDP requiert en moyenne 1094 secondes, soit le double du temps requis par l'algorithme PBIP.

Les performances de l'algorithme PBIP les plus impressionnantes par rapport à l'algorithme MBDP concernent celles du pourcentage de nœuds développés. Pour la résolution du problème MA-TIGER à horizon 10, l'algorithme PBIP développe seulement 7.83% du nombre de nœuds (solutions candidates) que l'algorithme MBDP considère dans le cadre de la routine de génération exhaustive de toutes les solutions possibles à chaque horizon. Malheureusement, nous constatons que lorsque l'horizon de planification augmente ce pourcentage augmente également puis se stabilise selon le problème et les différents paramètres utilisés. La résultat sur le problème cooperative box-pushing souligne la capacité des deux approches (PBIP et MBDP) à passer à l'échelle. Comme l'algorithme MBDP est incapable de résoudre ce dernier problème même pour de très petits horizons, nous avons fait appel à des approches réduisant la complexité de MBDP en ne sélectionnant qu'un sous-ensemble des observations conjointes sur lesquelles la planification doit être effectuée. Il s'agit des approches IMBDP et MBDP-OC. En ne planifiant que sur un sous-ensemble des observations conjointes, ces algorithmes retournent plus rapidement une solution, cependant cette solution est de qualité réduite. à la différence des algorithmes IMBDP et MBDP-OC, l'algorithme PBIP est capable de retourner une solution de valeur maximale compte tenu des entrées. Bien que l'algorithme PBIP n'inclut pas la planification sur un sous-ensemble des observations conjointes, il parvient néanmoins à résoudre plus rapide et à retourner des solutions de meilleures qualités pour les instances de dimensions les plus élevés de la littérature des DEC-POMDPs, et cela en comparaison des algorithmes IMBDP et MBDP-OC. En moyenne, l'algorithme PBIP est 1.78 fois plus rapide que l'algorithme MBDP-OC et 10 fois plus rapide que l'algorithme IMBDP. Le lecteur notera que les performances des algorithmes IMBDP et MBDP-OC n'ont pu être définis en moyenne compte tenu du temps requis par une seule simulation. Il se pourrait alors que ces résultats diffèrent sensiblement d'une simulation à l'autre, sans pour autant remettre en cause les conclusions ci-dessus.

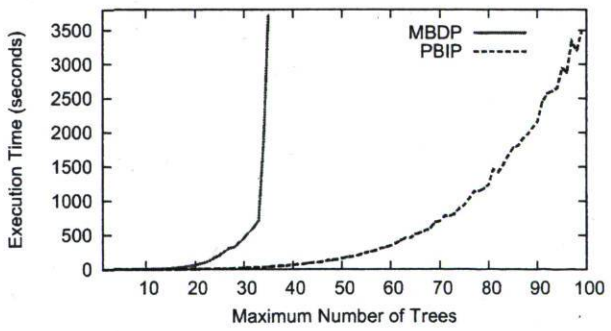


FIGURE 5.16 – Performances for MA-TIGER problem with $T = 10$.

Ces différences deviennent de plus en plus prononcés au fur et à mesure le paramètre K croît. Comme souligner à juste titre par [Seuken and Zilberstein, 2007a,b], en augmentant la valeur du paramètre K, il est possible d'accroître la valeur de la solution retour-

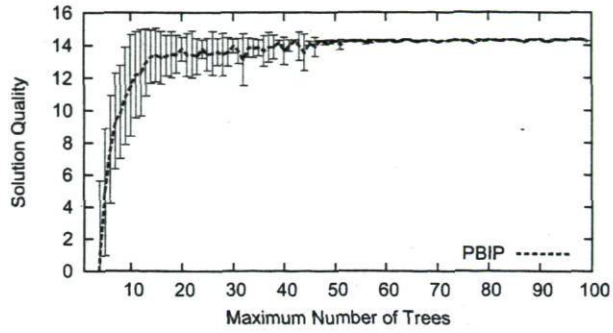


FIGURE 5.17 – Performances for MA-TIGER problem with $T = 10$.

née, mais cela à un prix. En effet, accroître K conduit nécessairement à l'accroissement de l'espace de recherche fonction du paramètre K. Nous illustrons les performances des algorithmes MBDP et PBIP lorsque le paramètre K croît. La Figure 5.16 montre les temps d'exécution des deux algorithmes pour le problème MA-TIGER. Tandis que la Figure 5.17 répertorie les valeurs des solutions pour les différentes valeurs du paramètre K. L'objectif des résultats répertoriés des Figures 5.16 et 5.17 est de montrer comment les performances des ces deux algorithmes sont influencées par différentes valeurs de K. Nous observons que l'algorithme MBDP excède la limite en temps fixée à 1 heure au-delà de $K = 35$ pour la résolution du problème MA-TIGER à horizon 10, tandis que l'algorithme PBIP parvient à résoudre ce problème au-delà de $K = 100$. Cela permet en particulier à l'algorithme PBIP d'améliorer la valeur des solutions retournées pour ce problème. Afin de confirmer la capacité de l'algorithme PBIP à passer à l'échelle, nous augmentons le paramètre K jusqu'à $K = 7$, pour le problème cooperative box-pushing à horizon 10, comme illustrer en Figures 5.18 et 5.19. Nous n'avons pu reporter que les performances de l'algorithme PBIP car les algorithmes MBDP, IMBDP et MBDP-OC excèdent très rapidement la limite en temps fixée à 10 heures. En somme, il apparaît que l'algorithme PBIP montre de bien meilleures performances que l'ensemble des autres approches approximatives et cela sur la totalité des instances testées. Néanmoins, en augmentant la valeur du paramètre K, nous pouvons nous rendre compte des limites de l'algorithme PBIP. Par exemple, lorsque $K = 7$ l'algorithme PBIP requiert approximativement 8 heures afin de venir à bout du problème, alors qu'il ne requiert qu'une heure et demi pour $K = 6$. Certaines des idées non testées ici mais introduite ci-dessus permettraient de palier à cet état de fait.

Nous examinons maintenant comment la suppression des sous politiques dominées dans l'algorithme PBIP-IPG accroît les performances de l'algorithme PBIP. Nous utilisons pour cela les mêmes domaines que ceux utilisés plus haut.

Les résultats expérimentaux sont exposés dans la Table 5.2. Nous pouvons voir que PBIP est incapable de résoudre le problème « Meeting Gird » de dimension 3×3 ou en-

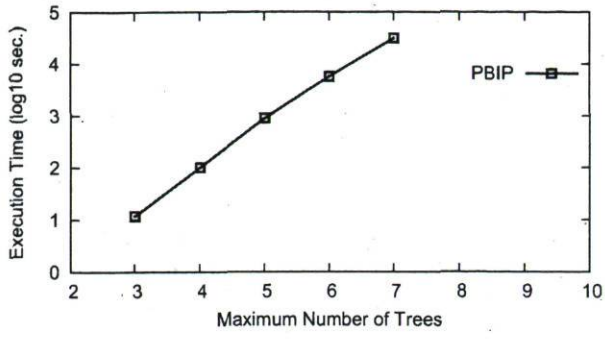


FIGURE 5.18 – Performances de l'algorithme PBIP pour le problème COOPERATIVE BOX-PUSHING à horizon $N = 10$.

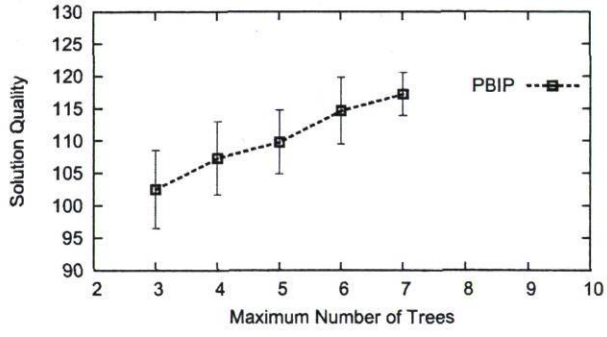


FIGURE 5.19 – Performances de l'algorithme PBIP pour le problème COOPERATIVE BOX-PUSHING à horizon $N = 10$.

core le problème « Mars Rover » pour plusieurs horizons dans les délais de 12 heures. En incorporant l'élimination des sous politiques, PBIP-IPG est capable de résoudre ces problèmes pour de très grands horizons. Sur le problème « BoxPushing », l'algorithme PBIP est capable de résoudre le problème pour tous les horizons testés, mais PBIP-IPG le fait toujours bien plus rapidement. Ces résultats soulignent l'importance de réduire l'espace de recherche dans lequel l'approche heuristique PBIP procède à la construction incrémentale des politiques.

La Figure 5.20 montre les temps de calculs pour différents choix du paramètre K (nombre de politiques individuelles par agent à chaque horizon) pour le problème « Box Pushing » à horizon $N = 10$. Alors que le temps de calculs croît naturellement pour les deux approches, le taux de croissance est plus accentué pour l'algorithme PBIP qu'il ne l'est pour l'algorithme PBIP-IPG. En particulier, pour un problème identique, il est possible de le résoudre via PBIP-IPG pour un paramètre K plus grand an comparaison avec PBIP. Cela est essentiellement dû à l'efficacité dans l'élimination des sous politiques par PBIP-IPG.

N	PBIP	PBIP-IPG	$v^{\delta}(h_0)$
Meeting Grid 3×3, $\|S\| = 81$, $\|A^i\| = 5$, $\|\Omega^i\| = 9$			
10	x	$352s$	3.85
100	x	$3084s$	92.12
200	x	$13875s$	193.39
Box Pushing, $\|S\| = 100$, $\|A^i\| = 4$, $\|\Omega^i\| = 5$			
100	$536s$	$181s$	598.40
1000	$5068s$	$2147s$	5707.59
2000	$10107s$	$4437s$	11392.03
Stochastic Mars Rover, $\|S\| = 256$, $\|A^i\| = 6$, $\|\Omega^i\| = 8$			
2	$106s$	$19s$	5.80
10	x	$976s$	21.18
20	x	$14947s$	37.81

TABLE 5.2 – Temps de calculs (en secondes) et valeur de la fonction de valeurs à la croyance initiale pour chaque horizon.

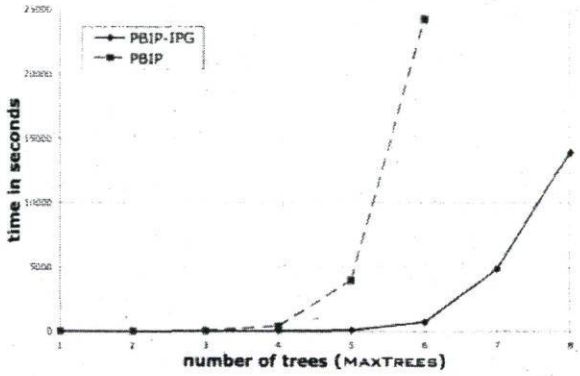


FIGURE 5.20 – Temps de calculs pour différents valeurs de K pour le problème « Box Pushing » à l'horizon $N = 10$.

5.8 Discussion

Ce chapitre propose deux types d'approches : celles basées sur la recherche des arbres de décisions et celles basées sur la recherche de règles de décisions. Il s'agit de deux stratégies fondamentalement différentes. Le tableau suivant donne un aperçu des complexités des différentes stratégies proposées dans ce chapitre, où $|A^*| = \max_{i \in \mathcal{I}} |A^i|$ et $|\Omega^*| = \max_{i \in \mathcal{I}} |\Omega^i|$.

La première approche est communément nommée recherche par chaînage arrière (ou selon le terme anglo-saxon *backward search*). Elle débute avec l'ensemble des actions in-

dividuelles de chaque agent, puis construit progressivement des arbres de décisions de profondeurs de plus en plus élevées, jusqu'à ce que ces arbres atteignent la profondeur désirée. Ces méthodes sont également connues comme étant des méthodes de génération exhaustive de politiques conjointes. Leur complexité est établie au Tableau 5.3.

La seconde approche est communément nommée recherche par chaînage avant (ou selon le terme anglo-saxon *forward search*). Elle débute avec un ensemble d'historiques initiaux, identifie la ou les règles de décisions pour cet ensemble. Par la suite, cet ensemble d'historiques est étendu, puis la ou les règles de décisions sont assignées pour cet ensemble. Et ainsi de suite, jusqu'à ce que l'horizon de planification soit atteint. La complexité en mémoire et en temps de ce type d'approche est également résumée au Tableau 5.3.

	Propriétés à chaque horizon τ	
Méthodes	Complexité en mémoire	Complexité en temps
Génération exhaustive de politiques conjointes	$\left(\lvert A^*\rvert^{\frac{\lvert\Omega^*\rvert^{\tau+1}-1}{\lvert\Omega^*\rvert-1}}\right)^{\lvert\mathscr{S}\rvert}$	$\left(\lvert A^*\rvert^{\frac{\lvert\Omega^*\rvert^{\tau+1}-1}{\lvert\Omega^*\rvert-1}}\right)^{\lvert\mathscr{S}\rvert}$
Induction rétrograde	$\left(\lvert A^*\rvert^{\lvert\Omega^*\rvert^{\tau}}\right)^{\lvert\mathscr{S}\rvert}$	$\left(\lvert A^*\rvert^{\lvert\Omega^*\rvert^{\tau}}\right)^{\lvert\mathscr{S}\rvert}$
Induction rétrograde à base de croyances	$\left(\lvert A^*\rvert^{\lvert\Omega^*\rvert^{\tau}}\right)^{\lvert\mathscr{S}\rvert}$	$\left(\lvert A^*\rvert^{\lvert\Omega^*\rvert^{\tau}}\right)^{\lvert\mathscr{S}\rvert}$

TABLE 5.3 – Propriétés des méthodes d'induction rétrograde.

Nous avons proposé un algorithme approximatif nommé PBIP inspiré de l'approche par chaînage arrière. Cet algorithme peut tout aussi bien être adapté dans le cadre de l'approche par chaînage avant. À noter cependant que l'algorithme PBIP servira alors de simple borne supérieure sur la valeur de la politique conjointe à calculer.

5.9 Conclusion

Ce chapitre offre dans un premier temps une analyse formelle de la planification centralisée dans le cadre de la prise de décisions séquentielles et distribuée. Cette analyse à conduit à la proposition d'une grande variété d'algorithmes exacts et approximatifs, dont la plupart ont été caractérisés par des erreurs bornées. Parmi ces algorithmes figure l'al-

gorithme PBIP pour la résolution des problèmes de contrôle distribué des processus décisionnels de Markov. Cet algorithme est basé sur l'observation selon laquelle la complexité de tout algorithme de programmation dynamique pour la résolution des problèmes de contrôle distribué peut être significativement amélioré en améliorant la résolution du mécanisme de mises à jour. En proposant une heuristique permettant d'éviter l'approche exhaustive jusqu'ici employée, nous sommes parvenus à améliorer considérablement les performances d'un grand nombre des approches auxquels cette heuristique à été greffée. En outre, ce chapitre propose d'explorer l'analyse d'accessibilité comme moyen de faire face à la complexité en mémoire mais aussi en temps des algorithmes, y compris PBIP. La méthode qui en résulte, nommée IPG-PBIP, permet en effet d'améliorer les performances de l'algorithme PBIP.

L'analyse théorique proposée dans ce chapitre à mis fin à tout espoir de résoudre de façon exacte le modèle général des DEC-POMDPs pour des instances de tailles moyennes. Même les approches approximatives ne parviendront à résoudre des instances de tailles moyennes qu'au prix d'une marge d'erreur non négligeable par rapport aux solutions optimales ou ε-optimales. Cependant, cette analyse sera très utile dans l'étude des cas munis de quelque hypothèse simplificatrice, en particulier les DEC-MDPs dont la complexité est également NEXP-complète.

Chapitre 6

DEC-POMDPs à Horizon Infini – « *avec récompenses décomptées* »

« Le désert est la seule chose qui ne puisse être détruite que par la construction. »

Boris Vian

La grande majorité de ce chapitre n'a été publiée nulle part. Seule une toute petite partie est apparue dans le papier : « Policy Iteration Algorithms for DEC-POMDPs with Discounted Rewards » [Dibangoye et al., 2009a] co-rédigé avec Brahim Chaib-draa et Abdel-Illah Mouaddib.

DANS ce chapitre, nous poursuivons la discussion sur la théorie de la planification centralisée pour le contrôle distribué. Cette discussion est débutée au chapitre 5 dans le cadre des problèmes de contrôle distribué à temps discret et à horizon fini. Nous l'étendons au cadre du contrôle distribué des processus de Markov à temps discret et horizon infini.

Malgré l'existence d'une multitude d'autres critères, nous appuyons cette analyse sur le critère d'optimalité le plus étudié à ce jour, à savoir *le total espéré des récompenses cumulées et décomptées* en Section 6.1. Nous nous appesantissons plus particulièrement sur l'établissement des équations d'optimalité en Section 6.2. Nous établissons les conditions d'existence d'une solution à ces équations. Encore plus important, nous introduisons des algorithmes simples de résolution exacte et approximative des équations d'optimalité en Section 6.3, 6.4, 6.5, et 6.6.

Hypothèses 5. *Nous supposons tout au long de ce chapitre que :*

1. *S, A, et Ω sont discrets et dénombrables,*
2. *l'horizon de planification est infini $N = \infty$,*
3. *les récompenses et les transitions sont stationnaires ;*
4. *les récompenses sont bornées, $|r(s,a)| \leqslant M < \infty$, pour tout $(s,a) \in S \times A$.*
5. *les récompenses futures sont décomptées suivant le facteur de décompte $0 \leqslant \lambda < 1$;*

6.1 Critères d'optimalité

À l'issue du choix puis de l'exécution d'une politique conjointe $\pi = (\pi^1, \pi^2, \cdots, \pi^{|\mathscr{I}|})$ d'un ensemble d'agents $\mathscr{I}$, nous pouvons observer une séquence infinie de paires de variables (ω_τ, r_τ), pour tout horizon $\tau = 0, 1, \cdots, N$. La première composante ω_τ de cette paire de variables représente les perceptions courantes de l'ensemble des agents $\mathscr{I}$, tandis que la seconde composante r_τ indique la récompense perçue mutuellement par ces agents à l'horizon $\tau = 0, 1, \cdots, N$. Comme nous l'avons vu au chapitre précédent, la perception courante ω_τ de l'ensemble des agents n'est en générale pas suffisante pour sélectionner l'action conjointe à exécuter. Seule la connaissance de l'ensemble des historiques H_τ collectés jusqu'a l'horizon courant τ permet de déterminer la règle de décisions conjointes $d_\tau \in \bar{D}^{\mathbf{HD}}$ à exécuter sachant la politique conjointe π :

$$\pi(H_\tau) = d_\tau$$

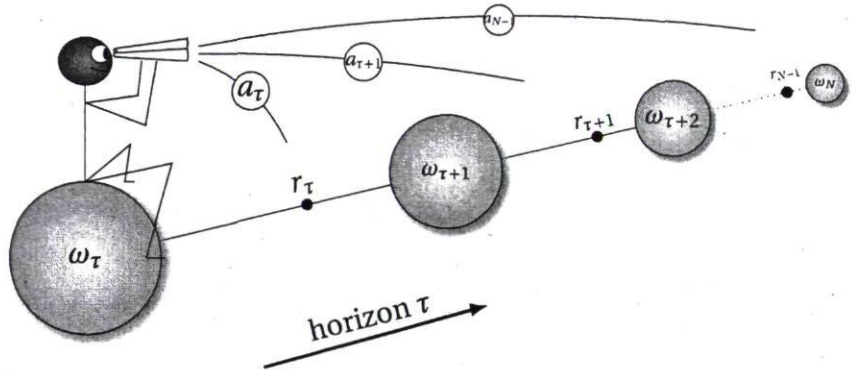


FIGURE 6.1 – Simulation d'un contrôle distribué des processus décisionnels de Markov.

6.1.1 Définitions

Pour toute politique conjointe π fixée, il est important d'y associer une fonction de valeurs v^π afin de légitimer le processus de sélection de la meilleure politique conjointe. L'ensemble des fonctions de valeurs suivantes permet d'associer à toute séquence infinie de récompenses $\{r_\tau\}_{\tau=0,1,\cdots,N}$ à l'issue de l'exécution d'une politique π une valeur, pour tout ensemble initial d'historiques H_0 :

1. la fonction de valeurs v^π relative au *total espéré des récompenses cumulées* pour toute politique conjointe $\pi \in \bar{\Pi}^{\mathbf{AH}}$, est donnée par :

$$v^\pi(H_0) \equiv \lim_{N\to\infty} \mathbb{E}_s^\pi \left\{ \sum_{\tau=1}^{N} r(H_\tau, d_\tau) \right\} = \lim_{N\to\infty} v_{N+1}^\pi(H_0),$$

où v^{π}_{N+1} dénote la fonction de valeurs d'un processus à horizon fini et dont la dernière récompense est nulle ; et N est fini. Le lecteur notera que $v^{\pi}(H_0)$ peut être $-\infty$ ou $+\infty$, on dit alors que la fonction de valeurs diverge. De plus, il existe des cas où cette limite n'existe pas, pour un état donné. Ce critère de mesure de performance d'une politique est par conséquent peu approprié en général.

2. la fonction de valeurs v^{π} relative au *total espéré des récompenses cumulées et décomptées* pour toute politique conjointe $\pi \in \bar{\Pi}^{\mathbf{AH}}$, est donnée par :

$$v^{\pi}(H_0) \equiv \lim_{N\to\infty} \mathbb{E}_{\pi}\left\{\sum_{\tau=1}^{N} \lambda^{\tau-1} r(H_\tau, d_\tau)\right\}, \tag{6.1}$$

avec $0 \leqslant \lambda < 1$. La limite de l'équation 6.1 existe si les récompenses sont bornées :

$$\sup_{s\in S}\sup_{a\in A} |R(s,a)| = M < \infty, \tag{6.2}$$

dans ce cas, nous obtenons alors $|v^{\pi}_{\lambda}(H)| \leqslant (1-\lambda)^{-1} \cdot M$ pour tout ensemble d'historiques H et toute politique conjointe $\pi \in \bar{\Pi}^{\mathbf{AH}}$. Lorsque la limite existe, nous pouvons substituer la limite avec l'opérateur d'espérance. Ainsi si l'expression 6.2 est valide, nous pouvons écrire :

$$v^{\pi}_{\lambda}(H_0) \quad = \quad \mathbb{E}_{\pi}\left\{\sum_{\tau=1}^{\infty} r(H_\tau, d_\tau)\right\}$$

3. la fonction de valeurs v^{π} relative à *la moyenne des récompenses ou gains* pour toute politique conjointe $\pi \in \bar{\Pi}^{\mathbf{AH}}$, est donnée par :

$$v^{\pi}(H_0) \quad = \quad \lim_{N\to\infty} \frac{1}{N} \mathbb{E}_{\pi}\left\{\sum_{\tau=1}^{N} r(H_\tau, d_\tau)\right\} \tag{6.3}$$

$$= \quad \lim_{N\to\infty} \frac{1}{N} v^{\pi}_{N+1}(H_0) \tag{6.4}$$

Lorsque la limite Équation 6.4 n'existe pas, nous définissons la limite inf notée v^{π}_{-} et sup notée v^{π}_{+} de la fonction de valeurs v^{π}, telles que :

$$v^{\pi}_{-}(H_0) \quad = \quad \lim_{N\to\infty} \inf \frac{1}{N} v^{\pi}_{N+1}(H_0)$$

$$v^{\pi}_{+}(H_0) \quad = \quad \lim_{N\to\infty} \sup \frac{1}{N} v^{\pi}_{N+1}(H_0)$$

Les fonctions de valeurs v^{π}_{-} et v^{π}_{+} offrent des bornes supérieure et inférieure sur la fonction de valeurs v^{π} pour toute politique conjointe distributive $\pi \in \bar{\Pi}^{\mathbf{AH}}$. Le lecteur notera que v^{π} n'existe que si $v^{\pi}_{-}(H_0) = v^{\pi}_{+}(H_0)$ pour tout ensemble initial d'historiques H_0.

6.1.2 Évaluation des machines à états finis

Dans cette section, nous développons une méthode d'évaluation de toute politique conjointe et distributive $\pi \in \bar{\Pi}^{\mathbf{AH}}$, pour le critère du total espéré des récompenses cumulées et décomptées. Cette méthode constitue la base des algorithmes et des preuves proposées dans ce chapitre.

Comme conséquence du Théorème 5.58, nous pouvons restreindre l'espace des politiques conjointes considérées. En effet, pour chaque état $s \in S$, il existe toujours une politique conjointe et distributive déterministe $\hat{\pi} \in \bar{\Pi}^{\mathbf{HD}}$ dont le total des récompenses cumulées et décomptées est supérieur ou égal à celui d'une quelconque politique conjointe et distributive $\pi \in \bar{\Pi}^{\mathbf{AH}}$. De sorte que,

$$\begin{aligned} v_\lambda^*(s) &\equiv \sup_{\pi \in \bar{\Pi}^{\mathbf{AH}}} v_\lambda^\pi(s) \\ &= \sup_{\hat{\pi} \in \bar{\Pi}^{\mathbf{HD}}} v_\lambda^{\hat{\pi}}(s). \end{aligned}$$

En particulier, comme nous l'avons indiqué aux chapitres précédents, une politique $\pi_\varepsilon^* \in \bar{\Pi}^{\mathbf{HD}}$ peut parfois être représentée au moyen de machines déterministes à états finis $\delta \equiv (x_0, \alpha, \eta, X)$. Pour tout état $x \in X$, nous définissons $r_x(s)$, $p_x(\bar{s}|s)$ et $o_x(\omega|\bar{s})$ par :

$$r_x(s) \equiv R(s, \alpha(x))$$

puis pour tout état $\bar{s}, s \in S$,

$$p_x(\bar{s}|s) \equiv T(\bar{s}|s, \alpha(x))$$

et enfin pour état $s \in S$, toute observation $\omega \in \Omega$, et tout état $\bar{x} \in X$,

$$o_x(\bar{x}, \omega|\bar{s}) \equiv O(\omega|\alpha(x), \bar{s}) \cdot \eta(\bar{x}|x, \omega)$$

Soit r_x une matrice de dimension $|S|$, dont la s-ème composante est donnée par $r_x(s)$, $\mathbb{P}_x$ la matrice $|S| \times |S|$ avec comme entrée $(s, \bar{s})$ donnée par $p_x(\bar{s}|s)$, et enfin $\mathbb{O}_x$ la matrice $|S| \times |\Omega| \times |X|$ avec comme entrée $(\bar{s}, \omega, \bar{x})$ donnée par $o_x(\omega, \bar{x}|\bar{s})$.

Nous nous référerons à $r_\delta \equiv \{r_x\}_{x \in X}$ comme étant *la matrice des récompenses*; $\mathbb{P}_\delta \equiv \{\mathbb{P}_x\}_{x \in X}$ représente *la matrice de transitions de probabilités*; et $\mathbb{O}_\delta = \{\mathbb{O}_x\}_{x \in X}$ dénote *la matrice des observations.* Toutes ces matrices sont associées à la machine déterministe à états finis $\delta \equiv (x_0, \alpha, \eta, X)$.

Nous associons à chaque état $x \in X$, un hyperplan $v_x \in \mathbb{R}^S$, de sorte que l'ensemble des hyperplans de δ soit donné par $\Lambda^\delta \equiv \{v_x\}_{x \in X}$. Nous dirons également que la fonction de valeurs v^δ d'une machine déterministe à états finis est représentée sous forme d'un ensemble fini d'hyperplans Λ^δ. De façon général, à toute fonction de valeurs v d'une politique, nous associons un ensemble d'hyperplans Λ.

Soit $\mathcal{V}$ un ensemble des fonctions de valeurs v à valeurs bornées, et de dimension $|X||S|$ où $|X|$ est le nombre d'hyperplans associés à v. $\mathcal{V}$ est muni d'un ordre partielle sur les composantes $v(x,s)$ des fonctions de valeurs. De plus, $\mathcal{V}$ dispose d'une norme $\|v\| \equiv \sup_{v_x \in \Lambda} \sup_{s \in S} |v_x(s)|$.

