Utilisation de croyances heuristiques pour la planification multi-agent dans le cadre des Dec-POMDP

Gabriel Corona

Nancy-Université
Université Henri Poincaré

Département de formation doctorale en informatique **École doctorale IAEM Lorraine**
Sciences et Technologies

Utilisation de croyances heuristiques pour la planification multi-agent dans le cadre des Dec-POMDP

THÈSE

présentée et soutenue publiquement le 11 avril 2011

pour l'obtention du

Doctorat de l'université Henri Poincaré – Nancy 1

(spécialité informatique)

par

Gabriel Corona

Composition du jury

Rapporteurs :	Abdel-Illah Mouaddib, Professeur à l'Université de Caen Basse-Normandie Brahim Chaib-Draa, Professeur à l'Université de Laval
Examinateurs :	Shlomo Zilberstein, Professeur à l'Université du Massachusetts Amherst Nadine Piat, Professeur à l'ENSMM René Mandiau, Professeur de l'Université de Valenciennes René Schott, Professeur à l'Université Henri-Poincarré
Directeur de thèse :	François Charpillet, Directeur de recherche INRIA

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Loria

Mis en page avec la classe thloria.

Table des matières

Table des figures

Liste des tableaux

Liste des Algorithmes

Chapitre 1

Introduction

Sommaire

Nous nous intéressons dans cette thèse à la décision séquentielle dans les systèmes multi-agents. Ces agents sont typiquement des entités robotiques ou logicielles. Ils sont placés dans un environnement dynamique avec lequel ils peuvent interagir : chaque agent a une vision locale ou imparfaite de son environnement et peut agir localement pour influencer son évolution. Cette problématique a des applications dans de nombreux domaines tel que le routage de paquets dans un réseau, le partage d'un canal de communication, la coordination d'une équipe de robots, le contrôle de réseaux de capteurs, etc.

Dans ce chapitre, nous présentons succinctement le domaine de l'intelligence artificielle et la décision séquentielle. Nous explicitons ensuite la notion d'agent, une entité capable d'interagir avec son environnement et de prendre des décisions. Enfin, nous présentons l'organisation de la thèse en évoquant les éléments importants de l'état de l'art ainsi que les contributions de la thèse.

1.1 Intelligence artificielle

Le problème de la prise de décision séquentielle dans l'incertain se place dans le cadre général de l'intelligence artificielle dont un des objectifs principaux est de comprendre comment on peut concevoir des systèmes intelligents artificiels.

Une branche de l'intelligence artificielle, s'est d'abord intéressée à la création d'entités capables de raisonnements intelligents, en particulier par le biais du raisonnement symbolique et des logiques formelles (*brain in a box*, [Brooks et Stein, 1994]). Cette approche a conduit notamment à la création des systèmes experts à base de règle, des systèmes de preuve de théorèmes, de la planification déterministe par manipulation de symboles, etc.

Une autre branche plus récente de l'intelligence artificielle s'intéresse à une intelligence dite située, cherchant à créer directement des systèmes qui agissent de manière intelligente *sans nécessairement penser de manière intelligente*. L'intelligence de ces « cerveaux incarnés » (*brain-for-a-body*, [Brooks et Stein, 1994]) est plus liée à l'adéquation de leur comportement vis à vis de leur environnement, aux résultats de leurs actions qu'à leur capacité de raisonnement. Ceci nous amène à expliciter la notion d'agent, une entité située dans un environnement et qui interagit avec elle par le biais d'actions et de perceptions.

1.2 Agent

Un agent est une entité, physique ou virtuelle, qui est capable d'interagir avec un environnement grâce à une boucle sensori-motrice (figure 1.1) : ses actions influencent l'évolution de l'environnement ; ses perceptions lui fournissent en rétroaction une information sur l'évolution de l'environnement qui lui permet de décider des actions ultérieures. Un agent peut être une machine, un robot, une personne, un animal, une personne morale, un composant logiciel, etc. Un exemple typique d'agent est un robot doté d'un certain nombre de capteurs qui lui fournissent des informations (observations) sur son environnement et d'un certain nombre d'effecteurs qui permettent d'agir sur cet environnement.

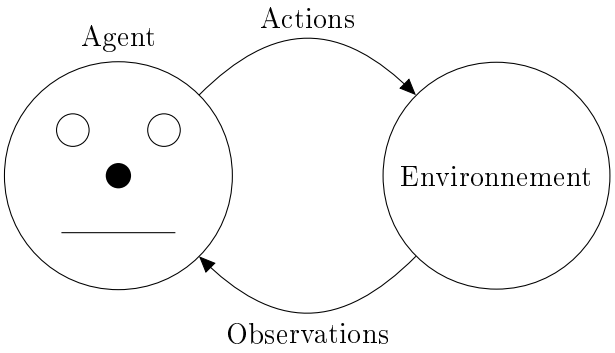


FIGURE 1.1 – Système mono-agent

Tout au long de cette thèse, nous illustrons notre propos avec un exemple dans lequel des pompiers doivent lutter contre la propagation d'un incendie. Cet exemple est inspiré du problème multi-agent des pompiers [Oliehoek *et al.*, 2008].

Exemple 1.2.1*:* Considérons le problème d'un agent pompier (humain ou robotique) devant lutter contre la propagation d'un incendie dans une rangée de maisons. L'incendie se propage entre

maisons voisines et à l'intérieur d'une même maison. Quand le pompier est dans une maison donnée, il peut tenter de limiter l'ampleur de l'incendie dans cette maison. Cependant, le pompier ne peut être que dans une seule maison à la fois. Le pompier se déplace de maison en maison pour limiter la propagation de l'incendie et, si possible, l'éteindre en minimisant les dégâts causés. Le problème pour le pompier consiste à décider à chaque instant dans quelle maison son action aura la meilleure influence probable (espérée) sur l'évolution ultérieure, à long terme, de l'incendie.

1.2.1 Prise de décision séquentielle

La décision séquentielle a pour but de déterminer quelles actions les agents doivent entreprendre pour atteindre un but donné, en particulier lorsque les effets des actions sont non déterministes ou incertaines. Son étude a des applications dans un grand nombre de disciplines telles que la robotique, l'informatique l'automatique, la télécommunications, l'économie, l'éthologie (comportement des animaux et en particulier apprentissage), la psychologie et la sociologie (idem mais pour les humains), etc. Dans le cas particulier, où l'on connaît un modèle du processus dans lequel l'agent évolue, il s'agit d'un problème de planification dans l'incertain.

L'agent peut être confronté au problème de la planification dans l'incertain. Il s'agit d'élaborer des plans d'action dont les effets ne sont pas déterministes afin d'atteindre un objectif : ce problème peut être un état à atteindre, un état à éviter, un critère de performance à optimiser, etc. À chaque instant t, le système est dans un état s_t donné (figure 1.2). L'agent doit alors choisir une action a_{t+1} parmi l'ensemble des actions qu'il peut effectuer. Le système réagit alors en passant dans un nouvel état s_{t+1}.

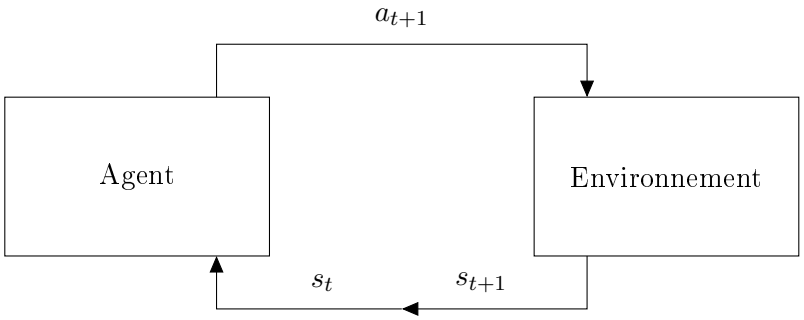


FIGURE 1.2 – Système mono-agent avec observabilité totale

De manière générale, résoudre un problème complexe demande de prévoir les conséquences à long terme des actions, de planifier un « chemin », c'est-à-dire une séquence d'actions, vers une solution. À un instant donné, l'agent est confronté au choix d'une action. Ce choix influe sur l'évolution de l'environnement et donc sur les choix ultérieurs de l'agent : après avoir effectué son action, l'agent se trouve devant un nouveau choix. Le problème de décision séquentielle consiste à déterminer quelle est la meilleur séquence de décision qui permette à un agent de réaliser au mieux la tâche qui lui est dévolue. La difficulté du problème est que ces choix ne sont pas indépendants.

Dans le cas simple des problèmes déterministes tel que les tours de Hanoï ou le *Rubic's cube*, l'agent doit d'exécuter une séquence d'actions élémentaires qui modifient la configuration générale (l'état) du problème pour le conduire d'un état initial à un l'état solution. Cependant, dans de nombreux problèmes que nous voudrions résoudre, il n'est pas possible de prédire avec exactitude le résultat des actions : celles-ci peuvent avoir un effet non déterministe. L'agent ne

peut donc pas se contenter de planifier une séquence prédéterminée d'actions, comme il pourrait le faire pour les problèmes simples des tours de Hanoï et le *Rubic's cube*. Il doit prévoir des plans d'actions qui prennent en compte l'évolution non déterminisme de l'environnement lorsqu'il agit sur celui-ci.

De plus, dans de nombreux cas, l'agent ne dispose pas d'une connaissance parfaite de l'état courant du problème mais uniquement d'observations partielles ou bruitées : l'agent doit donc estimer l'état courant du système. Pour cela, il peut utiliser la dynamique du système, les actions qu'il a effectué par le passé et les observations qu'il a reçu.

Nous sommes quotidiennement confrontés à ce genre de problèmes de décision séquentielle : nous résolvons ces problèmes en choisissant à chaque instant l'option qui nous semble la meilleure en nous basant sur notre connaissance imparfaite de l'environnement et en tenant compte de son évolution probable.

1.2.2 Objectif du problème

1.2.2.1 Critère

L'objectif du ou des agents peut être représenté par un critère de performance à optimiser. Ce peut être l'espérance du temps pris pour atteindre un état but fixé, la probabilité d'atteindre un état, l'espérance d'une quantité à optimiser, etc.

Exemple 1.2.2*:* Revenons sur l'exemple du pompier. Pour une même dynamique, différents objectifs peuvent être envisagés. Dans l'exemple du pompier, on peut utiliser une mesure pour quantifier les dégâts causés par l'incendie. Le pompier cherche à minimiser l'espérance de cette mesure de dégâts. D'autres critères plus évolués pourraient être utilisés. Par exemple, on pourrait chercher à maximiser la probabilité que les dégâts soient inférieurs à un seuil donné.

1.2.2.2 Récompenses

Les théories psychologiques et physiologiques du conditionnement peuvent servir d'inspiration à la création de systèmes capables d'apprendre un comportement. La théorie du conditionnement opérant s'intéresse à la manière dont les conséquences du comportement d'organismes (animaux, humaines) influent sur la reproduction ou non de ce comportement. Ces conséquences peuvent être de deux formes : les renforcements rendent plus probable l'émission du comportement et les punitions les rendent moins probables. Ainsi l'agent tend, par apprentissage, à adopter un comportement qui lui permet de maximiser les renforcements et minimiser les punitions. Le signal de récompense est une indication donnée à l'agent sur l'intérêt immédiat de l'action qu'il a choisi d'effectuer vis a vis de la tache à effectuer.

L'apprentissage par renforcement [Sutton et Barto, 1998, Bertsekas et Tsitsiklis, 2006, Watkins, 1989, Rummery et Niranjan, 1994] cherche à reproduire ce comportement pour des agents artificiels (figure 1.3) : l'agent reçoit un signal de récompense r_{t+1} qui dépend de l'évolution du système et donc de ses actions a_{t+1}. On peut alors concevoir des algorithmes d'apprentissage pour l'agent qui vont privilégier dans une situation donnée les actions qui maximisent les récompenses ultérieures. Résoudre un problème dans ce contexte revient à calculer la meilleure séquence d'action au sens d'un critère qui prend en compte l'espérance mathématique des récompense que va revoir l'agent tout au long de son évolution.

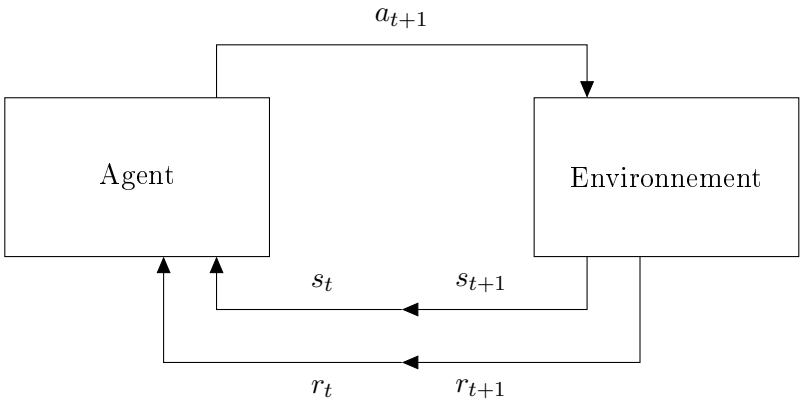


FIGURE 1.3 – Système mono-agent avec observabilité totale et récompenses

De manière plus générale, ce modèle à base de récompenses permet de spécifier l'objectif d'un agent : on cherche un comportement qui va maximiser les récompenses reçues par l'agent. Ces systèmes de récompenses permettent de formaliser les objectifs de nombreux problèmes, aussi bien pour l'apprentissage que pour la planification.

Exemple 1.2.3*:* Revenons sur le problème du pompier pour présenter un objectif sous forme de récompenses. À chaque instant, le pompier reçoit par exemple une récompense négative (une pénalité) proportionnelle à l'intensité de l'incendie dans chaque maison.

1.2.3 Problème

Le domaine de la prise de décision séquentielle cherche à déterminer une *politique* de l'agent, qui définit son comportement, afin de réaliser au mieux une tâche donnée. On peut distinguer deux cas :

- si on a un modèle précis de la dynamique du système, on peut calculer *a priori* le comportement de l'agent et on peut formuler le problème comme un problème de *planification* ;
- si on ne dispose pas d'un tel modèle, l'agent va devoir *apprendre* un comportement au fur et à mesure qu'il interagit avec l'environnement.

Nous nous intéressons dans cette thèse au problème de la planification qui constitue déjà un problème très difficile dans le cas décentralisé.

L'introduction de performance permet de comparer la qualité de plusieurs politiques et de définir une notion d'optimalité : une politique est optimale quand elle est au moins aussi bonne que toute autre politique. Le but de la planification est de déterminer une politique optimale (cas exacte) ou le plus proches possible de l'optimal (cas approché).

1.2.4 Observations

1.2.4.1 Observabilité totale

Certains problèmes sont à observabilité totale : l'agent connaît avec exactitude l'état courant du système. L'état du système permet à l'agent de prévoir l'évolution future du système et constitue ce qu'on appelle un signal markovien. Si le modèle du système est connu, l'agent peut

fonder sa décision sur l'état courant du système.

Ce type d'approche est largement étudié et utilisé dans de nombreuses applications telles que l'approvisionnement des repas pour les avions de ligne [Goto *et al.*, 2001], la gestion des flux de gaz dans un complexe sidérurgique [Geist, 2009], le jeu de Tetris [Dutech *et al.*, 2008, Gordon, 1999], la présentation et la résolution intelligente des incidents pour les avions [Carlin *et al.*, 2010], la gestion de ressources naturelles [Remon *et al.*, 2010], etc. Ce cadre est d'ailleurs très proche de la théorie du contrôle optimal et les méthodes pour résoudre ces deux types de problèmes sont largement similaires [Todorov, 2006].

Exemple 1.2.4*:* Nous revenons sur le problème du pompier pour expliquer la notion d'observabilité totale. Dans l'hypothèse de l'observabilité totale, le pompier peut connaître à chaque instant l'état général de l'incendie. Il sait quelles maisons sont en feu et quelle est l'intensité de l'incendie dans ces maisons. Son problème de décision est donc : « Étant donné l'état courant de l'incendie, dans quelle maison dois-je me rendre afin de lutter au mieux contre la propagation de l'incenvie ? »

1.2.4.2 Observabilité partielle

Cependant de manière générale, l'agent ne connaît pas avec exactitude l'état courant du système. Si on considère un robot devant effectuer une tâche, ses capteurs (caméra, capteurs de distance, GPS, etc.) ne lui fournissent en général qu'une perception limitée et bruitée de son environnement proche.

L'agent n'est pas capable de connaître l'état global du système mais peut, au mieux, émettre des hypothèses sur celui-ci en utilisant ses observations o_t (figure 1.4). En particulier, s'il dispose d'un modèle probabiliste du système et de ses capteurs, il peut déterminer une distribution de probabilité sur l'état du système par inférence bayésienne : cette distribution de probabilité, qui représente la croyance de l'agent sur l'état du système, peut être utilisée pour décider de l'action à effectuer [Smallwood et Sondik, 1973].

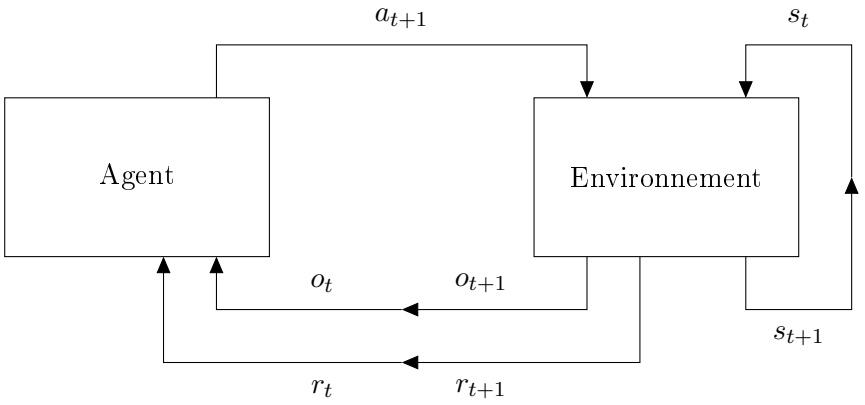


FIGURE 1.4 – Système mono-agent avec observabilité partielle

Ce type de problème trouve de nombreuses applications [Cassandra, 2008] comme la maintenance de machine dans des contextes industriels, l'inspection de structures (routes, ponts, etc) pour détecter leur détérioration, le contrôle d'ascenseurs, le contrôle de robots autonomes, etc.

Exemple 1.2.5*:* Nous revenons sur le problème du pompier pour illustrer la notion d'observabilité partielle. Sosu hypothèse d'observabilité partielle, le pompier ne connaît pas à chaque instant l'état global de l'incendie ce qui complique sa tâche. Par exemple, le pompier peut ne percevoir que l'incendie dans la maison dans laquelle il se trouve (perception locale) : il peut estimer l'état de l'incendie dans les autres maisons en utilisant toutes les observations passées. Ses observations peuvent de plus être bruitées et l'agent doit alors aussi estimer l'état de la maison dans laquelle il se trouve.

1.2.5 État interne

Dans le cadre des problèmes à observabilité partielle, l'agent ne perçoit pas à chaque instant toute l'information sur le système. Afin de prendre une bonne décision, il peut maintenir un état interne z_t qui représente une mémoire sur le passé du processus. Quand l'agent reçoit une observation o_t (voir figure 1.5), il met à jour cet état interne z_t et ce dernier influence l'émission des actions ultérieures de l'agent (a_{t+1}, ...).

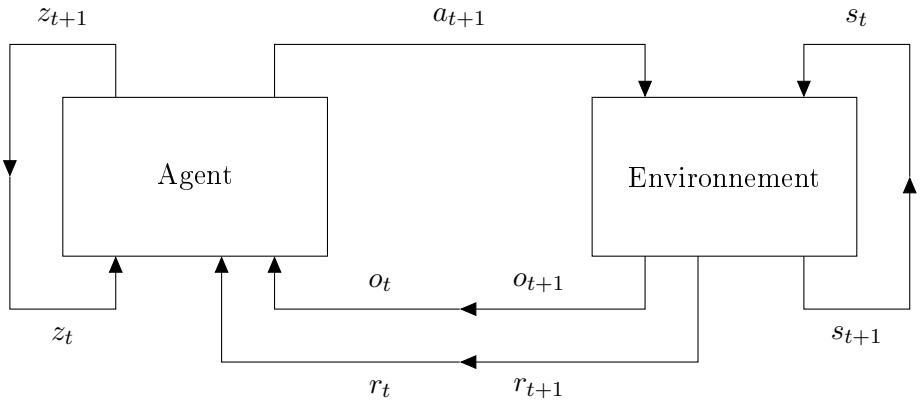


FIGURE 1.5 – Système mono-agent avec état interne de l'agent

Comme nous l'avons vu, dans le cas particulier des observations totales, l'agent n'a pas besoin de conserver un état interne. Dans le cas général des observations partielles, l'agent peut maintenir une mémoire parfaite en retenant l'historique complet constitué des observations et action passées. Cependant, l'agent peut alternativement maintenir une distribution de probabilité sur les états du système qui représente sa croyance sur les états qui prend en compte les actions qu'il a effectuées par le passé ainsi que les observations qu'il a perçu. Cette croyance permet de prendre la décision optimale à chaque instant et constitue aussi une information suffisante pour la prise de décision de l'agent qui est généralement plus pratique à manipuler que l'historique complet.

D'autres types d'états internes peuvent être utilisés : par exemple, l'état interne de l'agent peut être un état d'un automate à états finis. Ce type d'approches est notamment utilisé quand on cherche une politique de l'agent parmi une classe donnée ou lorsque l'on a besoin de raisonner explicitement sur l'état de l'agent. Cette approche est plus particulièrement utilisée dans le cadre multi-agent où, comme nous le verrons par la suite, prendre la croyance comme état interne pose problème.

1.2.6 Décision décentralisée

De nombreux systèmes sont décentralisés et ne sont pas contrôlés par une entité unique ayant une vue globale du problème mais par une équipe d'entités (agents) indépendantes ayant chacune une vue locale et partielle du problème. Nous nous intéressons dans cette thèse au cas particulier des systèmes coopératifs dans lesquels les agents ne sont pas en compétition mais coopèrent pour réaliser un objectif commun de façon coordonnée.

Ces systèmes peuvent être *décentralisés par nature* : des entités locales peuvent influencer sur son évolution mais il n'est pas possible d'introduire un contrôleur centralisé qui récolterait toutes les informations disponibles pour prendre la meilleure action jointe.

Exemple 1.2.6*:* Nous considérons ici un problème naturellement décentralisé. Considérons le problème du partage d'un canal d'accès [Ooi et Wornell, 1996]. Plusieurs ordinateurs sont reliés par un câble à accès partagé : afin de pouvoir communiquer un message sur ce lien, un seul ordinateur doit parler en même temps ; si plusieurs ordinateurs envoient un message simultanément sur le lien, les différents messages entrent en collision et ne peuvent pas être reçus. Chaque ordinateur reçoit de temps en temps des paquets à transmette sur le lien qu'il place dans un *buffer*. Les ordinateurs doivent trouver un moyen de se coordonner afin de maximiser le débit (nombre de message envoyés en un temps donné) sur le canal tout en évitant les collisions de messages. Pour cela, les ordinateurs ne peuvent pas communiquer directement puisque tout l'objet du problème est de gérer le lien qui les relie. La coordination se fait donc de manière complètement décentralisée.

Ils peuvent aussi être *décentralisés par conception* pour de nombreuses raisons : l'acheminement, l'analyse et la mise en correspondance des informations peut être trop complexe ; un système sans décision centralisée peut être plus robuste aux pannes ; les différents composants peuvent être développés indépendamment et être ajoutés dynamiquement au système (typiquement des agents logiciels qui interagissent à travers le réseau) [Poslad *et al.*, , Bellifemine *et al.*, 2001].

Chaque agent reçoit ses propres observations et a donc une connaissance de l'environnement qui lui est propre et utilise cette informations pour décider d'actions locales (figure 1.6). Un problème crucial dans ce type de systèmes est la difficulté de coordination entre les agents qui résulte du manque d'une vision globale du système partagée par l'ensemble des agents : chaque agent a sa propre vision de l'état du système basée sur ses propres observations et cette vision peut être fortement différente de celle des autres agents ; afin d'estimer l'état courant du système et de prévoir son évolution futur, un agent aurait besoin de prendre en compte les actions des autres agents et donc d'estimer et prévoir leurs états internes.

Exemple 1.2.7*:* Cette fois, plusieurs pompiers coopèrent pour lutter contre l'incendie. Ils doivent réussir à coordonner leurs actions avec des moyens de communications limités. Ils peuvent essayer d'aller dans des maisons différentes pour éviter la propagation de l'incendie ou se réunir pour un incendie difficile à éteindre. Comme ils ne partagent pas les même observations, chaque pompier a sa propre vue de l'incendie : un pompier donné a donc une incertitude à la fois sur l'état courant du système mais aussi sur l'état de ses collègues et donc de leurs actions à venir.

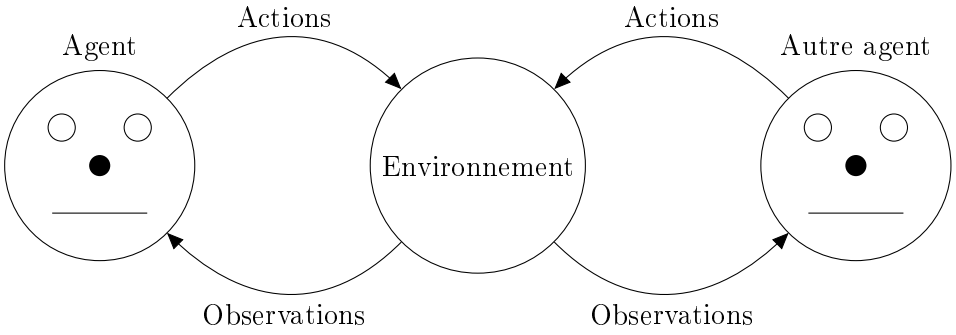


FIGURE 1.6 – Système multi-agent

La croyance d'un agent n'est plus uniquement sur l'état du système mais aussi sur la croyance des autres agents. Cependant cette croyance des autres agents inclut la croyance de l'agent considéré. Ainsi, l'utilisation de la croyance des agents comme état interne dans le cas multi-agent pose problème car il nécessite un raisonnement récursif infini de type « je crois que mon collègue croit que je crois que ... ».

Dans cette thèse, nous nous intéressons au cas le plus général des problèmes à observations partielles décentralisées car un grand nombre d'applications appartiennent naturellement à cette classe de problème.

1.2.7 Architecture de l'agent

1.2.7.1 Agent cognitif

Un agent cognitif est un agent qui dispose de capacités de raisonnement et utilise un processus délibératif. Il utilise une représentation de son environnement pour raisonner sur ses objectifs. Un exemple typique sont les architectures BDI (*Belief, Desire, Intention* - Croyance, Désir, Intention) [Rao et Georgeff, 1991] où l'agent raisonne explicitement sur sa connaissance, potentiellement erronée, de l'environnement (les croyances), ses objectifs (désirs) et son plan d'actions pour réaliser ces derniers (intentions). Ces différents étaux mentaux sont typiquement représentés sous forme symbolique et manipulés avec une logique modale.

Dans le cadre multi-agent, une approche de type cognitive consiste à créer des systèmes de communication entre les agents. Les agents peuvent raisonner sur ce que croient les autres agents, sur leurs buts et leurs plans respectifs et en déduire des communications entre les agents au moyen de langages de représentations, de communications et de protocoles associés. Les systèmes basés sur les langages de communication d'agents de la FIPA (*Foundation for Intelligent Physical Agents* – Fondation pour des agents physiques intelligents) constituent un exemple de ce type d'approche [Poslad *et al.*, , Bellifemine *et al.*, 2001].

1.2.7.2 Agent réactif

Une autre approche s'oppose à la construction de l'agent cognitif. Cette autre approche cherche à concevoir une « intelligence sans représentation » [Brooks, 1991] explicite de l'environnement avec des agents réactifs ayant un comportement de type stimulus-réaction. Cette approche se fonde sur l'idée qu'une architecture simple d'un agent peut produire un comportement complexe, résultat de l'interaction entre l'agent et son environnement : l'agent n'a pas besoin d'utiliser explicitement un modèle du monde mais peut « *utiliser le monde comme son*

propre modèle » [Brooks, 1991]. Ainsi Brooks propose-t-il une construction incrémentale des agents par agrégation de comportements simples, de même que l'intelligence humaine et animale a été construite incrémentalement par l'évolution par l'interaction de composantes simples.

L'observation de systèmes complexes et en particulier de systèmes biologiques, montre que des comportements complexes peuvent émerger de l'interaction entre un environnement et un grand nombre d'entités au comportement simple. On voit un comportement collectif complexe émerger de comportements individuels simples où l'intelligence « en essaim » n'est pas située dans les individus mais résulte de l'interaction des individus et de l'environnement. Ceci amène à l'approche des systèmes multi-agent réactifs où chaque agent a un comportement simple.

Exemple 1.2.8*:* Les colonies de fourmis tendent à emprunter de préférence un chemin qui minimise la distance entre les sources de nourriture et la fourmilière. Ce phénomène peut être expliqué par un comportement individuel simple des fourmis, sans faire l'hypothèse qu'elles construisent une représentation interne complexe de l'environnement. Les fourmis se déplacent initialement de manière aléatoire. Quand elles trouvent de la nourriture, elles retournent à la fourmilière [1] en déposant sur leur chemin une phéromone, formant ainsi un sentier. Quand d'autres fourmis rencontrent ce sentier de phéromones, elles vont avoir tendance à l'emprunter préférentiellement et ainsi renforcer le chemin en revenant à la fourmilière. Les chemins les plus courts vers une source de nourriture ont tendance à avoir une densité de phéromones plus élevée parce qu'ils sont parcourus plus souvent. Ce comportement a inspiré la famille des algorithmes par colonie de fourmis [Colorni *et al.*, 1991, Dorigo et Caro, 1999] : ce sont des algorithmes heuristique de recherche de plus court chemin dans un graphe qui simulent le comportement d'une colonie de fourmis se déplaçant dans le graphe à la recherche des nœuds buts.

Dans cette thèse, nous nous plaçons plutôt dans le cadre réactif : dans le cadre multi-agent le plus général, quand les communications sont très restreintes, il est difficile de maintenir une représentation interne du système qui comprend une représentation de l'état interne des autres agents. Nous cherchons à créer des agents simples qui n'utilisent pas une représentation explicite de l'environnement. Toute la complexité a lieu à la conception et consiste à concevoir *a priori* un comportement simple pour les agents.

1.3 Organisation de la thèse

Dans cette thèse, nous nous intéressons au cadre de la prise de décision séquentielle dans l'incertain avec observabilité partielle. Nous considérons les problèmes décentralisés car ils possèdent un domaine applicatif important. Ces problèmes sont modélisés par le formalisme Dec-POMDP [Bernstein *et al.*, 2000].

Nous ne nous intéressons pas dans cette thèse au problème de l'*apprentissage* multi-agent mais au cas plus simple de la *planification* multi-agent : nous cherchons à concevoir des agents simples représentés par un contrôleur à états finis ou plus particulièrement dans le cas des problèmes à horizon fini par des arbres de politique. La complexité à l'exécution est faible mais la complexité se trouve dans l'étape se planification. et la complexité se trouve donc au moment de la planification. Cependant la planification en horizon infini dans le cas multi-agent est au mieux NEXP-complet [Bernstein *et al.*, 2000] (alors qu'elle est PSPACE-difficile pour le cas mono-agent

1. Une autre phéromone permet de retrouver le chemin de la fourmilière.

[Mundhenk *et al.*, 2000]) et nous nous concentrons donc sur la planification approchée. Nous nous demandons comment utiliser au mieux une information heuristique sur le problème, représentée sous la forme d'une distribution de probabilité sur les croyances, pour guider la recherche approchée d'une politique décentralisée. Nous argumentons que pour une planification approchée de problèmes de contrôle décentralisée, on peut trouver des politiques de meilleure qualité que les méthode de l'état de l'art comme les algorithmes de la famille MBDP (*Memory Bounded Dynamic Programming*) en utilisant plus d'information heuristique que ces méthodes. L'approche que nous proposons peut aussi être considérée comme une généralisation de ces méthodes.

1.3.1 État de l'art

La première partie de la thèse se consacre à l'état de l'art de la planification en commençant par les cas mono-agent avec observations totales et partielles, qui sont des cas particulier plus simples, avant de s'intéresser au cas général qui nous intéresse (multi-agent). Nous concentrons notre étude sur une famille d'algorithmes qui utilisent une méthode appelée *programmation dynamique* à laquelle appartiennent les algorithmes de planification multi-agent de l'état de l'art les plus performantes ainsi que l'approche que nous proposons.

Le chapitre 2 présente l'état de l'art pour les problèmes mono-agent avec observabilité totale. Il présente les concepts de fonction de valeur, qui permet de quantifier la qualité d'une politique, et de fonction de valeur optimale, qui permet de caractériser une politique optimale. Nous nous intéressons plus particulièrement à deux méthodes de planification par programmation dynamique :

- une méthode classique détermine la *fonction de valeur optimale*, qui peut ensuite être utilisée pour déterminer les actions optimales ;
- une méthode moins connue, PSDP (*Policy Search Dynamic Programming*) [Bagnell *et al.*, 2003], utilise une information heuristique sur le problème (sous la forme d'une distribution de probabilité sur les états du système) pour chercher directement une politique approchée.

La première méthode sert de fondement aux approches de planification par programmation dynamique que ce soit dans le cadre des observations complètes où dans les cas plus généraux qui nous intéressent. La deuxième méthode est fortement liée à l'approche que nous proposons qui consiste à essayer d'utiliser au mieux une information heuristique similaire à celle de PSDP pour chercher une politique approchée dans le cas décentralisé.

Le chapitre 3 présente l'état de l'art pour les problèmes mono-agent avec observabilité partielle et introduit le concept de croyance d'un agent. Les résultats du chapitre précédent peuvent être exploités pour résoudre ce type de problèmes : en particulier, une fonction de valeur optimale peut être définie et calculée par programmation dynamique ; elle peut ensuite être utilisée pour déterminer les actions optimales. La *structure des fonctions de valeur* considérées et la manière d'*exploiter ces structures* sont utilisées par les algorithmes de planification multi-agent de l'état de l'art qui nous intéressent ainsi que par l'approche que nous proposons.

Le chapitre 4 présente l'état de l'art dans le cas décentralisé qui constitue l'objet de notre étude. Les méthodes de planification par programmation dynamique approchée ou exacte dans ce cadre sont des généralisations des approches du chapitre précédent. En particulier, les algorithmes approchés de type MBDP [Seuken et Zilberstein, 2007b] peuvent être vus comme une généralisation des algorithmes à base de points du chapitre précédent : ils utilisent une heuristique pour construire une politique. La taille de ces politiques est contraintes afin de limiter les ressources (temps de calcul et espace mémoire) nécessaires à la planification. Cependant, du fait de la complexité apporté par la décentralisation, les approximations effectuées sont plus

grossières que dans le cas centralisé. Une autre limite de ces méthodes est la complexité d'une opération fondamentale de ces méthodes, l'opération de *lookahead*, qui est bien supérieure que dans le cas centralisé [Dibangoye *et al.*, 2009]. Dans la suite de la thèse, nous proposons des solutions partielles à ces deux problèmes.

1.3.2 Contributions

La deuxième partie de la thèse présente nos contributions. Nous utilisons comme référence les approches de planification approchée par programmation dynamique de type MBDP et nous explorons deux directions opposées qui répondent aux deux limites que nous venons d'évoquer :

- utiliser une planification plus rapide mais de moins bonne qualité (en faisant une approximation de l'opération coûteuse de *lookahead*) ;
- effectuer une planification de meilleure qualité mais éventuellement plus lente en remplaçant l'utilisation de *lookahead* par une méthode plus complexe pour effectuer les approximations.

Dans le chapitre 6, nous nous intéressons au premier point. L'opération de *lookahead.* consiste à résoudre un problème d'optimisation combinatoire coûteux en terme de complexité : nous proposons une méthode efficace pour trouver un maximum local à ce problème. Il s'agit plus précisément d'un équilibre de Nash (voir chapitre 9 en annexe). Nous utilisons la méthode classique de recherche d'équilibre de Nash, par calcul itéré de la meilleure réponse d'un agent, pour trouver une bonne solution au problème de *lookahead* pour un coût en complexité bien inférieur.

Le chapitre 7 présente la contribution principale de cette thèse. Nous montrons comment une information heuristique (une distribution de probabilité sur les croyances centralisées) peut être utilisée, de manière similaire à PSDP, pour guider la recherche de politique approchée dans le cas décentralisé. Notre approche, PSMBDP, est ainsi capable d'utiliser plus d'information heuristique que les approches de type MBDP en cherchant à faire une bonne approximation de la fonction de valeur optimale à chaque étape de la planification. Les résultats du chapitre précédent peuvent être utilisés dans cette méthode.

Enfin, le chapitre 8 présente un certain nombre de perspectives.

Notations

Nous rappellons ici les notations utilisées au long de cette thèse ainsi que certaines formules importantes.

Notations mathématiques

Probabilités

X	Variables aléatoires (majuscules)
x	Valeur d'une variable aléatoire (minuscules)
$X \sim \mathcal{X}(Y)$	X suit la loi $\mathcal{X}$ conditionnée par la valeur de la variable aléatoire Y
$(X\|Y = y) \sim \mathcal{X}(y)$	X sachant $Y = y$ suit la loi $\mathcal{X}(y)$
$\mathcal{X}(x\|y)$	Probabilité conditionnelle donnée par la loi $\mathcal{X}$, $\Pr(X = x\|Y = y)$
$X = \mathcal{X}(Y)$	X est une fonction $\mathcal{X}$ (déterministe) de Y
$\Delta\mathcal{Z}$	Ensemble des distributions de probabilité sur l'ensemble $\mathcal{Z}$

Ensembles

$\mathcal{X} \times \mathcal{Y} = \{(x, y)\|x \in \mathcal{X}, y \in \mathcal{Y}\}$	Ensemble des paires $\|\mathcal{X} \times \mathcal{Y}\| = \|\mathcal{X}\| \times \|\mathcal{Y}\|$ pour des ensembles finis
$\mathcal{Y}^{\mathcal{X}} = \{f : \mathcal{X} \mapsto \mathcal{Y}\}$	Ensemble des fonctions de $\mathcal{X}$ dans $\mathcal{Y}$ $\|\mathcal{Y}^{\mathcal{X}}\| = \|\mathcal{Y}\|^{\|\mathcal{X}\|}$ pour des ensembles finis

Problèmes de décision

Nous présentons les notations dans le cas du modèle le plus général (Dec-POMPD). Le cas POMDP est obtenu en posant $|\mathcal{I}| = 1$. Le cas MDP est obtenu en remplaçant de plus les termes de gauches par les termes de droites dans les égalités suivantes : $O_t = S_t$, $B_t = S_t$, $\Omega = \mathcal{S}$, On a donc dans le cas MDP :

$$\mathcal{O}(o|s, a, s') = \begin{cases} 1 \text{ si } s' = o \\ 0 \text{ sinon} \end{cases}$$

$$\mathcal{O}(o|s, a = \mathcal{T}(o|s, a)$$

$$\mathcal{J}(s', o|s, a) = \begin{cases} \mathcal{T}(s'|s, a) \text{ si } s' = o \\ 0 \text{ sinon} \end{cases}$$

Ensembles

$\mathcal{I}$	Ensemble des agents
$\mathcal{S}$	Ensemble des états
$\mathcal{A}_i$	Ensemble des actions locales de l'agent i
$\mathcal{A}$	Ensemble des actions jointes, $\mathcal{A} = \prod_{i\in\mathcal{I}} \mathcal{A}_i$
$\mathcal{A}_{-i}$	Ensemble des actions conjointes de l'agent i, $\mathcal{A}_{-i} = \prod_{j\neq i} \mathcal{A}_j$
Ω_i	Ensemble des observations locales de l'agent i
Ω	Ensemble des observations jointes, $\Omega = \prod_{i\in\mathcal{I}} \Omega_i$
Ω_{-i}	Ensemble des observations conjointes de l'agent i à l'instant t, $\mathcal{Z}_t = \prod_{j\neq i} \mathcal{Z}_t^j$

Variables aléatoires

$S_t \in \mathcal{S}$	État du système à l'instant t
$A_t^i \in \mathcal{A}_i$	Action locale de l'agent i à l'instant t
$A_t \in \mathcal{A}$	Action joint) à l'instant t
$A_t^{-i} \in \mathcal{A}_{-i}$	Action conjoint de l'agent i à l'instant t
	$A_t^{-i} = (A_t^j)_{j\neq i}$, $A_t = \langle A_t^i, A_t^{-i}\rangle$
$O_t^i \in \Omega$	Observation locale de l'agent i à l'instant t
$O_t \in \Omega$	Observation jointe à l'instant t
$O_t^{-i} \in \Omega_{-i}$	Observation conjoint de l'agents i à l'instant t
	$O_t^{-i} = (O_t^j)_{j\neq i}$, $O_t = \langle O_t^i, O_t^{-i}\rangle$
$R_t \in \mathbb{R}$	Récompense à l'instant t
$H_t = (B_0, A_1, O_2, \ldots, A_t, O_t)$	Historique (joint) à l'instant t
$H_t^i = (B_0, Z_0^{-i}, A_1^i, O_2^i, \ldots, A_t^i, O_t^i)$	Historique local de l'agent i à l'instant t
Z_t^i	État interne (local) de l'agent i à l'instant t
Z_t	État interne (joint) à l'instant t
Z_t^{-i}	État interne conjoint de l'agent i à l'instant t
	$Z_t^{-i} = (Z_t^j)_{j\neq i}$, $Z_t = \langle Z_t^i, Z_t^{-i}\rangle$
$B_t \in \Delta\mathcal{S}$	Croyance centralisée à l'instant t
	$B_t(s) = \Pr(S_t = s \mid H_t)$
$S_t^i = (S_t, Z_t^{-i})$	État subjectif de l'agent i
$B_t^i \in \Delta(\mathcal{S} \times \mathcal{Z}_t^{-i})$	Croyance de l'agent i à l'instant t
	$B_t^i(s, z_{-i}) = \Pr(S_t^i = (s, z_{-i}) \mid H_t) = \Pr(S_t = s, Z_t^{-i} = z_{-i} \mid H_t)$

Problème

$\mathcal{T}$ Loi de transition, $S_{t+1} \sim \mathcal{T}(S_t, A_{t+1})$
$\mathcal{T}(s'|s,a) = \Pr(S_{t+1} = s'|S_t = s, A_{t+1} = a)$
$\mathcal{T}(s'|b,a) = \Pr(S_{t+1} = s'|O_t b, A_{t+1} = a) = \sum_{s\in\mathcal{S}} b(s)\mathcal{T}(s'|s,a)$

$\mathcal{O}$ Loi des observations, $O_{t+1} \sim \mathcal{O}(S_t, O_{t+1} = o, A_{t+1} = a)$
$\mathcal{O}(o|s,a,s') = \Pr(O_{t+1} = o|S_t = s, A_{t+1} = a, S_{t+1} = s')$
$\mathcal{O}(o|s,a) = \Pr(O_{t+1} = o|S_t = s, A_{t+1} = a) = \sum_{s'\in\mathcal{S}} \mathcal{O}(o|s,a,s')$
$\mathcal{O}(o|b,a) = \sum_{s\in\mathcal{S}} b(s)\mathcal{O}(o|s,a)$

$\mathcal{R}$ Loi (espérance) des récompenses, $R_t \sim \mathcal{R}(S_t, A_{t+1})$
$\mathcal{R}(s,a) = \mathrm{E}[R_{t+1}|S_t = s, R_{t+1} = a]$
$\mathcal{R}(b,a) = \mathrm{E}[R_{t+1}|B_t = b, R_{t+1} = a] = \sum_{s\in\mathcal{S}} b(s)\mathcal{R}(s,a)$

$\mathcal{J}$ Loi jointe (tansition et observations), $(S_{t+1}, O_{t+1}) \sim \mathcal{J}(S_t, A_{t+1})$
$\mathcal{J}(s',o|s,a) = \Pr(S_{t+1} = s', O_{t+1} = o|S_t = s, A_{t+1} = a) = \mathcal{T}(s'|s,a)\mathcal{O}(o|s,a,s')$
$\mathcal{J}(s',o|b,a) = \Pr(S_{t+1} = s', O_{t+1} = o|B_t = b, A_{t+1} = a) = \sum_{s\in\mathcal{S}} b(s)\mathcal{J}(s',o|s,a)$

τ Fonction de mise à jour de croyance, $B_{t+1} = \tau(B_t, A_{t+1}, O_{t+1})$
$[\tau(b,a,o)](s) = \lambda\mathcal{J}(s',o|b,a)$ soit $\tau(b,a,o) \propto \mathcal{J}_{o|a} b$

τ_i Fonction de mise à jour de croyance de l'agent i, $B^i_{t+1} = \tau_i(B^i_t, A^i_{t+1}, O^i_{t+1})$

Lois de l'état interne

ν Loi de l'état interne joint initiale, $Z_0 \sim \nu(B_0)$
$\nu(z|b_0) = \Pr(Z_0 = z_0|B_0 = b_0)$

φ_i Loi des actions locales de l'agent i, $A^i_{t+1} \sim \varphi_i(Z^i_t)$
$\varphi_i(a_i|z_i) = \Pr(A^i_{t+1} = a_i|Z^i_t = z_i)$

ψ_i Loi de mise à jour de l'état interne local de l'agent i, $Z^i_{t+1} \sim \psi_i(Z^i_t, A^i_{t+1}, O^i_{t+1})$
$\psi_i(z'_i|z_i,a_i,o_i) = \Pr(Z^i_{t+1} = z'_i|Z^i_t = z_i, A^i_{t+1} = a_i, O^i_{t+1} = o_i)$

φ Loi des actions jointes, $A_{t+1} \sim \varphi(Z_t)$
$\varphi(a|z) = \Pr(A_{t+1} = a|Z_t = z)$
$\varphi(a|z) = \prod_{i\in\mathcal{I}} \varphi_i(a_i|z_i)$ (décentralisation)

ψ Loi de mise à jour de l'état interne joint, $Z_{t+1} \sim \psi(Z_t, A_{t+1}, O_{t+1})$
$\psi(z'|z,a,o) = \Pr(Z_{t+1} = z'|Z_t = z, A_{t+1} = a, O_{t+1} = o)$
$\psi(z'|z,a,o) = \prod_{i\in\mathcal{I}} \psi_i(z'_i|z_i,a_i,o_i)$ (décentralisation)

φ_{-i} Loi des actions cojointes de l'agent i, $A^{-i}_{t+1} \sim \varphi_{-i}(Z^{-i}_t)$
$\varphi_{-i}(a_{-i}|z_{-i}) = \Pr(A^{-i}_{t+1} = a_{-i}|Z^{-i}_t = z_{-}i) = \prod_{j\neq i} \varphi_j(a_j|z_j)$

ψ_{-i} Loi de mise à jour de l'état interne cojoint de l'agent i, $Z^{-i}_{t+1} \sim \psi_{-i}(Z^{-i}_t, A^{-i}_{t+1}, O^{-i}_{t+1})$
$\psi_{-i}(z'_{-i}|z_{-i},a_{-i},o_{-i}) = \Pr(Z^{-i}_{t+1} = z'_i|Z^{-i}_t = z_{-i}, A^{-i}_{t+1} = a_i, O^{-i}_{t+1} = o_i)$
$\psi_{-i}(z'_{-i}|z_{-i},a_{-i},o_{-i}) = \prod_{j\neq i} \psi_j(z'_j|z_j,a_j,o_j)$

Arbre de décision

$\varphi_i(z_i)$	Action locale racine de l'arbre local z
$\psi_i(z_i,o_i)$	Sous-arbre d'observation o de l'abre local z_i
$\varphi(z)$	Action racine de l'arbre joint z
	$\varphi(z) = (\varphi_1(z_1), \ldots, \varphi_n(z_n))$ (décentralisation)
$\psi(z,o)$	Sous-arbre d'observation o de l'arbre joint z_i
	$\psi(z,o) = (\psi_1(z_1,o_1), \ldots, \psi_n(z_n,o_n))$ (décentralisation)
$\varphi_{-i}(z_{-i})$	Action de l'arbre conjoint z_{-i}
$\psi_{-i}(z_{-i},o_{-i})$	Sous-arbre d'observation o de l'abre conjoint z_{-i}

Règles de décision

π_t Règle de décision à l'instant t
$\pi_t(a|h) = \Pr(A_{t+1} = a|H_t = h)$ ou $\pi_t(a|b) = \Pr(A_{t+1} = a|B_t = b)$ (stochastique)
$A_{t+1} = \pi_t(H_t)$ ou $A_{t+1} = \pi_t(B_t)$ (déterministe)

π Régles de décisions pour tout t, $\pi = (\pi_t)_{\forall t}$

π Règle de décision stationnaire, $\forall (t, t'), \pi_t = \pi_{t'}$
$\pi(a|h) = \Pr(A_{t+1} = a|H_t = h)$ ou $\pi(a|b) = \Pr(A_{t+1} = a|B_t = b)$ (stochastique)
$A_{t+1} = \pi(H_t)$ ou $A_{t+1} = \pi(B_t)$ (déterministe)

Valeur

T Horizon de planification

γ Facteur de décompte

V Fonction de valeur, $V(s)$

Q Fonction de valeur d'action, $Q(s, a)$

$V_t^\pi(b)$ Valeur de la politique π pour la croyance b à l'instant t
$V_t^\pi(b) = \mathrm{E}\left[\sum_{t'=t+1}^{T} \gamma^{t'-t-1} R_{t'} | S_t \sim b, B_t = b, \pi\right]$

$V^\pi(b)$ Valeur de la politique π pour la croyance (horizon infini)

H_{π_t} Opérateur de Bellman pour la règle de décision π_t ; $V_{t+1}^\pi = H_{\pi_t} V_t^\pi$ et $V^\pi = H_\pi^\infty 0$

$V_t^\star(b)$ Valeur optimale pour la croyance b à l'instant t

$V^\star(b)$ Valeur optimale pour la croyance (horizon infini)

H Opérateur d'optimalité de Bellman ; $V_{t+1}^\star = H V_t^\star$ et $V^\star = H^\infty 0$

Fonction de valeur PWLC et vecteurs α

$\alpha^z \in \mathbb{R}^{\mathcal{S}}$ Vecteur α

Γ Représentation d'une fonction de valeur V PWLC
$V(b) = \max_{\alpha \in \Gamma} \alpha \cdot b = \sum_{s \in \mathcal{S}} b(s)\alpha(s)$

$H_{\text{màj}}$ Opérateur de mise à jour exhaustive

$V^z(b)$ Valeur de l'arbre joint z dans la croyance b

$\alpha \in \mathbb{R}^{\mathcal{S}}$ Vecteur α qui représente la valeur V^z d'un arbre de politique *joint*
$V^z(s) = \alpha^z(s)$; $V^z(b) = b \cdot \alpha^z = \sum_{s \in \mathcal{S}} b(s)\alpha^z(s)$

$\mathcal{R}_a$ Vecteur α représentant les récompenses immédiates pour l'action a, $\mathcal{R}_a(s) = \mathcal{R}(s, a)$

$\bar{\mathcal{J}}_{o|a}\alpha'$ Contribution d'un vecteur α' à l'instant $t+1$ à un vecteur α à l'instant t
$\left[\bar{\mathcal{J}}_{o|a}\alpha'\right](s) = \sum_{s' \in \mathcal{S}} \mathcal{J}(s', o|s, a)\alpha'(s')$
$\alpha^z = \mathcal{R}_{\varphi(z)} + \gamma \sum_{o \in \Omega} \bar{\mathcal{J}}_{o|\varphi(z)} \alpha^{\psi(z,o)}$ (déterministe)
$\alpha^z = \sum_{a \in \mathcal{A}} \varphi(a|z) \left[\mathcal{R}_a + \gamma \sum_{o \in \Omega} \sum_{z'} \psi(z'|z, a, o) \bar{\mathcal{J}}_{o|a} \alpha^{z'}\right]$ (stochastique)

$\perp$ État interne terminal de l'agent, arbre de politique vide
$\alpha^\perp = 0$

Distributions heuristiques

p_t Distribution de probabilité heuristique sur les états
$p_t(s) = \Pr(S_t = s|B_0 = b_0, \tilde{\pi})$

μ_t Distribution de probabilité heuristique sur les croyances
$\mu_t(b) = \Pr(B_t = b|B_0 = b_0, \tilde{\pi})$

μ Ensemble des distributions de probabilité heuristiques sur les croyances
$\mu = (\mu_0, \ldots, \mu_T)$

Première partie

État de l'art

Chapitre 2

MDP

Sommaire

Nous nous intéressons dans ce chapitre au problème de la décision séquentielle dans l'incertain pour des processus complètement observables mono-agent. On considère dans ce chapitre les cas constitués d'un agent unique en interaction avec un système dynamique. L'agent peut influencer l'évolution du système par le biais de ses actions. L'évolution du système est cependant stochastique : l'évolution du processus après une action donnée n'est pas déterministe et l'action ne fait qu'influer la probabilité de l'état résultant. On considère de plus dans ce chapitre que l'agent peut à tout instant observer de manière exacte et complète l'état de l'environnement.