Pour tout $0 \leqslant \lambda \leqslant 1$, l'expression $v_x \equiv (r_x + \lambda \mathbb{P}_x \cdot \mathbb{O}_x \cdot v)$ représente le total espéré des récompenses décomptées à l'issue de l'exécution de l'action $\alpha(x)$ plus les récompenses terminales en provenance de la fonction de valeurs v. Il est alors facile de montrer le Lemme suivant.

Lemme 3. *Soient S un ensemble discret; $|r(s,a)| \leqslant M$, pour tout $a \in A$ et tout $s \in S$; et $0 \leqslant \lambda \leqslant 1$. Pour toute fonction de valeurs v et toute politique $\delta \equiv (x_0, \alpha, \eta, X)$. Il vient alors que $v^\delta \in \mathcal{V}$.*

Démonstration. Par conséquence de l'hypothèse sur $r(s,a)$,

$$\begin{aligned} \|r_\delta\| &= \sup_{x \in X} \|r_x\| \\ &\leqslant M, \end{aligned}$$

donc $r_\delta \in \mathcal{V}$. Lorsque $\mathbb{P}_\delta$ est une matrice de transitions, $\|\mathbb{P}_\delta\| = 1$, de même $\|\mathbb{O}_\delta\| = 1$, de sorte que

$$\begin{aligned} \|\mathbb{P}_\delta \cdot \mathbb{O}_\delta \cdot v\| &\leqslant \|\mathbb{P}_\delta\| \cdot \|\mathbb{O}_\delta\| \cdot \|v\| \\ &= \|v\| \\ &= \sup_{x \in X} \sup_{s \in S} |v_x(s)| \end{aligned}$$

Par conséquent, $\mathbb{P}_\delta \cdot \mathbb{O}_\delta \cdot v \in \mathcal{V}$ pour toute fonction de valeurs v, et donc $v^\delta \in \mathcal{V}$.

□

À l'exécution d'une machine à états finis δ correspond une séquence $\{x_0, \omega_1, x_1, \omega_2, x_2, \cdots\}$, où l'état x_τ dénote l'état $x \in X$ dans lequel la machine se trouve à l'issue de τ prises de décisions x et perceptions d'observations ω. La composante $(s, \bar{s})$ de la matrice de transitions

à l'étape τ notée $\mathbb{P}_\delta^\tau$ satisfait :

$$\begin{aligned}\mathbb{P}_\delta^\tau(\bar{s}|s) &= \left[\mathbb{P}_{x_\tau}, \mathbb{P}_{x_{\tau-1}}, \cdots, \mathbb{P}_{x_1}\right](\bar{s}|s) \\ &= \mathbb{P}_\delta(s_{\tau+1} = \bar{s}|s_\tau = s)\end{aligned}$$

de même, la composante $(\bar{x}, \omega, \bar{s})$ de la matrice des observations à l'étape τ notée $\mathbb{O}_\delta^\tau$ satisfait :

$$\begin{aligned}\mathbb{O}_\delta^\tau(\bar{x}, \omega|\bar{s}) &= \left[\mathbb{O}_{x_\tau}, \mathbb{O}_{x_{\tau-1}}, \cdots, \mathbb{O}_{x_1}\right](\bar{x}, \omega|\bar{s}) \\ &= \mathbb{O}_\delta(x_{\tau+1} = \bar{x}, \omega_{\tau+1} = \omega|s_{\tau+1} = \bar{s})\end{aligned}$$

L'espérance correspondant à la chaîne de Markov associée à la machine à états finis δ est calculée comme suit :

$$\mathbb{E}_s^\delta\{v(s_\tau)\} = \left[\mathbb{P}_\delta^{\tau-1} \cdot \mathbb{O}_\delta^{\tau-1} \cdot v\right](s),$$

pour toute fonction de valeurs v et $1 \leqslant \tau \leqslant \infty$.

Comme conséquence de cette représentation de l'espérance, et de la définition de la fonction de valeurs v_λ^π, pour tout $0 \leqslant \lambda \leqslant 1$,

$$v_\lambda^\delta = \sum_{\tau=1}^{\infty} \lambda^{\tau-1} \cdot \mathbb{P}_\delta^{\tau-1} \cdot \mathbb{O}_\delta^{\tau-1} \cdot r_{x_\tau} \tag{6.5}$$

Suivant l'équation 6.5, et posant $\mathbb{P}_\delta^0 \equiv \mathbf{1}$, l'équation 6.1 peut être réécrite sous forme vectorielle comme suit :

$$\begin{aligned}v_\lambda^\delta &= \sum_{\tau=1}^{\infty} \lambda^{\tau-1} \cdot \mathbb{P}_\delta^{\tau-1} \cdot \mathbb{O}_\delta^{\tau-1} \cdot r_{x_\tau} \\ &= r_{x_0} + \lambda \mathbb{P}_{x_0} \cdot \mathbb{O}_{x_0} \cdot r_{x_1} + \lambda^1 \mathbb{P}_{x_0} \cdot \mathbb{O}_{x_0} \cdot \mathbb{P}_{x_1} \cdot \mathbb{O}_{x_1} \cdot r_{x_2} + \cdots \\ &= r_{x_0} + \lambda \mathbb{P}_{x_0} \cdot \mathbb{O}_{x_0} \cdot \left(r_{x_1} + \lambda \mathbb{P}_{x_1} \cdot \mathbb{O}_{x_1} \cdot r_{x_2} + \cdots\right),\end{aligned}$$

de sorte que,

$$v_\lambda^\delta = r_{x_0} + \lambda \mathbb{P}_{x_0} \cdot \mathbb{O}_{x_0} \cdot v_\lambda^{\bar{\delta}} \tag{6.6}$$

où $\bar{\delta} = (x_1, \alpha, \eta, X)$. L'équation 6.6 montre que la récompense décomptée associée à la politique δ est donnée d'une part, par la récompense immédiate r_{x_0} reçue à la première étape $\tau = 0$ du problème, et d'autre part, par le total espéré des récompenses cumulées et décomptées associé à la politique $\bar{\delta}$. Cette interprétation est simple mais elle permet d'extraire l'équation suivante :

$$v_{x_0}(s) = r_{x_0}(s) + \lambda \sum_{\omega \in \Omega} \sum_{x_1 \in X} \sum_{\bar{s} \in S} p_{x_0}(\bar{s}|s) \cdot o_{x_0}(x_2, \omega|\bar{s}) \cdot v_{x_1}(\bar{s}) \tag{6.7}$$

Les équations 6.6 et 6.7 sont valides pour toute politique représentée sous forme d'une machine à états finis δ ; cependant, ces équations peuvent se simplifier un peu plus, en posant $v^\delta(x) = v_x$. Ainsi, l'équation 6.7 devient pour tout état $x \in X$,

$$v^\delta(x,s) = r_x(s) + \lambda \sum_{\omega\in\Omega} \sum_{\bar{x}\in X} \sum_{\bar{s}\in S} p_x(\bar{s}|s) \cdot o_x(\bar{x},\omega|\bar{s}) \cdot v^\delta(\bar{x},\bar{s}) \tag{6.8}$$

et l'équation 6.8 devient,

$$v^\delta = \mathbb{R}_\delta + \lambda \mathbb{P}_\delta \cdot \mathbb{O}_\delta \cdot v^\delta \tag{6.9}$$

où $\mathbb{R}_\delta = [r_{x_0}, r_{x_1}, \cdots, r_{x_{|X|}}]$, $\mathbb{P}_\delta = [\mathbb{P}_{x_0}, \mathbb{P}_{x_1}, \cdots, \mathbb{P}_{x_{|X|}}]$, et enfin $\mathbb{O}_\delta = [\mathbb{O}_{x_0}, \mathbb{O}_{x_1}, \cdots, \mathbb{O}_{x_{|X|}}]$. Donc, v^δ satisfait le système d'équations,

$$v = \mathbb{R}_\delta + \lambda \mathbb{P}_\delta \cdot \mathbb{O}_\delta \cdot v \tag{6.10}$$

En fait, nous montrons ci-dessous que pour $0 \leqslant \lambda < 1$, v^δ est l'unique solution de l'équation 6.10.

Pour toute fonction de valeurs $v \in \mathcal{V}$, nous définissons la transformation linéaire $\mathbb{L}_\delta$ par :

$$\mathbb{L}_\delta \cdot v \equiv \mathbb{R}_\delta + \lambda \mathbb{P}_\delta \cdot \mathbb{O}_\delta \cdot v.$$

Comme conséquence du Lemme 3, nous pouvons établir que l'application $\mathbb{L}_\delta : \mathcal{V} \to \mathcal{V}$ est une application qui associe à toute fonction de valeurs dans $\mathcal{V}$ une fonction de valeurs dans $\mathcal{V}$. Suivant cette notation, l'équation 6.9 devient alors,

$$v_\lambda^\delta = \mathbb{L}_\delta \cdot v_\lambda^\delta$$

Cela signifie que v_λ^δ est *un point fixe* pour l'application $\mathbb{L}_\delta$ dans l'espace des fonctions de valeurs $\mathcal{V}$. Les théorèmes sur les points fixes nous offrent d'excellents outils pour l'analyse de convergence.

La discussion ci-dessus ainsi que le corollaire suivant mènent à un important résultat, énoncé juste après :

Corollary 1. *Soit Q un opérateur de transformations linéaires défini sur un espace de Banach $\mathcal{V}$, supposons que le déterminant $\rho(Q)$ soit inférieur à 1. Alors $(\mathbf{1} - Q)^{-1}$ existe et satisfait,*

$$(\mathbf{1} - Q)^{-1} = \lim_{N\to\infty} \sum_{n=0}^{N} Q^n$$

Théorème 20. *Supposons que* $0 \leqslant \lambda < 1$. *Alors pour toute machine déterministe à états finis* δ, v_λ^δ *est l'unique solution dans l'espace* $\mathcal{V}$ *des fonctions de valeurs de l'équation,*

$$v = \mathbb{R}_\delta + \lambda \mathbb{P}_\delta \cdot \mathbb{O}_\delta \cdot v. \tag{6.11}$$

De plus, v_λ^δ *peut s'écrire comme suit :*

$$v_\lambda^\delta = (\mathbf{1} - \lambda \mathbb{P}_\delta \cdot \mathbb{O}_\delta)^{-1} \cdot \mathbb{R}_\delta.$$

Démonstration. En réécrivant l'équation 6.11, nous obtenons

$$(\mathbf{1} - \lambda \mathbb{P}_\delta \cdot \mathbb{O}_\delta) \cdot v = \mathbb{R}_\delta. \tag{6.12}$$

Comme $\|\mathbb{P}_\delta\| = 1$ et $\lambda = \|\lambda \mathbb{P}_\delta\| \geqslant \rho(\lambda \mathbb{P}_\delta)$, le Corollaire 1 établit pour $0 \leqslant \lambda < 1$ que $(\mathbf{1} - \lambda \mathbb{P}_\delta \cdot \mathbb{O}_\delta)^{-1}$ existe, de sorte qu'à partir de l'équation 6.12 nous ayons,

$$\begin{aligned} v &= (\mathbf{1} - \lambda \mathbb{P}_\delta \cdot \mathbb{O}_\delta)^{-1} \cdot \mathbb{R}_\delta \\ &= \sum_{\tau=1}^{\infty} \lambda^{\tau-1} \mathbb{P}_\delta^{\tau-1} r_{x_\tau} \\ &= v_\lambda^\delta \end{aligned}$$

□

6.2 Équations d'optimalité

Les équations optimalité et leurs solutions jouent un rôle central dans la théorie du contrôle distribué des processus décisionnels de Markov à récompenses cumulées et décomptées. Dans cette section, nous montrons successivement que : (1) Les équations d'optimalité ont une unique solution dans l'espace des fonctions de valeurs $\mathcal{V}$; (2) La fonction de valeurs v_λ^δ d'un problème de contrôle distribué des processus décisionnels de Markov satisfait les équations d'optimalité.

6.2.1 Motivation et définitions

Sous les hypothèses de ce chapitre, l'équation d'optimalité 5.23 peut être exprimé dans le cas à horizon infini comme suit : pour tout $\delta_n \equiv (\bar{x}_0, \alpha, \eta, X_n)$,

$$v_\lambda^{\delta_{n+1}}(x,s) = \sup_{\delta_{n+1}} \left\{ r_x(s) + \lambda \sum_{\omega \in \Omega} \sum_{\bar{x} \in X_n} \sum_{\bar{s} \in S} p_x(\bar{s}|s) \cdot o_x(\bar{x}, \omega|\bar{s}) \cdot v_\lambda^{\delta_n}(\bar{x}, \bar{s}) \right\} \tag{6.13}$$

pour tout état $x \in X_{n+1}$, où $\delta_{n+1} \equiv (x_0, \alpha, \eta, X_{n+1})$ est une politique résultant du développement de δ_n, nous y reviendrons plus en détails ci-dessous.

Le passage à la limite de l'équation 6.13 suggère que les équations de la forme suivante caractériseront les ensembles de fonctions de valeurs et les politiques conjointes optimales pour les modèles infinis des problèmes de contrôle distribué des processus décisionnels de Markov :

$$v(x,s) = \sup_{\delta} \left\{ r_x(s) + \lambda \sum_{\omega \in \Omega} \sum_{\bar{x} \in X} \sum_{\bar{s} \in S} p_x(\bar{s}|s) \cdot o_x(\bar{x}, \omega|\bar{s}) \cdot v(\bar{x}, \bar{s}) \right\} \tag{6.14}$$

Nous nous référerons à ce système d'équations 6.14 comme étant *les équations d'optimalité* ou encore *les équations de Bellman* pour les modèles infinis des problèmes de contrôle distribué des processus décisionnels de Markov.

Nous utilisons en outre la représentation vectorielle suivante dans le reste du chapitre. Pour toute fonction de valeurs $v \in \mathcal{V}$, nous définissons l'opérateur non linéaire $\mathscr{L}$ sur $\mathcal{V}$ par :

$$\mathscr{L} \cdot v \equiv \sup_{\delta} \{\mathbb{R}_\delta + \lambda \mathbb{P}_\delta \cdot \mathbb{O}_\delta \cdot v\} \tag{6.15}$$

où le supremum est déterminé en accord avec l'ordre partielle sur $\mathcal{V}$. En d'autres termes, ce dernier est déterminé en comparant la valeur $v_\lambda^\delta(s)$ pour chaque état $s \in S$ séparément pour toutes les politiques δ. Lorsque le supremum est atteint, il est facile d'établir que $\mathscr{L} \cdot v \in \mathcal{V}$, pour toute fonction de valeurs $v \in \mathcal{V}$. Nous définissons également l'opérateur $\mathbb{L}$ par :

$$\mathbb{L} \cdot v \equiv \max_{\delta} \{\mathbb{R}_\delta + \lambda \mathbb{P}_\delta \cdot \mathbb{O}_\delta \cdot v\} \tag{6.16}$$

L'opérateur $\mathbb{L}$ correspond à l'opérateur $\mathscr{L}$ dans le cas où le supremum à l'équation (6.15) est atteint. C'est en particulier le cas sous les hypothèses du chapitre en recherchant une politique ε-optimale δ_ε^*.

6.2.2 Opérateur de mise à jour

Cette section propose une implémentation de l'opérateur de mise à jour $\mathscr{L}$ de la politique δ d'un problème de contrôle distribué des processus décisionnels de Markov via sa fonction de valeurs $v \in \mathcal{V}$. En particulier, nous proposons le détail des mécanismes de calcul des hyperplans $\Lambda^{\delta_{n+1}}$ sachant les hyperplans Λ^{δ_n}, donnée par : $\forall x \in X_{n+1}$, $\forall s \in S$,

$$\begin{aligned}\Lambda^{\delta_{n+1}} &\equiv \mathscr{L} \cdot \Lambda^{\delta_n} \\ v_\lambda^{\delta_{n+1}}(x,s) &\equiv \sup_{\delta_{n+1}} \left\{ r_x(s) + \lambda \sum_{\omega \in \Omega} \sum_{\bar{x} \in X_n} \sum_{\bar{s} \in S} p_x(\bar{s}|s) \cdot o_x(\bar{x}, \omega|\bar{s}) \cdot v_\lambda^{\delta_n}(\bar{x}, \bar{s}) \right\}\end{aligned}$$

La politique δ_{n+1} correspond ainsi à la politique de valeur maximale dont les décisions immédiates de valeurs maximales sont celles associées aux états X_{n+1}, et les décisions suivantes de valeurs maximales sont celles de la politique δ_n. Cela signifie que l'ensemble des états X_{n+1} de la politique δ_{n+1} est construit sur la base des états X_n de la politique δ_n. De sorte que tout état $x \in X_{n+1}$ soit défini comme suit :

1. l'action associée à l'état x est telle que $\alpha(x) \in A$,
2. l'état successeur, pour toute observation conjointe $\omega \in \Omega$, est $\eta(x, \omega) \in X_n$,
3. la politique sous forme de machine à états finis $\delta_x \equiv (x, \alpha, \eta, X_{n+1})$ doit être distributive.

La dernière condition est centrale pour la garantie de la distributivité de la politique δ_{n+1} finale. Cette condition est également la principale différence avec les opérateurs de mises à jour développés dans le cadre des problèmes de contrôle centralisé des processus décisionnels de Markov.

Préservation de la distributivité

Préserver la distributivité des politiques au fur et à mesure des mises à jour est essentiel pour garantir la distributivité de la politique retournée. Nous établissons une condition nécessaire et suffisante pour la préservation de la distributivité des politiques lors de l'application de l'opérateur de mises à jour $\mathscr{L}$. Pour y parvenir, nous établissons un lien entre la représentation des politiques à horizon fini et celles à horizon infini. Ainsi, nous pourrons utiliser les résultats établis dans le cas fini pour établir des résultats similaires dans le cas infini.

Rappelons qu'une politique déterministe et distributive $\pi \in \bar{\Pi}^{\mathbf{HD}}$ à horizon fini est composée de règles de décisions $d \in \bar{D}^{\mathbf{HD}}$ également distributives, comme discuté à la Section 5.4.2. Or, toute politique représentée sous forme d'une machine à états finis δ peut être associée à un arbre de décisions de hauteur quelconque. En particulier, une arbre de décisions où les τ premières décisions sont des actions atomiques, puis les décisions suivantes sont sous politiques à horizon infini. La Figure 6.3 illustre un tel encodage. Fort de cette astuce, il est facile d'endéduire une condition nécessaire et suffisante de préservation de la distributivité.

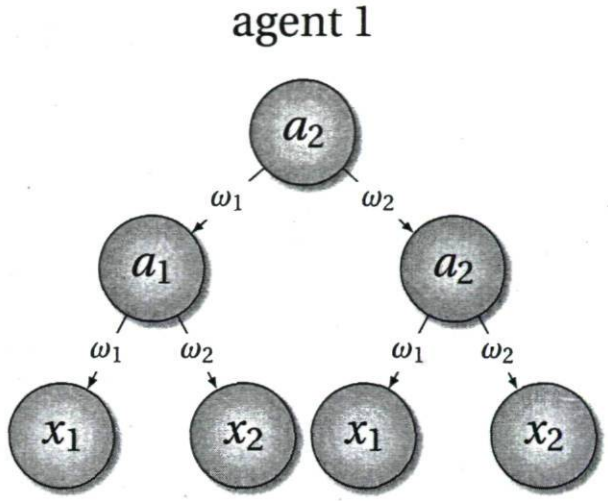


FIGURE 6.2 – Représentation arborescente et finie d'une politique à horizon infini.

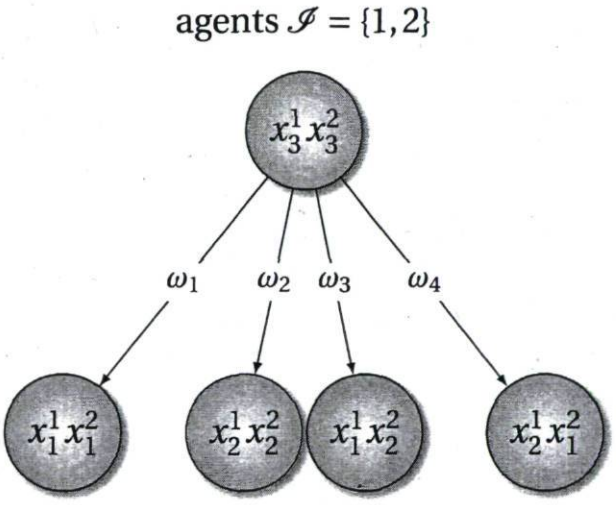


FIGURE 6.3 – Arbre de décisions $\delta_{x_3^1 x_3^2}$ correspondant à une politique à horizon infini.

En particulier, nous appelons *l'arbre de décision d'un état* $x \in X_{n+1}$, l'arbre de décisions à horizon 2 noté δ_x dont la racine est l'état x et l'arc étiqueté par ω transite vers l'état $\eta(x,\omega)$, pour tout $\omega \in \Omega$, comme illustré à la Figure 6.3. Le lemme suivant offre des conditions nécessaires et suffisantes pour qu'une politique produite par $\mathscr{L}$ soit distributive.

Lemme 4. *Soit $\delta_n \equiv (\bar{x}_0, \alpha, \eta, X_n)$ une machine distributive à états finis. Soit X_{n+1} l'ensemble des états de δ_{n+1} donné pour tout $x \in X_{n+1}$ par,*

$$x : \begin{cases} \alpha(x) \in A \\ \eta(x,\omega) \in X_n \end{cases}$$

Si pour tout $x \in X_{n+1}$, l'arbre de décisions δ_x est distributive, alors la machine à états finis $\delta_{n+1} \equiv (x_0, \alpha, \eta, X_{n+1})$ est distributive.

Démonstration. Toutes les machines à états finis $(x, \alpha, \eta, X_{n+1})$ sont indépendantes les unes des autres, pour tout $x \in X_{n+1}$. Par analogie, le lecteur peut les assimiler à un ensemble d'arbres de décisions indépendants. De plus chaque machine à états finis $(x, \alpha, \eta, X_{n+1})$ est distributive car δ_x est distributive. En effet, la règle de décisions $d_0(\emptyset) \equiv \alpha(x)$ à la première étape de décisions est distributive ; en outre la règle de décisions $d_1(\omega) \equiv \eta(x, \omega)$ à la seconde étape de décisions est également distributive car δ_x est distributive ; enfin les règles de décisions aux étapes suivantes sont également distributives car δ_n est distributive.

□

Algorithme de mises à jour

Nous sommes désormais prêts à proposer une implémentation de l'opérateur $\mathscr{L}$.

Algorithme de mises à jour de la politique conjointe δ_n

1. Nous commençons par créer les ensembles intermédiaires $\Lambda^{a,*}$ et $\Lambda^{a,\omega}$, pour toute action $a \in A$, et pour toute observation $\omega \in \Omega$:

$$\Lambda^{a,*} \leftarrow v^{a,*}(s) = R(s,a)$$

et

$$\Lambda^{a,\omega} \leftarrow v_{\bar{x}}^{a,\omega}(s) = \sum_{\bar{s}} p_a(\bar{s}|s) \cdot o_a(\omega|\bar{s}) \cdot v_{\bar{x}}(\bar{s}), \quad \forall \bar{x} \in X_n$$

2. Par la suite, nous créons un ensemble Λ^a ($\forall a \in A$) de vecteurs de fonctions de valeurs $(v_{x_1}^{a,\omega_1}, v_{x_2}^{a,\omega_2}, \cdots, v_{x_{\dot{\Omega}}}^{a,\omega_{|\dot{\Omega}|}})$, le produit-croisé sur l'ensemble des observations de base $\omega \in \dot{\Omega}$, qui inclut un hyperplan $v_{\bar{x}}^{a,\omega}$ issue d'un ensemble $\Lambda^{a,\omega}$. À chaque vecteur d'hyperplans $(v_{x_1}^{a,\omega_1}, v_{x_2}^{a,\omega_2}, \cdots, v_{x_{|\dot{\Omega}|}}^{a,\omega_{|\dot{\Omega}|}})$ nous associons une politique de base $\dot{\delta}_x$ sous forme d'une arbre de décisions, où l'état x est donné par

$$\begin{aligned} \alpha(x) &= a \\ \eta(x,\omega_j) &= x_j \end{aligned}$$

pour tout $\omega_j \in \dot{\Omega}$. À partir de l'arbre de décisions de base $\dot{\delta}_x$, nous pouvons construire une arbre de décisions complet δ_x. Notons que l'ensemble des états X_{n+1}^a correspond à l'ensemble des états x associés aux politiques δ_x ainsi construites.

3. Enfin, nous considérons l'union sur l'ensemble des états X_{n+1}^a pour toute action conjointe $a \in A$ et l'ensemble des états précédents X_n :

$$X_{n+1} \leftarrow X_n \cup_{a \in A} X_{n+1}^a.$$

et pour tout état $x \in X_{n+1}$,

$$v^{\delta_{n+1}} \leftarrow v_x = v^{\alpha(x),*} + \lambda \oplus_{\omega \in \Omega} v_{\eta(x,\omega)}^{\alpha(x,\omega)}$$

FIGURE 6.4 – Algorithme de mises à jour de la politique conjointe δ_n.

Avant d'aller plus loin, il est utile de considérer un exemple simple de DEC-POMDP à horizon infini. Initialement introduit par [Nair et al., 2003], il nous permettra d'illustrer l'ensemble des concepts de ce chapitre.

FIGURE 6.5 – Problème du tigre multi-agents [Nair et al., 2003].

Exemple 5. *Considérons deux agents en face de deux portes closes s_1 et s_2. Derrière une des portes se tient un tigre affamé et derrière l'autre porte se trouve un trésor. Les agents ne savent pas derrière laquelle des portes se tiennent le tigre. Cependant, en écoutant conjointement (action a_2) une des portes, les agents peuvent accroître leur croyance sur la position du tigre. Mais procéder à l'écoute a un coût $R(s,a)$ et une telle écoute ne fournie pas une absolue certitude sur la position du tigre. De plus, les agents ne peuvent partager leurs observations l'un avec l'autre. À chaque étape, chaque agent peut indépendamment soit écouter soit ouvrir une des portes (actions a_1 ou a_2). Si un des agents ouvre la porte derrière laquelle se trouve le trésor, ils reçoivent tous deux une récompense. Si un des agents ouvre la porte derrière laquelle se tient le tigre, une pénalité leur est attribuée. Afin de résoudre ce problème, les agents doivent alors se prémunir d'une politique conjointe permettant d'écouter puis d'ouvrir finalement une des portes. Dès l'ouverture d'une des portes et l'attribution d'un gain ou d'une pénalité, le problème est initialisé à nouveau.*

En observant le problème du tigre, on peut extraire les ensembles d'observations indépendantes $\{(\omega_1,\omega_1),(\omega_2,\omega_2)\}$ et $\{(\omega_1,\omega_2),(\omega_2,\omega_1)\}$ – ce sont des ensembles de base par définition. Considérons par exemple l'ensemble $\{(\omega_1,\omega_2),(\omega_2,\omega_1)\}$. Nous observons que toutes les observations individuelles de chacun des agents y sont représentées une et une seule fois : c'est donc un ensemble de base. À noter que la cardinalité, de ces ensembles est de 2 soit le nombre d'observations individuelles de chaque agent.

L'exemple Figure 6.6 décrit la construction d'une politique distributive. Observons que la politique de base $\dot{\delta}_{x_3^1 x_3^2}$ (états en traits pleins) permet de déterminer la politique $\delta_{x_3^1 x_3^2}$ distributive et complète.