Exemple 2.0.1*:* Revenons sur notre exemple. Une action de l'agent consiste à tenter d'éteindre l'incendie dans une maison donnée. À chaque instant, le pompier connaît l'état global de l'incendie dans l'ensemble de ces maisons (observabilité totale) et utilise cette information pour décider dans quelle maison il doit agir. Ce choix influe sur l'évolution de l'incendie. L'incertitude réside dans l'évolution stochastique de l'incendie et en particulier dans l'effet des actions du pompier.

Dans ce chapitre, nous présentons l'état de l'art pour ce type de problème modélisés par le formalisme MDP (*Markov Decision Process*). Ce modèle est présenté dans l'optique de son

adaptation dans les chapitres ultérieurs aux problèmes plus généraux (observabilité partielles et agents multiples) qui nous intéressent.

Nous rappelons dans ce chapitre les principaux résultats théoriques :

- on peut restreindre notre recherche aux politiques markoviennes déterministes ;
- la théorie de la valeur permet de déterminer la qualité d'une politique, grâce au concept de fonction de valeur, ainsi que de caractériser une politique optimale, grâce au concept de fonction de valeur optimale.

Nous concentrons ensuite notre présentation sur deux méthodes de planification par programmation dynamique.

L'approche classique consiste à calculer la fonction de valeur optimale. Nous verrons dans les prochains chapitre que les approches classiques de l'état de l'art pour la planification décentralisée dérivent fortement de la méthode de planification par calcul de la fonction de valeur optimale.

L'autre approche que nous présentons, PSDP (*Policy Search Dynamic Programming* – Programmation dynamique par recherche de politique), utilise une information heuristique sur le problème sous la forme d'une distribution de probabilité sur les états du système pour chercher une solution approchée au problème de planification. L'approche que nous proposons dans cette thèse pour la planification approchée multi-agent utilise, de manière similaire, des informations heuristiques sur le problème, représentées comme des distributions de probabilités.

2.1 Formalisme

Le cadre des Processus Décisionnels de Markov (MDP, *Markov Decision Process*) [Puterman, 1994, Bellman, 1957] est un formalisme mathématique à temps discret pour modéliser les problèmes de décision à observabilité totale. Ce modèle de base est étendu dans les chapitres suivants pour les problème à observations partielles (modèle POMDP [Sondik, 1971]) et les problèmes décentralisés (modèle Dec-POMDP [Bernstein *et al.*, 2000]).

Nous commençons par présenter et expliquer les différentes variables aléatoires qui composent un processus décisionnel de Markov et nous aboutissons à la définition d'un processus décisionnel de Markov.

2.1.1 Présentation

Dynamique du système Un processus décisionnel de Markov est une chaîne de Markov $(S_t)_{t\in\mathbb{N}}$ dont les transitions résultent des actions A_t effectuées par l'agent. À chaque instant $t \in \mathbb{N}$, le système est dans un *état* noté $S_t \in \mathcal{S}$, et l'agent doit choisir une *action* $A_{t+1} \in \mathcal{A}$ (cf. figure 2.1). Cette action A_{t+1} induit une *transition* stochastique du système dans un nouvel état S_{t+1}. On suppose que l'état du système possède la *propriété de Markov* [Sutton et Barto, 1998] c'est-à-dire que le nouvel état S_{t+1} ne dépend pas de l'historique complet des actions et états passés mais uniquement de l'état précédent S_t et de l'action effectuée A_{t+1},

$$\Pr(S_{t+1} = s' | S_0, A_1, S_1, \ldots, S_t = s, A_{t+1} = a) = \Pr(S_{t+1} = s' | S_t = s, A_{t+1} = a) = \mathcal{T}(s'|s,a) \tag{2.1}$$

Observabilité totale L'hypothèse d'observabilité totale signifie que l'agent connaît à chaque instant t l'état S_t dans lequel il est situé. Sa prochaine action A_{t+1} peut dépendre de manière la plus générale possible des états $S_{t'}$ et des actions $A_{t'}$ passés (pour $t' \leq t$). On appelle *historique* à l'instant t le tuple composé de ces variables : $H_t = (S_0, A_1, S_1, \ldots A_t, S_t)$.

Dynamique de l'agent Afin d'effectuer ce type de comportement très général, l'agent a besoin de maintenir un *état interne* Z_t : cet état interne peut être l'historique H_t ou une autre

donnée qui « résume » l'historique. Un nouvel état interne Z_{t+1} de l'agent ne peux dépendre que de l'état interne précédent Z_t, de la nouvelle l'action A_{t+1} et du nouvel état du système S_{t+1} :

$$\Pr(Z_{t+1} = z' | Z_t = z, A_{t+1} = a, S_{t+1} = s) = \psi(z' | z, a, s) \tag{2.2}$$

L'état interne Z_t permet de décider de l'action A_{t+1} à effectuer :

$$\Pr(A_{t+1} = a | Z_t = z) = \varphi(a | z) \tag{2.3}$$

On appelle une *politique* π une définition du comportement de l'agent : cette notion sera détaillée à la section 2.1.4.

Remarque: Comme nous le verrons à la section 2.2.1, du fait de l'observabilité totale et de la propriété de Markov, l'agent n'a pas besoin de retenir tout l'historique H_t ni de maintenir un état interne Z_t pour choisir une action A_{t+1} optimale mais uniquement d'utiliser l'état courant S_t. Nous présentons ce modèle général pour justifier que l'on peut se limiter, dans le cadre MDP, au modèle simplifié des agents sans état interne et car l'agent a besoin de maintenir une état interne pour les problèmes plus généraux avec observabilité partielle et agents multiples.

Récompense Afin de représenter l'objectif de l'agent, un critère de performance est introduit pour quantifier les résultats de l'agent. Après chaque action A_t l'agent reçoit une *récompense* R_t : il s'agit d'une grandeur réelle positive (récompense, renforcement, gain) ou négative (coût, pénalité, perte). De manière informelle, l'objectif de l'agent est de maximiser les récompenses obtenues : nous présenterons dans la section 2.1.3 des critères de performance. De manière générale, la récompense R_{t+1} peut dépendre des états S_t et S_{t+1} ainsi que de l'action A_{t+1}. Les critères de performance couramment utilisés ne dépendent que de l'espérance de la récompense obtenue après avoir effectué une actions $A_{t+1} = a$ dans un état $S_t = s$:

$$\mathrm{E}\left[R_{t+1} | S_t = s, A_{t+1} = a\right] = \mathcal{R}(s, a) \tag{2.4}$$

La figure 2.1 résume la relation entre les différentes variables aléatoires sous la forme d'un DBN (réseau bayésien dynamique – *Dynamic Bayesian Network*) [Murphy, 2002]. La partie inférieure représente la dynamique interne de l'agent : elle n'est dans un premier temps pas formulée de manière classique (pour un MDP). Cette forme générale de politique de l'agent est expliquée plus formellement à la section 2.1.4 et est particulièrement utile dans les problèmes plus généraux étudiés dans les chapitres ultérieurs de la thèse. La formulation simplifiée est ensuite introduite (section 2.2.1).

2.1.2 Définition

Dans un processus décisionnel de Markov, la dynamique du système et les récompenses sont fixées et on cherche un comportement pour l'agent afin de maximiser un critère de performance. Un processus décisionnel de Markov est donc défini par les lois de transition $\mathcal{T}$ et de récompenses $\mathcal{R}$ (définition 2.1.1).

Définition 2.1.1. Un *processus décisionnel de Markov* (MDP) est défini par un tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ où :

– $\mathcal{S}$ est l'*ensemble des états* possibles de l'environnement, $\forall t, S_t \in \mathcal{S}$;

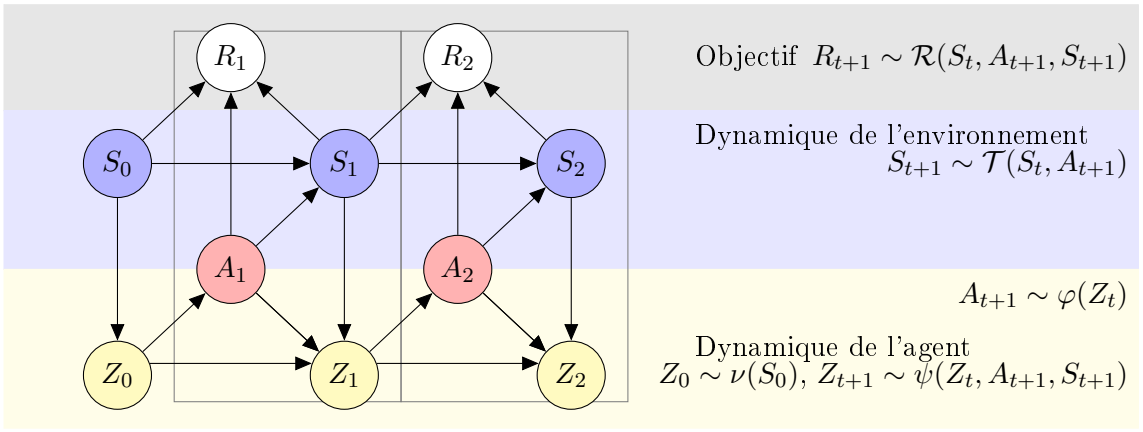


FIGURE 2.1 – Modèle MDP

- $\mathcal{A}$ est l'*ensemble des actions* possibles de l'agent, $\forall t, A_t \in \mathcal{A}$;
- $\mathcal{T}$ est la *loi de transition* qui définit les probabilités $\mathcal{T}(s'|s,a)$ de transition de l'état $S_t = s$ à l'état $S_{t+1} = s'$ conditionnée par l'action $A_{t+1} = a$

$$\mathcal{T}(s'|s,a) = \Pr(S_{t+1} = s'|S_t = s, A_{t+1} = a) = \Pr(S_{t+1} = s'|S_0, A_1, S_1, \ldots, S_t = s, A_{t+1} = a) \tag{2.5}$$

- $\mathcal{R}$ est la *fonction de récompense* qui définit l'espérance $\mathcal{R}(s,a)$ de la récompense R_{t+1} obtenue en effectuant l'action $A_{t+1} = a$ dans l'état $S_t = s$

$$\mathcal{R}(s,a) = \mathrm{E}\left[R_{t+1}|S_t = s, A_{t+1} = a\right] \tag{2.6}$$

Pour un MDP, deux types de problèmes sont envisagés :

- la *planification* où les lois du problème ($\mathcal{T}$ et $\mathcal{R}$) sont connues et où l'on cherche à calculer à l'avance une politique π qui maximise le critère ;
- l'*apprentissage* où les lois du problème ($\mathcal{T}$ et $\mathcal{R}$) ne sont pas connues, et où l'agent perçoit les récompenses R_t et peut les utiliser pour apprendre un comportement par essais et erreurs.

Nous nous intéressons dans cette thèse au problème de la planification et la présentation que nous avons faite du processus stochastique se place du point de vue de la planification.

Exemple 2.1.1*:* Formalisons le problème du pompier. Chaque maison est identifiée par une valeur $x \in \mathcal{X} = \{1, \ldots, maxHouses\}$. À un instant t, chaque maison x est dans un état $S_t^x \in \mathcal{L} = \{0, 1, \ldots, maxFire\}$ qui définit l'intensité de l'incendie dans cette maison : la valeur 0 indique qu'elle ne brûle pas. On introduit les conditions aux limites : $\forall t, \forall x \notin \mathcal{X}, S_t^x = 0$. L'état global est défini par l'état de chaque maison :

$$S_t = (S_t^1, \ldots, S_t^{maxHouses}) \in \mathcal{S} = \mathcal{L}^{\mathcal{X}} \tag{2.7}$$

À chaque instant, l'agent peut choisir de se rendre dans une maison : $A_t \in \mathcal{A} = \mathcal{X}$. La probabilité de l'état S_{t+1} dépend de l'état S_t et de l'action A_t. On fait l'hypothèse que l'évolution d'une

maison ne dépend que de son voisinage :

$$\mathcal{T}(s'|s,a) = \Pr(S_{t+1} = s'|S_t = s, A_{t+1} = a) \tag{2.8}$$
$$= \prod_{x\in X} \Pr(S^x_{t+1} = s'_x|S^{x-1}_t = s_{x-1}, S^x_t = s_x, S^{x+1}_t = s_{x+1}, P^x_{t+1} = \delta_x(a)) \tag{2.9}$$
$$= \prod_{x\in X} \mathcal{T}_x(s'_x|s_{x-1}, s_x, s_{x+1}, \delta_x(a)) \tag{2.10}$$

où $P^x_t = \delta_x(A_t)$ (0 ou 1) indique si l'agent se trouve dans la maison x dans la maison x à l'instant t avec δ_x la fonction caractéristique de x définie par :

$$\delta_x(a) = \begin{cases} 1 \text{ si } a = x \\ 0 \text{ sinon} \end{cases} \tag{2.11}$$

On considère que les $\mathcal{T}_x$ sont identiques. La dynamique de l'incendie sans influence du pompier est définie dans [Oliehoek *et al.*, 2008] par :

$$\mathcal{T}_x(s_x + 1|s_{x-1}, s_x, s_{x+1}, 0) = 0.8 \qquad \text{si } s_x \neq 0, s_{x-1} + x_{x+1} \neq 0 \tag{2.12}$$
$$\mathcal{T}_x(s_x|s_{x-1}, s_x, s_{x+1}, 0) = 0.2 \qquad \text{si } s_x \neq 0, s_{x-1} + x_{x+1} \neq 0 \tag{2.13}$$
$$\mathcal{T}_x(s_x + 1|s_{x-1}, s_x, s_{x+1}, 0) = 0.4 \qquad \text{si } s_x \neq 0, s_{x-1} + x_{x+1} = 0 \tag{2.14}$$
$$\mathcal{T}_x(s_x|s_{x-1}, s_x, s_{x+1}, 0) = 0.6 \qquad \text{si } s_x \neq 0, s_{x-1} + x_{x+1} = 0 \tag{2.15}$$
$$\mathcal{T}_x(1|s_{x-1}, 0, s_{x+1}, 0) = 0.8 \qquad \text{si } s_{x-1} + x_{x+1} \neq 0 \tag{2.16}$$
$$\mathcal{T}_x(0|s_{x-1}, 0, s_{x+1}, 0) = 0.2 \qquad \text{si } s_{x-1} + x_{x+1} \neq 0 \tag{2.17}$$
$$\mathcal{T}_x(0|0, 0, 0, p_x) = 1.0 \tag{2.18}$$

L'influence du pompier est donnée par :

$$\mathcal{T}_x(0|s_{x-1}, 0, s_{x+1}, 1) = 1.0 \tag{2.19}$$
$$\mathcal{T}_x(s_x - 1|0, s_x, 0, 1) = 1.0 \qquad \text{si } s_x \neq 0 \tag{2.20}$$
$$\mathcal{T}_x(s_x - 1|s_{x-1}, s_x, s_{x+1}, 1) = 0.6 \qquad \text{si } s_x \neq 0, s_{x-1} + x_{x+1} \neq 0 \tag{2.21}$$
$$\mathcal{T}_x(s_x|s_{x-1}, s_x, s_{x+1}, 1) = 0.4 \qquad \text{si } s_x \neq 0, s_{x-1} + x_{x+1} \neq 0 \tag{2.22}$$

Le pompier cherche à éteindre le plus rapidement les incendies ou à minimiser leur intensité. La récompense utilisée est l'opposé de la somme de l'intensité l'incendie dans chaque maison :

$$R_t = -\sum_{x\in\mathcal{X}} S^x_t \tag{2.23}$$

Remarque: Nous supposons en général que les ensembles $\mathcal{A}$ d'actions et $\mathcal{S}$ d'états sont finis. Les algorithmes qui suivent nécessitent ces hypothèses pour être effectués de manière exacte. En effet, ils supposent un parcours exhaustif des ensembles $\mathcal{S}$ et $\mathcal{A}$. Cependant, les résultats théoriques sont en général valides pour des ensembles infinis ou continus, en remplaçant les sommes par des intégrales. En particulier, le chapitre 3 sur les POMDP montre comment ces résultats peuvent s'appliquer pour une classe particulière de problèmes MDP avec espace d'états continus.

2.1.3 Critère

Informellement, le but de l'agent est de maximiser la somme des récompenses. Le *critère de performance* généralement utilisé quand le nombre T de pas de temps (l'*horizon*) est fini est l'espérance de la somme des récompenses

$$\mathrm{E}\left[\sum_{t=1}^{T} R_t | S_0, \pi\right] = \sum_{t=1}^{T} \mathrm{E}\left[R_t | S_0, \pi\right] \tag{2.24}$$

Une politique est dite optimale, notée $\pi^\star$, si elle maximise ce critère pour toute valeur de S_0.

En horizon infini, la somme diverge dans le cas général et on utilise généralement comme critère l'espérance de la somme des récompenses actualisées. On introduit un *facteur d'actualisation* $\gamma \in]0,1[$ pour faire converger le critère :

$$\mathrm{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_t | S_0, \pi\right] \tag{2.25}$$

Ce facteur traduit une éventuelle préférence pour les récompenses immédiates plutôt que les récompenses différées. Plus γ est petit, plus l'agent préfère les récompenses à court termes aux récompenses à long terme ; inversement plus γ est proche de 1, plus les conséquences à long terme sont importantes.

Remarque: Le terme de facteur d'actualisation vient de l'économie où il permet de comparer deux flux financiers non comparables car situés à des dates différentes. Il traduit une préférence pour « la jouissance immédiate ». Ce facteur est lié au taux d'actualisation τ défini par $\gamma = \frac{1}{1+\tau}$. Le taux d'actualisation est généralement fixé au taux d'intérêt d'un placement « sûr » ou au taux d'inflation.

Pour généraliser ces deux critères, on peut noter le critère de performance :

$$V^\pi(s) = \mathrm{E}\left[\sum_{t=1}^{T} \gamma^{t-1} R_t | S_0 = s, \pi\right] \tag{2.26}$$

avec $\gamma \in]0,1]$ et $T \in \mathbb{N} \cup \{\infty\}$. Étant donné les lois d'un problème, on cherche à déterminer une politique qui maximise ce critère soit pour un état initial $S_0 = s$ donné, soit pour tout état initial $S_0 = s$.

2.1.4 Politique

À chaque instant t, l'agent ne peut choisir son action A_{t+1} qu'à partir de ce qu'il connaît c'est-à-dire des états et actions passés : $H_t = (S_0, A_1, S_1, \ldots, A_t, S_t)$. Une politique, notée π, définit le comportement de l'agent c'est-à-dire les A_{t+1} de l'agent étant donné l'historique H_t à chaque instant t. Le problème de la planification consiste à déterminer une politique pour l'agent afin d'atteindre un objectif représenté par un critère de performance.

On peut définir une politique π comme les lois π_t des actions A_{t+1} conditionnée par les historiques H_t :

$$\pi_t(a|h) = \Pr(A_{t+1} = a | H_t = h) \tag{2.27}$$

On peut aussi définir une politique π comme un automate défini par une variable aléatoire Z_t, l'état interne de l'agent (voir figure 2.1) prenant des valeurs dans un ensemble $\mathcal{Z}$. L'agent prend ses décisions en se basant sur cet état interne. L'état interne de l'agent est associé à :

– des *probabilités d'action*, notées φ

$$\varphi(a|z) = \Pr(A_{t+1} = a|Z_t = z) \tag{2.28}$$

– des *probabilités de transition d'état interne*, notées ψ

$$\psi(z'|z, a, s) = \Pr(Z_{t+1} = z'|Z_t = z, A_{t+1} = a, S_{t+1} = s) \tag{2.29}$$

– des *probabilités de l'état interne initial*, notées ν

$$\nu(z|s) = \Pr(Z_0 = z|S_0 = s) \tag{2.30}$$

Remarque: Il s'agit cependant d'une forme de politique trop générale pour ce type de problèmes. Aussi, dans le cas des MDP, le terme de politique est généralement utilisée pour désigner une politique markovienne, c'est à dire qui ne dépend que de l'état courant. Ce type de politique est défini plus bas (section 2.2.1) et conduit à un modèle simplifié de MDP (voir figure 2.2) qui est le modèle généralement présenté. Cependant considérer d'abord le type le plus général de politique permet de justifier que les formes plus simples sont bien suffisantes. De plus, présenter le type le plus général de politique permet de faire le lien avec les problèmes plus généraux qui nous intéressent et qui sont développés dans les prochaines chapitre. Ce type de politique devient intéressant dans ces cas plus généraux, pour les problèmes à observabilité partielle et plus particulièrement pour les problèmes multi-agent où ce type de politique est utilisé explicitement.

2.2 Résultats

Nous présentons dans cette section, les principaux résultats théoriques pour les MDP :

– l'action optimale A_{t+1} ne dépend que de l'état S_t et il n'est pas nécessaire de maintenir un état interne ;
– on peut se restreindre aux politiques déterministes ;
– la politique optimale peut être déterminée en calculant une fonction appelée *fonction de valeur optimale*.

2.2.1 Politique markovienne

Dans le cas des observations totales, l'agent n'a pas besoin de retenir tout l'historique H_t pour choisir A_{t+1}. Grâce à la propriété de Markov, l'état courant S_t est une statistique suffisante pour la prise de décision de l'agent (théorème 2.2.1). Nous pouvons donc considérer un modèle plus simple d'agent sans mémoire où l'action A_{t+1} ne dépend que de l'état courant S_t : on appelle politique markovienne une politique qui respecte cette propriété. Le théorème d'équivalence 2.2.1 justifie que, pour les critères que nous avons envisagés, il n'est pas nécessaire de considérer la forme général des politiques mais uniquement les politiques markoviennes. La figure 2.2 présente sous forme de DBN le modèle simplifié de MDP avec des politiques markoviennes qui est la représentation classique des MDP (comparer à la figure 2.1).

Définition 2.2.1. Une *politique markovienne* π est une politique pour laquelle A_t dépend uniquement de S_t et de t :

$$\pi_t(a|s) = \Pr(A_{t+1} = a|H_t = (\ldots, s)) = \Pr(A_{t+1} = a|S_t = s) \tag{2.31}$$

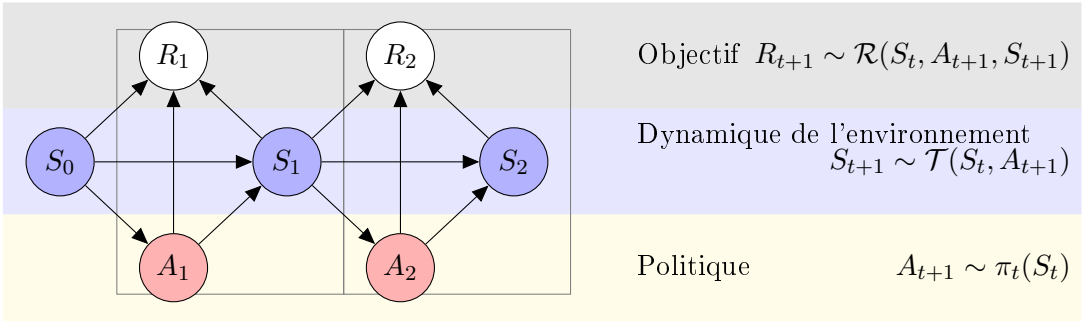


FIGURE 2.2 – Modèle MDP classique (politique markovienne)

Théorème 2.2.1. *(Théorème d'équivalence pour un MDP) Soit une politique π, pour tout état initial $s \in \mathcal{S}$, il existe une politique markovienne π' telle que $V^{\pi}(s) = V^{\pi'}(s)$ [Garcia, 2008].*

Démonstration. voir [Garcia, 2008] □

2.2.2 Politique stationnaire

En horizon infini, le problème de décision dans l'état $S_t = s$ à l'instant t est le même pour toute valeur de t. L'agent n'a pas besoin de prendre en compte l'instant t pour prendre une décision. Pour les problèmes à horizon infini, on considère des politiques stationnaires qui ne dépendent pas du temps.