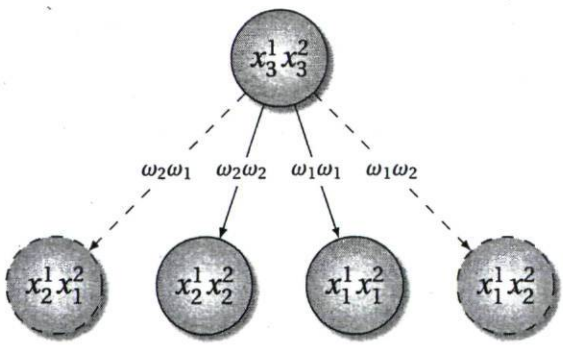


FIGURE 6.6 – Construction d'un hyperplan valide à partir d'un hyperplan de base (en traits plein) et un hyperplan hors base (en tirets).

Afin d'illustrer le mécanisme de mise-à-jour d'une fonction de valeur à travers l'ensemble des hyperplans, considérons à nouveau le problème du tigre. L'ensemble d'hyper-

plans initiaux est extrait directement de la fonction de récompense :

$$v^{\delta_0} \leftarrow R(a), \quad \forall a \in A$$

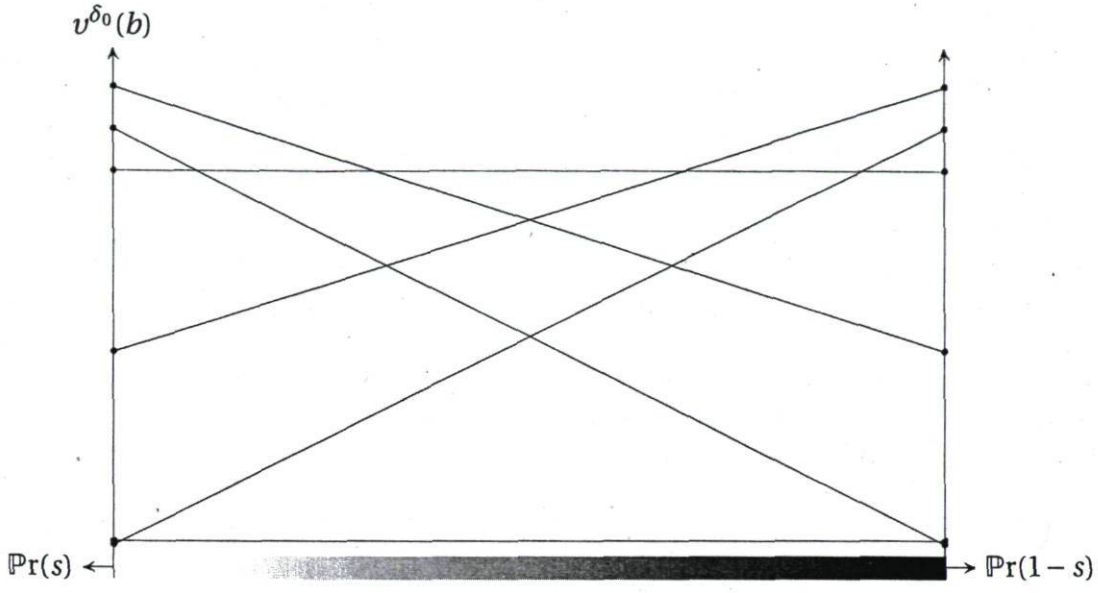


FIGURE 6.7 – Fonctions de valeurs extraites directement de la fonction de récompenses.

La Figure 6.7 montre la représentation géométrique de la fonction de valeur associée à l'ensemble v^{δ_0}. Compte tenu de la quantité d'hyperplans qui résulte de l'application de l'opérateur $\mathscr{L}$, nous n'expliquons que les étapes de génération d'une seule fonction de valeurs.

1. La première étape consiste en la génération des ensembles intermédiaires $\Lambda^{a,*}$ et $\Lambda^{a,\omega}$ pour toute action conjointe $a \in A$ et toute observation conjointe $\omega \in \Omega$:

$$
\begin{aligned}
v^{a_1 a_1,*} &= (20,-50) \\
v^{a_1 a_1,\omega_1\omega_1}_{a_1 a_3} &= (-50,-50) \\
v^{a_1 a_1,\omega_2\omega_2}_{a_1 a_2} &= (-7.5,-7.5) \\
v^{a_1 a_2}_{a_1 a_1,\omega_1\omega_2} &= (-7.5,-7.5) \\
v^{a_1 a_3}_{a_1 a_1,\omega_2\omega_1} &= (-50,-50) \\
\cdots &= \cdots
\end{aligned}
$$

2. Calculer l'ensemble Λ^a des fonctions de valeurs v^a correspondant à l'ensemble des états x tels que : $\dot{\delta}_x$ est donné par,

$$
\begin{aligned}
\alpha(x) &= a \\
\eta(x,\omega_2\omega_2) &= a_1 a_2 \\
\eta(x,\omega_1\omega_1) &= a_1 a_3
\end{aligned}
$$

et δ_x est distributive. La Figure 6.8 montre la politique associé à la fonction de valeurs $v^{a_1 a_1}$ calculée comme suit :

$$
\begin{aligned}
v^{a_1 a_1} &= v^{a_1 a_1,*} + \lambda\left(v^{a_1 a_1,\omega_1\omega_2}_{a_1 a_2} + v^{a_1 a_1,\omega_2\omega_1}_{a_1 a_3} + v^{a_1 a_1,\omega_1\omega_1}_{a_1 a_3} + v^{a_1 a_1,\omega_2\omega_2}_{a_1 a_2}\right) \\
&= (-83.5,-153.5)
\end{aligned}
$$

plus précisément,

$$
\begin{aligned}
v^{a_1 a_1}(s_1) &= 20 + 0.9\cdot(-50-7.5-50-7.5) \\
v^{a_1 a_1}(s_2) &= -50 + 0.9\cdot(-50-7.5-50-7.5)
\end{aligned}
$$

Nous laissons le lecteur le soin de vérifier la distributivité de cette politique δ_x.

3. La Figure 6.9 décrit une partie des fonctions de valeurs $v^{\delta_1} \leftarrow \mathscr{L}\cdot v^{\delta_0}$ construite après une application de l'opérateur $\mathscr{L}$. Elle correspond à l'ensemble des fonctions de valeurs associées aux états $X_1 = X_0 \cup_{a\in A} X_1^a$.

L'une des premières remarques est que $\mathscr{L}$ génère un très grand nombre $|\Lambda_1|$ de fonctions de valeurs en comparaisons à l'ensemble Λ_0 de fonctions de valeurs initiales. Mais plus important encore, si l'on ne regarde que l'enveloppe supérieure de cet ensemble de fonctions de valeurs, on constate qu'il est bien plus élevé que celui de l'ensemble initial. Cette dernière observation tend à conforter un de nos objectifs à savoir la construction d'un opérateur de contraction – reste à vérifier cette hypothèse.

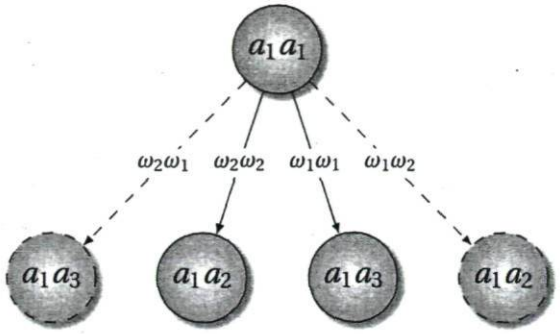


FIGURE 6.8 – Politique correspondant à la fonction de valeurs $v^{a_1a_1}$.

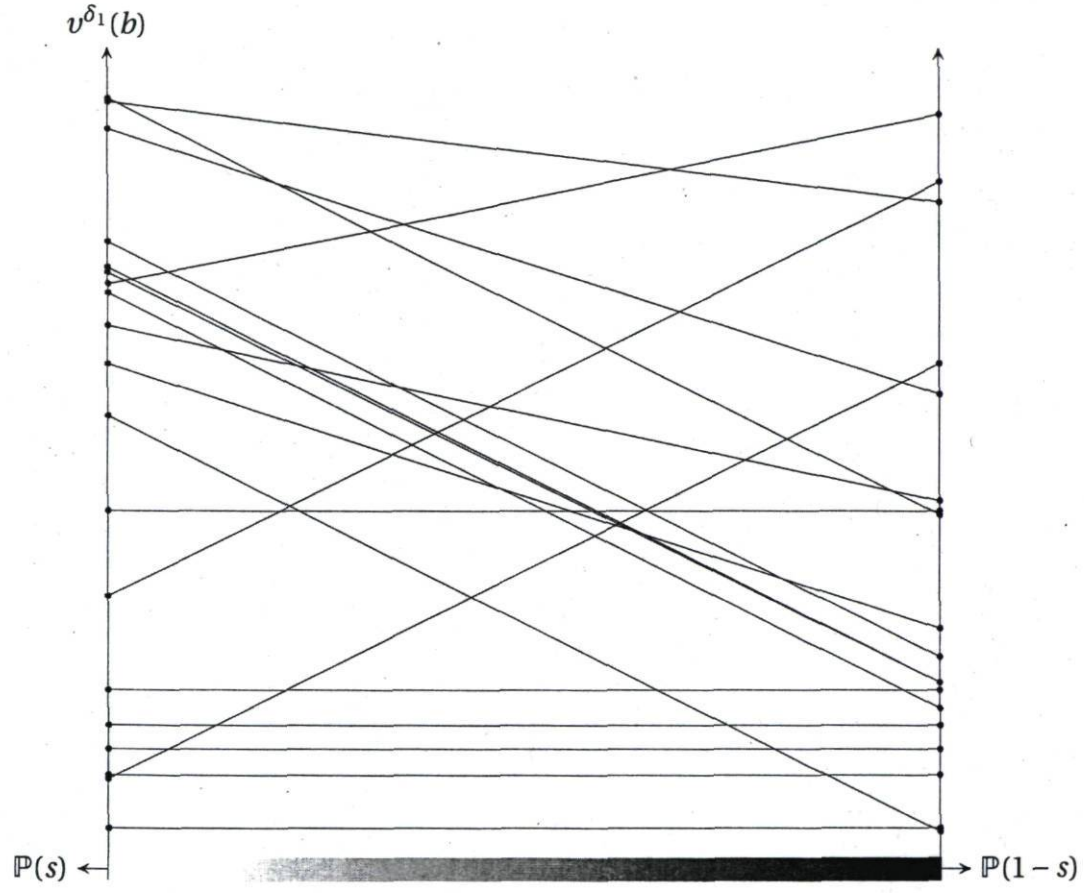


FIGURE 6.9 – Vingt des fonctions de valeurs générées à l'issue de $\mathscr{L} \cdot v^{\delta_0}$.

Élagation des états individuels dominés

En pratique, bon nombre des états $x \in X_{n+1}^a$ sont dominés par un état $\hat{x} \in X_{n+1}$. Il est crucial de supprimer ces états, afin de garantir l'efficacité de l'opérateur de mises à jour. Cependant, il faut garder à l'esprit que contrairement aux problèmes de contrôle centralisé, lors du contrôle distribué nous devons préserver la distributivité de la politique. Plutôt

que de supprimer les états une fois construits, nous suggérons de la faire en amont. C'est à dire bien avant leur construction explicite. En d'autres termes, nous souhaitons supprimer les états individuels $\bar{x}^i \in X_n^i$ inutiles pour la construction des états $x \in X_{n+1}$ non dominés, et cela pour tout agent $i \in \mathscr{I}$. Muni de la représentation sous forme d'arbre de décisions $\delta_{\bar{x}}$ de tout état $\bar{x} \in X_n$, nous pouvons utiliser le même critère introduit dans le cas à horizon fini à la Section 5.6.2.

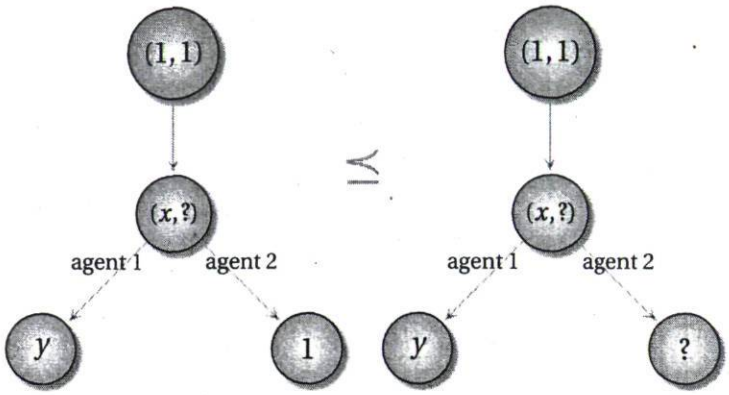


FIGURE 6.10 – Critère de domination dans le cas à horizon infini. Ces deux structures représentent des sous politiques à horizon infini.

L'algorithme 6.11 propose un adaptation de l'algorithme 5.6 introduit dans le cas à horizon fini. L'élimination d'états individuels dominés se fait de façon incrémentale, en effet nous utilisons d'une part l'algorithme 6.11 afin d'éliminer les états individuels de l'ensemble X_n^i pour chaque ensemble de fonctions de valeurs $\Lambda^{a,\omega}$, et d'autre part l'algorithme 6.12 afin d'éliminer les états individuels $x^i \in X_{n+1}^i$ dominés.

Programme linéaire d'élimination de sous politiques dominées

1. Soient $a \in A$, $x^i \in X_n^i$ et $\omega^i \in \Omega^i$.
2. Posons $\omega = \omega^i \omega^{\neq i}$, $\bar{x} = \bar{x}^i x^{\neq i}$ et $x = x^i x^{\neq i}$.
3. Maximiser ξ, sachant : $\forall \bar{x}^i \in X_n^i \backslash \{x^i\}$, $\forall \omega^{\neq i} \in \Omega^{\neq i}$, $\forall s \in S$,

$$\left(\sum_{x^{\neq i}} \zeta(x^{\neq i}) \cdot v_{\bar{x}}^{a,\omega}(s)\right) + \xi \leqslant \left(\sum_{x^{\neq i}} \zeta(x^{\neq i}) \cdot v_{x}^{a,\omega}(s)\right) \qquad (6.17)$$

   où $\sum_{x^{\neq i}} \zeta(x^{\neq i}) = 1$ et $\zeta(x^{\neq i}) \geqslant 0$.
4. Si la variable ξ est négative ($\xi \leqslant 0$) alors supprimer x^i de X_n^i.

FIGURE 6.11 – Programme linéaire d'élimination d'états individuels $\bar{x}^i \in X_n^i$ dominées.

Programme linéaire d'élimination de sous politiques dominées

1. Soit $x^i \in X_n^i$. Posons $\bar{x} = \bar{x}^i x^{\neq i}$ et $x = x^i x^{\neq i}$.
2. Maximiser ξ, sachant : $\forall \bar{x}^i \in X_n^i \backslash \{x^i\}$, $\forall s \in S$,

$$\left(\sum_{x^{\neq i}} \zeta(x^{\neq i}) \cdot v_{\bar{x}}(s)\right) + \xi \leqslant \left(\sum_{x^{\neq i}} \zeta(x^{\neq i}) \cdot v_x(s)\right) \tag{6.18}$$

   où $\sum_{x^{\neq i}} \zeta(x^{\neq i}) = 1$ et $\zeta(x^{\neq i}) \geqslant 0$.
3. Si la variable ξ est négative ($\xi \leqslant 0$) alors supprimer x^i de X_n^i.

FIGURE 6.12 – Programme linéaire d'élimination d'états individuels $\bar{x}^i \in X_{n+1}^i$ dominées.

Comme dans le cas à horizon fini, nous offrons des garanties quant à la préservation de la distributivité et la qualité de la politique construite à l'issue de l'application de la méthode d'élagation d'états individuels.

Lemme 5. *Tout état individuel $x^i \in X_n^i$ supprimé par l'algorithme d'élimination des états individuels dominés est un état individuel qui produit des états $x \in X_{n+1}$ dominés.*

Démonstration. En considérant chaque état individuel x^i comme une arbre de décisions δ_{x^i} et chaque état x comme une arbre de décisions δ_x, il est facile de démontrer le résultats ci-dessus de manière similaire à celle utilisée au Lemme 2.

□

6.2.3 Solutions aux équations d'optimalité

Dans cette section, nous utilisons le théorème de point fixe de Banach afin d'établir l'existence d'une solution aux équations d'optimalité. Nous y mentionnons également une preuve constructive car elle offre les bases des certains des algorithmes développés par la suite.

Tout d'abord rappelons que si $\mathcal{U}$ est un espace de Banach, alors $\mathcal{U}$ est un espace linéaire muni d'une norme $\|\cdot\|$. Il est dit qu'un opérateur $\mathbb{H}$: $\mathcal{U} \to \mathcal{U}$ est *un opérateur de*

contraction s'il existe un réel λ, $0 \leqslant \lambda < 1$ tel que :

$$\|\mathbb{H}v - \mathbb{H}u\| \leqslant \lambda \|v - u\|$$

pour tout u et v appartenant à $\mathscr{U}$.

Théorème 21. ***(Point fixe de Banach)***
Soit $\mathscr{U}$ un espace de Banach et $\mathbb{H}$: $\mathscr{U} \to \mathscr{U}$ un opérateur de contraction. Il vient alors,

1. *qu'il existe une unique solution $v^* \in \mathscr{U}$ telle que $\mathbb{H}v^* = v^*$; et*
2. *que pour toute solution arbitraire $v^0 \in \mathscr{U}$, la séquence $\{v^n\}$ définie par*

$$v^{n+1} = \mathbb{H}v^n = \mathbb{H}^{n+1}v$$

converge vers v^*.

Afin d'appliquer ce théorème dans notre cas, nous montrons maintenant que l'opérateur $\mathscr{L}$ est un opérateur de contraction sur l'ensemble $\mathcal{V}$, espace des ensembles des fonctions de valeurs définis sur S et muni de la norme supremum.

Théorème 22. *Soit $0 \leqslant \lambda < 1$. Il vient alors que les opérateurs $\mathbb{L}$ et $\mathscr{L}$ sont des opérateurs de contractions sur $\mathcal{V}$.*

Démonstration. L'opérateur $\mathbb{L}$ associe à tout ensemble de fonctions de valeurs dans $\mathcal{V}$ un ensemble de fonctions de valeurs dans $\mathcal{V}$. Soit $u \equiv v^{\delta}$ et $v \equiv \mathcal{V}^{\bar{\delta}}$ deux ensembles de fonctions de valeurs appartenant à $\mathcal{V}$, où $v(h) = \max_{x \in X_v} \sum_{s \in S} \mathbb{P}(s|h) \cdot v_x(s)$ pour tout historique conjoint $h \in H$. Fixons $h \in H$, et supposons que $\mathbb{L} \cdot v(h) \geqslant \mathbb{L} \cdot u(h)$, et soit

$$x_h^* \in \underset{x:\, \delta_x \in \bar{\Pi}^{\mathbf{HD}}}{\arg\max} \{\mathbb{P}(h) \cdot (r_x + \lambda \mathbb{P}_x \cdot \mathbb{O}_x \cdot v_{\bar{x}})\}.$$

Il vient alors d'une part que,

$$0 \leqslant \mathbb{L} \cdot v(h) - \mathbb{L} \cdot u(h)$$

et d'autre part que,

$$\begin{aligned}
\mathbb{L} \cdot v(h) - \mathbb{L} \cdot u(h) &\leqslant \mathbb{P}(h) \cdot \left(r_{x_h^*} + \lambda \mathbb{P}_{x_h^*} \cdot \mathbb{O}_{x_h^*} \cdot v_{x_v} - r_{x_h^*} - \lambda \mathbb{P}_{x_h^*} \cdot \mathbb{O}_{x_h^*} \cdot v_{x_u} \right) \\
&= \lambda \mathbb{P}(h) \cdot \mathbb{P}_{x_h^*} \cdot \mathbb{O}_{x_h^*} \cdot \left(v_{x_v} - v_{x_u} \right) \\
&\leqslant \lambda \mathbb{P}(h) \cdot \mathbb{P}_{x_h^*} \cdot \mathbb{O}_{x_h^*} \cdot \| v_{x_v} - v_{x_u} \| \\
&= \lambda \| v_{x_v} - v_{x_u} \|
\end{aligned}$$

où $x_v \in X_v$ et $x_u \in X_u$. En répétant l'argument dans le cas où $\mathbb{L} \cdot v(h) \leqslant \mathbb{L} \cdot u(h)$, nous déduisons que :

$$|\mathbb{L} \cdot v(h) - \mathbb{L} \cdot u(h)| \leqslant \lambda \| v_{x_v} - v_{x_u} \|$$

pour tout historique $h \in H$. Le supremum sur l'ensemble des historiques dans l'expression ci-dessus donne le résultat escompté. Une preuve que l'opérateur $\mathscr{L}$ soit également un opérateur de contraction suit le même argument que celui utilisé pour l'opérateur $\mathbb{L}$.

□

Nous établissons maintenant le résultat le plus important de cette section. Ce résultat est fondamental pour la théorie de la décision dans le cadre des problèmes de contrôle distribué des processus de Markov à horizon infini et à récompenses décomptées.

Théorème 23. *Soient* $0 \leqslant \lambda < 1$, S *un ensemble d'états finis, et* $r(s,a)$ *une fonction de récompenses bornées.*

1. *Dès lors, il existe une solution* $v^* \in \mathcal{V}$ *satisfaisant* $\mathbb{L}v^* = v^*$. *De plus,* v^* *est le seul élément de* $\mathcal{V}$ *muni de cette propriété et égal à* v^*_λ.
2. *Pour chaque politique* $\delta \in \bar{\Pi}^{\mathbf{HD}}$, *il existe une unique solution* $v \in \mathcal{V}$ *satisfaisant* $\mathbb{L}_\delta v = v$. *De plus,* v *est l'unique solution et équivaut à* v^δ_λ.

Démonstration. Comme $\mathcal{V}$ est un espace linéaire complet muni d'une norme, le Théorème 22 établit que $\mathbb{L}$ et $\mathscr{L}$ sont des opérateurs de contractions de sorte que les hypothèses du Théorème 21 soient satisfaites. Dès lors, il existe une unique solution $v^* \in \mathcal{V}$ à l'équation $\mathbb{L}v = v$ ou $\mathscr{L}v = v$. En utilisant $\mathscr{L}v = v$, il est facile d'établir que $v^* = v^*_\lambda$. La partie (2) se démontre de façon similaire en choisissant $\bar{\Pi} = \{\delta\}$.

□

6.2.4 Existence d'une politique optimale

Aussi bien du point de vue des calculs que de l'implémentation des politiques, nous aimerions restreindre notre attention aux politiques à horizon infini représentées sous forme de machines à états finis lorsque nous recherchons les politiques optimales. Bien entendu, une politique optimale dans l'espace des machines à états finis peut être sous optimale dans l'espace des politiques à horizon infini $\bar{\Pi}^{\mathbf{HD}}$. Cette section, vise à montrer que l'existence d'une politique à horizon infini qui atteint le supremum équation 6.13 pour $v = v^*_\lambda$, implique l'existence d'une politique à horizon infini représentée sous forme d'une machine à états finis δ. Elle vise également à montrer comment identifier une telle politique et établir les conditions sur les fonctions de récompenses et de transitions qui garantissent que le supremum soit atteint.

Nous commençons cette section par un résultat proposant une méthode triviale d'identification des politiques optimales pour les modèles de contrôle distribué des processus décisionnels de Markov. Afin de l'implémenter, nous considérons une politique δ, l'évaluons, puis vérifions si sa valeur v^{δ} satisfait l'équation d'optimalité.

Théorème 24. *Une politique $\pi^* \in \bar{\Pi}^{\mathbf{HD}}$ est optimale si et seulement si $v_\lambda^{\pi^*}$ est une solution de l'équation d'optimalité.*

Démonstration. Soit π^* une politique optimale. Il vient alors que $v_\lambda^{\pi^*} = v_\lambda^*$. D'après le Théorème 23 (1) il s'ensuit que $v_\lambda^{\pi^*}$ satisfait $\mathscr{L}v = v$. Supposons que $\mathscr{L}v_\lambda^{\pi^*} = v_\lambda^{\pi^*}$. Cela implique alors que $v_\lambda^{\pi^*} = v_\lambda^*$, donc π^* est optimale.

□

Pour tout $v \in \mathcal{V}$, nous appelons $\delta_v \in \bar{\Pi}^{\mathbf{HD}}$ une politique *v-améliorante* si

$$\delta_v \in \underset{\delta \in \bar{\Pi}^{\mathbf{HD}}}{\operatorname{argmax}} \{r_\delta + \lambda \mathbb{P}_\delta \cdot \mathbb{O}_\delta \cdot v\}$$

De façon équivalente,

$$r_{\delta_v} + \lambda \mathbb{P}_{\delta_v} \cdot \mathbb{O}_{\delta_v} \cdot v \quad = \quad \max_{\delta \in \bar{\Pi}^{\mathbf{HD}}} \{r_\delta + \lambda \mathbb{P}_\delta \cdot \mathbb{O}_\delta \cdot v\}$$

ou encore

$$\mathbb{L}_{\delta_v} v = \mathbb{L} v$$

Sous forme non matricielle, δ_v est v-améliorante si pour toute croyance $b \in \mathscr{P}(S)$,

$$\sum_{s \in S} b(s) \cdot \left(r_{x_v}(s) + \sum_{\omega \in \Omega} \sum_{x \in X} \sum_{\bar{s} \in S} \lambda p_{x_v}(\bar{s}|s) \cdot o_{x_v}(x, \omega|\bar{s}) \cdot v_x(\bar{s}) \right) =$$

$$\max_{x:\, \delta_x \in \bar{\Pi}^{\mathbf{HD}}} \left\{ \sum_{s \in S} b(s) \cdot \left(r_x(s) + \sum_{\omega \in \Omega} \sum_{\bar{x} \in X} \sum_{\bar{s} \in S} \lambda p_x(\bar{s}|s) \cdot o_x(\bar{x}, \omega|\bar{s}) \cdot v_{\bar{x}}(\bar{s}) \right) \right\}$$

où $\delta_v \equiv (x_v, \alpha, \eta, X_v)$.

Les politiques δ qui sont v_λ^*-améliorantes sont d'un ordre particulier. Nous parlerons plutôt de politiques *conservatrices*. En d'autres termes, une politiques δ^* est conservatrice si

$$\mathbb{L}_{\delta^*} v_\lambda^* \quad \equiv \quad r_{\delta^*} + \lambda \mathbb{P}_{\delta^*} \cdot \mathbb{O}_{\delta^*} \cdot v_\lambda^* = v_\lambda^* \tag{6.19}$$

ou encore de façon équivalente,

$$\delta^* \in \underset{\delta \in \bar{\Pi}^{\mathbf{HD}}}{\operatorname{argmax}} \{r_\delta + \lambda \mathbb{P}_\delta \cdot \mathbb{O}_\delta \cdot v_\lambda^*\}$$

Le théorème suivant prouve que ces politiques sont optimales. Il est l'un des plus importants résultats de ce chapitre. Il offre l'un des principaux outils de la théorie de la décision pour le contrôle distribué des processus décisionnels de Markov : c'est à dire une méthode d'identification des politiques optimales.