Définition 2.2.2. Une *politique stationnaire* est une politique markovienne qui ne dépend pas de t :

$$\pi(a|s) = \Pr(A_{t+1} = a|H_t = (\ldots, s)) = \Pr(A_{t+1} = a|S_t = s) \tag{2.32}$$

2.2.3 Valeur d'une politique

Afin d'évaluer la qualité d'une politique π markovienne, on introduit la fonction de valeur V_t^{π}. La valeur d'une politique π à l'instant t dans l'état s est l'espérance des récompenses à venir quand on suit cette politique à partir de $S_t = s$. En d'autres termes, V_{τ}^{π} est la fonction valeur de la sous-politique $\pi_{\tau:T}$ exécutée à partir de l'instant τ jusqu'à l'instant terminal T : $V_{\tau}^{\pi} = V^{\pi_{\tau:T}}$.

Définition 2.2.3. La *fonction de valeur* d'une politique markovienne π à l'instant t est la fonction qui à tout état $S_t = s \in \mathcal{S}$ associe le critère :

$$V_t^{\pi}(s) = \mathrm{E}\left[\sum_{t'=t+1}^{T} \gamma^{t'-t-1} R_{t'} | S_t = s, \pi\right] \tag{2.33}$$

La fonction de valeur peut être déterminée par programmation dynamique [Bellman, 1953]. L'idée est qu'il est possible de décomposer la fonction de valeur V^{π} en une contribution provenant de la récompense immédiate liée à π_0 et une contribution qui vient de la sous-politique $\pi' =$

$\pi_{|1:T-1}$ utilisée à partir de $t = 1$. On peut donc décomposer le problème du calcul de $V_t^\star$ en deux sous-problèmes : le sous-problème du calcul de la fonction de valeur $V_{t+1}^\star$ de la sous-politique $\pi_{t+1:T-1}$ et le calcul de la fonction de valeur $V_t^\star$ de $\pi_{t:T}$ à partir de la fonction de valeur $V_{t+1}^\star$ de la sous-politique.

Théorème 2.2.2 (Évaluation de Bellman)**.** *La fonction de valeur d'une politique π peut être décomposé en un terme de récompense immédiate et un terme de valeur du sous-problème :*

$$V_t^\pi(s) = \sum_{a\in\mathcal{A}} \pi_t(a|s) \left[\mathcal{R}(s,a) + \gamma \sum_{s'\in\mathcal{S}} \mathcal{T}(s'|s,a) V_{t+1}^\pi(s')\right] \tag{2.34}$$

soit, pour les politiques déterministes,

$$V_t^\pi(s) = \mathcal{R}(s,\pi_t(s)) + \gamma \sum_{s'\in\mathcal{S}} \mathcal{T}(s'|s,\pi_t(s)) V_{t+1}^\pi(s') \tag{2.35}$$

Démonstration. [Garcia, 2008] □

On introduit la fonctionnelle H_{π_t}, appelée opérateur de Bellman de π_t, qui à toute fonction de valeur $V \in \mathbb{R}^{\mathcal{S}}$ associe une fonction de valeur $H_{\pi_t} V \in \mathbb{R}^{\mathcal{S}}$:

$$\left[H_{\pi_t} V\right](s) = \sum_{a\in\mathcal{A}} \pi_t(a|s) \left[\mathcal{R}(s,a) + \gamma \sum_{s'\in\mathcal{S}} \mathcal{T}(s'|s,a) V(s')\right] \tag{2.36}$$

L'équation d'évaluation de Bellman prend alors la forme :

$$V_t^\pi = H_{\pi_t} V_{t+1}^\pi \tag{2.37}$$

2.2.4 Valeur optimale

Le concept de fonction de valeur est important non seulement parce qu'il permet de comparer la qualité de plusieurs politiques mais parce qu'il permet de déterminer la politique optimale du MDP. Toute politique optimale $\pi^\star$ a la même fonction de valeur, appelée *fonction de valeur optimale*, notée $V_t^\star$:

$$V_t^\star = V_t^{\pi^\star} \tag{2.38}$$

Celle-ci peut être utilisée pour déterminer l'action optimale à chaque instant.

Intuitivement, le principe d'optimalité de Bellman dit que pour une politique optimale $\pi^\star$ en horizon T, toutes les sous-politiques $\pi_{t:T}^\star$ pour les instant t à T sont des politiques optimales du sous-problème constitué de ces instants. La recherche de la politique optimale peut se faire par programmation dynamique rétrograde en décomposant le problème en deux étapes successives :

1. la recherche d'une sous-politique optimale $\pi_{1:T-1}^\star$ (sous-problème) ;
2. la construction d'une politique optimale $\pi^\star$ à partir de la sous-politique optimale $\pi_{1:T-1}^\star$.

Théorème 2.2.3 (Optimalité de Bellman)**.** *La fonction de valeur $V_t^\star$ optimale est donnée par*

$$V_t^\star(s) = \max_{a\in\mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \sum_{s'\in\mathcal{S}} \mathcal{T}(s'|s,a) V_{t+1}^\star(s')\right] \tag{2.39}$$

et une action optimale à l'instant t est une action gourmande pour $V_{t+1}^\star$:

$$A_{t+1} = \pi_t^\star(S_t) \in \arg\max_{a\in\mathcal{A}} \left[\mathcal{R}(S_t,a) + \gamma \sum_{s'\in\mathcal{S}} \mathcal{T}(s'|S_t,a) V_{t+1}^\star(s')\right] \tag{2.40}$$

Démonstration. [Garcia, 2008] □

On introduit la fonctionnelle H, appelée *opérateur (d'optimalité) de Bellman*, qui à toute fonction de valeur $V \in \mathbb{R}^{\mathcal{S}}$ associe une fonction de valeur $HV \in \mathbb{R}^{\mathcal{S}}$:

$$[HV](s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s'|s,a) V(s') \right] \tag{2.41}$$

L'équation d'optimalité de Bellman peut se noter de manière plus concise par :

$$V_t^\star = HV_{t+1} \tag{2.42}$$

De même, une action optimale est donnée par :

$$A_{t+1} = \pi_t^\star(S_t) \in \arg\max_{a \in \mathcal{A}} [H_a V_{t+1}](S_t) \tag{2.43}$$

avec la fonctionnelle H_a définie par :

$$[H_a V](s) = \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s'|s,a) V(s') \tag{2.44}$$

Grâce à (2.40), il n'est pas nécessaire de conserver explicitement les $\pi_t^\star$ mais uniquement les fonction de valeur $V_t^\star$: si on connaît les lois $\mathcal{R}$ et $\mathcal{T}$ du problème, l'action optimale A_t à l'instant t peut être déterminée. Il s'agit de l'action gourmande par rapport à la fonction de valeur $V_t^\star$ de la sous-politique optimale. De nombreuses approches de planification cherchent donc à déterminer ou approcher les fonctions $V_t^\star$: ceci peut en particulier se faire par programmation dynamique de manière similaire au calcul de la fonction de valeur d'une politique donnée (ceci est expliqué dans la section 2.3.1).

2.2.5 Politique déterministe

Si on dispose d'une fonction de optimale valeur $V_t^\star$, on peut construire une politique optimale déterministe définie par (2.40). On peut donc restreindre notre étude aux politiques déterministes markoviennes. On notera une politique déterministe comme une fonction :

- $A_{t+1} = \pi_t(S_t)$ pour les politiques markoviennes ;
- $A_{t+1} = \pi(S_t)$ pour les politiques stationnaires.

Définition 2.2.4. Une politique déterministe est une politique où les probabilités de A_{t+1} conditionnées par l'historique sont déterministes :

$$\forall t < T, \forall h \in (\mathcal{A} \times \Omega)^t, \exists a \in \mathcal{A}, \pi(a|h) = 1 \tag{2.45}$$

2.3 Programmation dynamique

Nous présentons ici deux approches par programmation dynamique pour la planification dans le cas des observations totales. En horizon fini, ces approches construisent la politique optimale en horizon par horizon en commençant par la fin de la politique (la dernière action) et en construisant des politiques de plus en plus grandes. De manière plus générale, ces politiques construisent les politiques dans l'ordre inverse de l'ordre chronologique : on parle de méthodes *bottom-up* par opposition aux approches *top-down* qui raisonnent dans l'ordre chronologique.

L'approche classique consiste à calculer la fonction de valeur optimale en appliquant de manière itérée l'opérateur H d'optimalité de Bellman. Cette méthode est le fondement des approches par programmation dynamique, aussi bien dans le cadre des observations totales que dans les cas plus généraux. En particulier, les algorithmes de la famille de MBDP (*Memory Bounded Dynamic Programming*), qui constituent l'état de l'art de la planification dans le cas décentralisés, ainsi que l'approche que nous proposons se basent sur cette approche.

L'autre approche, PSDP (*Policy Search Dynamic Programming*), construit explicitement une politique en recherchant les règles de décision π_t dans un espace donné. Une distribution heuristique sur les états du système est utilisée pour guider cette recherche. L'approche que nous proposons se comporte de manière similaire dans le cas décentralisé.

2.3.1 Calcul de la fonction de valeur

Les théorèmes de Bellman et d'optimalité de Bellman peuvent être utilisés pour le calcul de la fonction de valeur d'une politique ou de la fonction de valeur optimale. Dans le cas, du calcul de la fonction de valeur optimale, ces méthode sont appelées *induction rétrograde*, en horizon fini, et *itération de la valeur*, en horizon infini [Bellman, 1957].

En horizon fini, les fonctions de valeur sont données par :

$$V_t^\pi = H_{\pi_t} H_{\pi_{t-1}} \dots H_{\pi_{T-1}} 0 \tag{2.46}$$
$$V_t^\star = H^{T-t} 0 \tag{2.47}$$

et peuvent être calculée par programmation dynamique :

$$V_T^\pi = 0 \qquad V_{t-1}^\pi = H_{\pi_{t-1}} V_t^\pi \tag{2.48}$$
$$V_T^\star = 0 \qquad V_{t-1}^\star = H V_t^\star \tag{2.49}$$

En horizon infini, on s'intéresse aux politiques stationnaires. La fonction de valeur d'une politique stationnaire π et la fonction de valeur optimale ne dépendent pas du temps et sont solutions des systèmes respectifs :

$$V^\pi = H_\pi V^\pi \tag{2.50}$$
$$V^\star = H V^\star \tag{2.51}$$

Comme les opérateurs H_π et H sont des γ-contractions [Bertsekas, 1987], le théorème du point fixe donne une méthode pour déterminer le point fixe :

$$V^\pi = H_\pi^\infty V \qquad \forall V \in \mathbb{R}^{\mathcal{S}} \tag{2.52}$$
$$V^\star = H^\infty V \qquad \forall V \in \mathbb{R}^{\mathcal{S}} \tag{2.53}$$

Les fonctions de valeur peuvent être calculées par programmation dynamique. L'application itérée de l'opérateur de Bellman permet de déterminer des approximations de plus en plus précises de la fonction de valeur en partant d'une valeur initiale quelconque :

$$\hat{V}^\pi \leftarrow H_\pi \hat{V}^\pi \tag{2.54}$$
$$\hat{V}^\star \leftarrow H \hat{V}^\star \tag{2.55}$$

Ces résultats sont importants pour la planification dans les cas plus généraux. En effet, les résultats sur les fonctions de valeur peuvent être utilisés pour les problèmes centralisés à

observations partielles : la planification peut y être formulée comme un problème de calcul d'une fonction de valeur optimale. Le passage aux problèmes décentralisés est moins immédiat car la fonction de valeur ne peut pas être utilisée.

D'autres méthodes pour calculer la fonction de valeur optimale ne sont pas développées ici. L'*itération de politique* peut être vu comme une variante de l'itération de la valeur qui travaille dans l'espace des politiques en alternant une phase d'évaluation de politique et une phase d'amélioration de politique [Littman *et al.*, 1995]. Une autre solution pour déterminer la fonction de valeur optimale en horizon infini et de la chercher comme solution d'un programme linéaire [Littman *et al.*, 1995].

2.3.2 PSDP

L'algorithme PSDP [Bagnell *et al.*, 2003] (*Policy Search Dynamic Programming* – Programmation dynamique par recherche de politique) est un algorithme de recherche d'une politique approchée en horizon fini. Il cherche les règles de décision π_t dans un sous-espace donné $\hat{\Xi}$ de l'ensemble $\Xi = \mathcal{A}^{\mathcal{S}}$ des règles de décisions déterministes, c'est-à-dire des fonctions des états dans les actions. Typiquement, il s'agit d'une classe paramétrée de politiques. Cela est en particulier utile quand le nombre d'états est très important ou continu : dans ces cas, le calcul exact de la fonction de valeur par programmation dynamique n'est pas réalisable.

Pour cela, des distributions de probabilité *a priori* heuristiques μ_t sur les états S_t sont introduites. Intuitivement, ces distributions indiquent quels états sont importants pour la planification, parce qu'ils sont probablement visités quand l'agent suit une bonne politique. Une telle distribution de probabilité sur les états peut représenter une connaissance experte sur le problème [Kakade, 2003].

L'espace de recherche $\hat{\Pi} = \hat{\Xi}^T \subset \Pi = \Xi^T$ de $\pi = (\pi_0, \ldots, \pi_{T-1})$ ne contient pas forcément de politique optimale $\pi^\star$ de Π. De plus, il n'existe pas forcément de politique de $\hat{\Pi}$ au moins aussi bonne que toutes les autres *pour tous les états* $s \in \mathcal{S}$. On cherche donc une politique optimale pour un état $s_0 \in \mathcal{S}$ donné :

$$\hat{\pi}^\star \in \arg\max_{\pi \in \hat{\Pi}} V^\pi(s_0)$$

PSDP cherche une politique par programmation dynamique *bottom-up*. Étant donné les règles de décision π_{t+1} jusqu'à π_{T-1}, il cherche la règle π_t. Contrairement au cas non-contraint, il n'y a en général pas de politique $(\pi_t, \ldots, \pi_{T-1})$ qui soit au moins aussi bonne que toutes les autres $(\pi'_t, \ldots, \pi_{T-1})$ et on ne sait donc pas comment choisir π_t. Pour résoudre ce problème, PSDP introduit une distribution heuristique *a priori* $\mu_t \in \Delta\mathcal{S}$ de chaque état S_t. Ceci permet de définir un ordre entre les différents π_t :

$$\pi_t \leq \pi'_t \Leftrightarrow \mathop{\mathrm{E}}_{s\sim\mu_t} V^{(\pi_t,\ldots,\pi_{T-1})}(s) \leq \mathop{\mathrm{E}}_{s\sim\mu_t} V^{(\pi'_t,\ldots,\pi_{T-1})}(s) \tag{2.56}$$

PSDP utilise cette heuristique pour chercher le meilleur π_t étant donné cette probabilité de S_t :

$$\pi_t \in \arg\max_{\xi\in\Xi} \mathop{\mathrm{E}}_{s\sim\mu_t} \left[V^{\xi,\pi_{t+1},\ldots\pi_{T-1}}(s) \right] \tag{2.57}$$

Une version ϵ-approchée de la recherche peut être utilisée :

$$\mathop{\mathrm{E}}_{s\sim\mu_t} \left[V^{(\pi_t,\ldots,\pi_{T-1})}(s) \right] \geq \max_{\xi\in\Xi} \mathop{\mathrm{E}}_{s\sim\mu_t} \left[V^{(\xi,\ldots,\pi_{T-1})}(s) \right] - \epsilon \tag{2.58}$$

Une garantie de performance théorique indique que si la distribution heuristique μ utilisée est proche en norme 1 de la distribution $\mu^\star$ obtenue en suivant une politique $\hat{\pi}^\star$ optimale *dans l'ensemble de recherche* Π pour s_0, c'est-à-dire

$$\hat{\pi}^\star \in \arg\max_{\pi \in \hat{\Pi}} V^\pi(s_0) \tag{2.59}$$

alors la valeur de la politique trouvée est proche de celle de $\hat{\pi}^\star$.

Théorème 2.3.1. *Si π est obtenue par PSDP ϵ-approchée alors pour toute politique $\pi' \in \hat{\Pi}$,*

$$V^\pi(s_0) \geq V^{\pi'}(s_0) - T(R_{\max} - R_{\min}) d(\mu, \mu') - T\epsilon \tag{2.60}$$

où μ' est la distribution obtenue en suivant π' depuis s_0 :

$$\mu'_t(s') = \Pr(S_t = s' | S_0 = s_0, \pi') \tag{2.61}$$

en notant d la distance en norme 1,

$$d(\mu, \mu') = \|\mu - \mu'\|_1 = \sum_{t=0}^{T-1} \|\mu_t - \mu'_t\|_1 = \sum_{t=0}^{T-1} \sum_{s \in \mathcal{S}} |(\mu_t - \mu'_t)(s)|_1 \tag{2.62}$$

et avec

$$R_{\min} = \inf_{(s,a) \in \mathcal{S} \times \mathcal{A}} \mathcal{R}(s, a) \tag{2.63}$$

$$R_{\max} = \sup_{(s,a) \in \mathcal{S} \times \mathcal{A}} \mathcal{R}(s, a) \tag{2.64}$$

Plus généralement,

$$V^\pi(\mu'_0) \geq V^{\pi'}(\mu'_0) - T(R_{\max} - R_{\min}) d(\mu, \mu') - T\epsilon \tag{2.65}$$

avec

$$\mu'_t(s') = \Pr(S_t = s' | S_0 \sim \mu'_0, \pi') \tag{2.66}$$

Démonstration. [Bagnell *et al.*, 2003] □

Corollaire. *En particulier, pour $\pi' = \hat{\pi}^\star$, on obtient une borne sur l'erreur commise par rapport à la meilleure politique $\hat{\pi}^\star$:*

$$V^\pi(s_0) \geq V^{\hat{\pi}^\star}(s_0) - T(R_{\max} - R_{\min}) d(\mu, \mu^\star) - T\epsilon \tag{2.67}$$

PSDP montre comment une distribution heuristique sur les états peut être utilisée pour guider la recherche d'une politique sous-optimale appartenant à une classe donnée de politiques. Ceci est intéressant car nous proposons d'interpréter les méthodes de planification décentralisée de l'état de l'art comme des algorithmes de recherche de politique dans un sous-espace donné. Dans cette thèse, nous utilisons une distribution heuristique pour guider la recherche d'une politique décentralisée de manière similaire à ce que fait PSDP dans le cas centralisé avec observabilité totale.

2.4 Conclusion

Le modèle MDP permet de représenter des problèmes de décision séquentielle dans lesquels l'agent connaît à chaque instant l'état actuel du système. Dans ce cadre, la théorie de la valeur indique que, pour connaître l'action optimale à effectuer à chaque instant dans un MDP, il suffit de déterminer la fonction de valeur optimale. Celle-ci peut être déterminée par programmation dynamique : les approches de planification par programmation dynamique pour les problèmes plus généraux que nous allons étudier dans les chapitre ultérieurs se fondent sur cette approche et la généralisent. PSDP est un algorithme de planification approché dans un sous-espace de politiques donné. Pour ce genre de problème, l'introduction d'une distribution heuristique sur les états permet de guider la recherche en induisant un ordre total sur les politiques. Dans cette thèse, nous proposons d'utiliser ce type d'approche pour la planification multi-agent approchée plutôt que les approches classiques à base de points.

Chapitre 3

POMDP

Sommaire

Le modèle MDP fait l'hypothèse forte de l'*observabilité totale* : l'agent connaît à chaque instant et avec exactitude l'état du système qui constitue donc un signal markovien et lui permet de prendre la décision optimale. Cette propriété n'est cependant pas vérifiée dans de nombreux problèmes : citons les exemples d'applications robotiques où l'agent n'a qu'une visibilité de son entourage immédiat, généralement fortement bruitée, plutôt qu'une connaissance exacte du système dans sa totalité. Afin de traiter les problèmes qui nous intéressent, nous sommes amenés à remplacer l'hypothèse d'observabilité totoale par une hypothèse plus générale d'*observabilité partielle*. L'agent ne connaît plus forcément à chaque instant t l'état S_t du système mais reçoit une observation O_t qui dépend de manière stochastique de l'état du système. L'observation peut être une information partielle, une mesure bruitée, etc.

Exemple 3.0.1*:* Revenons sur le problème du pompier. Il ne peut observer directement que l'état de la maison dans laquelle il se trouve. De plus, cette observation peut être imprécise ou bruitée. Le pompier ne connaît donc pas avec exactitude l'état global de l'incencie mais peut essayer de l'inférer à partir d'un modèle de la dynamique de l'incendie, des observations ponctuelles qu'il a reçues sur l'état local de l'incendie.

Dans ce chapitre, nous présentons l'état de l'art pour ce type de problèmes modélisés par le formalisme POMDP. Les concepts de fonction de valeur et de fonction de valeur optimale peuvent être généralisés dans ce cadre mais sont définis sur un espace continu. La fonction de valeur possède cependant une structure particulière qui peut être exploitée pour la représenter en mémoire ainsi qu'effectuer les divers calculs. La planification revient de nouveau au calcul d'une fonction de valeur optimale. Nous rappelons de plus le lien entre le calcul de la fonction de valeur optimale et la construction de politiques en horizon fini : c'est ce qui permet de généraliser la planification par programmation dynamique au cas multi-agent (voir prochain chapitre).

Cependant, la taille nécessaire pour représenter les fonctions de valeur augmente considérablement après chaque application de l'opérateur de Bellman. Ceci limite l'utilisation de l'opérateur de Bellman exact. Des versions approchées de cet opérateur sont donc généralement utilisées. Les méthodes à base de points échantillonnent des points de l'espace de définition des fonctions de valeur et calculent des approximations qui sont bonnes pour ces points. Les méthodes de l'état de l'art de planification multi-agent approchée généralisent les méthodes à base de point du cas mono-agent.

3.1 Problème

3.1.1 Modèle

Le modèle POMDP (*Partially Observable Markov Decision Process* – Processus Décisionnel de Markov Partiellement Observable) [Sondik, 1971] généralise le modèle MDP pour les problèmes à observabilité partielle. Des variables d'observations O_t sont donc ajoutées. Ces observations permettent à l'agent d'inférer l'état S_t réel du système. L'observation O_t peut dépendre de l'ancien état S_{t-1}, du nouvel état S_t et de l'action A_t (figure 3.1) :

$$\Pr(O_{t+1} = o | S_t = s, A_{t+1} = a, S_{t+1} = s') = \mathcal{O}(o|s, a, s') \tag{3.1}$$

Initialement, l'agent a une croyance initiale B_0 sur les états S_0 :

$$B_0(s) = \Pr(S_0 = s) \tag{3.2}$$

Le choix de l'action A_{t+1} ne peux dépendre que de la croyance initiale B_0 ainsi que des actions et observations $A_1, O_1, \ldots A_t, O_t$. Nous appelons historique $H_t = (B_0, A_1, O_1, \ldots, A_t, O_t)$ le tuple constitué de ces variables. La figure 3.3 représente le modèle POMDP sous forme de DBN (comparer au modèle MDP, figure 2.2).

Définition 3.1.1. Un POMDP est défini par un tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O} \rangle$ où :
- $\mathcal{S}$, $\mathcal{A}$, $\mathcal{T}$, $\mathcal{R}$ sont définis comme pour un MDP ;
- Ω est l'ensemble des observations que peut recevoir l'agent, $\forall t, O_t \in \Omega$;
- $\mathcal{O}$ définit la probabilité d'observer o lors d'une transition de s à s' avec l'action a :

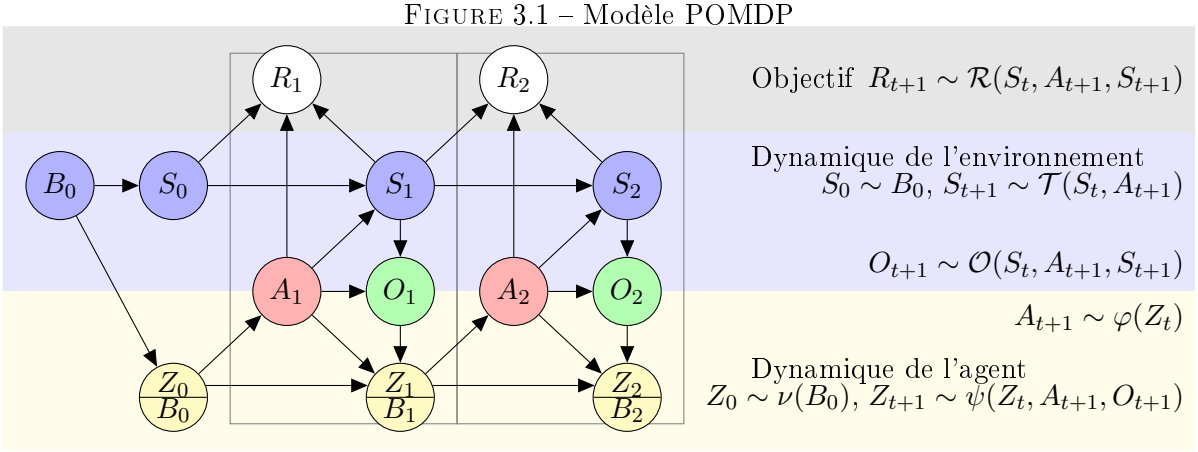
$$\mathcal{O}(o|s, a, s') = \Pr(O_{t+1} = o | S_t = s, A_{t+1} = a, S_{t+1} = s') \tag{3.3}$$

Théorème 3.1.1 (MDP sous-classe des POMDP). *Un MDP* $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ *est un POMDP* $\mathcal{M}' = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O} \rangle$ *où l'agent observe toujours l'état courant :*

$$\Omega = \mathcal{S} \tag{3.4}$$

$$O_t = S_t \text{ } c'est\text{-}à\text{-}dire \text{ } \mathcal{O}(o|s,a,s') = \begin{cases} 1 \text{ } si \text{ } s' = o \\ 0 \text{ } sinon \end{cases} \tag{3.5}$$

FIGURE 3.1 – Modèle POMDP



Exemple 3.1.1*:* Revenons sur l'exemple des pompiers. L'observabilité locale consiste à poser : $O_t = S_t(A_t)$, $\Omega = \mathcal{L}$. On peut ajouter du bruit en ajoutant une probabilité d'observation erronée. On peut aussi considérer un modèle où l'agent n'a pas accès à l'intensité de l'incendie mais peut uniquement percevoir si il y a un incendie ($O_t = 1$) ou non ($O_t = 0$). On introduit par exemple les probabilités (tirées de [Oliehoek *et al.*, 2008]) :

$$\Pr(O_{t+1} = 1 | S_{t+1}^{A_{t+1}} = 0) = 0.2 \qquad \Pr(O_{t+1} = 0 | S_{t+1}^{A_{t+1}} = 0) = 0.8 \tag{3.6}$$

$$\Pr(O_{t+1} = 1 | S_{t+1}^{A_{t+1}} = 1) = 0.5 \qquad \Pr(O_{t+1} = 0 | S_{t+1}^{A_{t+1}} = 1) = 0.5 \tag{3.7}$$

$$\Pr(O_{t+1} = 1 | S_{t+1}^{A_{t+1}} > 1) = 1 \qquad \Pr(O_{t+1} = 0 | S_{t+1}^{A_{t+1}} > 1) = 0 \tag{3.8}$$

Par abus de notation, nous notons aussi $\mathcal{O}$ la loi de O_{t+1} sachant S_t et A_{t+1} :

$$\mathcal{O}(o|s,a) = \Pr(O_{t+1} = o | S_t = s, A_{t+1} = a) = \sum_{s' \in \mathcal{S}} \mathcal{O}(o|s,a,s') \tag{3.9}$$

Nous notons $\mathcal{J}$ la loi jointe de (S_{t+1}, O_{t+1}) sachant (S_t, A_{t+1}) :

$$\mathcal{J}(s', o|s,a) = \Pr(S_{t+1} = s', O_{t+1} = o | S_t = s, A_{t+1} = a) = \mathcal{T}(s'|s,a)\mathcal{O}(o|s,a,s') \tag{3.10}$$

Contrairement au cas des observations totales, le signal que l'agent perçoit (les observations) ne constitue pas un signal markovien. Les méthodes du chapitre précédent ne peuvent donc pas être utilisée directement : une fonction de valeur définie sur les états ne peut pas être utilisée car l'agent n'a pas accès aux états et on ne peut pas définir une fonction de valeur sur les observations.