Théorème 25. *Soit S un ensemble fini d'états, et supposons que le supremum soit atteint à l'équation (6.15) pour tout $v \in \mathcal{V}$. Il vient alors,*

1. *qu'il existe une politique conservatrice $\delta^* \in \bar{\Pi}^{\mathbf{HD}}$;*
2. *que si δ^* est conservatrice, alors δ^* est une politique optimale ; et*
3. *que $v_\lambda^* = \sup_{\delta \in \bar{\Pi}^{\mathbf{HD}}} v_\lambda^{\delta^*}$.*

Démonstration. Le point (1) se déduit en notant que $v_\lambda^* \in \mathcal{V}$ et le supremum est atteint pour l'équation (6.15). Nous déduisons que v_λ^* est l'unique solution de l'équation $\mathbb{L}v = v$. Il s'ensuit alors de l'équation 6.19, que

$$v_\lambda^* = \mathbb{L}v_\lambda^* = r_{\delta^*} + \lambda \mathbb{P}_{\delta^*} \cdot \mathbb{O}_{\delta^*} \cdot v_\lambda^* = \mathbb{L}_{\delta^*} v_\lambda^*.$$

Donc, il vient d'après le Théorème 20 que

$$v_\lambda^{\delta^*} = v_\lambda^*.$$

Le résultat (3) est une conséquence immédiate de (2). □

Notons que la partie (3) implique que le supremum du total espéré des récompenses cumulées et décomptées sur l'ensemble des politiques déterministes est égale à celui sur l'ensemble des politiques ; c'est à dire, pour tout $s \in S$,

$$\sup_{\delta \in \bar{\Pi}^{\mathbf{HD}}} v_\lambda^\delta = \sup_{\pi \in \bar{\Pi}^{\mathbf{HA}}} v_\lambda^\pi$$

Notons également que la même politique déterministe est optimale pour tout $s \in S$.

Nous réétablissons le théorème ci-dessus sous sa forme non matricielle.

Corollary 2. *Supposons que pour tout $v \in \mathcal{V}$ et pour toute croyance $b \in \mathscr{P}(S)$, il existe un état $x_b^v \in X_v$, tel que*

$$\sum_{s \in S} b(s) \cdot \left(r_{x_b^v}(s) + \sum_{\omega \in \Omega} \sum_{x \in X} \sum_{\bar{s} \in S} \lambda p_{x_b^v}(\bar{s}|s) \cdot o_{x_b^v}(x, \omega|\bar{s}) \cdot v_x(\bar{s}) \right) =$$

$$\sup_{x:\, \delta_x \in \bar{\Pi}^{\mathbf{HD}}} \left\{ \sum_{s \in S} b(s) \cdot \left(r_x(s) + \sum_{\omega \in \Omega} \sum_{\bar{x} \in X} \sum_{\bar{s} \in S} \lambda p_x(\bar{s}|s) \cdot o_x(\bar{x}, \omega|\bar{s}) \cdot v_{\bar{x}}(\bar{s}) \right) \right\}$$

Il existe alors une politique déterministe et distributive δ^*. De plus, si $\delta^*(h) = x_h^*$ où

$$x_h^* \in \underset{x:\, \delta_x \in \bar{\Pi}^{\mathbf{HD}}}{\operatorname{argmax}} \left\{ \sum_{s \in S} b(s) \cdot \left(r_x(s) + \sum_{\omega \in \Omega} \sum_{\bar{x} \in X} \sum_{\bar{s} \in S} \lambda p_x(\bar{s}|s) \cdot o_x(\bar{x}, \omega|\bar{s}) \cdot v_{\bar{x}}(\bar{s}) \right) \right\},$$

alors δ^* est optimale.

Lorsque le supremum équation (6.15) n'est pas atteint, il n'existe pas de politique optimale. Dans ce cas, nous recherchons une politique ε-optimale. Nous disons qu'une politique π_ε^* est *ε-optimale* pour $\varepsilon > 0$ si, pour toute croyance $b \in \mathscr{P}(S)$,

$$v_\lambda^{\pi_\varepsilon^*}(b) \quad = \quad v_\lambda^*(b) - \varepsilon$$

Théorème 26. *Soit S un ensemble fini d'états. Alors pour tout $\varepsilon > 0$, il existe une politique déterministe distributive et ε-optimale.*

Démonstration. D'après le Théorème 23, $\mathscr{L} v_\lambda^* = v_\lambda^*$. Choisissons $\varepsilon > 0$ et sélectionnons une politique déterministe $\pi_\varepsilon \in \bar{\Pi}^{\mathbf{HD}}$ satisfaisant

$$\begin{aligned} r_{\pi_\varepsilon} + \lambda \mathbb{P}_{\pi_\varepsilon} \cdot \mathbb{O}_{\pi_\varepsilon} \cdot v_\lambda^* &\geqslant \sup_{\pi \in \bar{\Pi}^{\mathbf{HD}}} \{ r_\pi + \lambda \mathbb{P}_\pi \cdot \mathbb{O}_\pi \cdot v_\lambda^* \} - (1-\lambda)\varepsilon e \\ &= v_\lambda^* - (1-\lambda)\varepsilon e \end{aligned}$$

Comme $v_\lambda^{\pi_\varepsilon^*} = (\mathbf{1} - \lambda \mathbb{P}_{\pi_\varepsilon^*})^{-1} \cdot r_{\pi_\varepsilon^*}$, en réarrangeant les termes et en les multipliant par $(\mathbf{1} - \lambda \mathbb{P}_{\pi_\varepsilon^*})^{-1}$ nous montrons que

$$v_\lambda^{\pi_\varepsilon^*} \quad \geqslant \quad v_\lambda^* - \varepsilon e,$$

de sorte que π_ε^* soit ε-optimale.

□

6.3 Algorithme d'itération de valeurs

L'algorithme d'itération de valeurs est l'algorithme de mieux connu et certainement le plus utilisé dans le cadre des problèmes de contrôle centralisé des processus décisionnels de Markov (complètement ou partiellement observables) : c'est à dire pour les modèles MDPs, MMDPs, POMDPs, ou encore MPOMDPs. Cependant, il n'existe pour ainsi dire aucun algorithme d'itération de valeurs pour les problèmes de contrôle distribué des processus décisionnels de Markov. Il s'agit d'un algorithme permettant de déterminer une

politique ε-optimale δ_ε^* en calculant des approximations successives de la fonction de valeurs $v^{\delta_\varepsilon^*}$. Le succès de cet algorithme réside dans la simplicité de son implémentation bien que son efficacité dans le cadre des modèles de contrôle distribué ne soit pas prouvée.

Cette section se propose d'exposer une implémentation de l'algorithme d'itération de valeurs pour les modèles de contrôle distribué, dans sa forme la plus basic. Nous supposons dans cette section que le maximum en 6.14 et de façon équivalente le maximum en 6.15 soit atteint pour toute fonction de valeurs $v \in \mathcal{V}$. Ce qui est possible par exemple lorsque le nombre de croyances $b \in \mathscr{P}(S)$ accessibles est fini. Bien que cette hypothèse ne soit pas requise pour garantir la convergence de notre algorithme d'itération de valeurs, nous l'adoptons pour simplifier l'exposition de l'algorithme. Par conséquence, nous serons amenés à résoudre le système d'équations suivantes, pour tout $b \in \mathscr{P}(S)$,

$$v(x,b) \quad = \quad \max_{\delta} \left\{ \sum_{s \in S} b(s) \left(r_x(s) + \lambda \sum_{\omega \in \Omega} \sum_{\bar{x} \in X} \sum_{\bar{s} \in S} p_x(\bar{s}|s) \cdot o_x(\bar{x}, \omega | \bar{s}) \cdot v(\bar{x}, \bar{s}) \right) \right\}$$

où $\delta \equiv (x_0, \alpha, \eta, X)$.

6.3.1 L'algorithme

L'algorithme suivant détermine une politique déterministe et distributive ε-optimale $\delta_\varepsilon^* \in \bar{\Pi}^{\mathbf{HD}}$, ainsi qu'une approximation de sa fonction de valeurs.

Algorithme 29 Algorithme d'itération de valeurs (DEC-POMDP).

1: **procedure** VI
2: Sélectionner $v^0 \in \mathcal{V}$.
3: Spécifier $\varepsilon > 0$, et poser $n = 0$.
4: **répéter**
5: Calculer v^{n+1} comme suit :

$$v^{n+1} \quad \leftarrow \quad \mathbb{L} v^n \tag{6.20}$$

6: Incrémenter n de 1.
7: **jusqu'à** $\|v^{n+1} - v^n\| < \frac{\varepsilon(1-\lambda)}{2\lambda}$
8: $\forall b_0 \in \mathscr{P}(S)$, sélectionner δ_{b_0} :

$$\delta_\varepsilon^* \quad \leftarrow \quad \delta_{b_0} \in \arg\max_{\delta_x} \left\{ v^{n+1}(x, b_0) \right\} \tag{6.21}$$

9: **fin procedure**

6.3.2 Convergence et borne sur l'erreur

Le théorème suivant offre des garanties quant à la convergence de l'algorithme d'itération de valeurs.

Théorème 27. *Soient $v^0 \in \mathcal{V}$, $\varepsilon > 0$, et soit $\{v^n\}$ une séquence satisfaisant l'équation 6.20 pour tout $n \geqslant 1$. Il vient alors que,*

1. *v^n converge vers v^*_λ,*
2. *il existe un entier N pour lequel l'équation*

$$\|v^{n+1} - v^n\| < \frac{\varepsilon(1-\lambda)}{2\lambda} \tag{6.22}$$

   est satisfait, pour tout $n \geqslant N$.

3. *la politique δ^*_ε définie en 6.21 est ε-optimale.*
4. *nous avons $\|v^{n+1} - v^*_\lambda\| < \frac{\varepsilon}{2}$ si l'équation 6.22 est satisfaite.*

Démonstration. Les parties (1) et (2) se déduisent immédiatement du Théorème 21. Supposons par ailleurs que l'équation 6.22 soit satisfaite pour un n et δ_ε satisfasse l'équation 6.21. Alors,

$$\|v^{\delta_\varepsilon}_\lambda - v^*_\lambda\| \leqslant \|v^{\delta_\varepsilon}_\lambda - v^{n+1}\| + \|v^{n+1} - v^*_\lambda\|. \tag{6.23}$$

Comme $v^{\delta_\varepsilon}_\lambda$ est un point fixe de l'opérateur $\mathbb{L}_{\delta_\varepsilon}$, et par conséquent de l'équation 6.21, $\mathbb{L}_{\delta_\varepsilon} v^{n+1} = \mathbb{L} v^{n+1}$, la première expression à droite de l'équation 6.23 satisfait,

$$\begin{aligned}
\|v^{\delta_\varepsilon}_\lambda - v^{n+1}\| &= \|\mathbb{L}_{\delta_\varepsilon} v^{\delta_\varepsilon}_\lambda - v^{n+1}\| \\
&\leqslant \|\mathbb{L}_{\delta_\varepsilon} v^{\delta_\varepsilon}_\lambda - \mathbb{L} v^{n+1}\| + \|\mathbb{L} v^{n+1} - v^{n+1}\| \\
&= \|\mathbb{L}_{\delta_\varepsilon} v^{\delta_\varepsilon}_\lambda - \mathbb{L}_{\delta_\varepsilon} v^{n+1}\| + \|\mathbb{L} v^{n+1} - \mathbb{L} v^n\| \\
&\leqslant \lambda \|v^{\delta_\varepsilon}_\lambda - v^{n+1}\| + \lambda \|v^{n+1} - v^n\|
\end{aligned}$$

où les dernières inégalités proviennent du fait que les opérateurs $\mathbb{L}$ et $\mathbb{L}_{\delta_\varepsilon}$ soient des opérateurs de contraction sur l'ensemble $\mathcal{V}$. En réarrangeant les termes, nous obtenons alors,

$$\|v^{\delta^*_\varepsilon}_\lambda - v^{n+1}\| \leqslant \frac{\lambda}{1-\lambda} \|v^{n+1} - v^n\|.$$

Dans le second argument de l'expression à droite de l'équation 6.23, il en résulte l'inégalité suivante :

$$\|v^{n+1} - v^*_\lambda\| \leqslant \frac{\lambda}{1-\lambda} \|v^{n+1} - v^n\|.$$

Donc lorsque l'équation 6.22 est satisfaite, chacune des expressions à la droite de l'équation 6.23 est bornée par $\frac{\varepsilon}{2}$, de sorte que,

$$\|v_\lambda^{\delta_\varepsilon} - v_\lambda^*\| \leqslant \varepsilon.$$

Cela établit les partie (3) et (4).

□

Ainsi l'algorithme d'itération de valeurs pour les modèles de contrôle distribué à horizon infini et récompenses décomptées, détermine une politique δ_ε^* déterministe et distributive ε-optimale en un nombre fini d'itérations.

6.4 Algorithme d'itération de politiques

L'algorithme d'itération de politiques 30 est une méthode générale de résolution de processus de Markov. Elle recherche la politique optimale directement dans l'espace des politiques. Contrairement à l'algorithme d'itération de valeurs présenté ci-dessus, il existe un algorithme d'itération de politiques pour les modèles de contrôle distribué des processus décisionnels de Markov. À la différence de cet algorithme, celui présenté ici est en tout point identique à ces homologues introduits pour la résolution de modèles de contrôle centralisé (MDPs, POMDPs). Cette grande similitude est un atout pour une plus grande facilité de compréhension et d'implémentation. Outre cela, cet algorithme et surtout ses versions optimisées sont bien plus performantes que ceux proposées récemment par [Bernstein et al., 2009].

6.4.1 L'algorithme

L'algorithme d'itération de politiques 30 pour les problèmes de contrôle distribué des processus décisionnels de Markov est décrit comme suit :

Cet algorithme construit une séquence de politiques déterministes et distributives $\{\delta_n\}$ et de fonctions de valeurs $\{v^n\}$. La séquence est finie si le critère d'arrêt $\delta_{n+1} = \delta_n$ est satisfait, et infinie sinon. Nous rappelons qu'une telle séquence n'est finie que si l'ensemble des croyances $b \in \mathscr{P}(S)$ accéssible est fini – en addition des hypothèses du chapitre.

Algorithme 30 Algorithme d'itération de politiques (DEC-POMDP).

1: **procedure** PI
2: Posons $n = 0$, et sélectionnons une politique arbitraire $\delta_0 \in \bar{\Pi}^{\mathbf{HD}}$.
3: **répéter**
4: (**Évaluation de la politique**) Déterminer v^n en résolvant

$$(\mathbf{1} - \lambda \mathbb{P}_{\delta_n}) \cdot v \;=\; r_{\delta_n}. \tag{6.24}$$

5: (**Amélioration de la politique**) Choisir δ_{n+1} tel que

$$\delta_{n+1} \in \underset{\delta \in \bar{\Pi}^{\mathbf{HD}}}{\arg\max} \left\{ r_\delta + \lambda \mathbb{P}_\delta \cdot \mathbb{O}_\delta \cdot v^n \right\} \tag{6.25}$$

posons $\delta_{n+1} \leftarrow \delta_n$ si possible.
6: Incrémentons n de 1.
7: **jusqu'à** $\delta_{n+1} = \delta_n$
8: Posons $\delta^* \leftarrow \delta_n$.
9: **fin procedure**

L'étape 2 de l'algorithme 30 est appelé communément *évaluation de la politique* car elle correspond via la résolution de l'équation 6.24, au calcule du total espéré des récompenses cumulées et décomptées de la politique δ_n. Cela pourrait s'implémenter en utilisant la méthode Gaussienne d'élimination ou tout autre méthode de résolution d'équation linéaire.

L'étape 3 de l'algorithme 30 consiste en la sélection d'une politique δ_{n+1} dite v^n-améliorante. Afin d'éviter des phénomènes cycliques, nous vérifions que δ_{n+1} ne soit égale à δ_n. À noter que, l'implémentation de l'équation 6.25 est faite de sorte que seules les politiques δ_{n+1} composées d'une décision immédiate $a \in A$ et de sous politiques extraites de δ_n sont explorés. Cela permet d'éviter l'énumération exhaustive de toutes les politiques possibles à la fois. Ainsi, l'équation 6.25 permet d'identifier l'ensemble des politiques δ qui soit v^{δ_n}-améliorante pour au moins une croyance. En d'autres termes, il s'agit de ne garder que l'ensemble des états $x \in X_{n+1}$ tels que,

$$r_x + \lambda \mathbb{P}_x \cdot \mathbb{O}_x \cdot v^n \geqslant r_{\delta_n} + \lambda \mathbb{P}_{\delta_n} \cdot \mathbb{O}_{\delta_n} \cdot v^n$$

avec au moins une inégalité stricte pour une croyance $b \in \mathscr{P}(S)$.

6.4.2 Monotonie

Le résultat suivant établit la monotonie de la séquence $\{v^n\}$. C'est une propriété essentielle des algorithmes d'itération de politiques dans les modèles de contrôle centralisé. Nous aimerions l'étendre au cadre des modèles de contrôle distribué.

Théorème 28. *Soit v^n et v^{n+1} des fonctions de valeurs successives générées par l'algorithme d'itération de politiques. Il vient alors $v^{n+1} \geqslant v^n$.*

Démonstration. Soit δ_{n+1} une politique satisfaisant l'équation 6.24 c'est à dire

$$r_{\delta_{n+1}} + \lambda \mathbb{P}_{\delta_{n+1}} \cdot \mathbb{O}_{\delta_{n+1}} \cdot v^n \geqslant r_{\delta_n} + \lambda \mathbb{P}_{\delta_n} \cdot \mathbb{O}_{\delta_n} \cdot v^n = v^n$$

En réarra ngeant les termes, nous obtenons

$$r_{\delta_{n+1}} \geqslant (\mathbf{1} - \lambda \mathbb{P}_{\delta_{n+1}}) v^n$$

En multipliant les deux côtés par $(\mathbf{1} - \lambda \mathbb{P}_{\delta_{n+1}})^{-1}$, nous obtenons

$$v^{n+1} = (\mathbf{1} - \lambda \mathbb{P}_{\delta_{n+1}})^{-1} r_{\delta_{n+1}} \geqslant v^n$$

□

6.5 Algorithme modifié d'itération de politiques

Une des implémentations des plus efficaces de l'algorithme d'itération de politiques dans le cadre des MDPs, est celle de l'algorithme modifié d'itération de politiques. Elle consiste essentiellement à remplacer l'étape d'évaluation *exacte* de la politique en une évaluation *approximative* utilisant par exemple l'algorithme d'itération de valeurs jusqu'à un certain seuil. Bien que cette modification soit intéressante dans le cadre de la résolution des MDPs, en DEC-POMDP cette astuce est pour le moins inefficace. En effet, la complexité principale de l'algorithme d'itération de politiques provient véritablement de l'opérateur de mise à jour de la politique et de la fonction de valeurs associées. Le principal intérêt de cette version réside dans le critère d'arrêt permettant de garantir la terminaison systématique de l'algorithme sous les hypothèses exclusives du chapitre et cela avec une garantie quant à l'erreur produite.

6.5.1 L'algorithme

Soit $\{m_n\}$ une séquence d'entiers non négatifs.

L'algorithme modifié d'itération de politiques illustré à la Figure 6.13, combine à la fois des aspects de l'algorithme d'itération de politiques et ceux de l'algorithme d'itération de valeurs. Tout comme l'algorithme d'itération de valeurs, c'est un algorithme qui débute (étape 1) par une fonction de valeurs arbitraire v^0. De même le critère d'arrêt utilisé à l'étape 3(b) est identique à celui de l'algorithme d'itération de valeurs. Lorsque ce critère est satisfait la politique résultant est ε-optimale. Le déterminisme de la fonction de valeurs u_n^0 à l'étape 3(a) ne requiert guère de calcul supplémentaire, car elle a déjà été établit à l'étape 2 lors du calcul de l'argmax à l'équation 6.30.

Tout comme l'algorithme d'itération de politiques, l'algorithme 6.13 contient l'étape d'amélioration de la politique, étape 2, et une étape d'évaluation de la politique, étape 3. Cependant, l'évaluation de la politique n'est pas exacte, nous nous contentons d'une évaluation approximative. En effet, cette évaluation est effectuée de façon itérative à l'étape 3(c), et répétée m_n fois à l'itération n de l'algorithme. En conséquence de l'égalité 6.29, l'étape 3 peut être représenté par :

$$v^{n+1} = (\mathbb{L}_{\delta_{n+1}})^{m_n+1} v^n$$

La séquence $\{m_n\}$ peut être définie de diverses manières : en fixant $m_n = m$ pour tout n ; ou en sélectionnant m_n de sorte que $\| u_n^{m_n+1} - u_n^{m_n} \| < \varepsilon$. Il nous est possible de prouver que l'algorithme 6.13 converge quelque soit la séquence $\{m_n\}$ en adaptant l'argument utilisé dans le cadre des processus décisionnels de Markov [Putterman, 1994].

6.6 Algorithme d'itération de politiques à mémoire limité

Compte tenu de la complexité souvent prohibitive des problèmes de contrôle distribué, il est essentiel de développer des méthodes capables de faire face aux applications de grandes tailles. L'un des moyens communément utilisés est celui relatif à l'approximation de la fonction de valeurs optimales. En particulier, il est possible de calculer une fonction de valeurs optimales sur un sous-ensemble des croyances $B \subset \mathscr{P}(S)$. De tels méthodes sont dites *à base de points*; lorsque qu'il existe un entier positif K tel que $|B| \leqslant K$, nous parlerons de méthodes à mémoire limitée.

Algorithme modifié d'itération de politiques

1. Sélectionner $v^0 \in \mathcal{V}$, spécifier $\varepsilon > 0$, et fixons $n = 0$.
2. (**Amélioration de la politique**) Choisir une politique δ_{n+1} satisfaisant,

$$\delta_{n+1} \in \arg\max_{\delta \in \bar{\Pi}^{\mathbf{HD}}} \{r_\delta + \lambda \mathbb{P}_\delta \cdot \mathbb{O}_\delta \cdot v^n\} \tag{6.26}$$

   posant $\delta_{n+1} = \delta_n$ si possible, (quand $n > 0$).
3. (**Évaluation partielle de la politique**).
   (a) Poser $k = 0$ et

$$u_n^0 \equiv \max_{\delta \in \bar{\Pi}^{\mathbf{HD}}} \{r_\delta + \lambda \mathbb{P}_\delta \cdot \mathbb{O}_\delta \cdot v^n\} \tag{6.27}$$

   (b) Si $\|u_n^0 - v^n\| \leqslant \frac{\varepsilon(1-\lambda)}{2\lambda}$, aller à l'étape 4. Autrement aller à l'étape (c).
   (c) Si $k = m_n$, aller à l'étape (e). Autrement, calculer u_n^{k+1} donné par

$$u_n^{k+1} = r_{\delta_{n+1}} + \lambda \mathbb{P}_{\delta_{n+1}} \cdot \mathbb{O}_{\delta_{n+1}} \cdot v_n^k \tag{6.28}$$

$$= \mathbb{L}_{\delta_{n+1}} u_n^k \tag{6.29}$$

   (d) Incrémenter k de 1 et retourner à l'étape (c).
   (e) Poser $v^{n+1} = u_n^{m_n}$, incrémenter n de 1, et aller à l'étape 2.
4. Poser $\delta_\varepsilon = \delta_{n+1}$, et arrêter.

♣

FIGURE 6.13 – Algorithme modifié d'itération de politiques.

6.6.1 L'algorithme

L'algorithme à base de points (ou à mémoire limitée selon les cas) illustré à la Figure 6.14, correspond au calcul d'une fonction de valeurs approximatives sur un ensemble fini de croyances $b \in B$. Cela peut aussi se faire dans le cadre de l'algorithme d'itération de valeurs, ou celui d'itération de politiques. Nous avons choisit d'implémenter le calcul d'une fonction de valeurs optimales sur un ensemble fini de croyances dans le cadre de l'algorithme modifié d'itération de politiques. À la différence de l'algorithme modifié d'itération de politiques, nous construisons à chaque itération n et chaque croyance b, une politique δ_{n+1}^b, comme défini à l'équation 6.30. L'ensemble des politiques δ_{n+1}^b constitue par la suite une même politique δ_{n+1}. Outre l'étape 2 d'amélioration de la politique, l'ensemble des

Algorithme modifié d'itération de politiques et à mémoire limité

1. Soit $B \subset \mathscr{P}(S)$. Sélectionner $v^0 \in \mathcal{V}$, spécifier $\varepsilon > 0$, et fixons $n = 0$.
2. (**Amélioration de la politique**) Choisir une politique δ_{n+1} satisfaisant,

$$\delta_{n+1} \leftarrow \delta^b_{n+1} \in \operatorname*{argmax}_{\delta \in \bar{\Pi}^{\mathbf{HD}}} \{r_\delta + \lambda \mathbb{P}_\delta \cdot \mathbb{O}_\delta \cdot v^n\} \cdot b, \quad \forall b \in B \tag{6.30}$$

   posant $\delta_{n+1} = \delta_n$ si possible, (quand $n > 0$).
3. (**Évaluation partielle de la politique**).
   (a) Poser $k = 0$ et

$$u^0_n \equiv \max_{\delta \in \bar{\Pi}^{\mathbf{HD}}} \{r_\delta + \lambda \mathbb{P}_\delta \cdot \mathbb{O}_\delta \cdot v^n\} \tag{6.31}$$

   (b) Si $\|u^0_n - v^n\| \leqslant \frac{\varepsilon(1-\lambda)}{2\lambda}$, aller à l'étape 4. Autrement aller à l'étape (c).
   (c) Si $k = m_n$, aller à l'étape (e). Autrement, calculer u^{k+1}_n donné par

$$\begin{aligned} u^{k+1}_n &= r_{\delta_{n+1}} + \lambda \mathbb{P}_{\delta_{n+1}} \cdot \mathbb{O}_{\delta_{n+1}} \cdot v^k_n \\ &= \mathbb{L}_{\delta_{n+1}} u^k_n \end{aligned}$$

   (d) Incrémenter k de 1 et retourner à l'étape (c).
   (e) Poser $v^{n+1} = u^{m_n}_n$, incrémenter n de 1, et aller à l'étape 2.
4. Poser $\delta_\varepsilon = \delta_{n+1}$, et arrêter.

FIGURE 6.14 – Algorithme modifié d'itération de politiques et à mémoire limité.

autres étapes reste inchangé.

Il est possible de modifier les algorithmes d'itération de valeurs et de politiques de sorte à incorporer cette amélioration de la politique courante sur la base d'un ensemble fini de croyances B. L'opérateur $\mathbb{L}_B$ qui permet la construction d'une telle politique δ_{n+1} est dit v^n-améliorante sur la base des croyances B. Nous proposons deux implémentations possibles de cet opérateur. La première génère l'ensemble des fonctions de valeurs possibles puis en sélectionne la meilleure suivant les croyances B. La seconde construit pour chaque croyance $b \in B$ la meilleure politique δ^b_{n+1} de façon heuristique.

6.6.2 Opérateur de mise à jour $\mathbb{L}_B$

Nous commençons par créer les ensembles intermédiaires $\Lambda^{a,*}$ $\Lambda^{a,\omega}$ (*étape* 1) : $\forall a \in A$, $\forall \omega \in \Omega$,

$$\Lambda^{a,*} \leftarrow v^{a,*}(s) = r(s,a)$$

et $\forall v_x \in v^n$,

$$\Lambda^{a,\omega} \leftarrow v_x^{a,\omega}(s) = \lambda \sum_{\tilde{s} \in S} P(\tilde{s}|s,a) O(\omega|\tilde{s},a) v_x(\tilde{s})$$

Puis, nous créons l'ensemble des fonctions de valeurs Λ^a ($\forall a \in A$). Pour y parvenir nous générons l'ensemble des vecteurs $(v^{a,*}, v^{a,\omega_1}, \cdots, v^{a,\omega_{|\Omega|}})$ où $v^{a,\omega} \in \Lambda^{a,\omega}$ et $v^{a,*} \in \Lambda^{a,*}$. Seulement la fonction de valeurs $v^a = v^{a,*} \oplus_{\omega \in \Omega} v^{a,\omega}$ ne correspond pas nécessairement à une politique distributive. Pour pallier cet état de fait nous avons recours aux objets de base. Nous construisons tout d'abord un ensemble de vecteurs $(v^{a,*}, v^{a,\omega_1}, \cdots, v^{a,\omega_{|\dot{\Omega}|}})$ dits de base, où $v^{a,\omega} \in \Lambda^{a,\omega}$ et $\omega \in \dot{\Omega}$. Ces vecteurs de base correspondent exactement à des politiques de base $\dot{\delta}$. En conséquence, il est possible d'y associer un vecteur complet $(v^{a,*}, v^{a,\omega_1}, \cdots, v^{a,\omega_{|\Omega|}})$ correspondant à une politique déterministe et distributive $\delta \in \bar{\Pi}^{\mathbf{HD}}$. Ainsi la fonction de valeurs $v^a = v^{a,*} \oplus_{\omega \in \Omega} v^{a,\omega}$ est une fonction de valeurs d'une politique déterministe et distributive.

Par la suite, nous considérons l'union des ensembles Λ^a pour toute action conjointe $a \in A$.

$$\Lambda_{n+1} \leftarrow \cup_{a \in A} \Lambda^a.$$

Finalement, nous incluons l'ensemble Λ_n des hyperplans correspondant à la fonction de valeurs v^n dans Λ_{n+1} afin de préserver l'intégrité de la politique δ_{n+1}. Il est ainsi possible de sélectionner la politique δ_{n+1}^b maximale pour chaque croyance $b \in B$, et par conséquent de construire la politique δ_{n+1}.

6.6.3 Implémentation B&B de l'opérateur $\mathbb{L}_B$

Cette méthode a pour objectif de construire une fonction de valeurs v_{n+1} d'une politique dite v_n-améliorante pour un ensemble de croyance B. Contrairement au cas précédent, il s'agit d'éviter l'étape d'énumération exhaustive de l'ensemble des hyperplans

possibles avant l'élimination des hyperplans dominés pour un ensemble de croyances B donné. Pour ce faire, nous construisons directement l'hyperplan maximal pour chaque croyance $b \in B$. En d'autres termes partant de l'ensemble d'hyperplans Λ_n représentant la fonction de valeurs v_n, nous construisons l'ensemble d'hyperplans Λ_{n+1} représentant la fonction de valeurs v_{n+1} mais exclusivement pour les croyances B.

Le problème du déterminisme de l'ensemble des hyperplans Λ_{n+1} sachant l'ensemble des croyances B et l'ensemble des hyperplans intermédiaires $\Lambda^{a,*}$ et $\Lambda^{a,\omega}$ pour toute observation conjointe $\omega \in \Omega$, correspond au problème du déterminisme du vecteur d'hyperplans de base $(v^{a,*}, v^{a,\omega_1}, \cdots, v^{a,\omega_{|\Omega|}})$ de sorte l'hyperplan $v_b \in \Lambda_{n+1}$, soit maximal pour une croyance b, et cela pour toute croyance $b \in B$.

Afin de décrire cette méthode, les définitions suivantes sont requises :

1. $\vec{v}^a$ est un vecteur d'hyperplans, un pour chaque action conjointe $a \in A$.
2. $\vec{v}^a(\omega)$ est un hyperplan $v^{a,o} \in \Lambda^{a,\omega}$ assigné pour l'observation conjointe $\omega \in \Omega$.

La méthode de recherche par chaînage avant dans l'espace des vecteurs d'hyperplans peut être considérée comme une méthode de construction incrémentale du vecteur d'hyperplans maximal pour une croyance $b \in B$, basée sur une heuristique admissible sur la valeur de vecteurs d'hyperplans partiellement définis. À chaque étape de la recherche, le vecteur d'hyperplans partiellement définis le plus prometteur selon l'estimation heuristique est sélectionnée et développé. Lorsque qu'un vecteur d'hyperplans $\vec{v}^a$ est complètement défini, nous sommes alors capable de déterminer l'hyperplans correspondant $v^a = v^{a,*} + \gamma \sum_{\omega \in \Omega} \vec{v}^a(\omega)$. Nous pouvons dès lors définir notre problème d'optimisation comme suit :

$$v_b = \arg\max_{v^a} (v^a \cdot b)$$

Notons qu'un vecteur d'hyperplans dits partiellement défini correspond à un vecteur d'hyperplans où certaine des observations conjointes n'ont pas de représentant dans le vecteur. Afin d'en estimer néanmoins la valeur, nous remplaçons ces valeurs nulles par les hyperplans $v^{a,\omega} \in \Lambda^{a,\omega}$ de valeurs maximales pour la croyance $b \in B$ concernée. La valeur d'un tel vecteur d'hyperplans est une estimation heuristique optimiste sur la valeur exacte de l'ensemble des vecteurs d'hyperplans qui résulterait de ce vecteur partiellement défini. Ainsi, nous avons recours à cette valeur afin de déterminer si oui ou non nous devons développer un nœud correspondant à un vecteur partielle d'hyperplans, pour toute croyance $b \in B$. Nous définissons l'estimation heuristique construite comme la somme de deux estimations. La première, $G(\vec{v}^a, b)$, correspond à l'estimé exacte en provenance des hyperplans $\vec{v}^a(\omega)$ déjà définis. La seconde, $H(\vec{v}^a, b)$, correspond à la borne supérieure sur une assignation possible d'hyperplans aux observations conjointes pour lesquelles il n'y a pas encore d'hyperplans $\vec{v}^a(\omega)$ d'assignés. Nous proposons deux ensembles évolutifs,

Ω_1 et Ω_2 tels que $\Omega = \Omega_1 \cup \Omega_2$. Ces ensembles correspondent à l'ensemble des observations conjointes pour lesquelles un hyperplan a été assigné, et l'ensemble d'observations conjointes pour lesquelles il faut assigner des hyperplans, respectivement.

$$\hat{v}(\vec{v}^a, b) = \underbrace{\left(v^{a,*} + \lambda \sum_{\omega \in \Omega_1} \vec{v}^a(\omega)\right) \cdot b}_{G(\vec{v}^a, b)} + \underbrace{\left(\lambda \sum_{\omega \in \Omega_2} \max_{v^{a,\omega}} (v^{a,\omega} \cdot b)\right)}_{H(\vec{v}^a, b)}$$

Par la suite, nous proposons un détail de l'exécution de l'algorithme pour une seule croyance $b \in B$, et une seule action $a \in A$. Cet algorithme d'implémentation heuristique de l'opérateur $\mathbb{L}$ pour un ensemble de croyances B (noté $\mathbb{L}_B$) prend en entrée : une croyance $b \in B$; une action conjointe $a \in A$; ainsi que les ensembles intermédiaires $\Lambda^{a,*}$ et $\Lambda^{a,\omega}$, voir l'Algorithme 6.15.

Nous commençons par initialiser un ensemble de nœuds notés *Live* avec un vecteur partiellement défini $\vec{v}_0^a$ où aucun des hyperplans $\vec{v}_0^a(\omega)$ n'est défini, pour toute observation conjointe $\omega \in \Omega$, et la valeur associée à la *Solution* initiale est nommée *Incumbent* (étape 1). À chaque itération de l'heuristique, le nœud $\vec{v}_i^a$ possédant l'estimation heuristique la plus élevée est extrait de l'ensemble *Live*, puis développer (étapes 2, 3 et 4). Le développement d'un nœud correspond à la complétion d'une des observations de base $\omega \in \mathring{\Omega}$. De plus, pour chaque nœud fils généré $\vec{v}_{i_p}^a$, les hyperplans $\vec{\alpha}_i^a(\omega)$ qui peuvent-être définis sur la base de ceux déjà définis, sont complétés et l'estimation heuristique correspondante mise à jour. Lorsqu'un nœud correspond à un vecteur complètement défini, son estimation heuristique correspond à sa valeur exacte. Cette dernière est alors comparée à l'*Incumbent*, et la meilleure solution des deux est retenue (étape 5). Si le vecteur complètement défini dispose d'une estimation heuristique inférieure à l'*Incumbent*, ce nœud est supprimé, car aucun des nœuds fils de ce nœuds ne disposera d'une valeur supérieure à l'*Incumbent*. Le cas échéant, le nœud est inséré dans l'ensemble *Live* (étape 6). Lorsque l'arbre de recherche a été complètement exploré, l'algorithme est rappelé avec une autre action conjointe et d'autres ensembles intermédiaires, mais en maintenant la solution courante. Et ainsi de suite, jusqu'à ce que l'ensemble des actions conjointes ait été exploré pour chaque croyance $b \in B$.

6.6.4 Sélection des croyances B

Comme la sélection de l'ensemble des croyances B est cruciale pour la qualité de la solution des algorithmes à base de croyances, nous avons recours à une méthode qui à

Implémentation B&B de l'opérateur $\mathbb{L}_B$

1. Initialiser Incumbent = $\hat{v}(\emptyset, b)$, Live = $\{\vec{v}_0^a\}$
2. Sélectionner un vecteur d'hyperplans $\vec{v}_j^a(\omega) \in$ Live maximal pour b, c'est à dire :

$$\forall \vec{v}_j^a \in \text{Live}: \hat{v}(\vec{v}_j^a, b) \leqslant \hat{v}(\vec{v}_i^a, b)$$

3. Poser Live := Live $\setminus \{\vec{v}_i^a\}$
4. Développer $\vec{v}_i^a$ en générant $\vec{v}_{i_1}^a, \cdots, \vec{v}_{i_k}^a$
5. Si vecteur $\vec{v}_{i_p}^a$ est tel que :
   (a) $\hat{v}(\vec{v}_{i_p}^a, b) >$ Incumbent,
   (b) $\vec{v}_{i_p}^a$ est complètement défini
   alors

$$\begin{aligned} \text{Incumbent} &= \hat{v}(\vec{v}_{i_p}^a, b) \\ \text{Solution} &= v^{a,*} + \lambda \sum_{\omega \in \Omega} \vec{v}_{i_p}^a(\omega) \end{aligned}$$

   pour tout $p = 1, 2, \cdots, k$.
6. Sinon Live = Live $\cup \{\vec{v}_{i_p}^a\}$
7. Si Live $\neq \emptyset$, retourner à l'étape 2, autrement retourner Solution.

♣

FIGURE 6.15 – Implémentation B&B de l'opérateur $\mathbb{L}_B$.

fait ses preuves. En particulier, [Pineau et al., 2003b] a décrit une méthode de simulation centralisée générant un ensemble de croyances. La procédure débute par la sélection d'un ensemble de croyances initiales $B_0 := \{b_0\}$ incluant la croyance initiale b_0 à l'instant $\tau = 0$. Puis, à chaque instant $\tau = 1, 2, \cdots$, elle étend B_τ à $B_{\tau+1}$ en y ajoutant toute les croyances $b_{a,\omega}$ produites, et cela $\forall b \in B_\tau$, $\forall a \in A$, $\forall o \in \Omega$, de sorte que $\mathbb{P}(\omega|b, a) > 0$. L'ensemble des croyance $\bar{\Delta} := \cup_{\tau=0}^{\infty} B_\tau$ ainsi construit est l'ensemble de croyances accessibles durant la simulation centralisée. Il suffit alors de planifier de façon centralisée sur ces croyances afin de construire soit une politique conjointe exacte ou approximative, pour toute croyance initiale b_0. Malheureusement, il est fort probable que l'ensemble $\bar{\Delta}$ soit trop large. Dès lors, plutôt que de retenir l'ensemble des croyances $b_{a,\omega}$, nous pouvons retenir une seule croyance $b^{a,*}$. Celle dont la distance $\mathbb{L}_1$ à l'ensemble courant B est maximale. Puis nous l'ajoutons à l'ensemble B, si et seulement si la distance $\mathbb{L}_1$ est au-delà d'un certain seuil.

6.6.5 Propriétés théoriques

Dans cette section, nous présentons certaines des propriétés théoriques de l'algorithme d'itération de politiques utilisant l'opérateur B&B $\mathbb{L}_B$, cet algorithme sera noté $\bar{\mathbb{L}}_B$-PI. En particulier, nous abordons les questions relatives à la convergence, la borne sur erreur ainsi que la complexité de l'algorithme $\bar{\mathbb{L}}_B$-PI.

Convergence

Les algorithmes d'itération de politiques basés sur un ensemble de croyances B, bien qu'étant des algorithmes approximatifs, héritent plusieurs des propriétés désirables des algorithmes d'itération de politiques dans le cadre du contrôle centralisé [Hansen, 1997, Ji et al., 2007], y compris :

1. Il existe une fonction de valeurs v_{n+1} dite v_n-améliorante si l'algorithme n'a toujours pas convergé. Cette fonction de valeurs v_{n+1} est construite à l'issue de l'application de l'opérateur $\mathbb{L}_B$. De plus, elle améliore la valeur $v_n(b)$ d'au moins une croyance $b \in B$ et ne détériore la valeur d'aucune croyance $b \in B$.
2. L'algorithme $\bar{\mathbb{L}}_B$-PI converge vers une fonction de valeurs et donc une politique déterministe et distributive en un nombre fini d'itérations.
3. L'erreur produite par l'application de l'opérateur $\mathbb{L}_B$ afin de mettre à jour la fonction de valeurs est bornée et lorsque $B \equiv \mathscr{P}(S)$ cette erreur est inférieure à ε.

Borne sur l'erreur

Nous démontrons maintenant que l'erreur entre la fonction de valeurs $v_n \equiv \mathbb{L}_B^n v_0$ résultant de n applications de l'opérateur $\mathbb{L}_B$ et la fonction de valeurs optimales v_λ^* est bornée. La borne dépend de la densité de l'ensemble des croyances B échantillonné dans l'ensemble de toutes les croyances accessibles $\mathscr{P}(S)$: plus l'échantillonnage de B est dense, plus v_n se rapproche de v_λ^*. En arrêtant les itérations de l'algorithme $\bar{\mathbb{L}}_B$-PI à un nombre suffisamment large d'itérations n, nous savons que la différence entre la fonction de valeurs v_n et la fonction de valeurs optimales v_λ^* n'est pas très significative. L'erreur total de l'algorithme $\bar{\mathbb{L}}_B$-PI est alors donnée par :

$$\|v_n - v_\lambda^*\|_\infty \leqslant \|v_n - v_n^*\|_\infty + \|v_n^* - v_\lambda^*\|_\infty \tag{6.32}$$

Comme l'opérateur $\mathbb{L}_B$ au même titre que l'opérateur $\mathbb{L}$ est un opérateur de contraction, le second terme à droite de l'équation 6.32 est borné par $\lambda^n \|v_0^* - v_\lambda^*\|_\infty$ (voir [Bert-

sekas, 2005]). Dans le reste de cette section, nous établissons et prouvons la borne sur l'erreur de premier terme à droite de l'équation 6.32 $\|v_n - v_n^*\|_\infty$. Nous montrons que la borne de [Pineau et al., 2003b] est tout à fait applicable pour l'algorithme $\bar{\mathbb{L}}_B$-PI, bien que compte tenu de la nature « *itération de politiques* » de l'algorithme $\bar{\mathbb{L}}_B$-PI, il soit possible d'établir une borne plus serrée.

Dans un premier temps, nous prouvons tout d'abord le Lemme suivant :

Lemme 6. *L'erreur introduite par l'opérateur $\mathbb{L}_B$ lors d'une mise à jour de la fonction de valeurs sur l'ensemble des croyances B, plutôt que sur l'ensemble $\bar{\mathscr{P}}(S)$ de toutes les croyances accessibles, est bornée par :*

$$\eta \leqslant \frac{\|r\|_\infty}{1-\gamma}\varepsilon_B, \tag{6.33}$$

où $\|r\|_\infty = \max_{s,a} R(s,a)$ *et* $\varepsilon_B = \max_{b' \in \bar{\Delta}} \min_{b \in B} \|b - b'\|_1$.

Démonstration. Afin d'avoir une intuition sur cette preuve, nous proposons une illustration à la Figure 6.16. Soit $\bar{b} \in \bar{\mathscr{P}}(S) \backslash B$ une croyance pour laquelle l'opérateur $\mathbb{L}_B$ fait la plus large erreur lors de la mise à jour de la fonction de valeurs. Soit $b \in B$ la croyance la plus proche selon la norme $\mathbf{L}_1$ de la croyance $\bar{b}$. Soit $v_{\bar{b}}$ l'hyperplan qui aurait été maximal pour la croyance $\bar{b}$ si l'opérateur $\mathbb{L}$ avait été utilisé, et v_b l'hyperplan maximal pour la croyance b. En ne parvenant pas à insérer l'hyperplan $v_{\bar{b}}$ dans l'ensemble des hyperplans Λ_n représentant la fonction de valeurs v_n, l'opérateur $\mathbb{L}_B$ produit une erreur d'au plus,

$$v_{\bar{b}}^* \cdot \bar{b} - v_b \cdot \bar{b}$$

D'autre part, nous savons par hypothèse que

$$v_b \cdot b \geqslant v_{\bar{b}} \cdot b$$

Ainsi,

$$\begin{array}{rcll}
\eta & \leqslant & v_{\bar{b}} \cdot b' - v_b \cdot b' & \\
& = & v_{\bar{b}} \cdot b' - v_b \cdot b' + (v_{\bar{b}} \cdot b - v_{\bar{b}} \cdot b) & \text{ajout de zéro} \\
& \leqslant & v_{\bar{b}} \cdot b' - \alpha_x \cdot b' + v_b \cdot b - v_{\bar{b}} \cdot b & v_b \text{ est également maximal pour } \bar{b} \\
& = & (v_{\bar{b}} - v_b) \cdot (b' - b) & \text{collecte des termes} \\
& \leqslant & \|v_{\bar{b}} - v_b\|_\infty \|b' - b\|_1 & \text{inégalité de Hölder} \\
& \leqslant & \|v_{\bar{b}} - v_b\|_\infty \varepsilon_B & \text{définition de } \varepsilon_B \\
& \leqslant & \frac{\|r\|_\infty}{1-\lambda}\varepsilon_B &
\end{array}$$

□

Le reste de cette établit et prouve la borne sur l'erreur produite par l'algorithme $\bar{\mathbb{L}}_B$-PI

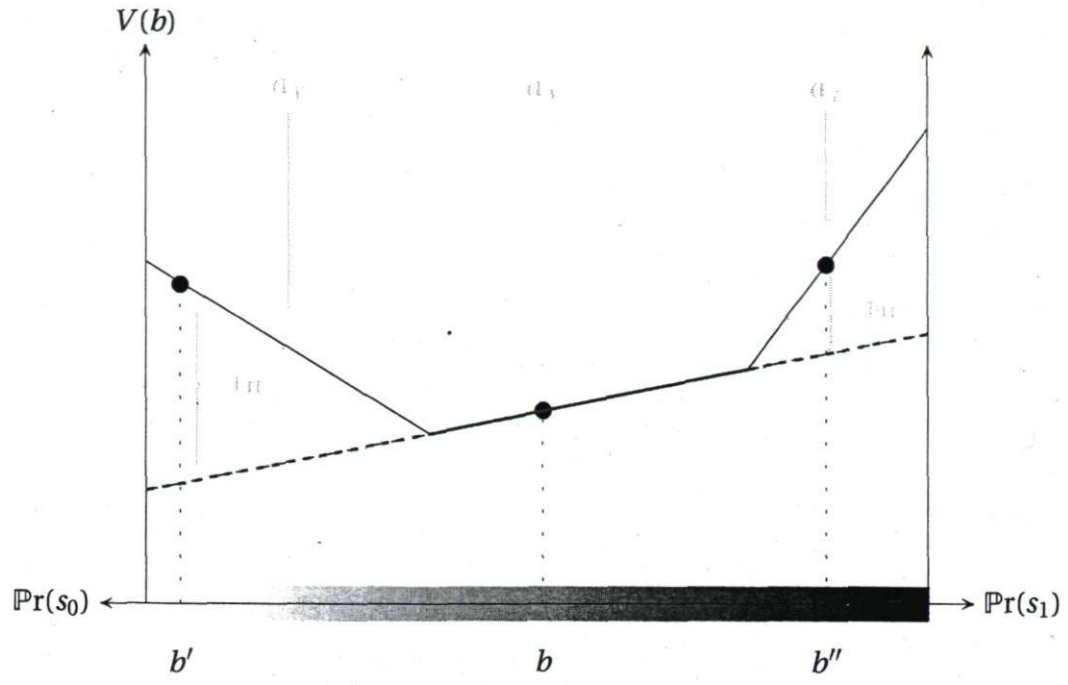


FIGURE 6.16 – Erreur réalisée par l'opérateur $\mathbb{L}_B$ pour la croyance $\bar{b} \notin B$ par rapport à son plus proche voisin $b \in B$.

Théorème 29. *Pour tout ensemble de croyance B, et toute itération n, l'erreur $\eta_\tau = \|v_n - v_n^*\|_\infty$ produite par l'algorithme $\bar{\mathbb{L}}_B$-PI est bornée par :*

$$\eta_\tau \leqslant \frac{\|r\|_\infty}{(1-\gamma)^2}\varepsilon_B,$$

où $\|r\|_\infty = \max_{s,a} R(s,a)$ et $\varepsilon_B = \max_{b' \in \bar{\Delta}} \min_{b \in B} \|b - b'\|_1$.

Démonstration.

$$\begin{aligned}
\eta_\tau &= \|v_n - v_n^*\|_\infty && \text{(définition de } \eta_\tau\text{)} \\
&= \|\mathbb{L}_B v_{n-1} - \mathbb{L} v_{n-1}^*\| && \text{(définition de } \mathbb{L}_B \text{ et } \mathbb{L}\text{)} \\
&\leqslant \|\mathbb{L}_B v_{n-1} - \mathbb{L} v_{n-1}\|_\infty + \|\mathbb{L} v_{n-1} - \mathbb{L} v_{n-1}^*\|_\infty && \text{inégalité du triangle} \\
&\leqslant \eta + \|\mathbb{L} v_{n-1} - \mathbb{L} v_{n-1}^*\|_\infty && \text{(définition de } \eta\text{)} \\
&\leqslant \eta + \lambda \|v_{n-1} - v_{n-1}^*\|_\infty && \text{(contraction)} \\
&\leqslant \eta + \lambda \eta_{n-1} && \text{(définition de } \eta_{n-1}\text{)} \\
&\leqslant \tfrac{\|r\|_\infty}{(1-\lambda)^2}\varepsilon_B && \text{(série des sommes)}
\end{aligned}$$

□

Notons qu'une borne sur l'erreur produite par l'algorithme $\bar{\mathbb{L}}_B$-PI un peu plus serrée peut être dérivée en exploitant la nature de l'algorithme. En effet, nous n'estimons uniquement la perte due à l'élagation à l'issue de l'application de l'opérateur $\mathbb{L}_B$, nous ne

prenons absolument pas en compte l'étape d'évaluation de la politique ainsi que le maintient d'hyperplans issues des itérations précédentes.

6.7 Expérimentations

Nous avons implémenté plusieurs algorithmes de résolution des processus décisionnels de Markov partiellement observables à horizon infini et muni du critère des récompenses décomptées :

Acronymes des algorithmes

1. L'algorithme exact d'itération de valeurs noté $\mathbb{L}$-VI, voir Algorithme 29.
2. L'algorithme exact d'itération de politiques noté $\mathbb{L}$-PI, voir Algorithme 30.
3. L'algorithme d'itération de politiques à base de croyances noté $\mathbb{L}_B$-PI, voir Algorithme 6.14.
4. L'algorithme PI à base de croyances avec B&B noté $\bar{\mathbb{L}}_B$-PI, voir Algorithme 6.15.

Les expérimentations proposées ici sont réalisées en accord avec les problèmes extraits de la littérature des DEC-POMDPs :

Problèmes testés

1. Le problème « *multi-access broadcast channel* » noté (MABC) [Hansen et al., 2004].
2. Le problème « *multi-agentstiger* » noté (TIGER) [Nair et al., 2005].
3. Le problème « *meeting grid* » noté (MEETING-GRID) [Bernstein et al., 2005].
4. Le problème « *recycling-robot* » noté (RECYCLING-ROBOT) [Sutton and Barto, 1998].

6.7.1 Resultats

Les tables 6.1, 6.2, et 6.3 ainsi que les figures 6.17 et 6.18 présentent les résultats de nos algorithmes ainsi que les algorithmes constituant l'état de l'art des méthodes de résolution des DEC-POMDPs à horizon infini. Il est important de signaler que les performances reportées ici sont extraites des divers articles de la littérature, voir la Table 6.1. Comme nous le verrons, il n'est pas nécessaire d'exécuter ces algorithmes compte tenu des différences

notables par rapport aux nôtres.

Pour chacun des problèmes et chacun de nos algorithmes, nous reportons :

1. la fonction de valeurs pour la croyance initiale b_0, c'est à dire $v(b_0)$. Nous distinguons la fonction de valeurs v^δ et v provenant respectivement des algorithmes d'itération de politiques et ceux d'itération de valeurs. Cette distinction peut sembler superflue, néanmoins elle reste essentielle dans le choix d'une des deux méthodes.
2. la taille $|\Lambda|$, de la politique conjointe retournée, ou de façon équivalente le nombre d'hyperplans dans les ensembles Λ^δ et Λ, selon les méthodes.
3. le temps CPU de calcul de la dite politique conjointe et reporté en secondes. Comme l'ensemble des méthodes n'est pas exécuté sur la même machine, ce critère ne servira qu'à comparer nos propres algorithmes.

6.7.2 Discussion

Dans cette section, nous nous proposons d'analyser les résultats expérimentaux collectés pour l'ensemble des algorithmes. Nous commençons tout d'abord par rappeler les performances des algorithmes de l'état de l'art. Puis, nous discuterons de nos algorithmes exacts. Enfin, nous terminerons cette section par les performances des approches approximatives que nous avons proposées.

Algorithmes à mémoire limité

La Table 6.1 présente les performances des algorithmes de l'état de l'art qui pour l'essentiel sont des algorithmes à mémoire limitée. Il s'agit plus précisément des algorithmes BPI et NLP. Rappelons que ces algorithmes à mémoire limité consistent à la recherche d'une politique dans l'espace des machines aléatoires à états finis. Le paramètre limité est le nombre d'états dont dispose la politique recherché. Comme abordé au Chapitre 3, ces méthodes à mémoire limité stagnent souvent dans un optimum local. Sur le problème le plus simple de la littérature, MABC, l'algorithme BPI est incapable de construire une politique optimale pour chacun des paramètres sélectionnés. Il en est de même lorsque le nombre d'états des machines considérées est restreint à un seul état. Bien que l'algorithme NLP détermine une politique optimale pour les machines à un seul état, le coût requis pour la construction des politiques de dimensions plus larges est prohibitif. En particulier, l'algorithme NLP est incapable de déterminer une politique de dimension 16 pour le problème MABC. Sur le problème TIGER, l'algorithme BPI produit des politiques de qualité médiocre tandis qu'à la surprise générale l'algorithme NLP produit des politiques de

bonnes qualités. En effet, nous constatons que NLP produit une politique à un état avec une valeur initiale de −2.66. Nous reviendrons sur cette performance en comparaison avec des approches que nous proposons. À noter d'ors et déjà que pour une machine déterministe à un seul état, il est impossible d'obtenir une telle performance. Cette observation a longtemps été à l'origine d'une fausse bonne idée selon laquelle l'espace des politiques aléatoires « *dominait* » celui des politiques déterministes. La différence est qu'un état d'une machine aléatoire équivaut au moins à $|A|$ états d'une machine déterministe. En d'autres termes, la comparaison ne doit pas se faire sur la dimension des politiques mais directement sur les meilleures politiques construites par chacune des méthodes. Sur le problème RECYCLING-ROBOT , les deux algorithmes BPI et NLP produisent des politiques avec sensiblement les mêmes performances à l'état initial, c'est à dire entre 24 et 32. De manière générale, ces algorithmes souffrent de plusieurs faiblesses :

1. ils sont souvent coincés dans des optimums locaux.
2. il n'existe aucune borne sur l'erreur des performances des politiques qu'ils construisent.
3. le temps qu'ils requièrent pour la construction de politiques de dimension raisonnable est prohibitif.