On pourrait faire l'approximation que le signal d'observations est markovien mais cette approximation n'est pas réaliste dans le cas général et on perdrait l'optimalité [Singh *et al.*, 1994]. La méthode classique de planification dans le cas POMDP consiste à introduire un signal markovien et à l'utiliser pour définir une fonction de valeur optimale.

3.1.2 Critère de performance

Le critère de performance du chapitre précédent devient dans le cas partiellement observable :

$$\text{maximiser } \mathrm{E}\left[\sum_{t=1}^{T} \gamma^{t-1} R_t | B_0 = b_0, \pi\right] \tag{3.11}$$

La planification cherche à calculer une politique qui maximise ce critère étant donné le modèle de transition $\mathcal{T}$, d'observations $\mathcal{O}$ et de récompenses $\mathcal{R}$, soit pour une croyance $B_0 = b_0$ initiale donnée, soit pour toute croyance initiale $B_0 = b_0$.

3.2 Politiques

3.2.1 État interne

De manière générale, le choix de l'action pour l'agent peut dépendre de la croyance initiale B_0, ainsi que des actions et observations passées. On peut définir une politique π comme l'ensemble des probabilités des actions A_{t+1} conditionnées par les historiques H_t :

$$\pi_t(a|h) = \Pr(A_{t+1} = a | H_t = h) \tag{3.12}$$

Ceci revient à maintenir un état interne Z_t associé à :
- des probabilités d'action conditionnées par l'état interne, que l'on notera φ

$$\varphi(a|z) = \Pr(A_{t+1} = a | Z_t = z) \tag{3.13}$$

- des probabilités de transition d'état interne conditionnées par les observations, que l'on notera ψ

$$\psi(z'|z, a, o) = \Pr(Z_{t+1} = z' | Z_t = z, A_{t+1} = a, O_{t+1} = o) \tag{3.14}$$

- des probabilités d'état interne initial conditionnées par la croyance initiale, que l'on notera ν

$$\nu(z|b) = \Pr(Z_0 = z | B_0 = b) \tag{3.15}$$

L'état interne peut être de diverses natures. Il peut s'agire de l'historique complet H_t mais aussi, comme nous allons le voir, d'un sous-arbre d'un arbre de politique, de la croyance B_t de l'agent, etc.

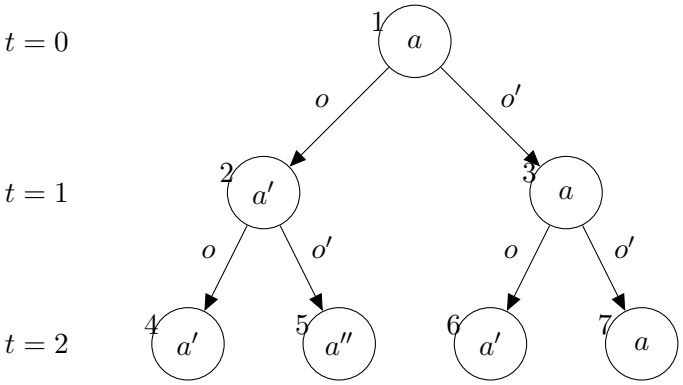
Remarque: Ce que nous appelons ici état interne de l'agent est proche de la notion d'*état d'information*. Les variables I_t sont des états d'information si la séquence $(I_t)_t$ est une chaîne de Markov conditionnée par les actions et observations [Hauskrecht, 1997, Dutch et Scherrer, 2008] :

$$\Pr(I_{t+1} | I_0, I_1, \ldots, I_t, O_{t+1}, A_{t+1}) = \Pr(I_{t+1} | I_t, O_{t+1}, A_{t+1}) \tag{3.16}$$

Un état interne Z_t est un état d'information associé à une probabilité d'émission d'action.

3.2.2 Arbre de politique

Dans le cas de politiques déterministes en horizon fini T, on peut représenter l'état interne Z_t d'un agent comme un nœud d'un arbre de profondeur T : chaque nœud z est associé à une action et a un arc sortant par observation $o \in \Omega$. On appelle un tel arbre, un arbre de politique [Littman, 1994] (voir figure 3.2). À l'instant $t = 0$, l'état Z_0 interne de l'agent est la racine de l'arbre. À un instant t quelconque, l'état Z_t interne est un nœud de l'arbre situé à la profondeur t. À l'instant $t+1$, l'agent effectue l'action A_{t+1} associé à son état interne Z_t et passe dans l'état Z_{t+1} associé à l'observation O_{t+1} reçue. Par la suite, on identifie un nœud d'un arbre de politique avec le sous-arbre dont il est la racine.



$T = 3, \Omega = \{o, o'\}, \mathcal{A} = \{a, a', a''\}, T = 3$

FIGURE 3.2 – Exemple d'arbre de politique

Exemple 3.2.1*:* Avec l'arbre de politique représenté par la figure 3.2, l'agent est initialement dans l'état $Z_0 = 1$ et émet l'action $A_1 = a$. S'il reçoit l'observation $O_1 = o$, son nouvel état interne est $Z_1 = 2$ et il exécute alors l'action $A_2 = a'$; s'il reçoit l'observation $O_1 = o'$, son nouvel état interne est $Z_1 = 3$ et il exécute l'action $A_2 = a$. Dans le premier cas (état interne $Z_1 = 2$), s'il reçoit ensuite l'observation $O_2 = o$, son nouvel état interne est $Z_2 = 4$ et il exécute l'action $A_3 = a'$; s'il reçoit l'observation $O_2 = o'$, son nouvel état interne est $Z_2 = 5$ et il exécute l'action $A_3 = a''$. Dans le second cas (état interne $Z_1 = 3$), s'il reçoit l'observation $O_2 = o$, son nouvel état interne est $Z_2 = 6$ et il exécute l'action $A_3 = a'$; s'il reçoit l'observation $O_2 = o'$, son nouvel état interne est $Z_2 = 7$ et il exécute l'action $A_3 = a$.

Comme précédemment, la politique est définie par une loi φ de sélection d'action, une loi ψ de transition d'état interne et une loi ν de sélection d'état interne initial. Comme ces lois sont déterministes, elle seront notées comme des fonctions :

- la loi φ de sélection d'action associe à un arbre Z_t, l'action A_{t+1} de son nœud racine

$$A_{t+1} = \varphi(Z_t) \tag{3.17}$$

- la loi ψ de transition associe à un arbre Z_t et à une observation O_{t+1}, le sous-arbre Z_{t+1} associé à l'observation O_{t+1} :

$$A_{t+1} = \psi(Z_t, O_t) \tag{3.18}$$

– une loi de sélection initiale qui, à une croyance initiale B_0, associe un état interne initial Z_0 parmi un ensemble $\mathcal{Z}_0$ d'arbre de politiques de profondeur T :

$$Z_0 = \nu(B_0) \tag{3.19}$$

3.2.3 *Belief*-MDP

L'agent dispose initialement d'une croyance B_0 sur l'état initial S_0. À chaque instant t, l'agent connaît la croyance initiale ainsi que les actions et observations passées. La croyance B_t de l'agent à l'instant t sur l'état courant S_t est la probabilité de S_t étant donné l'historique H_t :

$$B_t(s) = \Pr(S_t = s|H_t) \tag{3.20}$$

Le calcul de la nouvelle croyance B_{t+1} ne dépend que de la croyance actuelle B_t, de la nouvelle action A_{t+1} et de la nouvelle action O_{t+1}. On note τ la fonction de mise à jour de croyance :

$$B_{t+1} = \tau(B_t, A_{t+1}, O_{t+1}) \tag{3.21}$$

L'agent n'a donc pas besoin de retenir l'historique complet H_t pour calculer les croyances mais uniquement de conserver la croyance actuelle à chaque instant : la croyance B_t résume l'ensemble du passé.

Théorème 3.2.1 (Mise à jour de croyance)**.** *La croyance $B_{t+1} = b'$ à l'instant $t+1$ ne dépend que de la croyance $B_t = b$ à l'instant t, de la nouvelle action $A_{t+1} = a$ et de la nouvelles observation $O_{t+1} = o$ et est donnée par :*

$$b'(s') = [\tau(b, a, o)](s') = \lambda \sum_{s \in \mathcal{S}} b(s) \mathcal{J}(s', o|s, a) \tag{3.22}$$

soit sous forme vectorielle, $b' = \tau(b, a, o) = \lambda \mathcal{J}_{o|a} b$, où λ est un facteur de normalisation et τ la fonction de mise à jour de croyance.

Démonstration.

$$b'(s') = \Pr(S_{t+1} = s'|B_0, A_1, O_1, \ldots, A_{t+1} = a, O_{t+1} = o) \tag{3.23}$$

$$= \frac{\Pr(S_{t+1} = s', O_{t+1} = o|B_O, A_1, O_1, \ldots, A_{t+1} = a)}{\Pr(O_{t+1} = o|B_0, A_1, O_1, \ldots, A_{t+1} = a)} \tag{3.24}$$

$$= \frac{\Pr(S_{t+1} = s', O_{t+1} = o|B_0, A_1, O_1, \ldots, A_{t+1} = a)}{\sum_{s' \in \mathcal{S}} \Pr(S_{t+1} = s', O_{t+1} = o|B_0, A_1, O_1, \ldots, A_{t+1} = a)} \tag{3.25}$$

$$= \frac{\bar{b}'(s')}{\sum_{\bar{s}' \in \mathcal{S}} \bar{b}'(\bar{s}')} = \lambda \bar{b}'(s') \text{ avec } \lambda = \frac{1}{\sum_{s' \in \mathcal{S}} \bar{b}'(s')} \tag{3.26}$$

où $\bar{b}'$ est la croyance non-normalisée

$$\bar{b}'(s') = \Pr(S_{t+1} = s', O_{t+1} = o|B_0, A_1, O_1, \ldots, A_{t+1} = a) \tag{3.27}$$

$$= \sum_{s \in \mathcal{S}} \Pr(S_{t+1} = s', O_{t+1} = o|S_{t-1} = s, A_{t+1} = a) \Pr(S_t = s|B_0, A_1, O_1, \ldots, A_t, O_t) \tag{3.28}$$

$$= \sum_{s \in \mathcal{S}} \mathcal{J}(s', o|s, a) b(s) \tag{3.29}$$

□

Remarque: La croyance B_t est une variable aléatoire puisqu'elle est une fonction des variables aléatoires $A_1, O_1, \ldots, A_t, O_t$. Elle est représentée figure 3.1.

Par commodité, on étend les notations sur les croyances :

$$\mathcal{T}(s'|b,a) = \Pr(S_{t+1} = s'|S_t \sim b, A_{t+1} = a) = \sum_{s\in\mathcal{S}} \mathcal{T}(s'|s,a)b(s) \tag{3.30}$$

$$\mathcal{O}(o|b,a) = \Pr(O_{t+1} = o|S_t \sim b, A_{t+1} = a) = \sum_{s\in\mathcal{S}} \mathcal{O}(o|s,a)b(s) \tag{3.31}$$

$$\mathcal{J}(s',o|b,a) = \Pr(S_{t+1} = s', O_{t+1} = o|S_t \sim b, A_{t+1} = a) = \sum_{s\in\mathcal{S}} \mathcal{J}(s',o|s,a)b(s) \tag{3.32}$$

$$\mathcal{R}(b,a) = \mathrm{E}[R_{t+1}|S_t \sim b, A_{t+1} = a] = \sum_{s\in\mathcal{S}} \mathcal{R}(s,a)b(s) \tag{3.33}$$

La croyance B_t est une statistique suffisante pour l'historique passé complet [Smallwood et Sondik, 1973]. Elle représente l'état subjectif de l'environnement du points de vue de l'agent. Elle peut être considérée comme un état car elle possède la propriété de Markov. Un POMDP peut donc être vu comme un MDP dont les états S'_t sont les croyances B_t de l'agent (figure 3.3) : ce MDP est appelé *Belief*-MDP. La croyance de l'agent est un signal markovien et peut être utilisée pour la prise de décision et il existe une politique optimale $\pi^\star$ qui détermine chaque action A_{t+1} à partir de la croyance B_t :

$$A_{t+1} = \pi^\star_t(B_t) \tag{3.34}$$

Ceci revient à prendre comme état interne de l'agent la croyance : $Z_t = B_t$.

Théorème 3.2.2 (Équivalence avec un MDP)**.** *Un POMDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O}\rangle$ peut être reformulé de manière équivalente comme un MDP $\mathcal{M}' = \langle \mathcal{S}', \mathcal{A}', \mathcal{T}', \mathcal{R}'\rangle$ dont les états sont les croyances B_t de $\mathcal{M}$:*

$$\mathcal{S}' = \Delta\mathcal{S} \tag{3.35}$$

$$\mathcal{A}' = \mathcal{A} \tag{3.36}$$

$$\mathcal{T}'(b'|b,a) = \sum_{o\in\Omega \ t.q.\ b'=\tau(b,a,o)} \mathcal{O}(o|b,a) \qquad \forall (b,b') \in (\Delta\mathcal{S})^2, \forall a \in \mathcal{A} \tag{3.37}$$

$$\mathcal{R}'(b,a) = \mathop{\mathrm{E}}_{s\sim b}[\mathcal{R}(s,a)] = \sum_{s\in\mathcal{S}} \mathcal{R}(s,a)b(s) \qquad \forall a \in \mathcal{A}, \forall b \in \Delta\mathcal{S} \tag{3.38}$$

$$S'_t = B_t \qquad \forall t \tag{3.39}$$

$$A'_t = A_t \qquad \forall t \tag{3.40}$$

$$O'_t = O_t \qquad \forall t \tag{3.41}$$

$$R'_t = R_t \qquad \forall t \tag{3.42}$$

où $\Delta\mathcal{S}$ désigne l'ensemble des distributions de probabilité sur $\mathcal{S}$.

Les résultats sur les MDP peuvent se généraliser dans le cadre des POMDP en travaillant dans le MDP des croyances. En particulier, on cherche généralement des politiques markoviennes ou stationnaires et déterministes. De même, les méthodes de planification de construction de

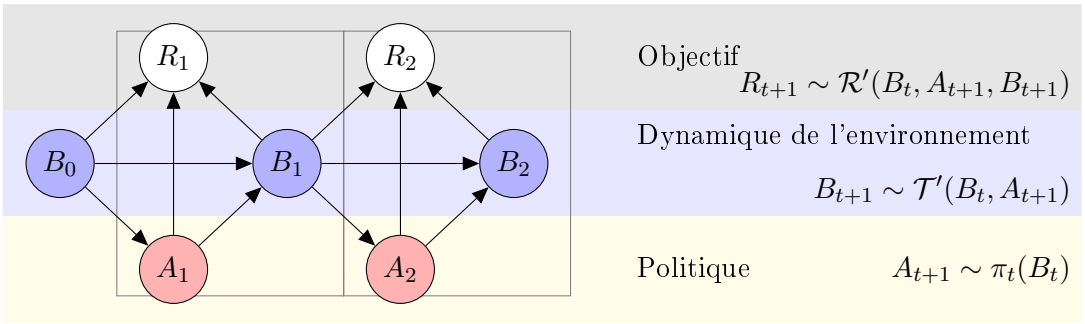


FIGURE 3.3 – POMDP vu comme un MDP sur les croyances

politique par calcul ou approximation de la fonction de valeur optimale sont utilisables dans le cadre POMDP. Cependant, l'espace des états du MDP des croyances est, dans le cas général, un ensemble continu ce qui empêche la représentation tabulaire des fonctions de valeur ou des politiques. Les fonctions de valeur POMDP possèdent cependant une forme spécifique qui peut être exploitée. pour représenter les fonctions de valeur et effectuer des calculs sur celles-ci.

Remarque: L'état interne Z_{t+1} de l'agent et la croyance B_{t+1} ont les mêmes dépendances : ils dépendent de la nouvelle action A_{t+1} et de la nouvelle observation O_{t+1} et respectivement de l'ancien état interne Z_t ou de l'ancienne croyance B_t. De plus, en général, la croyance B_t est prise comme état interne Z_t. Afin d'alléger la figure 3.3, ces variables sont représentées par le même nœud.

3.3 Fonction de valeur

3.3.1 Valeur d'une politique

Nous présentons ici la théorie de la valeur généralisée au cas des POMDP. Le théorème d'équivalence (théorème 3.2.2) implique que les fonctions de valeur dans un POMDP sont définies sur l'espace des croyances.

Définition 3.3.1. On définit la fonction de valeur V_t^π à l'instant t d'une politique π définie sur les croyances, comme l'espérance de la somme pondérée des récompenses :

$$V_t^\pi(b) = \mathrm{E}\left[\sum_{t'=t+1}^{T} \gamma^{t'-t-1} R_{t'} | S_t \sim b, B_t = b, \pi\right] \tag{3.43}$$

Les résultats concernant les fonctions de valeur des MDP se généralisent aux POMDP (équations de Bellman, propriétés des opérateurs de Bellman, optimalité). Pour une politique dont les actions ne dépendent que de la croyance courante, on peut écrire l'équation de Bellman :

$$V_t^\pi(b) = \left[H_{\pi_t} V_t^\pi(b)\right](b) = \sum_{a \in \mathcal{A}} \pi_t(a|b) \left[\mathcal{R}(b,a) + \gamma \sum_{b' \in \Delta\mathcal{S}} \mathcal{T}'(b'|b,a) V_{t+1}^\pi(b')\right] V_t^\pi(b) \tag{3.44}$$

D'après (3.37), on peut simplifier cette expression :

$$V_t^\pi(b) = \left[H_{\pi_t} V_{t+1}\right](b) = \sum_{a\in\mathcal{A}} \pi_t(a|b) \left[\mathcal{R}(b,a) + \gamma \sum_{o\in\Omega} \mathcal{O}(o|b,a) V_{t+1}^\pi(\tau(b,a,o))\right] \tag{3.45}$$

soit, pour les politiques déterministes,

$$V_t^\pi(b) = \left[H_{\pi_t} V_{t+1}\right](b) = \mathcal{R}(b,\pi_t(b)) + \gamma \sum_{o\in\Omega} \mathcal{O}(o|b,\pi_t(b)) V_{t+1}^\pi(\tau(b,\pi_t(b),o)) \tag{3.46}$$

Cependant, les fonctions de valeur considérées sont définies sur l'espace continu des croyances. Il n'est donc plus possible de les représenter de manière tabulaire.

3.3.2 Valeur optimale

D'après le théorème d'équivalence (théorème 3.2.2), on peut déterminer une fonction de valeur optimale $V_t^\star$ définie sur les croyances par programmation dynamique. Elle peut ensuite être utilisée pour déterminer les actions optimales.

Le principe d'optimalité de Bellman appliqué aux POMDP prend la forme :

$$V_t^\star(b) = \left[HV_{t+1}^\star\right](b) = \max_{a\in\mathcal{A}} \left[\mathcal{R}(b,a) + \gamma \sum_{b'\in\Delta\mathcal{S}} \mathcal{T}'(b'|b,a) V_{t+1}^\star(b')\right] \tag{3.47}$$

En utilisant (3.37), on a peut simplifier cette expression :

$$V_t^\star(b) = \max_{a\in\mathcal{A}} \left[\mathcal{R}(b,a) + \gamma \sum_{o\in\Omega} \mathcal{O}(o|b,a) V_{t+1}^\star(\tau(b,a,o))\right] \tag{3.48}$$

La politique optimale est obtenue en prenant l'action A_{t+1} gloutonne par rapport à la croyance B_t :

$$A_{t+1} = \pi_t^\star(B_t) \in \arg\max_{a\in\mathcal{A}} \left[H_a V_{t+1}^\star\right](B_t) \tag{3.49}$$

$$\in \arg\max_{a\in\mathcal{A}} \left[\mathcal{R}(B_t,a) + \gamma \sum_{o\in\Omega} \mathcal{O}(o|B_t,a) V_{t+1}^\star(\tau(B_t,a,o))\right] \tag{3.50}$$

Comme dans le cadre MDP, on peut chercher la fonction de valeur optimale $V^\star$ par programmation dynamique *bottom-up* en appliquant de manière itérée l'opérateur de Bellman H. Cependant, l'espace de définition de $V_t^\star$ est continu ce qui empêche d'utiliser une représentation tabulaire de la fonction de valeur.

3.3.3 Fonctions de valeur linéaires par morceaux convexes

Les fonctions de valeur linéaires par morceaux et convexes (PWLC - *Piecewise Linear and Convex*) peuvent être représentées par un ensemble Γ fini de vecteurs α à $|\mathcal{S}|$ dimensions, (figure 3.3.3) [Sondik, 1971],

$$\exists \Gamma \subset \mathbb{R}^{\mathcal{S}}, |\Gamma| \in \mathbb{N} \text{ t.q. } V(b) = \max_{\alpha\in\Gamma} \alpha \cdot b \tag{3.51}$$

Une telle fonction peut être représentée en mémoire par l'ensemble Γ. De plus, elles sont stables par l'application de l'opérateur de Bellman : l'application de l'opérateur de Bellman à une fonction PWLC reste PWLC. Les fonctions de valeur optimales $V_t^\star$ en horizon fini sont donc PWLC

et la fonction de valeur optimale $V^\star$ en horizon infini est convexe et peut donc être représentée par un ensemble Γ de taille éventuellement infinie de vecteurs α. On peut donc effectuer les algorithmes d'induction rétrograde et d'itération de la valeur (voir chapitre précédent, section 2.3.1) en déterminant les ensembles de vecteurs qui représentent les fonctions de valeur considérées.

Théorème 3.3.1. *Les fonctions de valeur linéaires par morceaux et convexes sont stables par H.*

Démonstration. Soit V' PWLC : $V'(b) = \max_{\alpha\in\Gamma'} \alpha \cdot b$. Alors la fonction $V = HV'$ prend la forme :

$$V(b) = \max_{a\in\mathcal{A}} \left[\mathcal{R}(b,a) + \gamma \sum_{o\in\Omega} \mathcal{O}(o|b,a)V'(\tau(b,a,o))\right] = \alpha_b \cdot b \tag{3.52}$$

en posant

$$\alpha_b(s) = \mathcal{R}(s,a) + \gamma \sum_{s',o} \mathcal{J}(s',o|s,a) \left(\left[\arg\max_{\alpha'\in\Gamma'} \left[\alpha' \cdot \tau(b,a,o)\right]\right)(s')\right] \tag{3.53}$$

$$a = \arg\max_{a\in\mathcal{A}} \left[\mathcal{R}(b,a) + \gamma \sum_{o\in\Omega} \mathcal{O}(o|b,a)V'(\tau(b,a,o))\right] \tag{3.54}$$

soit sous forme vectorielle,

$$\alpha_b = \mathcal{R}_a + \gamma \sum_{o\in\Omega} \bar{\mathcal{J}}_{o|a} \arg\max_{\alpha'\in\Gamma'} \left[\alpha' \cdot \tau(b,a,o)\right] \tag{3.55}$$

où $\mathcal{R}_a$ est le vecteur composé des récompenses immédiates composé des $\mathcal{R}(s,a)$:

$$\mathcal{R}_a(s) = \mathcal{R}(s,a) \tag{3.56}$$

et la fonction $\bar{\mathcal{J}}_{o|a}$ calcule la contribution d'un vecteur α'_o du sous-problème pour l'observation o :

$$[\bar{\mathcal{J}}_{o|a}\alpha'](s) = \sum_{s'\in\mathcal{S}} \mathcal{J}(s',o|s,a)\alpha'(s') \tag{3.57}$$

On a donc :

$$V(b) = [HV'](b) = \max_{\alpha\in\Gamma} b \cdot \alpha \tag{3.58}$$

avec

$$\Gamma = \left\{\mathcal{R}_a + \gamma \sum_{o\in\Omega} \bar{\mathcal{J}}_{o|a}\alpha'_o | a \in \mathcal{A}, \forall o \in \Omega, \alpha'_o \in \Gamma'\right\} \tag{3.59}$$

$|\Gamma|$ est fini et V est donc PWLC. □

Dans la figure 3.3.3, l'axe des abscisses représente l'espace des croyances : nous considérons ici un problème à deux états s_1 et s_2. L'axe des abcisses représente $b(s_1)$: comme $b(s_2) = 1-b(s_1)$, un point b de l'axe des abcisses représente une distribution de probabilité sur les états. En particulier, $b(s_1) = 0$ et $b(s_2) = 1$ à l'extrémité gauche ; $b(s_1) = 1$ et $b(s_2) = 0$ à l'extrémité droite. L'axe des ordonnées représente la valeur $V(b)$. Un vecteur α représente une fonction linéaire et est donc représentée par une droite. La fonction de valeur est représentée par l'enveloppe de ces différentes courbes.

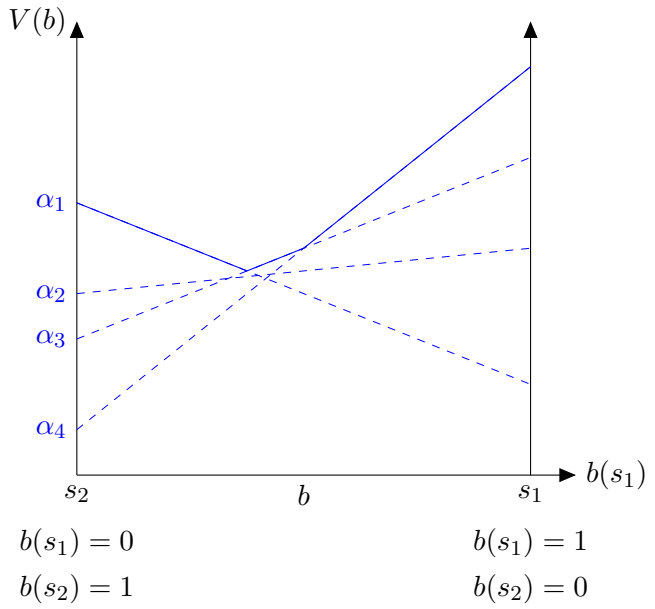


FIGURE 3.4 – Fonction de valeur linéaire par morceau et convexe

Théorème 3.3.2. *En horizon fini, les fonctions de valeur optimales pour chaque instant t sont PWLC :*

$$\exists \Gamma_t^\star \subset \mathbb{R}^{\mathcal{S}}, |\Gamma_t^\star| \in \mathbb{N} \ t.q. \ \forall b \in \Delta\mathcal{S}, V_t^\star(b) = \max_{\alpha \in \Gamma_t^\star} \alpha \cdot b \tag{3.60}$$

Démonstration. Preuve par induction en utilisant le théorème 3.3.1. □

Théorème 3.3.3. *En horizon infini, la fonction de valeur optimale $V^\star$ est convexe,*

$$\exists \Gamma^\star \subset \mathbb{R}^{\mathcal{S}}, \ \ t.q. \ \forall b \in \Delta\mathcal{S}, V^\star(b) = \max_{\alpha \in \Gamma^\star} \alpha \cdot b \tag{3.61}$$

Démonstration. L'ensemble des fonctions de valeur bornées convexes est un espace de Banach donc le point fixe $V^\star = HV^\star = H^\infty V$ (pour toute fonction V convexe) est une fonction de valeur convexe. □

3.3.4 MDP sous-jacent

Afin de guider la recherche d'une politique POMDP, il est parfois utile de résoudre d'abord le problème MDP obtenu en ignorant l'observabilité partielle : ce problème est plus simple à résoudre que le problème original et une politique optimale de ce problème est au moins aussi bonne qu'une politique optimale du problème original [Lovejoy, 1991]. Les politiques du MDP sous-jacent servent d'heuristique pour guider la recherche dans les approches de planification approchée décentralisées (par exemple, section 4.4 et chapitre 7).

Définition 3.3.2. Le MDP sous-jacent d'un POMDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O} \rangle$ est le MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ dans lequel l'agent observe directement l'état S_t de l'environnement.

Théorème 3.3.4. *Les fonctions de valeur optimales (notées $V^\star_{MDP}$, en horizon infini, et $V^\star_{MDP,t}$, en horizon fini) du MDP sous-jacent d'un POMDP sont supérieures ou égales aux fonctions de valeur optimales du POMDP :*

$$V^\star_{MDP} \geq V^\star \quad soit \quad V^\star_{MDP}(s) \geq V^\star(s) \quad \forall s \in \mathcal{S} \tag{3.62}$$

$$V^\star_{MDP,t} \geq V^\star_t \quad soit \quad V^\star_{MDP,t}(s) \geq V^\star_t(s) \quad \forall s \in \mathcal{S} \tag{3.63}$$

Démonstration. L'ensemble des politiques POMDP est un sous-ensemble des politiques (stochastiques) MDP. □

3.4 Programmation dynamique

Nous montrons ici comment les approches de planification par programmation dynamique se généralisent du cas MDP au cas POMDP en calculant une fonction de valeur définie sur les croyances : nous montrons comment appliquer l'opérateur de Bellman pour calculer pour chaque instant t la fonction de valeurs optimales $V^\star_t$. Nous montrons ensuite les version approchées de ces algorithmes qui forment la base des principaux algorithmes de planification approchée dans le cas décentralisé.

3.4.1 Mise à jour

D'après la preuve du théorème (3.59), le calcul d'une application de $V = HV'$ de l'opérateur de Bellman pour une fonction V' PWLC peut être fait en calculant sa représentation Γ en utilisant comme opérateur H l'opérateur de mise à jour $H_{\text{màj}}$ (algorithme 3.1) défini par :

$$\Gamma = H_{\text{màj}}\Gamma' = \left\{ \mathcal{R}_a + \gamma \sum_{o \in \Omega} \bar{\mathcal{J}}_{o|a} \alpha'_o \middle| a \in \mathcal{A}, \forall o \in \Omega, \alpha'_o \in \Gamma' \right\} \tag{3.64}$$

où les notations $\mathcal{R}_a$ et $\bar{\mathcal{J}}_{o|a}$ sont expliquées dans la preuve du théorème (3.59). L'opérateur de mise à jour construit $|\mathcal{A}||\Gamma'|^{|\Omega|}$ vecteurs α.

Algorithme 3.1 : Opérateur de Bellman sans élagage

Données : Fonction de valeur V' représentée par l'ensemble Γ'
Résultat : Fonction de valeur $V = HV'$ représentée par l'ensemble Γ
1 $\Gamma \leftarrow \left\{ \mathcal{R}_a + \gamma \sum_o \bar{\mathcal{J}}_{o|a} \alpha'_o | a \in \mathcal{A}, \forall o \in \Omega, \alpha'_o \in \Gamma' \right\}$ `// Mise à jour`

En horizon fini, les fonctions de valeur optimales sont données par :

$$V^\star_t = H^{T-t} 0 \tag{3.65}$$

et peuvent être déterminées par programmation dynamique par l'algorithme d'induction rétrograde *bottom-up* :

$$\Gamma^\star_T = 0 \tag{3.66}$$

$$\Gamma^\star_{t-1} = H\Gamma^\star_t \tag{3.67}$$

avec $H = H_{\text{màj}}$. L'ensemble $\Gamma^\star_0$ des vecteurs α construits pour représenter $V^\star_0$ est de taille doublement exponentielle avec l'horizon de planification ce qui limite le passage à l'échelle de cette approche :

$$|\Gamma^\star_0| = |\mathcal{A}|^{\frac{|\Omega|^{T-1}}{|\Omega|-1}} \tag{3.68}$$

En horizon infini, comme dans le cas MDP, la fonction de valeur $V^\star$ est le point fixe de l'opérateur de Bellman H quand $\gamma < 1$. On peut appliquer le même principe de programmation dynamique *bottom-up* pour la planification en horizon infini : $V^\star = H^\infty V$ pour tout $V \in \mathbb{R}^{\Delta\mathcal{S}}$. L'algorithme d'itération de la valeur part d'une fonction de valeur PWLC arbitraire et calcule à chaque itération une nouvelle approximation PWLC de la fonction de valeur optimale :

$$\hat{\Gamma}^\star \leftarrow H\hat{\Gamma}^\star \tag{3.69}$$

La fonction de valeur optimale $V^\star$ est convexe mais n'est pas linéaire par morceaux et sa représentation exacte peut nécessiter un nombre infini de vecteurs α : s'il est théoriquement possible de s'approcher arbitrairement de la fonction de valeur optimale, le nombre de vecteurs α nécessaires pour représenter une fonction de valeur d'une précision $\epsilon > 0$ donnée peut être arbitrairement grand.

3.4.2 Élagage

Certains des vecteurs α d'une représentation Γ d'une fonction de valeur PWLC V peuvent ne jamais contribuer au max : par exemple, pour la fonction de valeur représentée dans la figure 3.3.3, le vecteur α_2 ne contribue jamais à la fonction de valeur et peut donc être éliminé.

Après chaque application de l'opérateur de mise à jour, les vecteurs inutiles peuvent être éliminés de la représentation Γ pour produire une représentation minimale de la fonction de valeur V [Sondik, 1971, Zhang et Liu, 1996, Cassandra *et al.*, 1997, Feng et Zilberstein, 2004, Varakantham *et al.*, 2005] : on appelle ce procédé, l'*élagage* de l'ensemble Γ. Ceci limite le nombre de vecteurs α à construire pour les itérations suivantes. Cependant de manière générale, le nombre de vecteurs α à conserver reste du même ordre de grandeur.

Un vecteur α peut être supprimé de l'ensemble Γ, si, en tout point $b \in \Delta\mathcal{S}$, il existe un autre vecteur $\bar{\alpha} \in \Gamma$ tel que $\bar{\alpha} \cdot b \geq \alpha \cdot b$. Ceci peut être peut être effectué en résolvant le programme linéaire :

$$\begin{aligned} &\text{maximiser } \epsilon && && (3.70)\\ &\quad \text{sujet à } b \cdot (\alpha - \bar{\alpha}) \geq \epsilon && \forall \bar{\alpha} \in \Gamma \\ &\qquad\quad b \in \Delta\mathcal{S} \\ &\qquad\quad \epsilon > 0 \end{aligned}$$

Si aucun tel b n'existe, le vecteur α peut être supprimé.

L'opérateur de Bellman peut être effectué en deux phrases :

- la phase de mise à jour construit l'ensemble des vecteurs α possibles ;
- la phase d'élagage supprime les vecteurs α inutiles.

L'algorithme 3.2 [Sondik, 1971] est un exemple d'utilisation de cette méthode qui vérifie la condition sur tous les $\alpha \in \Gamma$.

Algorithme 3.2 : Opérateur de Bellman avec élagage

Données : Fonction de valeur V' à l'instant $t+1$ représentée par l'ensemble Γ'
Résultat : Fonction de valeur $V = HV'$ à l'instant t représentée par l'ensemble Γ

```
Γ ← {R_a + γ Σ_o J̄_{o|a} α'_o | a ∈ A, ∀o ∈ Ω, α'_o ∈ Γ'}        // Mise à jour
pour chaque α ∈ Γ faire
    Si α dominé, Γ ← Γ \ {α}                                    // Élagage
```

3.4.3 *Lookahead*

Une limite des approches à base de mise à jour exhaustive et élagage est qu'elles construisent tout d'abord l'ensemble Γ de tous les vecteurs α construits à partir de l'ensemble Γ' avant de faire l'élagage alors qu'une grande partie de Γ peut être complètement inutile. L'opération de *lookahead* peut être utilisée pour construire directement un ensemble Γ minimal sans passer par une mise à jour exhaustive.

L'opération de *lookahead* [Littman, 1994] consiste à déterminer un des vecteurs α construits à partir d'un ensemble Γ' qui est optimal pour une croyance b donnée :

$$\alpha = \mathcal{R}_a + \gamma \sum_{o \in \Omega} \bar{\mathcal{J}}_{o|a} \alpha'_o \tag{3.71}$$

où l'action a et les sous-vecteurs α'_o sont obtenus en maximisant le critère :

$$\max_{\substack{a \in \mathcal{A} \\ \alpha' \in (\Gamma')^\Omega}} \left[\mathcal{R}(b,a) + \gamma \sum_{o \in \Omega} \mathcal{O}(o|b,a)(\alpha'_o \cdot \tau(b,a,o)) \right] \tag{3.72}$$

Ce vecteur α peut être trouvé de manière efficace sans parcourir l'ensemble des α possibles, ou de manière équivalente, l'ensemble $\mathcal{A} \times (\Gamma')^\Omega$ [Littman, 1994] (algorithme 3.3) :

$$\max_{a \in \mathcal{A}} \left[\mathcal{R}(b,a) + \gamma \sum_{o \in \Omega} \mathcal{O}(o|b,a) \max_{\alpha'_o \in \Gamma'} (\alpha'_o \cdot \tau(b,a,o)) \right] \tag{3.73}$$

Intuitivement pour une action a donnée, le choix du vecteur α_o à utiliser pour une observation o donnée est indépendante des vecteurs $\alpha_{o'}$ à utiliser pour les autres observations o'. Pour chaque $a \in \mathcal{A}$, au lieu de parcourir l'ensemble $(\Gamma')^\Omega$ de taille exponentielle par rapport au nombre d'observations, l'ensemble Γ' est parcouru Ω fois (algorithme 3.4).

Algorithme 3.3 : *Lookahead*

Données : Fonction de valeur V' à l'instant $t+1$ représentée par l'ensemble Γ' ;
Croyance b à l'instant t
Résultat : Meilleur α à l'instant t pour b
1 $\alpha \leftarrow \arg\max_{\alpha \in \{\texttt{lookahead}(\Gamma', b, a) | a \in \mathcal{A}\}} \alpha \cdot b$, départager les égalités avec $\prec$

Algorithme 3.4 : *Lookahead* pour une action donnée

Données : Fonction de valeur V' représentée par l'ensemble Γ' ;
Croyance b ;
Action jointe a
Résultat : Meilleur α pour b construit à partir de a
1 **pour chaque** $o \in \Omega$ **faire**
2 $\quad \alpha'_o \leftarrow \arg\max_{\alpha' \in \Gamma'} \alpha' \cdot \tau(b,a,o)$, départager les égalités avec $\prec$
3 $\alpha \leftarrow \mathcal{R}_a + \gamma \sum_o \bar{\mathcal{J}}_{o|a} \alpha'_{b,a,o}$

Ceci permet de définir une représentation minimale de Γ :

$$\Gamma = \{\texttt{lookahead}(\Gamma', b) | b \in \Delta\mathcal{S}\} \tag{3.74}$$

Comme le nombre de vecteurs Γ est fini, le nombre de croyance b à considérer dans la pratique est fini. Toute une famille d'algorithmes [Sondik, 1971, Cheng, 1988, Littman, 1994] cherche des croyances b qui conduisent à la création de nouveau vecteurs α par *lookahead*. Ils construisent incrémentalement l'ensemble Γ en répétant itérativement deux étapes (algorithmes 3.5) :

1. recherche d'une croyance b qui peut conduire à la création d'un nouveau α utile ;
2. construction par *lookahead* du meilleur vecteur α pour b et ajout de celui-ci à Γ si il est meilleur pour b.

Remarque: Un choix arbitraire (toujours le même) doit être fait pour les α en cas d'égalité. Pour cela, on introduit un ordre $\prec$ entre vecteurs α. On peut par exemple utiliser l'ordre lexicographique.

Algorithme 3.5 : Opérateur de Bellman avec *lookahead* (principe général)

Données : Fonction de valeur PWLC V' représentée par Γ'
Résultat : Fonction de valeur PWLC $V = HV'$ représentée par Γ

$\Gamma \leftarrow \emptyset$
Choisir une croyance b
tant que $b \neq$ null **faire**
  $\alpha \leftarrow \texttt{lookahead}(b, \Gamma')$
  si $\alpha \cdot b > \max_{\bar{\alpha} \in \Gamma} \bar{\alpha} \cdot b$ **alors** $\Gamma \leftarrow \Gamma \cup \{\alpha\}$
  Choisir une croyance b

Outre l'utilisation pour la planification exacte, l'opération de *lookahead* est aussi utilisée pour la planification approchée à base de points que nous allons voir par la suite. Elle peut de plus se généraliser au cas multi-agent mais elle est plus complexe dans ce cas [Dibangoye *et al.*, 2009] : intuitivement le choix des α'_o n'est plus indépendant pour les différentes observations du fait du contrôle décentralisé. Cette généralisation constitue le cœur de l'algorithme PBIP, présenté dans le prochain chapitre (section 4.4.4).

3.4.4 *Witness*

L'algorithme *Witness* (témoin) [Littman, 1994] est un exemple d'un algorithme à base de *lookahead*. Nous le décrivons ici principalement pour illustrer le principe général des algorithmes de planification à base de *lookahead* qui constituent la base des algorithmes de planification approchée à base de points dans le cas centralisé mais aussi, comme noue le verrons dans le prochain chapitre, dans le cas décentralisé.

La mise à jour $V = HV'$ peut être décomposée en une mise à jour séparée pour chaque action $a \in \mathcal{A}$, en introduisant les fonction de valeur d'action [Sutton et Barto, 1998] définies comme :

$$Q_a(b) = \left[H_a V'\right](b) \tag{3.75}$$

$$= \mathcal{R}(b,a) + \gamma \sum_{o \in \Omega} \mathcal{O}(o|b,a) V'(\tau(b,a,o)) \tag{3.76}$$

$$= \mathcal{R}(b,a) + \gamma \sum_{o \in \Omega} \mathcal{O}(o|b,a) \max_{\alpha' \in \Gamma'} \left[\alpha' \cdot \tau(b,a,o)\right] \tag{3.77}$$

soit $Q_a = H_a V'$ où H_a est appelé opérateur de Bellman pour l'action a. Si V' est PWLC, les fonctions Q_a sont aussi PWLC et sont représentés par des ensembles Γ_a de vecteurs α :

$$\exists \Gamma_a \subset \mathbb{R}^{\mathcal{S}}, |\Gamma_a| \in \mathbb{N} \text{ t.q. } Q_a(b) = \max_{\alpha \in \Gamma_a} \alpha \cdot \Gamma \tag{3.78}$$

La fonction de valeur V est ensuite obtenue à partir des fonctions Q_a par :

$$V(b) = \max_{a \in \mathcal{A}} Q_a(b) = \max_{a \in \mathcal{A}} \max_{\alpha \in \Gamma_a} \alpha \cdot b = \max_{\alpha \in \bigcup_{a \in \mathcal{A}} \Gamma_a} \alpha \cdot b \tag{3.79}$$

La fonction V peut donc être représentée comme l'union Γ, éventuellement élaguée, des Γ_a :

$$\Gamma = \texttt{élaguer}\left(\bigcup_{a \in \mathcal{A}} \Gamma_a \right) \tag{3.80}$$

L'algorithme *Witness* [Littman, 1994] construit les ensembles Γ_a qui représentent les fonctions Q_a par *lookahead*. L'intérêt de construire les fonctions Q_a séparément est qu'il existe une méthode efficace pour trouver une croyance b pour laquelle la représentation Γ_a de la fonction Q_a n'est pas encore optimale : on appelle *témoin* une telle croyance b car il témoigne du fait que la représentation Γ_a n'est pas encore complète. Le théorème *Witness* fournit une méthode pour trouver un témoin : pour chaque tuple formé d'un vecteur $\alpha \in \Gamma$ déjà construit, d'une observation $o \in \Omega$ et d'un vecteur $\alpha' \in \Gamma'$ du sous-problème, l'algorithme de recherche de témoin cherche à déterminer une croyance b pour laquelle la contribution du vecteur $\alpha' \in \Gamma$ du sous-problème est meilleure que la contribution $\alpha'_o \in \Gamma'$ qui a contribué à α pour l'observation o. Cette croyance b peut être cherché par un programme linéaire :

$$\begin{aligned} &\text{maximise } c = \bar{\mathcal{J}}_{o|a} \left(\alpha' - \alpha'_o \right) \cdot b && && (3.81) \\ &\text{avec } b \cdot \alpha \geq b \cdot \bar{\alpha} && \forall \bar{\alpha} \in \Gamma_a \\ &\quad b \in \Delta \mathcal{S} \end{aligned}$$

La croyance b est un témoin si le critère est strictement positif. L'algorithme *Witness* 3.7 construit l'ensemble Γ_a incrémentalement en alternant la recherche d'un témoin b et le calcul par *lookahead* d'un nouveau vecteur α à ajouter à l'ensemble Γ_a.

Théorème 3.4.1. *(Théorème Witness) La fonction Q_a définie par $Q_a = H_a V'$ diffère de son approximation $\hat{Q}^a$ définie par un ensemble Γ incomplet*

$$\hat{Q}^a(b) = \max_{\alpha \in \Gamma_a} \alpha \cdot b \tag{3.82}$$

si et seulement si un vecteur $\alpha \in \Gamma_a$, une observation $o \in \Omega$, et une croyance $b \in \Delta \mathcal{S}$, et un vecteur $\alpha' \in \Gamma'$ tel que

$$\bar{\mathcal{J}}_{o|a} \alpha' \cdot b > \bar{\mathcal{J}}_{o|a} \alpha'_o \cdot b \tag{3.83}$$

où α_o désigne le vecteur de Γ' qui a été utilisé pour contribution de α pour l'observation o :

$$\alpha = \mathcal{R}_a + \gamma \sum_{o \in \Omega} \bar{\mathcal{J}}_{o|a} \alpha'_o \tag{3.84}$$

Démonstration. [Littman, 1994] □

Algorithme 3.6 : Recherche de témoin

Données : Fonction V' représentée par Γ' ; action a ; Fonction $Q_a = H_a V'$ incomplète représentée par Γ_a
Résultat : Témoin b

pour chaque $(\alpha, o, \alpha') \in \Gamma_a \times \Omega \times \Gamma'$ **faire**
  Résoudre le programme linéaire (3.81)
  si $b \neq$ null *et* $c > 0$ **alors retourner** b
retourner null

Algorithme 3.7 : Opérateur de Bellman pour une action donnée (Witness)

Données : Fonction de valeur PWLC V' représentée par Γ' ; a action
Résultat : Fonction de valeur PWLC Q_a représentée par Γ_a

$\Gamma_a \leftarrow \emptyset$
Choisir un b quelconque
tant que $b \neq$ null **faire**
  $\alpha_b \leftarrow \texttt{meilleurAlpha}(b, a, \Gamma')$
  $\Gamma_a \leftarrow \Gamma_a \cup \{\alpha_b\}$
  $b \leftarrow \texttt{rechercheTémoin}(\Gamma', a, \Gamma_a)$

3.4.5 Planification à base de points

Les algorithmes de type Witness montrent que l'on peut construire la fonction de valeur optimale en effectuant un nombre fini d'opérations de *lookahead* en plusieurs points b bien choisis de l'espace des croyances . Les algorithmes de planification par programmation dynamique approchée à base de points [Lovejoy, 1991, Pineau *et al.*, 2003, Poon, 2001, Spaan et Vlassis, 2005, Virin *et al.*, 2007, Smith et Simmons, 2004, Shani *et al.*, 2007] peuvent être vues comme des approximation de ce type de méthodes qui utilisent un échantillonnage de l'espace des croyances. Elles construisent une approximation de la fonction de valeur optimale en utilisant des mises à jour locales de la fonction de valeur. Ceci permet de limiter le nombre de vecteurs α dans les représentations Γ.

Le principe général est d'approcher la mise à jour $V = HV'$ et calculant les nouveaux vecteurs α optimaux pour les croyances (ou *points*) b appartenant à un ensemble $\mathcal{B}$ fini fixé :

$$\Gamma = \left\{ \texttt{lookahead}(b, \Gamma') | b \in \mathcal{B} \right\} \tag{3.85}$$

Le choix de l'ensemble $\mathcal{B}$ varie selon les algorithmes : une solution simple consiste à utiliser une discrétisation fixe en grille de l'espace de croyances [Lovejoy, 1991].

La majorité des approches de ce type utilise cependant un ensemble $\mathcal{B}$, qui change éventuellement à chaque itération, construit à partir de croyances b accessibles depuis une croyance initiale b_0 donnée [Pineau *et al.*, 2003, Poon, 2001, Spaan et Vlassis, 2005, Virin *et al.*, 2007, Smith et Simmons, 2004, Shani *et al.*, 2007] Par exemple, PBVI (*Point Based Value Iteration*) [Pineau *et al.*, 2003] fait croître à chaque itération l'ensemble $\mathcal{B}$ en ajoutant des croyances atteignables en un pas depuis les croyances de $\mathcal{B}$. Ceci permet de prendre en compte la croyance initiale b_0 pour concentrer la puissance de calcul pour obtenir une fonction valeur optimale $V^\star(b)$ plus précise sur des croyances b qui ont des chances d'être atteintes. Ces approches évitent ainsi de consommer du temps de calcul sur des croyances qui sont impossibles à atteindre et ne contribuent donc pas à la qualité de la solution trouvée.

Le choix de l'ensemble $\mathcal{B}$ permet de concentrer la précision de la fonction de valeur construite sur les parties importantes de l'espace des croyances en consacrant plus de vecteurs α pour les représenter. De telles approches sont utilisées par les principales méthodes de planification multi-agent approchée de l'état de l'art comme nous le verrons au chapitre suivant (section 4.4).