	BPI			NLP		
$\#i$	$v^{\delta}(b_0)$	$\|\Gamma\|$	CPU (s)	$v^{\delta}(b_0)$	$\|\Gamma\|$	CPU (s)
MABC						
1	4.68	1	1	9.1	1	1
2	4.06	4	1	9.1	4	3
3	8.63	9	2	9.1	9	764
4	7.85	16	5	9.1	16	4061
Tiger-A						
1	-125	1	t.e.	-2.66	1	1
2	-95	2	t.e.	-1.33	2	23
3	-195	3	t.e.	-1.33	3	300
4	-115	4	t.e.	-1.00	4	1700
Recycling-Robot						
1	6.20	1	500	18.0	1	1
2	18.3	2	1000	28.0	2	25
3	18.3	3	1700	32.0	3	700
4	24.0	4	2800	31.3	4	2200

TABLE 6.1 – Résultats expérimentaux des algorithmes BPI et NLP, où « t.e. » signifie temps requis excédé.

Nos algorithmes exacts : $\mathbb{L}$-VI et $\mathbb{L}$-PI

Nos résultats indiquent clairement que les politiques produites ont des performances bien plus élevées que celles des politiques produites par les méthodes précédentes.

En particulier, la Table 6.2 montre sur le problème RECYCLING-ROBOT que l'algorithme $\mathbb{L}$-PI produit des politiques dont les performances par rapport à la croyance initiale sont de l'ordre de 58.23. Le meilleur des algorithmes précédents n'obtient que 31.3 pour le même problème, voir la Table 6.1.

Cependant, il est à noter que l'algorithme NLP obtient des performances supérieures à celle de $\mathbb{L}$-PI, pour le problème TIGER. Nous devons avouer que nous ne nous l'expliquons pas. La seule explication serait que le modèle utilisé soit différent ou alors que ces valeurs ne constituent que des bornes supérieures sur la valeur exacte que produirait NLP. Comme nous pouvons le voir, l'algorithme $\mathbb{L}$-VI obtient la meilleure performance de tous −0.36.

Il est cependant fort probable que cette valeur ne soit qu'une borne supérieure sur la valeur exacte. En effet, à chaque itération de l'algorithme $\mathbb{L}$-VI la fonction de valeurs mises à jour ne correspond pas nécessairement à celle de la politique courante. Elle n'est qu'une estimation heuristique. C'est seulement à la convergence de cet algorithme que la fonction de valeurs est définie à ε près de la fonction de valeurs optimales si cette dernière existe. Ainsi, une politique qui a besoin de plusieurs mises à jour de la fonction de valeurs, disons 100, et qui débute avec une valeur initiale élevée par exemple +10, l'algorithme $\mathbb{L}$-VI nécessitera plusieurs itérations avant d'obtenir la fonction de valeurs exactes de cette politique. C'est pour cette raison, que parfois l'algorithme $\mathbb{L}$-VI maintient une borne supérieure sur la fonction de valeurs réelles pendant plusieurs itérations avant d'atteindre la fonction de valeurs exactes.

Néanmoins, sur l'ensemble des quatre problèmes traités, nous ne pouvons pas véritablement favorisé un de nos deux algorithmes exacts. Il faudra fort probablement les différentier sur d'autres benchmarks. Une chose est à noter néanmoins, ces algorithmes ont de biens meilleures performances et garanties que les algorithmes précédents. Le principal point faible des ces méthodes reste leur incapacité à passer à l'échelle. En effet, assez vite l'espace mémoire requis devient prohibitif. Il est alors crucial d'étudier les performances lorsque ces algorithmes sont à mémoire limité.

	𝕃-VI			𝕃-PI		
$\#i$	$v(b_0)$	$\|\Gamma\|$	CPU (s)	$v^{\delta}(b_0)$	$\|\Gamma\|$	CPU (s)
MABC						
1	1.00	4	0.01	9.1	1	0.04
2	2.59	30	0.05	9.1	4	0.11
3	3.87	80	0.45	9.1	24	0.20
4	4.40	144	2.87	9.1	90	0.53
5	5.11	238	13.1	9.1	195	3.96
6	5.83	272	47.9	9.1	340	25.3
7	6.12	528	113.7	9.1	529	109.3
Tiger-A						
1	-2.0	4	0.05	-19.99	1	0.04
2	-3.8	25	0.18	-19.99	9	0.12
3	-5.42	120	0.57	-19.99	49	0.27
4	-4.09	156	9.52	-10.93	306	1.58
5	-0.36	840	34.6	-10.66	1722	117.8
Recycling-Robot						
1	7.81	1	0.01	49.27	1	0.09
2	17.8	9	0.04	52.06	9	0.21
3	25.6	49	0.19	54.48	42	0.42
4	30.1	132	2.07	55.32	121	1.92
5	36.1	255	19.3	57.64	342	15.7
6	40.5	675	108.7	58.23	870	252.5
Meeting-Grid						
1	0.47	15	0.03	3.06	25	3.48
2	0.99	1995	9.39	4.75	10404	35.8

TABLE 6.2 – Performances des algorithmes 𝕃-VI et 𝕃-PI

Algorithme force brute à base de croyances : $\mathbb{L}_B$-PI

L'algorithme force brute à base de croyances, noté $\mathbb{L}_B$-PI, est basé sur la sélection des hyperplans maximaux pour un ensemble de croyances. Cela est possible d'une part, par la génération exhaustive, et d'autre part, par la sélection des hyperplans non dominés sur B. Comme nos algorithmes exacts ont été incapables de construire des politiques de très grande qualité compte tenu de l'exigence de l'optimalité, l'objectif de ces algorithmes est de sacrifier l'optimalité pour la faisabilité. Nous illustrons les performances de l'algorithme $\mathbb{L}_B$-PI à la Table 6.3. Il est utile de noter que cet algorithme produit des politiques de qualités très proches de celles produites par l'algorithme exact 𝕃-PI, pour les mêmes itérations.

Par exemple, l'algorithme $\mathbb{L}_B$-PI parvient à déterminer une politique dont les performances sont identiques à celles de l'algorithme exact $\mathbb{L}$-PI pour le problème RECYCLING-ROBOT. Et cela sur la totalité des 6 itérations au terme desquelles l'algorithme $\mathbb{L}$-PI s'arrête. Bien que nous soyons ainsi parvenus à réaliser quelques itérations de plus que celles réalisées par les algorithmes exacts, nous ne sommes toujours pas parvenus à la convergence. Cela est dû cette fois ci non plus à la mémoire mais plutôt à la complexité en temps résultant de la génération exhaustive. Pour pallier cet état de fait, nous discutons les performances d'une implémentation plus efficace utilisant la recherche heuristique : séparation et évaluation (selon l'acronyme anglais *branch-and-bound*).

Algorithme B&B à base de croyances : $\bar{\mathbb{L}}_B$-VI et $\bar{\mathbb{L}}_B$-PI

Les Figures 6.17 et 6.18 illustrent les performances de l'algorithme d'itération de politiques à base de croyances B. Il est développé suivant l'approche de séparation et évaluation. Selon qu'il s'agisse d'un algorithme d'itération de valeurs ou de politiques, il est noté $\bar{\mathbb{L}}_B$-VI et $\bar{\mathbb{L}}_B$-PI, respectivement.

Pour chaque problème nous proposons deux graphes : à droite, nous montrons les performances en temps des deux versions de l'algorithme ; à gauche, nous illustrons les performances des politiques construites par ces deux versions, et cela à chacune de leurs itérations. La bonne nouvelle est que toutes les deux versions convergent dans trois des quatre problèmes traités. De plus, de façon similaire à leurs versions « *force brute* », les performances des politiques à chaque itération sont très proches de celle des algorithmes exacts. La mauvaise nouvelle est qu'il reste difficile de départager les deux versions. La version « *itération de politiques* » converge après très peu d'itérations sur les problèmes MABC, MEETING-GRID et TIGER, comme nous pouvions nous attendre. Cependant, la version « *itération de valeurs* » est plus rapide sur le problème RECYCLING-ROBOT. En somme, tant que l'étape d'évaluation de la politique dans la version « *itération de politiques* » n'est pas très coûteuse en temps de calcul, l'algorithme $\bar{\mathbb{L}}_B$-PI est plus rapide. Or la complexité de cette étape dépend du nombre d'hyperplans conservés à chaque itération de l'algorithme. Ainsi, lorsque ce nombre devient trop grand, l'avantage de l'algorithme $\bar{\mathbb{L}}_B$-PI disparaît au profit de l'algorithme $\bar{\mathbb{L}}_B$-VI. Sur le problème RECYCLING-ROBOT par exemple, nous notons que bien que l'algorithme $\bar{\mathbb{L}}_B$-PI requiert moins d'itérations que l'algorithme $\bar{\mathbb{L}}_B$-VI, ce dernier requiert bien moins de temps que l'algorithme $\bar{\mathbb{L}}_B$-PI afin de converger.

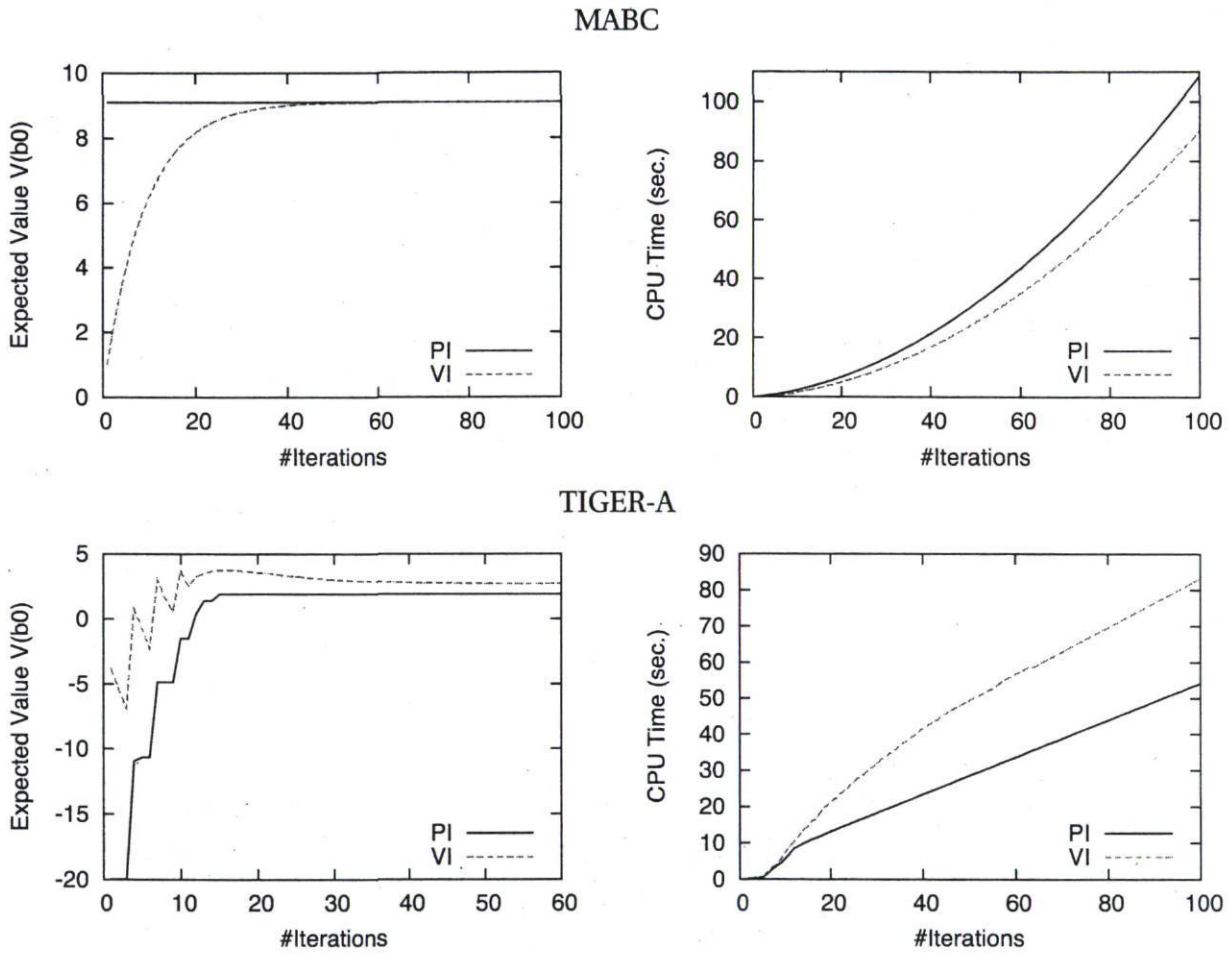


FIGURE 6.17 – Performances des algorithmes $\bar{\mathbb{L}}_B$-VI et $\bar{\mathbb{L}}_B$-PI (1)

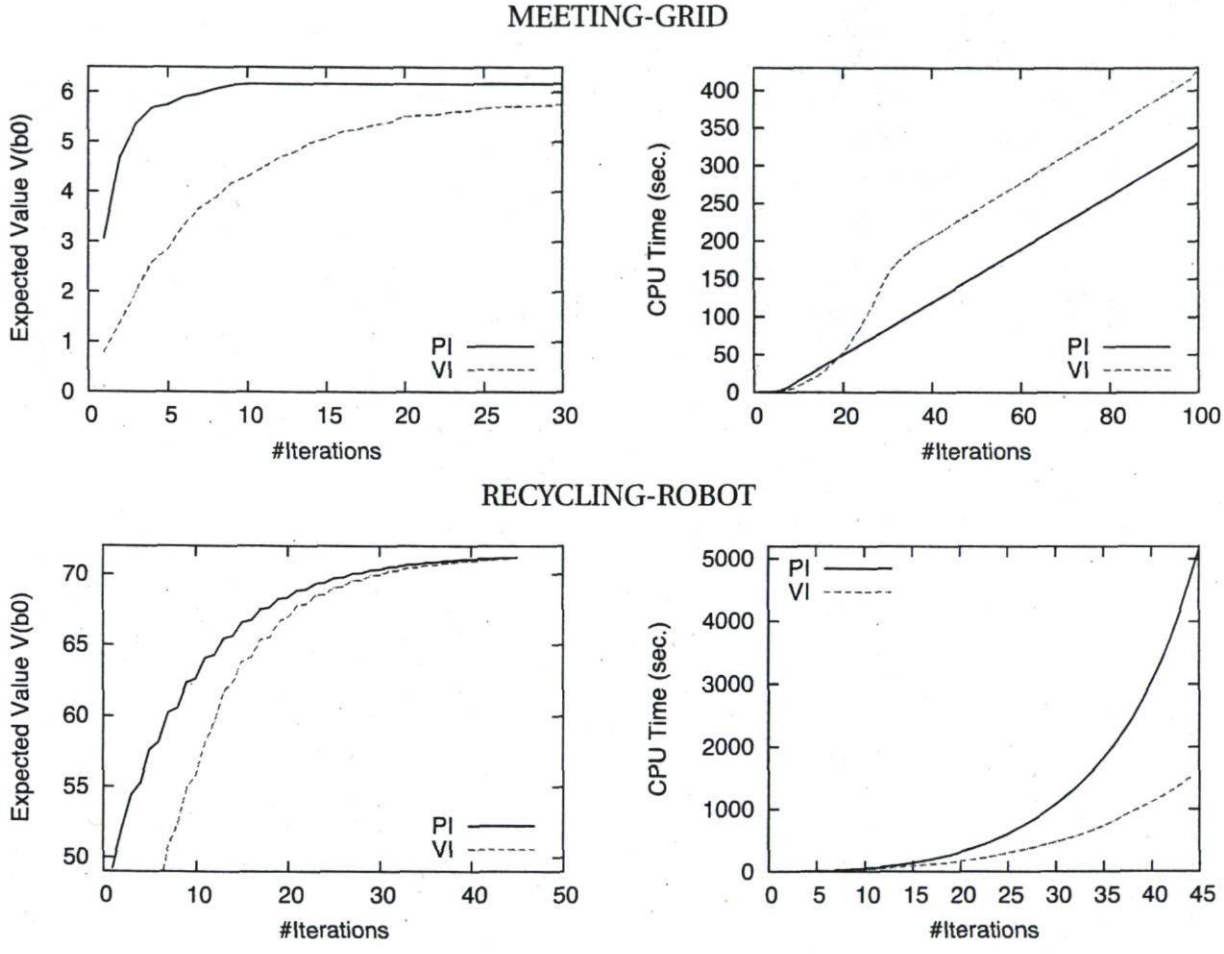


FIGURE 6.18 – Performances des algorithmes $\bar{\mathbb{L}}_B$-VI et $\bar{\mathbb{L}}_B$-PI (2)

6.8 Conclusion

Ce chapitre offre dans le cas à horizon infini la suite de l'analyse de la théorie de la planification centralisée pour le contrôle distribué des processus décisionnels de Markov. Nous avons proposé tout comme dans le cas à horizon fini, des approches de construction de politiques conjointes ε-optimales, ou à erreurs bornées. Sous l'hypothèse de la planification centralisée, nous avons établi les équations d'optimalité ; et proposé des méthodes de résolution y compris : les algorithmes d'itération de valeurs et de politiques. Nous avons également proposé un mécanisme d'élimination de politiques conjointes inutiles. Parmi l'ensemble des méthodes proposées, l'algorithme de recherche heuristique de politiques conjointes s'avère être le meilleur. Il étend l'algorithme PBIP dans le cadre de DEC-POMDPs à horizon infini. Les expérimentations sur l'ensemble des benchmarks montrent des performances remarquables, surpassant celles des tous les autres algorithmes approximatifs. De plus, cette approche offre des garanties quant à la qualité de la politique conjointe retournée.

	$\mathbb{L}_B$-VI			$\mathbb{L}_B$-PI		
$\#i$	$v(b_0)$	$\lvert\Gamma\rvert$	CPU (s)	$v^{\delta}(b_0)$	$\lvert\Gamma\rvert$	CPU (s)
MABC - $\lvert B\rvert = 19$						
1	1.00	1	0.01	9.1	1	0.04
2	2.59	2	0.03	9.1	1	0.11
3	3.29	3	0.09	9.1	1	0.29
4	3.92	4	0.16	9.1	1	0.41
5	4.48	3	0.27	9.1	1	0.53
6	5.33	5	0.39	9.1	1	0.67
7	5.73	5	0.54	9.1	1	0.81
8	6.09	6	0.71	9.1	1	0.97
9	6.41	5	0.91	9.1	1	1.14
Tiger-A - $\lvert B\rvert = 10$						
1	-2.0	4	0.01	-19.99	1	0.03
2	-3.8	16	0.06	-19.99	9	0.09
3	-5.42	28	0.27	-19.99	25	0.23
4	-3.00	36	0.82	-10.93	64	0.68
5	-0.17	100	1.93	-10.66	169	3.49
6	-2.04	121	9.01	-10.66	289	27.5
7	-3.07	121	25.0	-4.87	256	131.8
8	-3.86	121	49.7	-4.87	256	293.7
9	-4.57	121	82.9	t.e.	t.e.	t.e.
10	-5.20	121	125.1	t.e.	t.e.	t.e.
Recycling-Robot - $\lvert B\rvert = 66$						
1	7.81	1	0.01	49.27	1	0.09
2	17.8	9	0.05	52.06	9	0.21
3	24.0	36	0.28	54.48	25	0.51
4	30.3	81	2.80	55.32	49	1.87
5	37.8	144	17.0	57.64	90	7.39
6	41.2	132	70.0	58.23	132	27.3
7	44.7	272	154.5	60.26	182	80.1
8	48.2	272	401.6	60.59	210	197.4
9	50.4	418	803.7	62.37	255	406.3
10	t.e.	t.e.	t.e.	62.49	306	751.6
Meeting-Grid - $\lvert B\rvert = 12$						
1	0.47	6	0.03	3.06	9	3.47
2	0.99	56	0.69	4.75	25	8.24
3	1.43	126	51.9	5.43	72	22.9
4	1.99	368	371.9	5.82	143	122.4
5	t.e.	t.e.	t.e.	5.92	208	668.3

TABLE 6.3 – Performances des algorithmes $\mathbb{L}_B$-VI et $\mathbb{L}_B$-PI.

Chapitre 7

Planification centralisée des DEC-MDPs

Le modèle général de contrôle distribué des processus décisionnels de Markov est un formalisme aussi expressif que complexe, pour la modélisation des problèmes de contrôle distribué, multi-agents et coopératifs. En effet, la recherche a montré que résoudre un DEC-MDP pouvait être soit NEXP-difficile soit indécidable. Dans ce contexte, toutes les tentatives de résolution des DEC-MDPs ont été axées sur des modèles plus restreints, comme discuté au Chapitre 3. Et encore plus important, les méthodes de résolution du problème général, n'ont reçu aucune attention, aussi surprenant que cela puisse paraître.

Dans ce chapitre, nous exhibons un certain nombre de propriétés, y compris celles liées à la dimension bornée de la statistique suffisante pour la planification en DEC-MDPs – en Section 7.2. En outre, nous restreignons les équations d'optimalité, Section 7.4. Nous exploitons par la suite ces propriétés afin de concevoir le premier algorithme optimal et non trivial de résolution des DEC-MDPs, Section 7.5.

7.1 Le modèle des DEC-MDPs

Définition 16. *Le modèle de contrôle distribué des processus décisionnels de Markov (DEC-MDP) est défini par un tuple* $(\mathcal{I}, S, \{A^i\}, T, \{\Omega^i\}, O, R, N)$ *où :*

- $\mathcal{I}$ *est l'ensemble des agents* $i = 1, \cdots, n$ *;*
- S *symbolise l'ensemble fini des états ;*
- A^i *représente l'ensemble fini des actions individuelles de l'agent* $i \in \mathcal{I}$*.* $A = A^1 \times \cdots \times A^n$ *est l'ensemble fini des actions conjointes à l'ensemble des agents, où* $a = (a^1, \cdots, a^n)$

est une action conjointe ;

- $T(s'|s,a)$ *est une fonction de transitions, représentant la probabilité de transiter de l'état s à l'état s' lorsque l'action conjointe a est exécutée.*
- Ω^i *définit l'ensemble fini des observations individuelles disponibles pour l'agent i.* $\Omega = \Omega^1 \times \cdots \times \Omega^n$ *est un ensemble fini des observations conjointes, où* $\omega = (\omega^1, \cdots, \omega^n)$ *est une observation conjointe ;*
- $O(\omega|s',a)$ *est une fonction d'observation, qui décrit la probabilité de percevoir l'observation conjointe* ω *sachant que l'action conjointe a a été exécutée et a abouti à l'état s' ;*
- $R(s,a)$ *est une fonction de récompenses, décrivant la récompense perçue lors de l'exécution de l'action conjointe a dans l'état s ;*
- *N est l'horizon de planification. C'est à dire le nombre de décisions à prendre.*
- L'hypothèse d'observabilité conjointe totale *signifie que «* le $|\mathcal{I}|$-tuple d'observations individuelles pris ensemble détermine totalement l'état réel du système *». Plus formellement, si* $O(\omega|s',a) > 0$ *alors* $\mathbb{P}(s'|\omega) = 1$.

Soit $\pi = (d_0, d_1, \cdots, d_N)$ une politique conjointe aléatoire et non-Markovienne, où $d_\tau : H_\tau \to \mathcal{P}(A)$ est une règle de décisions et $H_\tau = S_0 \times A_1 \times \cdots \times A_{\tau-1} \times \Omega_\tau$ est un ensemble d'historiques de paires d'observations et d'actions générées avant l'étape τ. On note $v_N^\pi(H_0)$ le total espéré des récompenses décomptées et cumulées durant les N étapes de décisions, lorsque la politique conjointe π est exécutée et que le système est dans l'ensemble des historiques H_0 à l'étape de décision initiale $\tau = 0$. Pour toute politique aléatoire et non-Markovienne, cette valeur est donnée par :

$$v_N^\pi(H_0) \equiv \mathbb{E}_\pi \left\{ \sum_{\tau=0}^{N-1} \lambda^\tau r_\tau(H_\tau, \pi_\tau) + \lambda^N r_N(H_N) \right\} \tag{7.1}$$

où $1 \leqslant \lambda < 0$ est le facteur de décompte. Ainsi, résoudre un DEC-MDP revient alors à déterminer une politique conjointe π^* dont le valeur $v_N^{\pi^*}$ est la plus élevée. En d'autres termes, nous recherchons une politique conjointe aléatoire et non-Markovienne π^* telle que :

1. Si $N < \infty$,

$$v_N^{\pi^*}(H_0) \leqslant v_N^\pi(H_0) \tag{7.2}$$

2. Si $N = \infty$, par passage à la limite nous obtenons

$$v^{\pi^*}(H_0) \leqslant v^\pi(H_0) \tag{7.3}$$

7.2 Statistique suffisante

Cette section établit la principale propriété des DEC-MDPs, en opposition aux DEC-POMDPs. En particulier, nous définissons les conditions pour lesquelles les observations conjointes constituent la statistique suffisante pour la planification exacte en DEC-MDPs. Du point des applications réelles, qui motivent ce travail, nous trouvons confortable qu'en restreignant notre attention aux observations conjointes $\omega \in \Omega$, simplifiant l'implémentation et la recherche des politiques conjointes, nous puissions atteindre des politiques de valeurs aussi grandes que si nous utilisions les ensembles d'historiques H_τ, à chaque étapes de décisions $\tau = 0, 1, \cdots, N$.

Lemme 7. *Pour tout DEC-MDP* $(\mathscr{I}, S, \{A^i\}, T, \{\Omega^i\}, O, R, N)$*, il existe une application injective* $\varphi : \Omega \to S$*, qui à toute observation conjointe associe un état.*

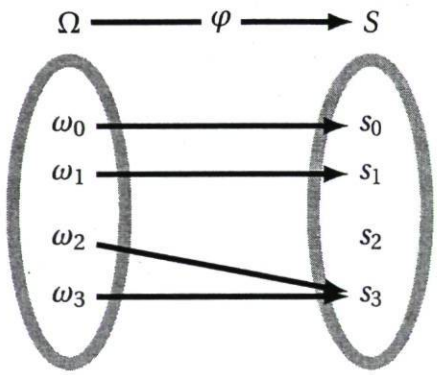


FIGURE 7.1 – Application injective φ pour les observations conjointes $\Omega = \{\omega_0, \omega_1, \omega_2, \omega_2\}$ et les états $S = \{s_0, s_1, s_2, s_3\}$.

Démonstration. Soit $\varphi^{-1} : S \to \Omega \cup \{\emptyset\}$ une application qui à tout état associe un élément dans $\Omega \cup \{\emptyset\}$ où $\emptyset$ représente une observation conjointe non définie. Cette application est donnée par :

$$\varphi^{-1}(s) = \begin{cases} \omega & \text{if } \sum_{a \in A} O(\omega | s, a) > 0, \\ \emptyset & \text{autrement.} \end{cases} \tag{7.4}$$

D'après l'hypothèse d'observabilité conjointe totale, chaque état s est associé à au plus une observation. Par définition, l'application φ^{-1} est alors injective. Par conséquent, $(\varphi^{-1})^{-1} : \Omega \cup \{\emptyset\} \to S$ est surjective. Donc l'application $\varphi : \Omega \to S$ donnée par $\varphi(\omega) = (\varphi^{-1})^{-1}(\omega)$, est une application injective. □

Nous sommes maintenant prêt à énoncer le résultat principal de la planification centralisée en DEC-MDPs.