3.5 Construction d'arbres de politique

Cette partie présente le lien entre la construction de la fonction de valeur optimale en horizon fini et la construction de l'ensemble des arbres de politique possibles : les vecteurs α de la fonction de valeur optimale représentent la valeur d'arbres de politiques. L'équivalence entre ces deux calculs est importante pour le passage aux problèmes multi-agent : nous verrons dans le prochain chapitre que dans le cas multi-agent, on ne sait pas définir une fonction de valeur optimale utilisable par les agents pour déterminer les actions optimales et les politiques sont donc construites explicitement.

3.5.1 Valeur d'un arbre de politique

Nous avons défini précédemment la fonction de valeur d'une politique qui utilise la croyance courante B_t pour décider de l'action A_{t+1} à effectuer. On peut aussi définir une fonction de valeur d'un arbre de politique qui quantifie la qualité d'un tel arbre de politique pour toute croyance b. Cette fonction de valeur est linéaire et peut donc être représentée par un vecteur $\alpha^z \in \mathbb{R}^{\mathcal{S}}$ à $|\mathcal{S}|$ dimensions, appelé vecteur α, composé des $V^z(s) = \alpha^z(s)$ (figure 3.5.1). La fonction de valeur V^z d'un arbre de politique z peut être calculée à partir des fonctions de valeur de ses sous-arbres (théorème 3.5.2).

Définition 3.5.1. La fonction de valeur d'un arbre de politique z de profondeur $T-t$ associe à toute distribution de probabilité b, l'espérance de la somme pondérée des récompenses, quand S_t est distribué selon b et $Z_t = z$:

$$V^z(b) = \mathrm{E} \sum_{t'=t+1}^{T} [R_{t'} | S_t \sim b, Z_t = z] \tag{3.86}$$

Théorème 3.5.1. *La fonction de valeur V^z est linéaire :*

$$V^z(b) = \mathrm{E}[V^z(s) | s \sim b] = \sum_{s \in \mathcal{S}} V^z(s) b(s) = \alpha^z \cdot b \tag{3.87}$$

Démonstration.

$$V^z(b) = \mathrm{E}\left[\sum_{t'=t+1}^{T} \gamma^{t'-t-1} R_{t'} | S_t \sim b, Z_t = z \right] \tag{3.88}$$

$$= \sum_{s \in \mathcal{S}} \mathrm{E}\left[\sum_{t'=t+1}^{T} \gamma^{t'-t-1} R_{t'} | S_t = s, Z_t = z \right] b(s) \tag{3.89}$$

$$= \sum_{s \in \mathcal{S}} V^z(s) b(s) \tag{3.90}$$

□

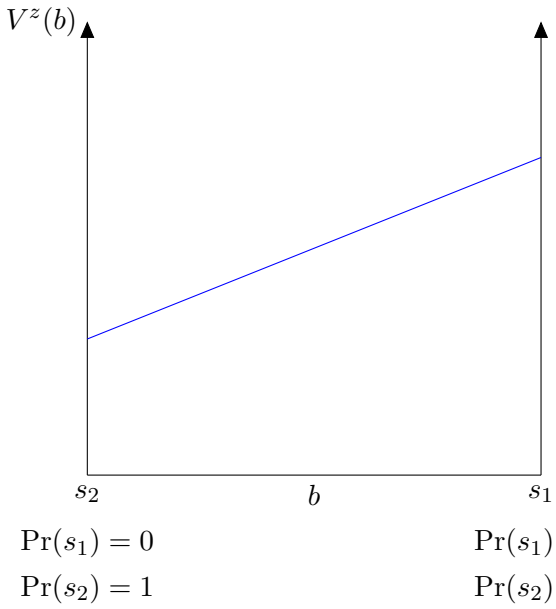


FIGURE 3.5 – Fonction de valeur V^z

Théorème 3.5.2. *La fonction V^z vérifie*

$$V^z(s) = \sum_{a \in \mathcal{A}} \varphi(a|z) \left[\mathcal{R}(s,a) + \gamma \sum_{\substack{s' \in \mathcal{S} \\ o \in \Omega}} \mathcal{J}(s', o|s, a) \sum_{z'} \psi(z'|z, a, o) V^{z'}(s') \right] \tag{3.91}$$

soit dans le cas déterministe :

$$V^z(s) = \mathcal{R}(s, \varphi(z)) + \gamma \sum_{\substack{s' \in \mathcal{S} \\ o \in \Omega}} \mathcal{J}(s', o|s, \varphi(z)) V^{\psi(z,o)}(s') \tag{3.92}$$

3.5.2 Programmation dynamique

L'ensemble $\mathcal{Z}_t$ des arbres de politique à l'instant t est l'ensemble des arbres de politique de profondeur $T - t$. Les arbres à l'instant $T - 1$ sont les actions terminales (équation (3.94), figure 3.6). Un arbre de politique z à l'instant $t-1$ est construit en choisissant une action racine $\varphi(z) = a \in \mathcal{A}$ et pour chaque observation $o \in \Omega$, un sous-arbre $\psi(z, o) = z' \in \mathcal{Z}_t$ associé à la branche correspondante (figure 3.6). Les ensembles $\mathcal{Z}_t$ peuvent être construits par programmation dynamique *bottom-up* :

$$\mathcal{Z}_T = \{\perp\} \text{ (arbre vide)} \tag{3.93}$$

$$\mathcal{Z}_{T-1} = \{z | \varphi(z) \in \mathcal{A}, \forall o \in \Omega, \psi(z, o) = \perp\} = \mathcal{A} \times \{\perp\}^{\Omega} = \mathcal{A} \tag{3.94}$$

$$\dots$$

$$\mathcal{Z}_{t-1} = \{z | \varphi(z) \in \mathcal{A}, \forall o \in \Omega, \psi(z, o) \in \mathcal{Z}_{t+1}\} = \mathcal{A} \times (\mathcal{Z}_t)^{\Omega} \tag{3.95}$$

Leurs fonctions de valeur peuvent être calculées en même temps que les arbres sont construits :

$$\alpha(s)^{\perp} = 0 \qquad \forall s \in \mathcal{S} \tag{3.96}$$

$$\alpha^q(s) = \mathcal{R}(s,a) + \gamma \sum_{o \in \Omega} \mathcal{O}(o|b,a) \left[0 \cdot \tau(b,a,o)\right] = \mathcal{R}(s,a) \qquad \forall q \in \mathcal{Z}_{T-1}, \forall s \in \mathcal{S} \tag{3.97}$$

...

$$\alpha^q(s) = \mathcal{R}(s,a) + \gamma \sum_{o \in \Omega} \mathcal{O}(o|b,a) \left[\alpha^{f(o)} \cdot \tau(b,a,o)\right] \qquad \forall q \in \mathcal{Z}_{T-t+1}, \forall f \in (\mathcal{Z}_{T-t})^{\Omega}, \forall s \in \mathcal{S} \tag{3.98}$$

Une politique optimale peut être obtenue en assignant à chaque valeur b_0 de la croyance initiale B_0 un des meilleurs arbres z pour b_0 :

$$Z_o = \nu(B_0) \in \arg\max_{z \in \mathcal{Z}_0} \alpha^z \cdot B_0 \tag{3.99}$$

La fonction de valeur optimale est la fonction de valeur de cette politique et est donc donnée par :

$$V_t^{\star}(b) = \max_{z \in \mathcal{Z}_t} V^z(b) = \max_{z \in \mathcal{Z}_t} \alpha_z \cdot b \tag{3.100}$$

$$= \max_{\alpha \in \Gamma_t} \alpha \cdot b \text{ avec } \Gamma_t = \{\alpha^z | z \in \mathcal{Z}_t\} \tag{3.101}$$

Les vecteurs α construits par l'algorithme d'induction rétrograde représentent les fonctions de valeur des arbres de politique z. L'action optimale a déterminée par $V_{t+1}^{\star}$ pour un b donnée est l'action du sous-arbre de politique z optimale :

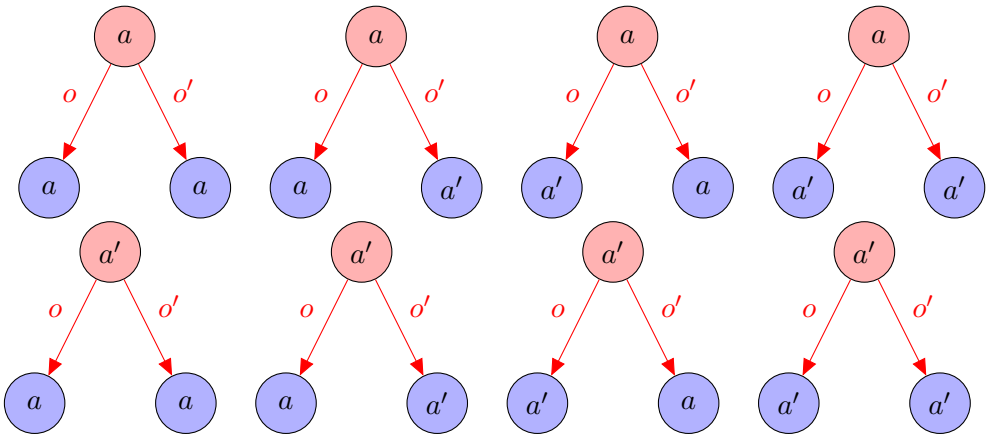
$$\arg\max_{a \in \mathcal{A}} \left[\mathcal{R}(b,a) + \gamma \sum_{o} \mathcal{O}(o|b,a) \max_{\alpha' \in \Gamma_{t+1}} \alpha' \cdot \tau(b,a,o)\right] = \varphi\left(\arg\max_{z \in \mathcal{Z}_t} V^z(b)\right) \tag{3.102}$$

L'algorithme d'induction rétrograde construit donc implicitement un ensemble d'arbres de politiques utiles : les vecteurs α de la fonction de valeur sont les arbres de politique correspondant. Toutes les opérations décrites dans ce chapitre comme opérant sur des vecteurs α, opérateur de Bellman, mise à jour, élagage, *lookahead*, etc peuvent être effectués sur les arbres de politiques. L'opérateur de mise à jour des ensembles de vecteurs α fait implicitement la mise à jour de l'ensemble $\mathcal{Z}_{t-1}$ des arbres de politique de profondeur $H - t + 1$ à partir de l'ensemble $\mathcal{Z}_t$ des arbres de politique de profondeur $H - t$. Ceci est important pour comprendre le passage de la planification mono-agent à la planification multi-agent car cette dernière utilise explicitement la représentation sous forme de politique.

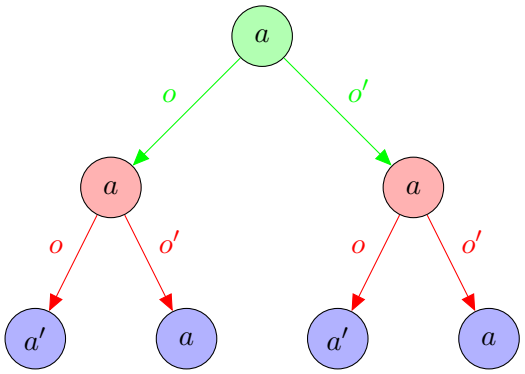
3.6 Conclusion

Les méthodes de planification pour les MDP peuvent être adaptées pour les problèmes à observabilité partielle en considérant comme état la croyance de l'agent définie comme la distribution de probabilité représentant la connaissance bayésienne de l'agent sur l'état réel du système. Les fonctions de valeur peuvent alors être définies sur l'espace des croyances de l'agent ce qui

Ensemble des arbres d'horizon 1



Ensemble des arbres d'horizon 2



Exemple d'arbre d'horizon 3

$$\mathcal{A} = \{a, a'\},\ \Omega = \{o, o'\},\ |\mathcal{Z}_{T-3}| = |\mathcal{A}||\mathcal{Z}_{T-2}|^{|\Omega|} = 128$$

FIGURE 3.6 – Construction d'arbres de politique par programmation dynamique

permet d'appliquer les méthodes d'itération de valeur et d'induction rétrograde. Les fonctions de valeur optimales sont linéaires par morceaux et convexes (respectivement convexes) en horizon fini (respectivement infini) et peuvent donc être représentées (respectivement approchées) par un nombre fini de vecteurs α. Le calcul de la fonction de valeur optimale se fait en déterminant ces ensembles de vecteurs α : le nombre de vecteurs α augmente cependant considérablement à chaque application de l'opérateur de Bellman ce qui fait exploser la complexité algorithmique. Un problème important de la planification POMDP est de déterminer une bonne représentation de ces fonctions de valeur :

- pour la planification optimale, il s'agit de déterminer un ensemble minimal permettant de représenter la fonction de valeur considérée ;
- pour la planification approchée, il s'agit de déterminer une bonne approximation de la fonction de valeur.

Le principe de la planification par construction d'arbres de politiques et son lien avec le calcul de la fonction de valeur optimale sont importants pour la planification multi-agent. Les approches de planification décentralisées de l'état de l'art présentées dans le prochain chapitre utilisent les concepts d'opérateur de Bellman, de mise à jour, d'élagage, de *lookahead*, de planification approchée à base de points présentées dans ce chapitre et les généralisent pour les problèmes à plusieurs agents.

Dans le cadre de la planification approchée, les algorithmes à base de points calculent des approximations de la fonction de valeur optimale. Ils échantillonnent des points sur l'espace des croyances et appliquent des mises à jour locales en ces points : en choisissant des points importants pour le problème, ils sont capables de concentrer la précision des approximations sur des parties utiles de la fonction de valeur. Cette idée est exploitée dans le cadre multi-agent par les approches de type MBDP dans le prochain chapitre ainsi que par notre approche dans le chapitre 7.

Chapitre 4

Dec-POMDP

Sommaire

Les modèles MDP et POMDP permettent de représenter le problème de choix séquentiel dans l'incertain d'un agent unique face à un environnement respectivement totalement et partiellement observable. Cependant, ces modèles ne permettent pas de représenter les problèmes qui comportent plusieurs agents. Nous nous intéressons ici à des problèmes complètement coopératifs où les agents partagent le même but (la fonction de récompense $\mathcal{R}$ est partagée par

tous les agents) : chaque agent reçoit ses propres observations (les observations locales) et choisit ses propres actions (actions locales) ; chaque agent a donc sa propre connaissance imparfaite de l'état global du système. De tels problèmes incluent des problèmes de coordination d'une équipe de robots autonomes [Amato et Zilberstein, 2009], des problèmes de routage de données dans un réseau [Bernstein, 2005], etc.

Exemple 4.0.1*:* On peut étendre le problème du pompier au cas multi-agent : plusieurs pompiers sont maintenant présents. Chaque pompier choisit dans quelle maison aller de manière indépendante et reçoit une observation lui donnant une idée de l'intensité de l'incendie dans la maison dans laquelle il se trouve. On suppose ici que les agents ne peuvent pas interagir directement mais uniquement via ces actions et observations. On pourrait cependant permettre des interactions plus directes en ajoutant des actions de communication, en permettant aux agents d'observer si un autre agent se trouve dans la même maison etc.

Dans ce chapitre nous présentons l'état de l'art pour la planification de problèmes multi-agent modélisés par le cadre formel Dec-POMDP en nous concentrant sur les algorithmes de programmation dynamique : ces derniers généralisent les algorithmes du chapitre précédent.

En particulier, les algorithmes de la famille de MBDP construisent des politiques de taille bornée. Ceci permet de limiter la complexité de la planification. Comme les méthodes à base de points du chapitre précédent, ces algorithmes utilisent une heuristique pour déterminer des points importants de l'espace des croyance et concentrent l'effort de planification sur ces points de croyance. Les approches de ce type, en particulier PBIP, sont importantes pour la seconde partie de cette thèse : l'approche que nous y proposons construit le même type de politique mais permet d'utiliser beaucoup plus d'information heuristique afin de faire un meilleur choix de politiques. Ceci est intéressant car la complexité introduite par la décentralisation limite fortement la taille des politiques construites ainsi que le nombre de points de croyance utilisés par les méthodes de l'état de l'art.

4.1 Formalisme

4.1.1 Modèle

Le modèle Dec-POMDP (*Decentralized Partially Observable Markov Decision Process* – Processus Décisionnel de Markov Partiellement Observable Decentralisé) [Bernstein *et al.*, 2000] généralise le modèle POMDP pour les problèmes multi-agent coopératifs. Initialement, les agents partagent une croyance initiale B_0 qui est une connaissance bayésienne sur l'état initial S_0 :

$$\forall s \in \mathcal{S}, b_0(s) = \Pr(S_0 = s | B_0 = b_0) \tag{4.1}$$

À l'instant t, chaque agent i choisit des actions locales A^i_{t+1} et reçoit des observations locales O^i_{t+1} : il peut utiliser ces informations pour mette à jour un état interne local Z^i_t. Le choix de l'action locale A^i_{t+1} à effectuer par l'agent i dépend de cet état interne Z^i_t. La figure 4.1 représente la relation entre les différentes variables aléatoires.

Exemple 4.1.1*:* Reprenons l'exemple du pompier. Chaque agent choisit indépendamment de se rendre dans une maison : $A^i_{t+1} \in \mathcal{A}_i = X$. L'évolution de l'état S^x_t d'une maison dépend de l'état

de ses voisins et du nombre $P_t^x = \sum_{x \in X} \delta_x(A_t)$ de pompiers présents : $\mathcal{T}_x(s'_x | s_{x-1}, s_x, s_{x+1}, p_x)$. Chaque agent i reçoit une observation O^i_{t+1} indépendamment des observations des autres agents selon la loi :

$$\Pr(O^i_{t+1} = o_i | S^{A^i_{t+1}}_{t+1} = s'_x) = \mathcal{O}_i(o_i | s'_x) \tag{4.2}$$

d'où

$$\Pr(O_{t+1} = o | S_{t+1} = s') = \prod_{i \in \mathcal{I}} \Pr(O^i_{t+1} = o_i | S^{A^i_{t+1}}_{t+1} = s'_{a_i}) \tag{4.3}$$

Nous appelons actions (respectivement observations et état interne) jointes les tuples d'actions (respectivement d'observations et d'état interne) locales : $A_t = (A^1_t, A^2_t \ldots)$ (respectivement $O_t = (O^1_t, O^2_t \ldots)$ et $Z_t = (Z^1_t, Z^2_t \ldots)$. Nous noterons aussi A^{-i}_t, O^{-i}_t, Z^{-i}_t les tuples composés respectivement des actions A^j_t, des observations O^j_t, des états internes Z^j_t pour tout agent j différent de i.

Définition 4.1.1. Un Dec-POMDP est un tuple $\mathcal{M} = \langle \mathcal{I}, \mathcal{S}, (\mathcal{A}_i)_{i \in \mathcal{I}}, \mathcal{T}, \mathcal{R}, (\Omega_i)_{i \in \mathcal{I}}, \mathcal{O} \rangle$ où :

- $\mathcal{I}$ est l'ensemble (fini) des agents ;
- $\mathcal{S}$ est l'ensemble des états de l'environnement, $\forall t, S_t \in \mathcal{S}$;
- $\mathcal{A}_i$ est l'ensemble des actions de l'agent i, $\forall i \in \mathcal{I}, \forall t, A^i_t \in \mathcal{A}$;
- $\mathcal{A} = \prod_{i \in \mathcal{I}} \mathcal{A}_i$ est l'ensemble des actions jointes, $\forall t, A_t = (A^1_t, A^2_t, \ldots) \in \mathcal{A}$;
- $\mathcal{T}$ définit les probabilité de transition entre états conditionnées par les actions jointes,

$$\mathcal{T}(s'|s,a) = \Pr(S_{t+1} = s' | S_0, A_1, S_1, \ldots, S_t = s, A_{t+1} = a) = \Pr(S_{t+1} = s' | S_t = s, A_{t+1} = a) \tag{4.4}$$

- Ω_i est l'ensemble des observations de l'agent i, $\forall i \in \mathcal{I}, \forall t, O^i_t \in \Omega$;
- $\Omega = \prod_{i \in \mathcal{I}} \Omega_i$ est l'ensemble des observations jointes, $\forall t, O_t = (O^1_t, O^2_t, \ldots) \in \Omega$;
- $\mathcal{O}$ définit les probabilités des observations jointes

$$\mathcal{O}(o|s,a,s') = \Pr(O_{t+1} = o | S_t = s, A_{t+1} = a, S_{t+1} = s') \tag{4.5}$$

- $\mathcal{R}$ définit l'espérance de la récompense $\mathcal{R}(s,a)$ obtenue en effectuant l'action jointe $A_{t+1} = a$ dans l'état $S_t = s$

$$\mathcal{R}(s,a) = \mathrm{E}\left[R_{t+1} | S_t = a, A_{t+1} = a\right] \tag{4.6}$$

Théorème 4.1.1 (POMDP sous-classe des Dec-POMDP). *Un POMDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O} \rangle$ est un Dec-POMDP $\mathcal{M}' = \langle \mathcal{I}, (\mathcal{S})_{i \in \mathcal{I}}, (\mathcal{A}_i)_{i \in \mathcal{I}}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O} \rangle$ avec un seul agent : $\mathcal{I} = \{i\}$, $\mathcal{A}_i = \mathcal{A}$ et $\Omega_i = \Omega$.*

4.1.2 Critère

Le critère de performance est identique au cas POMDP :

$$\text{maximiser } \mathrm{E}\left[\sum_{t=1}^{T} \gamma^{t-1} R_t | B_0 = b_0, \pi\right] \tag{4.7}$$

Les résultats de complexité indiquent que la planification dans le cas Dec-POMDP est un problème beaucoup plus difficile que dans les sous-classes POMDP et MDP. Le problème de la planification Dec-POMDP en horizon fini est NEXP-difficile alors que les problèmes de planifications MDP et POMDP sont P-difficile et PSPACE-difficile respectivement [Bernstein *et al.*, 2000].

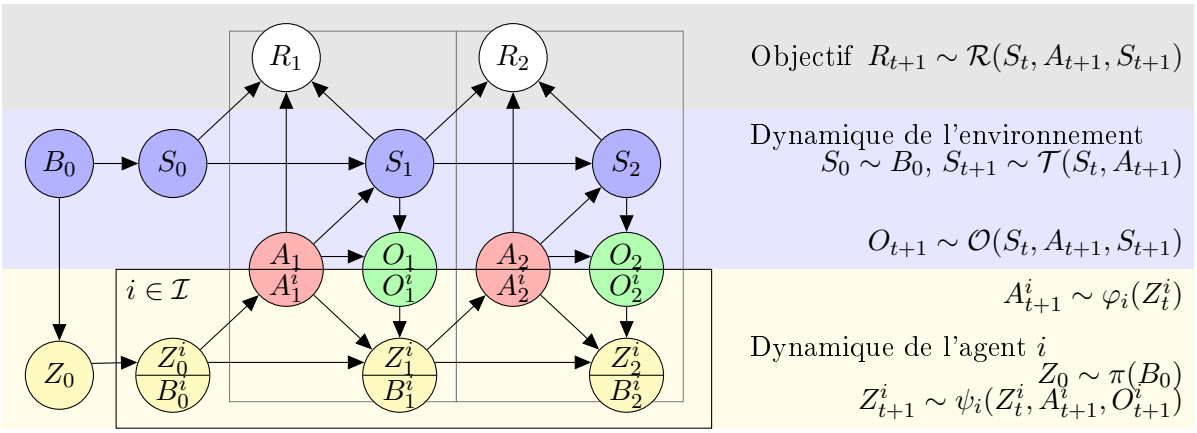


FIGURE 4.1 – Modèle Dec-POMDP

4.2 Politique

4.2.1 Politique locale

Le comportement d'un agent i donné est défini par son état interne Z^i_t associé à
- des probabilité d'actions, que l'on notera φ_i

$$\varphi_i(a_i|z_i) = \Pr(A^i_{t+1} = a_i | Z^i_t = z_i) \tag{4.8}$$

- des probabilités de transition d'état interne, que l'on notera ψ_i

$$\psi_i(z'_i|z_i, a_i, o_i) = \Pr(Z^i_{t+1} = z'_i | Z^i_t = z_i, A^i_{t+1} = a_i, O^i_{t+1} = o_i) \tag{4.9}$$

La politique joint π est définie par ces probabilités ainsi que la probabilité de l'état initial joint, que l'on notera ν :

$$\nu(z|b) = \Pr(Z_0 = z | B_0 = b) \tag{4.10}$$

où B_0 est une croyance, commune à tous les agents, *a priori* sur l'état initial S_0 :

$$\forall s \in \mathcal{S}, B_0(s) = \Pr(S_0 = s) \tag{4.11}$$

4.2.2 Politique jointe

L'état interne joint des agents est le tuple des états internes des agents $Z_t = (Z^1_t, Z^2_t, \ldots)$ et on note :
- la probabilité φ d'action,

$$\varphi(a|z) = \Pr(A_{t+1} = a | Z_t = z) = \prod_{i \in \mathcal{I}} \varphi_i(a_i|z_i) \tag{4.12}$$

- la probabilité ψ de transition de l'état interne joint,

$$\psi(z'|z, a, o) = \Pr(Z_{t+1} = z | Z_t = z, A_{t+1} = a, O_{t+1} = o) = \prod_{i \in \mathcal{I}} \psi_i(z'_i|z_i, a_i, o_i) \tag{4.13}$$

- la probabilité de l'état initial

$$\nu(z|b) = \Pr(Z_0 = z | B_0 = b) \tag{4.14}$$

Comparé au chapitre précédent (section 3.2.1), la seule différence entre le modèle POMDP et le modèle Dec-POMDP réside dans les contraintes de d'indépendance entre les agents, exprimées par les équations (4.12) et (4.13). Ces contraintes sont appelées *contraintes de décentralisation* [Dibangoye *et al.*, 2008]. La planification Dec-POMDP peut être interpretée comme une planification POMDP où la politique est cherchée dans le sous-espace obtenu en ajoutant ces contraintes de décentralisation.