Théorème 30. *L'ensemble des distributions de probabilités $\mathscr{P}(\Omega)$, sur l'ensemble des observations conjointes, constitue une statistique suffisante pour la planification centralisée exacte en DEC-MDPs.*

Démonstration. Durant la simulation centralisée d'une politique conjointe quelconque, à chaque réalisation d'un historique $h_\tau = \langle s_0, a_0, \omega_1, a_1, \cdots, \omega_\tau \rangle$ correspond une séquence $\langle s_0, a_0, \varphi(\omega_1), a_1, \cdots, \varphi(\omega_\tau) \rangle$. En d'autres termes, la simulation centralisée de l'historique h_τ révèle une séquence d'états et d'actions $\langle s_0, a_0, s_1, a_1, \cdots, s_\tau \rangle$. En effet, l'état $s_\tau = \varphi(\omega_\tau)$ est révélé pour toute observation conjointe ω_τ par l'hypothèse d'observabilité conjointe totale, et cela à chaque étape de décision $\tau = 0, 1, \cdots, N$. Ainsi, l'état courant s_τ dépend de l'historique h_τ seulement au travers de l'observation conjointe courante ω_τ du système. Finalement, la nature Markovienne du modèle suffit à conclure. □

7.3 Représentation des politiques

En conséquence du Théorème 30, il est facile de montrer qu'une politique individuelle d'un DEC-MDP est une application déterministe $\pi_\tau^i : \Omega^i \to A^i$ qui à toute observation individuelle associe une action individuelle, pour tout agent $i = 1, 2, \cdots, |\mathscr{I}|$, et à chaque étape de décisions $\tau = 0, 1, \cdots, N-1$. Ainsi, une politique conjointe d'un DEC-MDP est une application déterministe $\pi_\tau : \Omega \to A$, qui à toute observation conjointe associe une action conjointe, à chaque étape de décisions $\tau = 0, 1, \cdots, N-1$. Dès lors, la valeur d'une politique conjointe π_τ partant d'une distribution de probabilités s, notée $b(\omega) \equiv \mathbb{P}(\omega)$, est donnée par :

$$v^\pi(b) = \sum_\omega b(\omega) \left\{ r_\pi(\omega) + \lambda \sum_{\bar{\omega} \in \Omega} \sum_{\bar{s} \in S} o_\pi(\bar{\omega}|\bar{s})\, p_\pi(\bar{s}|\omega) \cdot v^\pi(\bar{\omega}) \right\}, \quad \forall b \in \mathscr{P}(\Omega), \tag{7.5}$$

où

$$\begin{aligned} s &= \varphi(\omega) \\ r_\pi(\omega) &= R(s, \pi(\omega)) \\ p_\pi(\bar{s}|\omega) &= T(\bar{s}|s, \pi(\omega)) \\ o_\pi(\bar{\omega}|\bar{s}) &= O(\bar{\omega}|\bar{s}, \pi(\omega)) \end{aligned}$$

Donc, v^π satisfait l'équation

$$v = r_\pi + \lambda P_\pi v \tag{7.6}$$

Pour chaque fonction de valeurs $v \in V$, nous définissons une transformation linéaire $\mathbb{H}_\pi$ par :

$$\mathbb{H}_\pi v \equiv r_\pi + \lambda P_\pi v \tag{7.7}$$

Sous cette notation, l'équation (7.10) devient :

$$v^{\pi} = \mathbb{H}_{\pi} v^{\pi} \tag{7.8}$$

Cela signifie que la fonction de valeurs v^{π} est un point fixe pour l'application $\mathbb{H}_{\pi}$. Nous rappelons que les théorèmes des points fixes constituent un outil important pour l'analyse des modèles à récompenses décomptées.

Par la suite, nous appelons politique d'un DEC-MDP, une ensemble de politiques $\Pi = \{\pi^b \mid b \in B\}$, où B est tout ou partie de l'ensemble $\mathscr{P}(\Omega)$. Il est possible de montrer que l'ensemble Π est fini lorsque $N < \infty$, et potentiellement infini lorsque $N = \infty$. Cela est notamment possible en s'inspirant de la théorie de planification en POMDPs. En effet, dans le cas fini, le nombre $|B|$ de distributions de probabilités accessibles partant d'une distribution initiale est fini. Il est alors possible d'associer à chaque distribution b_τ une politique π_τ, d'où la dimension fini de l'ensemble des politiques Π. Cette discussion s'applique bien entendu à la fonction de valeurs d'un DEC-MDP. En effet, elle correspond à l'ensemble $v^* \equiv \{v^{\pi} \mid \pi \in \Pi^*\}$. Comme l'ensemble Π^* est fini dans le cas $N < \infty$, la fonction de valeurs v^* est également finie dans le cas $N < \infty$. Un raisonnement similaire permet de conclure dans le cas infini.

7.4 L'équation d'optimalité

Dans cette section, nous introduisons l'équation d'optimalité pour la planification centralisée des DEC-MDPs. Nous montrons, en outre, que les solutions de cette équation sont les fonctions de valeurs optimales.

Pour toute fonction de valeurs $v_\tau \in V$, l'équation d'optimalité est donnée par :

$$v_{\tau+1} \equiv \max_{\pi} \{r_\pi + \lambda P_\pi v_\tau\} \tag{7.9}$$

Le passage à la limite de l'équation (7.9) suggère que l'équation suivante caractérise aussi bien les valeurs que les politiques conjointes optimales dans le cas $N = \infty$:

$$v \equiv \max_{\pi} \{r_\pi + \lambda P_\pi v\} \tag{7.10}$$

Pour toute fonction de valeurs $v \in V$, nous définissons une application (non linéaire) $\mathbb{H}$ sur l'ensemble V par :

$$\mathbb{H}v \equiv \max_{\pi} \{r_\pi + \lambda P_\pi v\} \tag{7.11}$$

$$= v \tag{7.12}$$

Nous pouvons maintenant appliquer le théorème de point fixe de Banach afin d'établir l'existence d'une solution à l'équation d'optimalité.

Lemme 8. *Supposons* $0 \leqslant \lambda < 1$ *; il vient alors que l'application* $\mathbb{H}$ *est un opérateur de contraction sur* V.

Démonstration. Comme Ω est un ensemble fini, $\mathbb{H}$ associe à toute fonction de valeurs dans V une fonction de valeurs dans V. Soit u et v deux fonctions de valeurs dans V, supposons $\mathbb{H}v \geqslant \mathbb{H}u$, et soit

$$\pi^* \equiv \arg\max_\pi \{r_\pi + \lambda P_\pi v\} \tag{7.13}$$

Alors,

$$\begin{aligned} 0 \leqslant \mathbb{H}v - \mathbb{H}u &\leqslant \lambda P_{\pi^*}(v-u) && (7.14)\\ &\leqslant \lambda P_{\pi^*}\|v-u\| && (7.15)\\ &\leqslant \lambda\|v-u\| && (7.16) \end{aligned}$$

En répétant cet argument dans le cas $\mathbb{H}u \geqslant \mathbb{H}v$, cela implique que

$$|\mathbb{H}v - \mathbb{H}u| \leqslant \lambda\|v-u\| \tag{7.17}$$

□

En conséquence du Lemme 8, il existe une fonction de valeurs $v^* \in V$ satisfaisant $\mathbb{H}v^* = v^*$ et une politique conjointe π^* est optimale si et seulement si $v^{\pi^*} = v^*$.

7.5 Algorithme d'itération de politiques

L'algorithme d'itération de politiques illustré Algorithme 31 est une méthode générale de programmation dynamique pour la résolution des processus décisionnels de Markov. L'algorithme d'itération de politiques pour le contrôle décentralisé des processus décisionnels de Markov avec récompenses décomptées suit.

A noter que la politique conjointe $\hat{\Pi}$ n'est pas nécessairement unique, d'où la spécification $\hat{\Pi} = \Pi$ si possible, afin d'éviter les cycles. Cet algorithme d'itération de politiques diffère dans son implémentation de l'opérateur max dans l'équation (7.19) qui est sous forme matricielle contrairement au cas mono agent. Cela est essentiellement dû à la contrainte

Algorithme 31 Algorithme d'itération de politiques (DEC-MDP).

procedure PI
  Sélectionner une politique conjointe arbitraire Π.
  répéter
    (**Évaluation de politique**) Obtenir v en résolvant

$$v^{\pi} \leftarrow (\mathbf{1} - \lambda P_{\pi})^{-1} r_{\pi} \tag{7.18}$$

    (**Amélioration de politique**) Choisir $\hat{\pi}$ satisfaisant

$$\hat{\pi} \in \arg\max_{\pi}(r_{\pi} + \lambda P_{\pi} v) \tag{7.19}$$

    Poser $\hat{\Pi} \leftarrow \Pi$ si possible.
  jusqu'à $\hat{\Pi} = \Pi$
  Retourner $\Pi^* \leftarrow \Pi$.
fin procedure

de décentralisation du contrôle. Cela signifie que nous devons évaluer toutes les politiques conjointes possibles π ; puis choisir la meilleure d'entre elles $\hat{\pi}$, pour chaque distribution $\mathbb{P}(\omega)$ possible. Malheureusement, le nombre $\otimes_{i=1}^{n} |A^i|^{|\Omega^i|}$ de toutes les politiques conjointes possibles est exponentiel en fonction du nombre d'observations et d'agents. Néanmoins, lorsque ce nombre n'est pas trop prohibitif, l'algorithme d'itération de politiques peut-être utilisé. Afin de résoudre des problèmes de dimensions supérieures, il faudra remplacer l'opérateur d'énumération exhaustive de toutes les politiques possibles par une opérateur de recherche heuristique dans l'espace de toutes les politiques possibles, comme nous avons discuter dans le cadre des DEC-POMDPs.

7.6 Conclusion

Nous avons tenté de montrer l'intérêt de l'analyse théorique de la planification centralisée pour le contrôle distribué débutée aux Chapitres 5 et 6. L'application de cette analyse au cadre des DEC-MDPs, offre des perspectives jusqu'ici inespérées. En effet, la difficulté majeur d'application des modèles de contrôle distribué (DEC-POMDPs, DEC-MDPs, ...) réside essentiellement dans quantité exorbitante des ressources qu'elles requièrent. En particulier, les politiques conjointes en DEC-POMDPs sont définies au mieux dans l'espace des politiques conjointes déterministes et non-Markoviennes ; la statistiques suffisante est de dimension variable, dont la croissance est exponentielle. A la différence, nous avons établit que dans le cadre des DEC-MDPs, l'hypothèse d'observabilité conjointe totale simplifie considérablement le problème. D'abord, la statistique suffisante pour la pla-

nification centralisée est l'ensemble des distributions de probabilités sur les observations conjointes, de dimension fixe ; de plus, les politiques conjointes sont définies sur l'espace des politiques pures (stationnaires et déterministe) ; enfin la fonction de valeurs associe une seule valeur pour chaque observation conjointe. Toutes ces propriétés ont permis la définition d'un opérateur de mise à jour d'une fonction de valeurs arbitrairement choisie, propice à la construction d'algorithme d'itération de politiques, ou valeurs, et bien d'autre encore. Encore plus important, ces propriétés permettent le transfert de presque la totalité des travaux réalisés dans le cadre des MDPs au cadre des DEC-MDPs, y compris les approches de recherche heuristique par chaînage avant ; de compression de l'espace des observations ; d'apprentissage des modèles ; .... Les perspectives du modèle général des DEC-MDPs sont bien plus prometteuses que celles que nous pourrions envisager dans le cadre des DEC-POMDPs. L'application de ce modèle sur un problème réel n'est certainement plus qu'une question de temps.

Chapitre 8

Conclusion

DANS cette thèse, nous avons abordé les problèmes de prise de décisions séquentielles. Nous avons exposé les modèles adoptés par la communauté des chercheurs en Intelligence Artificielle, en particulier les problèmes de contrôle centralisé ou distribué des processus décisionnels de Markov complètement observables (MDPs / DEC-MDPs), les processus décisionnels de Markov partiellement observables (POMDPs / DEC-POMDPs).

8.1 Planification topologique pour le contrôle centralisé

Le contrôle centralisé des processus décisionnels de Markov bénéficie d'une riche littérature comprenant une théorie complète, ainsi que des algorithmes exacts et approximatifs analysés en profondeur. Néanmoins, force est de constater que face à des problèmes de dimensions considérables, cette théorie et ces algorithmes sont souvent mis à défaut. En particulier, les algorithmes de programmation dynamique visant à calculer une succession d'approximations de la fonctions de valeurs optimales ou de la politique optimale, font systématiquement usage d'un opérateur dit de mises à jour. La complexité de résolution des problèmes de contrôle centralisé des processus décisionnels de Markov est ainsi fonction du nombre de fois que l'opérateur de mises à jour est utilisé. Afin d'améliorer significativement le temps de résolution de tels problèmes, l'un des moyens consiste à trouver une méthode utilisant le moins souvent l'opérateur de mises à jour tout en parvenant à calculer la politique désirée. C'est dans ce contexte que s'inscrit notre première contribution (Chapitre 4), permettant d'exploiter la structure topologique des problèmes afin de réduire le nombre d'application de l'opérateur de mises à jour.

8.1.1 Processus décisionnels de Markov

Dans le cadre des MDPs, nous avons proposé un algorithme de programmation dynamique prioritisé et nommé *i*TVI. Cet algorithme se propose d'exploiter des relations causales dans les domaines non acycliques, cela permet en particulier de faire face aux limites de l'algorithme TVI. En exploitant les dépendances causales entre états à l'intérieur d'un sous-ensemble d'états non acyliques, *i*TVI parvient à accroître empiriquement le taux de convergence de TVI, dans l'ensemble des domaines testés.

8.1.2 Processus décisionnels de Markov partiellement observables

Nous avons identifié les propriétés structurelles permettant de caractériser un POMDP, comme étant un problème topologique. Dans ce contexte, nous avons tout d'abord décrit un algorithme exact, permettant d'exploiter astucieusement cette structure particulière, afin de fournir une solution ε-optimale. Bien souvent le passage à l'échelle des approches optimales ou ε-optimales est limité. Pour cette raison, nous avons proposé un deuxième algorithme, approximatif nommé TOP, permettant de faire face à des domaines de dimensions plus conséquentes.

L'algorithme TOP se distingue des approches génériques de résolution des POMDPs, par sa capacité à identifier la structure plus ou moins topologique d'un problème. TOP procède par la suite, suivant un ordre topologique préétablis en construisant des séquences d'états et de croyances ordonnées. Cette organisation permet à TOP de choisir la croyance qu'il convient de mettre à jour et surtout à le moment où il faut la mettre à jour. En sélectionnant l'ordre et le moment où les croyances les plus probablement accessibles sont mises à jour, TOP parvient à réduire significativement le nombre de mises à jour gratuites. Cela à pour conséquence d'accélérer la convergence de l'algorithme surpassant ainsi l'ensemble des algorithmes génériques sur l'ensemble des benchmarks topologiques testés.

Cette approche à été appliquée avec succès dans le cadre des MDPs, puis des POMDPs. Elle peut s'appliquer également dans le cadre des DEC-POMDPs. En effet, dans le cadre de la planification centralisée, un DEC-POMDP peut être perçu comme un POMDP. Ainsi, les propriétés exploitables en POMDPs, peuvent également l'être en DEC-POMDPs, en particulier les propriétés structurelles telles que la topologie sur les états du système ou les agents.

8.2 Planification centralisée pour le contrôle distribué

Le contrôle distribué des processus décisionnels de Markov (DEC-POMDPs) est un champ de recherche récent, il a été formellement introduit par [Bernstein et al., 2000] il y a tout juste une dizaine d'années. Cependant, très vite la communauté des chercheurs en planification à reconnu la grande expressivité de ce modèle. L'analyse de complexité des DEC-POMDPs a révélé qu'il s'agit d'un problème dont la résolution est soit NEXP- difficile dans le cas horizon fini, soit indécidable dans le cas infini. Ce résultat a orienté la recherche vers l'identification de sous-classes de moindre complexité.

Nous avons proposé dans un premier temps une analyse rigoureusement de la planification centralisée pour le contrôle distribué. Nous avons défini les critères d'optimalité ; les équations d'optimalité ; la statistique suffisante pour la résolution centralisée des problèmes de contrôle distribué des processus décisionnels de Markov. Ces résultats tiennent pour l'ensemble des modèles de contrôle distribué des processus de Markov, à savoir DEC-POMDPs ; DEC-MDPs ; ND-POMDPs ; . . .

Nous avons en outre proposé un algorithme exact de planification centralisée pour le contrôle distribué. Cet algorithme recherche les règles de décisions maximales à chaque horizon de planification. Comme la dimensions de la statistique suffisante selon laquelle la sélection des règles de décisions se fait croît de façon exponentielle tout comme le nombre de règle de décisions à considérer, cette approche ne sert que de cadre pour la conception d'approche approximatif à erreur bornée.

Nous avons par la suite introduit plusieurs approches approximatives, à mémoire limitée. Parmi ces approches, l'algorithme PBIP à montré les meilleures performances. Cet algorithme sélectionne un ensemble de croyances sur lesquelles il base sa mise à jour de la politique conjointe. A chaque horizon, et pour chaque croyance, PBIP procède à une recherche heuristique dans l'espace de toutes les politiques conjointes possibles afin d'en extraire celle qui maximise la croyance. Cette recherche heuristique permet d'éviter l'opération de génération exhaustive de l'ensemble des politiques conjointes comme préconiser par l'ensemble des approches jusqu'ici.

Les tests expérimentaux réalisés sur l'ensemble des benchmarks montre que PBIP est le meilleur algorithme approximatif de résolution des DEC-POMDPs.

8.3 Travaux à venir

8.3.1 Planification topologique

Bien que les algorithmes topologiques aient démontré d'excellentes performances, leur applicabilité reste restreinte aux problèmes structurés, en particulier ceux disposant d'une structure topologique. Cette propriété n'est malheureusement pas partagé par l'ensemble des problèmes de prise de décisions séquentielles.

Il existe de récents travaux dans la littérature des processus décisionnels de Markov tendant à prouver l'intérêt de cette propriété même lorsque celle-ci n'existe pas. [Dai et al., 2009] montre qu'il est possible de construire des politiques de bonnes qualités sur des domaines sans aucune structure topologique. Pour y parvenir, certains états voient contraint à un plus petit nombre d'actions possibles, le domaine qui en résulte peut alors être à structure topologique quand bien même il n'en était rien auparavant. En appliquant une technique similaire, dans le cadre des processus décisionnels de Markov partiellement observables, il nous est possible de calculer des politiques exactes ou approximatives sur des problèmes topologiques extraits du problème initial.

En outre, cette approche peut s'appliquer dans le cadre de la résolution des problèmes de contrôle distribué des processus décisionnels de Markov partiellement observables. Tout comme dans le cadre du contrôle centralisé des processus décisionnels de Markov, les problèmes rencontrés lors du contrôle distribué sont parfois munis d'une structure topologique. Si tel est le cas, nous pouvons l'exploiter dans le cadre d'un algorithme de planification centralisé pour le contrôle distribué des processus décisionnels de Markov, comme le en discuterons au chapitre suivant. Cette perspective offre l'opportunité de résoudre des problèmes de dimensions encore plus grande.

8.3.2 Planification centralisée pour le contrôle distribué

L'approche de construction heuristique proposée dans le cadre de l'algorithme PBIP est applicable bien au-delà du cadre des DEC-POMDPs. En effet, le principale problème auquel cet approche fait face est celui de la mise à jour de la fonction de valeurs courante. Cette problématique est identique dans de nombreux problèmes de contrôle distribué. C'est notamment le cas en théorie des jeux, dans le cadre des jeux stochastiques à récompenses identiques. Il est possible de montrer que l'algorithme PBIP offre une solution optimale à tout problème de jeux bayésiens à récompenses identiques.

Au regard de l'analyse formelle réalisée aux Chapitres 5 et 6, il apparaît que la résolution des DEC-POMDPs est impraticable pour des applications réelles. Fort heureusement, l'analyse de sa sous-classe, DEC-MDP, indique un potentiel de passage à l'échelle non négligeable. Dans ce contexte, nous envisageons de développer des algorithmes exacts et approximatifs pour la résolution centralisée de DEC-MDPs. En particulier, nous comptons transférer la théorie de la décision développer dans le cadre des POMDPs et MDPs, pour la résolution centralisée des DEC-MDPs.

Afin d'augmenter la capacité de nos algorithmes à passer à l'échelle, nous comptons également exploiter les différentes dépendances relationnelles entre agents. Le modèle ND-POMDP offre un cadre propice à l'analyse de notre approche de planification centralisée pour le contrôle distribué dans ce cadre. En exploitant ces dépendances, nous pourrons augmenter considérablement le nombre d'agents interagissant durant le contrôle du processus.

8.4 Résumé

La majeure partie des applications réelles sont de dimensions si grandes qu'ils sont bien souvent difficile à résoudre de façons optimale. Cependant, bon nombre de ces applications disposent de propriétés structurelles sous-jacentes. En définissant des algorithmes qui permettent d'exploiter ces propriétés, il est possible de produire des solutions approximatives de très bonnes qualités.

Cette thèse offre un large éventail de stratégies d'exploitation des propriétés structurelles des modèles Markoviens, sous différents angles y compris : les propriétés topologiques ; les propriétés de la planification centralisées ; la propriété de distributivité ; ou encore l'hypothèse d'observabilité conjointe totale. Toutes ces propriétés offrent de puissants outils de résolution exacte ou approximative des modèles Markoviens. Conjointement, ces techniques constituent un arsenal de choix pour la planification et l'apprentissage centralisée afin de contrôler de façon centralisée ou distribuée des systèmes de très grandes dimensions.

Bibliographie

Mohammed Abbad and Hatim Boustique. A decomposition algorithm for limiting average Markov Decision problems. *Oper. Res. Lett.*, 31(3) :473–476, 2003.

Pieter Abbeel, Varun Ganapathi, and Andrew Y. Ng. Learning vehicular dynamics, with application to modeling helicopters. In *NIPS*, 2005.

Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. In *NIPS*, pages 1–8, 2006.

Philip Agre and David Chapman. Pengi : An implementation of a theory of activity. *In Proceedings of the AAAI National Conference*, 1987.

J.S. Albus. A new approach to manipulation control : The cerebelar model articulation controller. *Transaction of ASME, J. Dynamics Systems, Measurement and Control.*, pages 220–227, 1985.

Eitan Altman. Applications of Markov Decision Processes in communicating networks : A survey. Technical Report RR-3984, INRIA, 2000.

Eitan Altman. Constrained Markov Decision Processes. Technical Report RR-2574, INRIA, 1998.

Christopher Amato, Daniel S. Bernstein, and Shlomo Zilberstein. Optimizing memory-bounded controllers for decentralized POMDPs. In *UAI*, 2007.

Christopher Amato, Jilles S. Dibangoye, and Shlomo Zilberstein. Incremental policy generation for finite-horizon dec-POMDPs. *in Proceedings of the Ninetieth International Conference on Automated Planning and Scheduling*, 2009.

David Andre, Nir Friedman, and Ronald Parr. Generalized prioritized sweeping. In *Proceedings of the 10th conference on Advances in neural information Processing systems*, pages 1001–1007, Cambridge, MA, USA, 1998. MIT Press. ISBN 0-262-10076-2.

Raghav Aras, Alain Dutech, and François Charpillet. Mixed integer linear programming for exact finite-horizon planning in decentralized POMDPs. *CoRR*, 2007a.

Raghav Aras, Alain Dutech, and François Charpillet. Mixed integer linear programming for exact finite-horizon planning in decentralized POMDPs. In *ICAPS*, pages 18–25, 2007b.

A. Arnt, Shlomo Zilberstein, and J. Allen. Dynamic composition of information retrieval techniques, 2004.

K. J. Aström. Optimal control of Markov Decision Processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 10 :174–205, 1965.

Andrew G. Barto, Steven J. Bradtke, and Satinder P. Singh. Learning to act using real-time dynamic programming. Technical Report UM-CS-1993-002, University of Alberta, 1993.

J. Baxter and P. Bartlett. Direct gradient-based reinforcement learning. Technical report, Research School of Information Sciences and Engineering, Australian National University., July 1999.

Patrick Beaumont. Multi-platform coordination and resource management in command and control. Technical report, Laval University, 2004.

Raphen Becker, Shlomo Zilberstein, Victor R. Lesser, and Claudia V. Goldman. Transition-independent decentralized Markov Decision Processes. In *AAMAS*, pages 41–48, 2003.

Raphen Becker, Shlomo Zilberstein, and Victor R. Lesser. Decentralized Markov Decision Processes with event-driven interactions. In *AAMAS*, pages 302–309, 2004a.

Raphen Becker, Shlomo Zilberstein, Victor R. Lesser, and Claudia V. Goldman. Solving transition independent decentralized Markov Decision Processes. *Journal of Artificial Intelligence Research*, 22 :423–455, 2004b.

Richard E. Bellman. *Dynamic Programming*. Dover Publications, Incorporated, 1957.

Daniel S. Bernstein. *Complexity Analysis and Optimal Algorithms for Decentralized Decision Making*. PhD thesis, University of Massachusetts, Amherst, MA., 2005.

Daniel S. Bernstein, Shlomo Zilberstein, and Neil Immerman. The complexity of decentralized control of Markov Decision Processes. In *UAI*, pages 32–37, 2000.

Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of Markov Decision Processes. *Math. Oper. Res.*, 27(4), 2002.

Daniel S. Bernstein, Eric A. Hansen, and Shlomo Zilberstein. Bounded policy iteration for decentralized POMDPs. In *IJCAI*, pages 1287–1292, 2005.

Daniel S. Bernstein, Christopher Amato, Eric A. Hansen, and Shlomo Zilberstein. Policy iteration for decentralized control of Markov Decision Processes. *Journal of Artificial Intelligence Research*, 34 :89–132, 2009.

Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, 2005.

Aurélie Beynier and Abdel-Illah Mouaddib. A polynomial algorithm for decentralized Markov Decision Processes with temporal constraints. In *AAMAS*, pages 963–969, 2005.

Aurélie Beynier and Abdel-Illah Mouaddib. An iterative algorithm for solving constrained decentralized Markov Decision Processes. In *AAAI*, 2006.

D. Blackwell. Discounted dynamic programming, 1965.

Dale E. Blodgett, Michel Gendreau, François Guertin, Jean-Yves Potvin, and René Séguin. A tabu search heuristic for resource management in naval warfare. *J. Heuristics*, 9(2) : 145–169, 2003.

Avrim L. Blum and Merrick L. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90 :1636–1642, 1995.

Avrim L. Blum and John C. Langford. Probabilistic planning in the graphplan framework. In *In Proceedings of the Fifth European Conference on Planning*, pages 8–12, 1998.

Blai Bonet and Hector Geffner. Faster heuristic search algorithms for planning with uncertainty and full feedback. In *International Joint Conference Artificial Intelligence*, pages 1233–1238, 2003a.

Blai Bonet and Hector Geffner. Labeled RTDP : Improving the convergence of real-time dynamic programming, 2003b.

Blai Bonet and Hector Geffner. Solving large POMDPs by real time dynamic programming, 1998.

Abdeslam Boularias and Brahim Chaib-draa. Exact dynamic programming for decentralized POMDPs with lossless policy compression. In *ICAPS*, pages 20–27, 2008.

Craig Boutilier. Sequential optimality and coordination in multiagent systems. In *International Joint Conference Artificial Intelligence*, pages 478–485, 1999.

Craig Boutilier and Richard Dearden. Using abstractions for Decision-theoretic planning with time constraints. In *National Conference on Artificial Intelligence (AAAI)*, volume 2, pages 1016–1022, 1994.

Craig Boutilier, Richard Dearden, and Moise's Goldszmidt. Exploiting structure in policy construction. In Chris Mellish, editor, *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1104–1111, San Francisco, 1995. Morgan Kaufmann.

Craig Boutilier, Nir Friedman, Moises Goldszmidt, and Daphne Koller. Context-specific independence in Bayesian networks. In *Uncertainty in Artificial Intelligence*, pages 115–123, 1996.

Craig Boutilier, Ronen I. Brafman, and Christopher Geib. Prioritized goal decomposition of Markov Decision Processes : Towards a synthesis of classical and Decision theoretic planning. In Morgan Kaufman, editor, *International Joint Conference Artificial Intelligence*, pages 1156–1163, San Francisco, 1997.

Craig Boutilier, Ronen I. Brafman, and Christopher W. Geib. Structured reachability analysis for Markov Decision Processes. In *Conference on Uncertainty in Artificial Intelligence*, pages 24–32, 1998.

Craig Boutilier, Thomas Dean, and Steve Hanks. Decision-theoretic planning : Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11 : 1–94, 1999.

Craig Boutilier, Richard Dearden, and Moises Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1-2) :49–107, 2000.

Craig Boutilier, Kevin Regan, and Paolo Viappiani. Online feature elicitation in interactive optimization. In *ICML*, page 10, 2009.

Justin A. Boyan and Andrew W. Moore. Learning evaluation functions for large acyclic domains. In *International Conf. on Machine Learning*, pages 63–70, 1996.

Ronen I. Brafman. Reachability, relevance, resolution and the planning as satisfiability approach. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 976–981, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1-55860-613-0.

Rodney A. Brooks. Intelligence without representation. *Artif. Intell.*, 47(1-3) :139–159, 1991.

Zi-Xing Cai. *Intelligent control : principles, techniques and applications.* World Scientific, 1997.

Alan Carlin and Shlomo Zilberstein. Value-based observation compression for DEC-POMDPs. In *AAMAS*, 2008.

Anthony Cassandra, Michael L. Littman, and Nevin L. Zhang. Incremental Pruning : A simple, fast, exact method for partially observable Markov Decision Processes. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 54–61, San Francisco, CA, 1997. Morgan Kaufmann Publishers.

Fangruo Chen. Decentralized supply chains subject to information delays. *Manage. Sci.*, 45 :1076–1090, 1999.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms.* MIT Press and McGraw-Hill, Cambridge, MA, USA, 2001.

Nick Craswell and Martin Szummer. Random walks on the click graph. In *SIGIR '07 : Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 239–246, New York, NY, USA, 2007. ACM Press.

Peng Dai and Judy Goldsmith. Topological value iteration algorithm for Markov Decision Processes. In *International Joint Conference Artificial Intelligence*, pages 1860–1865, 2007.

Peng Dai, Mausam, and Daniel S. Weld. Focused topological value iteration. *in Proceedings of the Ninetieth International Conference on Automated Planning and Scheduling*, 2009.

Thomas Dean and Robert Givan. Model minimization in Markov Decision Processes. In *National Conference on Artificial Intelligence (AAAI/IAAI)*, pages 106–111, 1997.

Thomas Dean and Shieu-Hong Lin. Decomposition techniques for planning in stochastic domains. In *Proceedings of The 14th International Joint Conference Artificial Intelligence*, pages 1121–1129, 1995.

Thomas Dean, Leslie P. Kaelbling, Jak Kirman, and Ann Nicholson. Planning under time constraints in stochastic domains. *Artificial Intelligence*, 76(1–2) :35–74, july 1995.

Richard Dearden and Craig Boutilier. Abstraction and approximate Decision theoretic planning. *Artificial Intelligence 89*, pages 219–283, 1997.

Rina Dechter and Judea Pearl. Generalized best-first search strategies and the optimality of A*. *Journal of the ACM*, 32(3) :505–536, 1985.

Jilles S. Dibangoye. Topological dynamic programming algorithms for general MDPs. Technical report, Laval University, June 2007a.

Jilles S. Dibangoye. Topological dynamic programming algorithms for high-dimentional POMDPs. Technical report, Laval University, August 2007b.

Jilles S. Dibangoye, Abdell-Illah Mouaddib, and Brahim Chaib-draa. Periodic real-time resource allocation for teams of progressive Processing agents. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 768–770, Hawai, 2007.

Jilles S. Dibangoye, Brahim Chaib-draa, and Abdel-Illah Mouaddib. A novel prioritization technique for solving Markov Decision Processes. In *FLAIRS Conference*, pages 537–542, 2008a.

Jilles S. Dibangoye, Brahim Chaib-draa, and Abdel-Illah Mouaddib. Topological order based planning for solving POMDPs. *in Proceedings of the AAAI 2008 Workshop on Advancements in POMDP Solvers.*, July 2008b.

Jilles S. Dibangoye, Brahim Chaib-draa, and Abdel-Illah Mouaddib. Planification à base d'ordres topologiques pour la résolution des POMDPs. *dans les Actes des 34émes JFPDA.*, Juin 2008c.

Jilles S. Dibangoye, Abdel-Illah Mouaddib, and Brahim Chaib-draa. Incremental pruning heuristic for solving DEC-POMDPs. *in Proceedings of AAMAS 2008 Workshop on MSDM*, 2008d.

Jilles S. Dibangoye, Abdel-Illah Mouaddib, and Brahim Chaib-draa. Recherche incrémentale à base de points pour la résolution des DEC-POMDPs. *dans les Actes des quinzièmes Journées Françaises des Systèmes Multi-Agent*, 2008e.

Jilles S. Dibangoye, Brahim Chaib-draa, and Abdel-Illah Mouaddib. Policy iteration algorithms for DEC-POMDPs with discounted rewards. *in Proceedings of AAMAS 2009 Workshop on MSDM*, 2009a.

Jilles S. Dibangoye, Abdel-Illah Mouaddib, and Brahim Chaib-draa. Point-based incremental pruning heuristic for solving finite-horizon DEC-POMDPs. *in Proceedings of the Eighth International Joint Conference on Autonomous Agents and Multiagent Systems*, 2009b.

Jilles S. Dibangoye, Guy Shani, Brahim Chaib-draa, and Abdel-Illah Mouaddib. Topological order planner for POMDPs. *in Proceedings of the 21-th International Joint Conference on Artificial Intelligence*, July 2009c.

Thomas G. Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of Artificial Intelligence Research*, 13 :227–303, 2000.

Thomas G. Dietterich. The MAXQ method for hierarchical reinforcement learning. In *Proc. 15th International Conf. on Machine Learning*, pages 118–126. Morgan Kaufmann, San Francisco, CA, 1998.

Dmitri A. Dolgov and Edmund H. Durfee. Resource allocation among agents with preferences induced by factored mdps. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 297–304, 2006a.

Dmitri A. Dolgov and Edmund H. Durfee. Satisficing strategies for resource-limited policy search in dynamic environments. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, 3, pages 1325–1332, 2002.

Dmitri A. Dolgov and Edmund H. Durfee. Graphical models in local, asymmetric multi-agent Markov Decision Processes. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 956–963, July 2004a.

Dmitri A. Dolgov and Edmund H. Durfee. Optimal resource allocation and policy formulation in loosely-coupled Markov Decision Processes. In *Proceedings of The International Conference on Automated Planning and Scheduling*, pages 315–324, June 2004b.

Dmitri A. Dolgov and Edmund H. Durfee. Resource allocation among agents with mdp-induced preferences. *Journal of Artificial Intelligence Research*, 27 :505–549, December 2006b.

J. L. Doob. *Stochastic Processes.* John Wiley and Sons, Inc., New York, 1953.

M. Dorigo. *Optimization, Learning and Natural Algorithms.* PhD thesis, Politecnico di Milano., Milan, Italie., 1992.

Prashant Doshi. A framework for optimal sequential planning in multiagent settings. In *AAAI*, pages 985–986, 2004.

Rosemary Emery-Montemerlo, Geoffrey J. Gordon, Jeff G. Schneider, and Sebastian Thrun. Approximate solutions for partially observable stochastic games with common payoffs. In *AAMAS*, pages 136–143, 2004.

Kutluhan Erol, James A. Hendler, and Dana S. Nau. Htn planning : Complexity and expressivity. In *AAAI*, pages 1123–1128, 1994.

Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite Markov Decision Processes. In *National Conference on Artificial Intelligence (AAAI)*, pages 950–951, 2004.

Richard E. Fikes and Nils J. Nilsson. Strips : A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4) :189–208, 1971. doi : 10.1016/0004-3702(71)90010-5.

Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in Markov Decision Processes. *Artif. Intell.*, 147(1-2) :163–223, 2003.

Fred Glover. Tabu search — part i. *ORSA Journal on Computing*, 1(3) :190–206, 1989.

David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Kluwer Academic Publishers, Boston, MA., 1989.

Herbert Goldstein, Charles P. Poole, and John Safko. Classical mechanics, 2002.

G. Gottlob, N. Leone, and F. Scarcello. A comparison of structural CSP decomposition methods. In *Proceedings of The 18th International Joint Conference on Artificial Intelligence*, 1999.

Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research*, 19 :399–468, 2002.

Carlos Guestrin, Milos Hauskrecht, and Branislav Kveton. Solving factored mdps with continuous and discrete variables. In *Conference on Uncertainty in Artificial Intelligence*, Arlington, Virginia, 2004. AUAI Press.

B. Hamidzadeh and S. Shekhar. Dynoraii : A real-time planning algorithm. *International Journal on Artificial Intelligence Tools*, 2(1) :93–115, March 1993.

Eric A. Hansen. An improved policy iteration algorithm for partially observable mdps. In *NIPS*, 1997.

Eric A. Hansen. Solving POMDPs by searching in policy space. In *Conference on Uncertainty in Artificial Intelligence*, pages 211–219, 1998.

Eric A. Hansen and Shlomo Zilberstein. Lao* : A heuristic search algorithm that finds solutions with loops. *Artif. Intell.*, 129(1-2) :35–62, 2001.

Eric A. Hansen, Daniel S. Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, pages 709–715, 2004.

Chip Heath and Amos Tversky. Preference and belief : Ambiguity and competence in choice under uncertainty. *Journal of Risk and Uncertainty*, 4(1) :5–28, January 1991.

Martial Hebert, Chuck Thorpe, and Anthony Stentz. *Intelligent Unmanned Ground Vehicles : Autonomous Navigation Research at Carnegie Mellon.* KluwerAcademic Publishers, 1997.

Bernhard Hengst. Discovering hierarchy in reinforcement learning with hexq. In *International Conf. on Machine Learning*, pages 243–250, 2002.

Jesse Hoey, Robert St-Aubin, Alan Hu, and Craig Boutilier. Spudd : Stochastic planning using Decision diagrams. In *Conference on Uncertainty in Artificial Intelligence*, pages 279–28, San Francisco, CA, 1999. Morgan Kaufmann.

Jörg Hoffmann, Carla P. Gomes, and Bart Selman. Structure and problem hardness : Goal asymmetry and dpll proofs in sat-based planning. In *ICAPS*, pages 284–293, 2006.

John H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, 1975.

John E. Hopcroft and Robert Endre Tarjan. Dividing a graph into triconnected components. *SIAM J. Comput.*, 2(3) :135–158, 1973.

Ronald A. Howard. *Dynamic Programming and Markov Processes.* The M.I.T. Press, 1960.

Shihao Ji, Ronald Parr, Hui Li, Xuejun Liao, and Lawrence Carin. Point-based policy iteration. In *AAAI*, pages 1243–1249, 2007.

John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior.* Princeton University Press, 1944.

I. T. Jolliffe. *Principal Component Analysis.* Springer, New York, NY, USA, 2002.

Leslie P. Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101 :99–134, 1995.

Leslie P. Kaelbling, Michael Littman, and Andrew W. Moore. Reinforcement learning : A survey. *Journal of Artificial Intelligence Research*, 4 :237–285, 1996.

Michael J. Kearns, Michael L. Littman, and Satinder P. Singh. Graphical models for game theory. In *UAI*, pages 253–260, 2001.

Uwe Kiencke, Lars Nielsen, Robert Sutton, Klaus Schilling, Markos Papageorgiou, and Hajime Asama. The impact of automatic control on recent developments in transportation and vehicle systems. *Annual Reviews of Control*, 30(1) :81–89, 2006.

Kee-Eung Kim and Thomas Dean. Solving factored mdps using non-homogeneous partitions. *Artif. Intell.*, 147(1-2) :225–251, 2003.

David L. Klienman and S. Braon. Manned-vehicle system analysis by means of modern control. In *Theory Report.* Bolt Beranek and Newman Inc., 1970.

Daphne Koller and Ronald Parr. Computing factored value functions for policies in structured mdps. In *International Joint Conference Artificial Intelligence*, pages 1332–1339, 1999.

Richard E. Korf. Real-time heuristic search. *Artif. Intell.*, 42(2-3) :189–211, 1990.

Harold W. Kuhn. Extensive games and the problem of information. *Annals of Mathematics Studies*, 28, 1953.

Ailsa H. Land and Alison G. Doig. An automatic method for solving discrete programming problems. *Econometrica*, 28 :497–520, 1960.

Terran Lane and Leslie P. Kaelbling. Nearly deterministic abstractions of Markov Decision Processes. In *National Conference on Artificial Intelligence (AAAI/IAAI)*, pages 260–266, 2002.

P. Laroche, François Charpillet, and R. Schott. Decomposition of Markov Decision Processes using directed graphs, 1999.

William H. Levison. A control-theory model for human Decision-making. In *Seventh Annual Conference on Manual Control. NASA SP-281.*, page 23, 1972.

William H. Levison. Optimal control model of human response – parts i and ii. In *Automatica*, volume 6, pages 357–369, 1970.

Joseph A. Lewis and George F. Luger. A constructivist model of robot perception and performance. *In Proceedings of the Conference of the Cognitive Science Society*, 2000.

Michael L. Littman, Thomas Dean, and Leslie P. Kaelbling. On the complexity of solving Markov Decision problems. In *Conference on Uncertainty in Artificial Intelligence*, pages 394–402, 1995.

Michael L. Littman, Richard S. Sutton, and Satinder P. Singh. Predictive representations of state. In *NIPS*, pages 1555–1561, 2001.

W. S. Lovejoy. Computationally feasible bounds for partially observable Markov decison Processes. *Operations Research*, 39 :175–192, 1991.

Janusz Marecki and Milind Tambe. On opportunistic techniques for solving decentralized mdps with temporal constraints. In *AAMAS*, May 2007.

Janusz Marecki, Tapana Gupta, Pradeep Varakantham, Milind Tambe, and Makoto Yokoo. Not all agents are equal : scaling up distributed POMDPs for agent networks. In *AAMAS (1)*, pages 485–492, 2008.

M. Mataric and G. Sukhatme. Task-allocation and coordination of multiple robots for planetary exploration. In *Proc. of the International Conference on Advanced Robotics.*, 2001.

Andrew McCallum. Instance-based utile distinctions for reinforcement learning with hidden state. In *ICML*, pages 387–395, 1995.

H. Brendan McMahan and Geoffrey J. Gordon. Fast exact planning in Markov Decision Processes. In *The International Conference on Automated Planning and Scheduling.*, pages 151–160, 2005.

H. Brendan McMahan, Maxim Likhachev, and Geoffrey J. Gordon. Bounded real-time dynamic programming : RTDP with monotone upper bounds and performance guarantees. In *International Conf. on Machine Learning*, pages 569–576, 2005.

Nicolas Meuleau, Milos Hauskrecht, Kee-Eung Kim, Leonid Peshkin, Leslie Kaelbling, Thomas Dean, and Craig Boutilier. Solving very large weakly coupled Markov Decision Processes. In *National Conference on Artificial Intelligence (AAAI/IAAI)*, pages 165–172, 1998.

A. Meystel. Cognitive controller for autonomous systems. *IEEE Workshop on Intelligence Control.*, page 222, 1986.

Andrew W. Moore and Christopher G. Atkeson. Prioritized sweeping : Reinforcement learning with less data and less time. *Machine Learning*, 13 :103–130, 1993.

Abdel-Illah Mouaddib and Shlomo Zilberstein. Knowledge-based anytime computation. In *IJCAI*, pages 775–783, 1995.

Abdel-Illah Mouaddib and Shlomo Zilberstein. Optimal scheduling of dynamic progressive Processing. In *ECAI*, pages 499–503, 1998.

Abdel-Illah Mouaddib, Matthieu Boussard, and Maroua Bouzid. Towards a formal framework for multi-objective multiagent planning. In *AAMAS*, page 123, 2007.

Ranjit Nair, Milind Tambe, Makoto Yokoo, David V. Pynadath, and Stacy Marsella. Taming decentralized POMDPs : Towards efficient policy computation for multiagent settings. In *IJCAI*, pages 705–711, 2003.

Ranjit Nair, Pradeep Varakantham, Milind Tambe, and Makoto Yokoo. Networked distributed POMDPs : A synthesis of distributed constraint optimization and POMDPs. In *National Conference on Artificial Intelligence (AAAI)*, pages 133–139, 2005.

John Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 1(36) :48–49, 1950.

Dana S. Nau, Vipin Kumar, and Laveen Kanal. General branch and bound, and its relation to A* and AO*. *Artif. Intell.*, 23(1) :29–58, 1984. ISSN 0004-3702. doi : http://dx.doi.org/10.1016/0004-3702(84)90004-3.

Andrew Y. Ng and Michael Jordan. PEGASUS :A policy search method for large MDPs and POMDPs. In *Conference on Uncertainty in Artificial Intelligence*, pages 406–415, 2000.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations : theory and application to reward shaping. In *Proc. 16th International Conf. on Machine Learning*, pages 278–287. Morgan Kaufmann, San Francisco, CA, 1999.

Paul M. Nicola, Paul Morris, and Nicola Muscettola. Dynamic control of plans with temporal uncertainty. In *In IJCAI*, pages 494–502, 2001.

Nils J. Nilsson. Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research*, 1 :139–158, 1994.

Frans A. Oliehoek, Matthijs T. J. Spaan, and Nikos Vlassis. Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research*, 32 :289–353, 2008.

Frans A. Oliehoek, Shimon Whiteson, and Matthijs T J Spaan. Lossless clustering of histories in decentralized POMDPs. In *Proceedings of the Eighth International Joint Conference on Autonomous Agents and Multiagent Systems*, Budapest, Hungary, 2009.

James B. Orlin. The complexity of dynamic/periodic languages and optimization problems. Technical report, Massachusetts Institute of Technology (MIT), Sloan School of Management, April 1985.

Y.-H. Pao. Some views on analytic and artificial intelligence approaches. *IEEE Workshop on Intelligence Control.*, page 29, 1986.

Sébastien Paquet, Ludovic Tobin, and Brahim Chaib-draa. An online POMDP algorithm for complex multiagent environments. In *AAMAS*, pages 970–977, 2005.

Ron Parr. Flexible decomposition algorithms for weakly coupled Markov Decision problems. In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence*, pages 422–43, 1998.

Judea Pearl. *Heuristics : intelligent search strategies for computer problem solving.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1984.

Jing Peng and Ronald J. Williams. Efficient learning and planning within the dyna framework. *Adapt. Behav.*, 1(4) :437–454, 1993. ISSN 1059-7123.

Jing Peng and Ronald J. Williams. Incremental multi-step Q-learning. In *ICML*, pages 226–232, 1994.

Leonid Peshkin, Kee-Eung Kim, Nicolas Meuleau, and Leslie P. Kaelbling. Learning to cooperate via policy search. In *UAI*, pages 489–496, 2000.

Nils J. Nilsson Peter E. Hart and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics SSC*, 2 :100–107, 1968.

Joelle Pineau. *Tractable Planning Under Uncertainty : Exploiting Structure.* PhD thesis, Robotic Institute. Carnegie Mellon University., Pittsburgh., May 2004.

Joelle Pineau, Geoffrey J. Gordon, and Sebastian Thrun. Policy-contingent abstraction for robust robot control. In *Proceedings of UAI*, pages 477–484, 2003a.

Joelle Pineau, Geoffrey J. Gordon, and Sebastian Thrun. Point-based value iteration : An anytime algorithm for POMDPs. In *IJCAI*, 2003b.

Pascal Poupart and Craig Boutilier. Vdcbpi : an approximate scalable algorithm for large POMDPs. In *Conference on Advances in neural information Processing systems*, 2004.

Warren Buckler Powell. *Approximate Dynamic Programming : Solving the curse of dimensionality.* not published yet, july 2006.

Doina Precup, Richard S. Sutton, and Satinder P. Singh. Theoretical results on reinforcement learning with temporally abstract options. In *European Conference on Machine Learning*, pages 382–393, 1998.

Martin L. Putterman. *Markov Decision Processes : Discrete Stochastic Dynamic Programming.* John Wiley and Sons, New York, NY, 1994.

David V. Pynadath and Milind Tambe. Multiagent teamwork : analyzing the optimality and complexity of key theories and models. In *AAMAS*, pages 873–880, 2002.

Zinovi Rabinovich, Claudia V. Goldman, and Jeffrey S. Rosenschein. The complexity of multiagent systems : the price of silence. In *AAMAS*, pages 1102–1103, 2003.

Balaraman Ravindran and Andrew G. Barto. Smdp homomorphisms : An algebraic approach to abstraction in semi-Markov Decision Processes. In *International Joint Conference Artificial Intelligence*, pages 1011–1018, 2003.

Matthew Rosencrantz, Geoffrey J. Gordon, and Sebastian Thrun. Learning low dimensional predictive representations. In *ICML*, 2004.

Stéphane Ross and Brahim Chaib-draa. Aems : An anytime online search algorithm for approximate policy refinement in large POMDPs. In *IJCAI*, pages 2592–2598, 2007.

Nicholas Roy, Geoffrey J. Gordon, and Sebastian Thrun. Finding approximate POMDP solutions through belief compression. *Journal of Artificial Intelligence Research*, 23 :1–40, 2005.

Matthew R. Rudary and Satinder P. Singh. Predictive linear-gaussian models of controlled stochastic dynamical systems. In *International Conf. on Machine Learning*, pages 777–784, 2006.

Stuart Russell and Peter Norvig. *Artificial Intelligence : A Modern Approach.* Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition, 2003.

George N. Saridis. On the revised theory of intelligent machines. *IEEE/RSJ International Workshop on Intelligent Robots and Systems.*, pages 24–30, 1989.

George N. Saridis. Fondations of intelligent controls. *IEEE Workshop on Intelligence Control.*, page 23, 1986.

Bruno Scherrer. Asynchronous neurocomputing for optimal control and reinforcement learning with large state spaces. *Neurocomputing*, 63 :229–251, 2005.

Sven Seuken and Shlomo Zilberstein. Formal models and algorithms for decentralized Decision making under uncertainty. *JAAMAS*, 17(2) :190–250, 2008.

Sven Seuken and Shlomo Zilberstein. Improved Memory-Bounded Dynamic Programming for DEC-POMDPs. In *UAI*, 2007a.

Sven Seuken and Shlomo Zilberstein. Memory-bounded dynamic programming for DEC-POMDPs. In *IJCAI*, pages 2009–2015, 2007b.

Guy Shani, Ronen I. Brafman, and Solomon E. Shimony. Prioritizing point-based POMDP solvers. In *ECML*, pages 389–400, 2006.

Guy Shani, Ronen I. Brafman, and Solomon E. Shimony. Forward search value iteration for POMDPs. In *International Joint Conference Artificial Intelligence*, pages 2619–2624, 2007.

Guy Shani, Pascal Poupart, Ronen I. Brafman, and Solomon E. Shimony. Efficient add operations for point-based algorithms. In *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling*, pages 330–337, 2008.

Jennie Si, Andrew G. Barto, Warren Buckler Powell, and Don Wunsch. *Handbook of Learning and Approximate Dynamic Programming*. Wiley-IEEE Press, july 2004.

Satinder Singh and David Cohn. How to dynamically merge Markov Decision Processes. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. The MIT Press, 1998.

Satinder P. Singh, Michael L. Littman, Nicholas K. Jong, David Pardoe, and Peter Stone. Learning predictive state representations. In *ICML*, pages 712–719, 2003.

R. D. Smallwood and E. J. Sondik. The optimal control of partially observable Markov Decision Processes over a finite horizon. *Operations Research*, 21(5) :1071–1088, 1973.

Trey Smith and Reid Simmons. Heuristic search value iteration for POMDPs. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 520–527, Arlington, Virginia, United States, 2004.

Trey Smith and Reid G. Simmons. Focused real-time dynamic programming for MDPs : Squeezing more out of a heuristic. In *National Conference on Artificial Intelligence (AAAI)*, 2006.

Trey Smith and Reid G. Simmons. Point-based POMDP algorithms : Improved analysis and implementation. In *Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*, pages 542–547, 2005.

E. J. Sondik. The optimal control of partially observable Markov Decision Processes. Technical report, Standford, 1971.

E. J. Sondik. The optimal control of partially observable Markov Decision Processes over the infinite horizon : Discounted cost. *Operations Research*, 12 :282–304, 1978.

Matthijs T. J. Spaan and Nikos Vlassis. Perseus : Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24 :195–220, 2005.

Richard S. Sutton. Planning by incremental dynamic programming. In *Machine Learning*, pages 353–357, 1991.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning – An Introduction.* The MIT Press, 1998.

Richard S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. Technical report, ATT Labs– Research, 1999.

Daniel Szer. *Contribution à la résolution des Processus de décision Markoviens décentralisés.* PhD thesis, Université Henri Poincaré Nancy 1., Nancy, France., Decembre 2006.

Daniel Szer and François Charpillet. An optimal best-first search algorithm for solving infinite horizon DEC-POMDPs. In *ECML*, pages 389–399, 2005.

Daniel Szer and François Charpillet. Point-based dynamic programming for DEC-POMDPs. In *AAAI*, pages 16–20, July 2006.

Daniel Szer, François Charpillet, and Shlomo Zilberstein. MAA* : A heuristic search algorithm for solving decentralized POMDPs. In *UAI*, pages 568–576, 2005.

Hao Tang and Elise Miller-Hooks. Solving a generalized traveling salesperson problem with stochastic customers. *Comput. Oper. Res.*, 34(7) :1963–1987, 2007. ISSN 0305-0548.

Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2), June 1972.

Vincent Thomas, Christine Bourjot, and Vincent Chevrier. Interac-dec-mdp : Towards the use of interactions in dec-mdp. In *AAMAS*, pages 1450–1451, 2004.

Julius T. Tou and King S. Fu. *U.S.-Japan Seminar on Learning Control and Intelligent Control.* 1973.

Amos Tversky and Daniel Kahneman. Judgment under uncertainty : Heuristics and biases. *Science*, 185 :1124–31, 1974.

Pradeep Varakantham, Janusz Marecki, Yuichi Yabu, Milind Tambe, and Makoto Yokoo. Letting loose a spider on a network of POMDPs : generating quality guaranteed policies. In *AAMAS*, page 218, 2007.

Yan Virin, Guy Shani, Solomon E. Shimony, and Ronen I. Brafman. Scaling up : Solving POMDPs through value based clustering. In *National Conference on Artificial Intelligence (AAAI)*, 2007.

B. Walliser. Deux modes d'émergence. *Hors série Siences & Avenir – L'énigme de l'Emergence*, 143, 2000.

T. Walsh. Stochastic constraint programming, 2002.

Christopher Watkins and Peter Dayan. Q-learning. *Ninth Asian Computing Science Conference (ASIAN'04)*, 2004.

Christopher Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4) :279–292, 1992.

Gerhard Weiss. *Multiagent systems : a modern approach to distributed artificial intelligence.* MIT Press, 2000.

David Wingate and Kevin D. Seppi. Prioritization methods for accelerating mdp solvers. *J. Mach. Learn. Res.*, 6 :851–881, 2005. ISSN 1533-7928.

David Wingate and Satinder P. Singh. Mixtures of predictive linear gaussian models for nonlinear, stochastic dynamical systems. In *National Conference on Artificial Intelligence (AAAI)*, 2006.

Weihong Zhang and Nevin Lianwen Zhang. Value iteration working with belief subset. In *National Conference on Artificial Intelligence (AAAI/IAAI)*, pages 307–, 2002.

Shlomo Zilberstein and Abdel-Illah Mouaddib. Optimizing resource utilization in planetary rovers. In *Proceedings of the Second NASA International Workshop on Planning and Scheduling for Space*, pages 163–168, San Francisco, California, 2000.

Shlomo Zilberstein, Richard Washington, Daniel S. Bernstein, and Abdel-Illah Mouaddib. Decision-theoretic control of planetary rovers. In *Advances in Plan-Based Control of Robotic Agents*, pages 270–289, 2001.

Shlomo Zilberstein, Richard Washington, Daniel S. Bernstein, and Abdel-Illah Mouaddib. Decision-theoretic control of planetary rovers. In *Advances in Plan-Based Control of Robotic Agents.*, pages 270–289, London, UK, 2002. Springer-Verlag.