4.2.3 Historiques

On définit l'historique local H_t^i de l'agent i à l'instant t comme le tuple constitué de la croyance initiale, de l'état interne initial des autres agents ainsi que des actions et observations locales passées de l'agent :

$$H_t^i = (B_0, Z_t^{-i}, A_1^i, O_1^i, \dots A_t^i, O_t^i) \tag{4.15}$$

Une politique définit des lois π_t^i de probabilités d'actions locales conditionnées par les historiques locaux :

$$\pi_t^i(a_i|h_i) = \Pr(A_t^i = a_i|H_t^i = h_i) \tag{4.16}$$

4.2.4 Croyances

Pour un agent i, étant donné la politique jointe des autres agents, le Dec-POMDP peut être transformé en un POMDP en incluant l'état Z_t^{-i} des autres agents dans l'état de l'environnement [Hansen *et al.*, 2004] (figure 4.2) : le nouvel état $S_t^i = (S_t, Z_t^{-i})$ permet de définir un POMDP prenant en compte le comportement des autres agents (définition 4.2.1). La croyance B_t^i de l'agent i à l'instant t est la croyance de ce POMDP : l'agent i peut maintenir une croyance locale B_t^i qui est la densité de probabilité de cet état subjectif S_t^i, c'est-à-dire la densité de probabilité *jointe* de l'état de l'environnement S_t et des autres agents Z_t^{-i} [Szer et Charpillet, 2006a].

Remarque: Mettre à jour une telle croyance nécessite de connaître la politique des autres agents. Il n'est pas possible de maintenir une telle croyance sans connaître la politique des autres agents.

Définition 4.2.1. Le POMDP subjectif $\mathcal{M}^i = \langle \mathcal{S}^i, \mathcal{A}^i, \mathcal{T}^i, \mathcal{R}^i, \Omega^i, \mathcal{O}^i \rangle$ de l'agent i est défini par :

$$\mathcal{S}^i = \mathcal{S} \times \mathcal{Z}_t^{-i} \tag{4.17}$$

$$\mathcal{A}^i = \mathcal{A}_i \tag{4.18}$$

$$\Omega^i = \Omega_i \tag{4.19}$$

$$\begin{aligned}\mathcal{J}^i((s', z'_{-i}), o_i|(s, z_{-i}), a_i) = & \sum_{\substack{a_{-i}\in\mathcal{A}_{-i} \\ a=\langle a_i, a_{-i}\rangle}} \varphi_{-i}(a_{-i}|z_{-i}) \sum_{s'\in\mathcal{S}} \mathcal{T}(s'|s,a) \\ & \sum_{o_{-i}\in\Omega_{-i}} \left[\sum_{\substack{o_i\in\Omega_i \\ o=\langle o_i, o_{-i}\rangle}} \mathcal{O}(o|s,a,s') \right] \\ & \psi_{-i}(z'_{-i}|z'_{-i}, a_{-i}, o_{-i})\end{aligned} \tag{4.20}$$

$$\mathcal{T}^i((s', z'_{-i})|(s, z_{-i}), a_i) = \sum_{o_i\in\Omega_i} \mathcal{J}^i((s', z'_{-i}), o_i|(s, z_{-i}), a_i) \tag{4.21}$$

$$\mathcal{O}^i(o_i|(s, z_{-i}), a_i, (s', z'_{-i})) = \frac{\mathcal{J}^i((s', z'_{-i}), o_i|(s, z_{-i}), a_i)}{\mathcal{T}^i((s', z'_{-i})|(s, z_{-i}), a_i)} \tag{4.22}$$

$$\mathcal{R}^i((s, z_{-i}), a_i) = \sum_{\substack{a_{-i}\in\mathcal{A}_{-i} \\ a=\langle a_i, a_{-i}\rangle}} \varphi_{-i}(a_{-i}|z_{-i}) \mathcal{R}(s,a) \tag{4.23}$$

avec

- $\mathcal{Z}^j$, l'ensemble des états interne d'un agent $j \neq i$;
- $\mathcal{Z}^{-i} = \prod_{j\neq i} \mathcal{Z}^j$, l'ensemble des états internes conjoints de l'agent i ;
- la probabilité de transition des autres agents, notée ψ_{-i},

$$\psi_{-i}(z'_{-i}|z_{-i}, a_{-i}, o_{-i}) = \prod_{j\neq i} \psi_j(z_j|z_j, a_j, o_j) \tag{4.24}$$

- la probabilité d'action des autres agents, notée φ_{-i},

$$\varphi_{-i}(a_{-i}|z_{-i}) = \prod_{j\neq i} \varphi_j(a_j|z_j) \tag{4.25}$$

Définition 4.2.2. La croyance B_t^i d'un agent i à l'instant t est la distribution de probabilité jointe de l'état S_t de l'environnement et de l'état Z_t^{-i} des autres agents étant donné l'historique local H_t^i constitué de la croyance initiale B_0, de l'état initial des autres agents Z_0^i et des actions et observations locales passées :

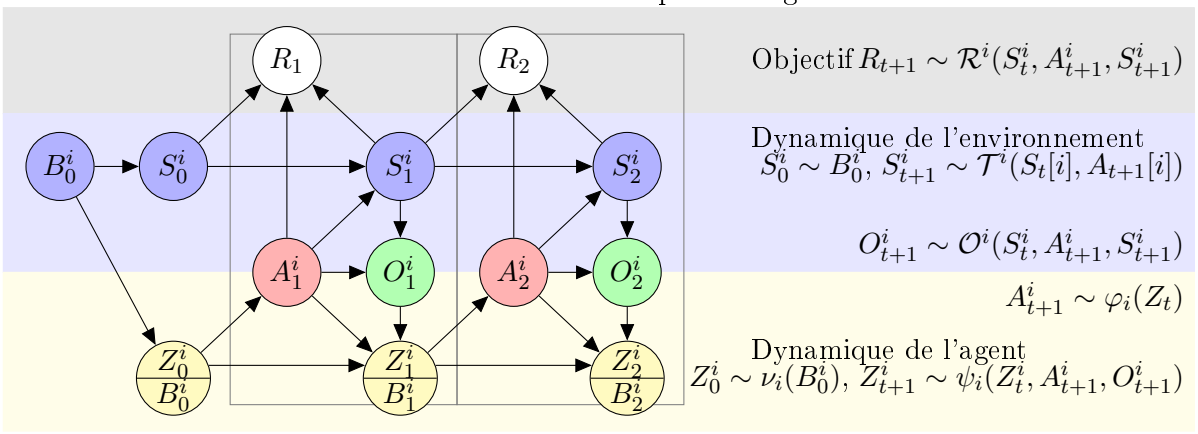
$$B_t^i(s, z_{-i}) = \Pr(S_t = s, Z_t^{-i} = z_{-i}|B_0, Z_0^{-i}, A_1^i, O_1^i, \ldots A_t^i, O_t^i) \tag{4.26}$$

$$= \Pr(S_t = s, Z_t^{-i} = z_{-i}|H_t^i) \tag{4.27}$$

Théorème 4.2.1 (Mise à jour de croyance multi-agent)**.** *La croyance $B_{t+1}^i = b'_i$ de l'agent i à l'instant $t+1$ ne dépend que de sa croyance $B_t^i = b_i$ à l'instant t, de la nouvelle action locale $A_{t+1}^i = a_i$ et de la nouvelle observation locale $O_{t+1}^i = o_i$ et est donnée par :*

$$b'_i(s', z'_{-i}) = [\tau_i(b_i, a_i, o_i)]\,(s', z'_{-i}) = \lambda \mathcal{J}^i((s', z'_{-i}), o_i|(s, z_{-i}), a_i) \tag{4.28}$$

FIGURE 4.2 – Modèle POMDP pour un agent i donné



Démonstration. Il s'agit du théorème de mise à jour des croyances POMDP (théorème 3.2.1) appliqué au POMDP défini sur les états S_t^i (définition 4.2.2). □

On note τ_i la fonction de mise à jour de croyance pour l'agent i :

$$b_i' = \tau_i(b_i, a_i, o_i) \tag{4.29}$$

4.2.5 Arbres de politique

En horizon fini T, l'état interne Z_t^i d'un agent i déterministe à l'instant t est un arbre de politique de profondeur $T-t$ dont les nœuds sont associés à des actions locales et les branches à des observations locales (figure 4.3). On note $\varphi_i(z_i)$ l'action associée à la racine de l'arbre z_i et $\psi_i(z_i, o_i)$ les sous arbres associés aux observations o_i :

$$A_{t+1}^i = \varphi_i(Z_t^i) \tag{4.30}$$
$$Z_{t+1}^i = \psi_i(Z_t^i, O_{t+1}^i) \tag{4.31}$$

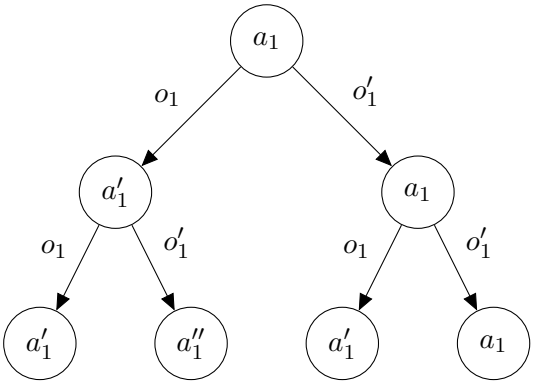
L'état interne joint peut être représentée par un tuple d'arbres de politiques locales (un par agent) ou de manière équivalente par un arbre de politique jointe (figure 4.3) dont les nœuds sont associés à des actions jointes et les branches à des observations jointes : cet arbre de politique vérifie les contraintes de décentralisation (4.12) et (4.13). On utilise les mêmes notations $\varphi(z)$ et $\psi(z, o)$ que pour les arbres locaux et les arbres POMDP pour représenter les actions et les sous-arbres :

$$A_{t+1} = \varphi(Z_t) \tag{4.32}$$
$$Z_{t+1} = \psi(Z_t, O_{t+1}) \tag{4.33}$$

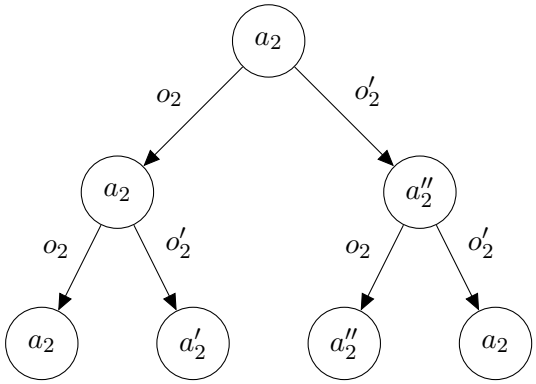
Les contraintes de décentralisation (4.12) et (4.13) prennent respectivement la forme (figure 4.3) :

$$\varphi(z) = (\varphi_1(z_1), \dots \varphi_n(z_n)) \tag{4.34}$$
$$\psi(z, o) = (\psi_1(z_1, o_1), \dots \psi_n(z_n, o_n)) \tag{4.35}$$

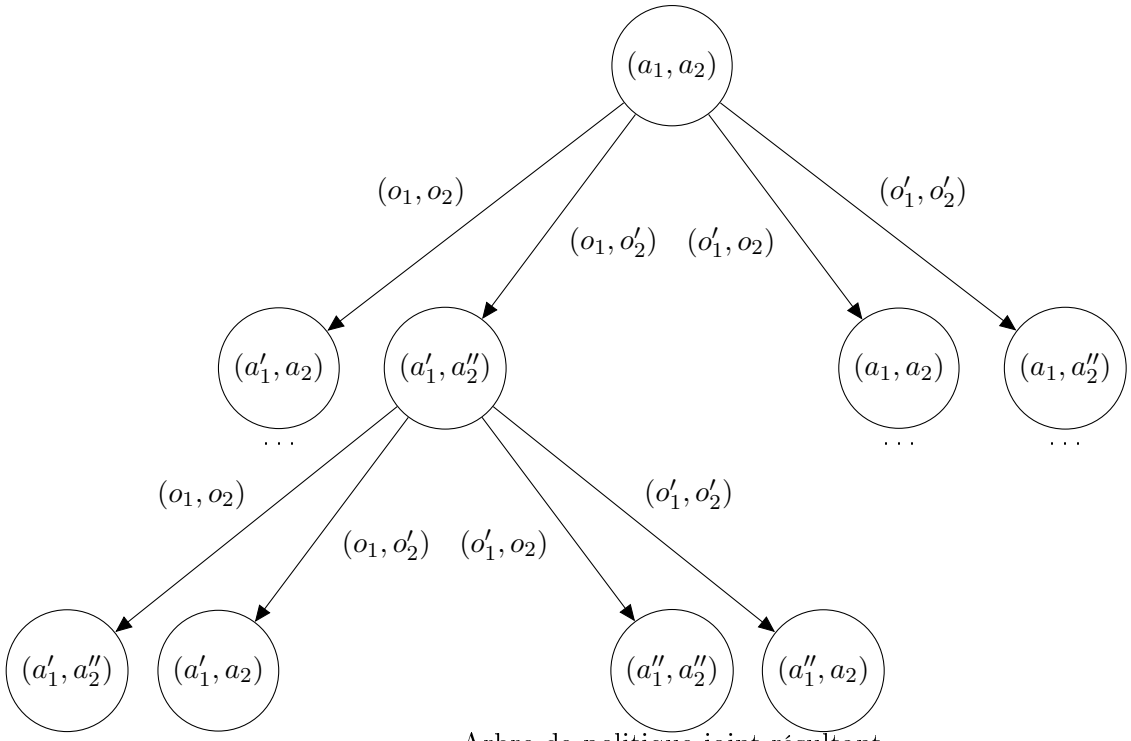
La figure 4.4 représente un exemple de politique qui n'est pas décentralisable : cette politique ne peux pas être exécutée de manière décentralisée car deux actions locales différentes (a_1' et a_1) sont associées à une même observation locale o_1 pour l'agent 1.



Arbre de politique de l'agent 1



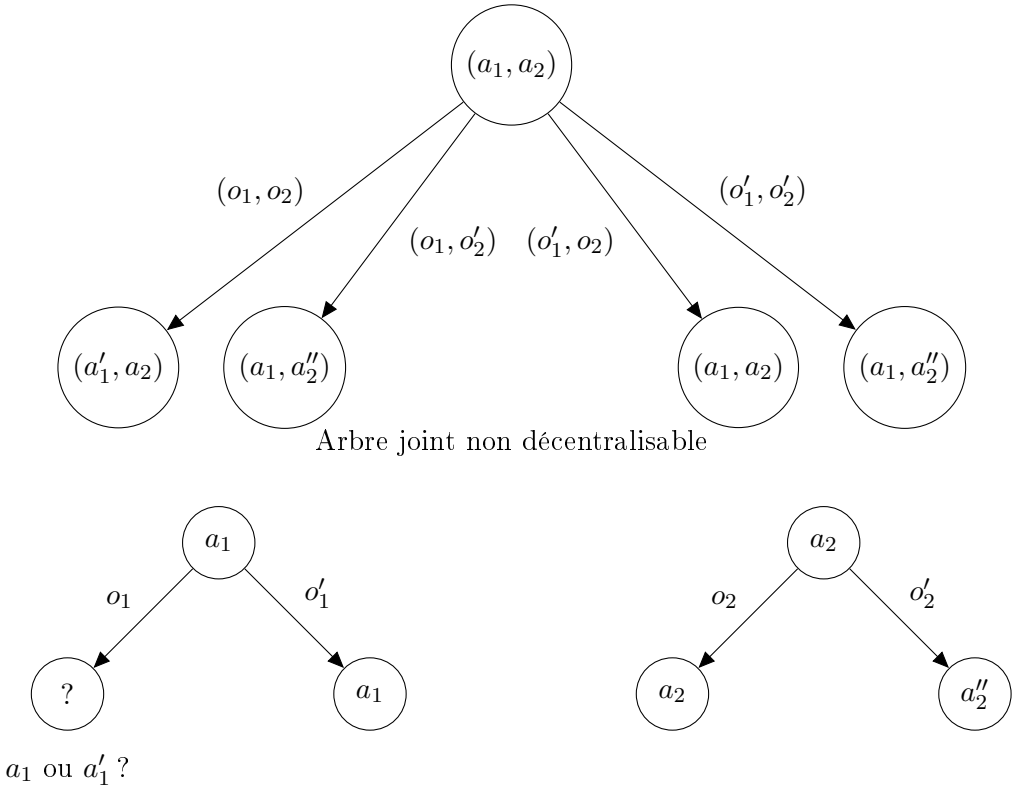
Arbre de politique de l'agent 2



Arbre de politique joint résultant

$T = 3, \mathcal{I} = \{1, 2\}, \Omega_i = \{o_i, o'_i\}, \mathcal{A}_i = \{a_i, a'_i, a''_i\}, \mathcal{I} = \{1, 2\}$

FIGURE 4.3 – Exemple de politiques locales et politique jointe déterministe en horizon fini



Arbre de politique de l'agent 1 Arbre de politique de l'agent 2

$T = 3, \mathcal{I} = \{1, 2\}, \Omega_i = \{o_i, o_i'\}, \mathcal{A}_i = \{a_i, a_i', a_i''\}, \mathcal{I} = \{1, 2\}$

FIGURE 4.4 – Exemple de politique non décentralisable

4.3 Programmation dynamique

Contrairement au cas MDP et POMDP, on ne sait pas définir une fonction de valeur optimale simple pour déterminer les actions optimales [2]. Pour un agent i donné, la croyance $b_i \in \Delta(\mathcal{S} \times \mathcal{Z}_t^{-i})$ est une distribution sur l'état du système et des autres agents. La fonction de valeur d'une politique locale est donc une fonction $V_t^i(b_i)$. Même s'il dispose de la fonction de valeur optimale $V_t^{\star,i}$, mettre à jour la croyance locale b_i nécessite de connaître la politique des autres agents : utiliser une fonction de valeur optimale définie sur les croyances de l'agent i pour déterminer la politique de l'agent i nécessite de connaître le comportement des autres agents ; ce type de raisonnement se mord la queue.

Les approches de planification par programmation dynamique *bottom-up* de l'état de l'art dans le cadre Dec-POMDP travaillent directement dans l'espace des politiques [Hansen *et al.*, 2004]. Ils construisent en parallèles des ensembles $\mathcal{Z}_t^i$ (exécutés de l'instant t à l'instant $T-1$) d'arbres de politiques locales pour les agents de manière *bottom-up* (algorithme 4.1), c'est-à-dire à partir des ensemble $\mathcal{Z}_{t+1}^i$ des sous-arbres de politique (exécutés de l'instant $t+1$ à $T-1$). Ces algorithmes généralisent l'approche présentée dans la section 3.5 pour le cas POMDP au cas multi-agent. Comme ceux-ci, ils sont souvent décomposés en deux phases : *mise à jour*, c'est-à-dire construction des ensembles d'arbres de politiques locaux à partir de sous-arbres, et *élagage* des ensembles d'arbres de politique.

Algorithme 4.1 : Planification par programmation dynamique

Résultat : Ensembles des arbres locaux $\forall i, \mathcal{Z}_0^i$ (et leurs vecteurs α)
1 **pour chaque** $i \in \mathcal{I}$ **faire** $\mathcal{Z}_T^i \leftarrow \{\bot\}$ (arbre vide)
2 **pour** $t = T$ **à** 1 **faire** $(\mathcal{Z}_{t-1}^i)_{i\in\mathcal{I}} \leftarrow \texttt{bellman}((\mathcal{Z}_t^i)_{i\in\mathcal{I}})$

4.3.1 Évaluation

De même que dans le cas POMDP, on définit la fonction de valeur d'un état interne *joint* z comme la fonction qui à toute distribution de probabilité b sur les états associe l'espérance de la somme des récompenses :

$$V_t^b(b) = \mathrm{E}\sum_{t'=t}^{T}\left[\gamma^{t'-t-1}R_{t'}|S_t \sim b, Z_t = z\right] = \sum_{s\in\mathcal{S}} V_t^b(s)b(s) = \alpha_t^z \cdot b \tag{4.36}$$

Cette fonction de valeur peut comme dans le cas POMDP être représentée comme une fonction définie sur les états, appelée vecteur α.

Dans le cas particulier des arbres de politique, la fonction de valeur d'un arbre de politique z peut être calculée à partir de celle de ses sous-arbres $\alpha^{\psi(z,o)}$ avec la même formule que dans le cas POMDP :

$$\alpha^z = \mathcal{R}_{\varphi(z)} + \gamma \sum_{o\in\Omega} \bar{\mathcal{J}}_{o|\varphi(z)} \alpha^{\psi(z,o)} \tag{4.37}$$

2. Une fonction de valeur optimale définie sur les historiques joints et permettant de déterminer les actions optimale est définie dans [Oliehoek *et al.*, 2008]. Cependant le nombre de politiques jointes est exponentiel avec le temps t et aucune méthode n'est proposée pour déterminer cette fonction : des heuristiques sont utilisées pour déterminer des majorants de la fonction de valeur optimale.

4.3.2 Mise à jour

Les ensembles des arbres de politiques joints peuvent être construit par programmation dynamique [Hansen *et al.*, 2004]. La mise à jour exhaustive (algorithme 4.2) construit, pour chaque agent $i \in \mathcal{I}$, l'ensemble des arbres $\mathcal{Z}_t^i$ des arbres de politiques locales d'horizon $T-t$ que l'on peut construire à partir de l'ensemble des sous-arbres $\mathcal{Z}_{t+1}^i$ des arbres de politiques locales d'horizon $T-t-1$ (ligne 1). Ce sont les arbres z_i dont la racine est une action $\varphi_i(z_i)$ de $\mathcal{A}_i$ et dont chaque sous-arbre $\psi_i(z_i, o_i)$ est dans $\mathcal{Z}_{t+1}^i$:

$$\mathcal{Z}_T^i = \{\perp\} \text{ (arbre vide)} \qquad \forall i \in \mathcal{I} \qquad (4.38)$$

$$\mathcal{Z}_{T-1}^i = \{z_i | \varphi_i(z_i) \in \mathcal{A}_i, \forall o_i \in \Omega_i, \psi_i(z_i, o_i) = \perp\} = \mathcal{A}_i \times \{\perp\}^{\Omega_i} = \mathcal{A}_i \qquad \forall i \in \mathcal{I} \qquad (4.39)$$

$$\dots$$

$$\mathcal{Z}_{t-1}^i = \left\{z_i | \varphi_i(z_i) \in \mathcal{A}_i, \forall o_i \in \Omega_i, \psi_i(z_i, o_i) \in \mathcal{Z}_{t+1}^i\right\} = \mathcal{A}_i \times (\mathcal{Z}_t^i)^{\Omega_i} \qquad \forall i \in \mathcal{I} \qquad (4.40)$$

Ensuite, les vecteurs α^z sont calculés (ligne 2), pour tous les arbres *joints*, $\forall z \in \prod_{i \in \mathcal{I}}, \mathcal{Z}_t^i$, avec la formule (4.37).

Algorithme 4.2 : Opérateur de Bellman par mise à jour exhaustive

Données : Ensembles des arbres locaux $\forall i, \mathcal{Z}_{t+1}^i$ (et leurs vecteurs α)
Résultat : Ensembles des arbres locaux $\forall i, \mathcal{Z}_t^i$ (et leurs vecteurs α)
1 **pour chaque** $i \in \mathcal{I}$ **faire** $\mathcal{Z}_t^i \leftarrow \mathcal{A}_i \times (\mathcal{Z}_{t+1}^i)^{\Omega_i}$
2 **pour chaque** $z \in \prod_{i \in \mathcal{I}} \bar{\mathcal{Z}}_t^i$ **faire** $\alpha^z \leftarrow \mathcal{R}_a + \gamma \mathcal{J}_{o|\varphi(z)} \alpha^{\psi(z,o)}$

L'application de cet opérateur de programmation dynamique permet de construire l'ensemble des arbres locaux d'horizon T. Pour un $B_0 = b_0$ donné, un arbre joint optimal est donc donné par :

$$Z_0 = \underset{q \in \prod_{i\mathcal{I}} \mathcal{Z}_0^i}{\arg\max}\, \alpha^i \cdot b_0 \qquad (4.41)$$

4.3.3 Élagage

Cependant, le nombre d'arbres de politique de l'agent i est exponentiel avec le nombre d'observations locales Ω [Hansen *et al.*, 2004]. Afin de limiter la croissance du nombre d'arbres de politiques, une phase d'élagage (10) peut être utilisée. Elle consiste à éliminer des arbres inutiles des ensembles $\mathcal{Z}_t^i$ [Hansen *et al.*, 2004]. Les arbres de politique dominés, qui ne seront jamais sélectionnés par (4.41), sont élagués. La vérification de dominance est itérée sur chaque agent jusqu'à ce que plus aucun arbre de politique ne puisse être élagué.

Définition 4.3.1. Un arbre de politique local z_i de l'agent i est dominé si :

$$\forall b_i \in \Delta(\mathcal{S} \times \mathcal{Z}^{-i}), \exists z_i' \in \mathcal{Z}_t^i \setminus \{z_i\}, V^{z_i'}(b_i) \geq V^{z_i}(b_i) \qquad (4.42)$$

Définition 4.3.2. Un arbre de politique local z_i de l'agent i est ϵ-dominé si :

$$\forall b_i \in \Delta(\mathcal{S} \times \mathcal{Z}^{-i}), \exists z_i' \in \mathcal{Z}_t^i \setminus \{z_i\}, V^{z_i'}(b_i) \geq V^{z_i}(b_i) + \epsilon \qquad (4.43)$$

Théorème 4.3.1 (Test de domination). *La propriété d'ε-domination de z_i peut être vérifiée en résolvant le programme linéaire :*

$$\begin{aligned} &\textit{maximiser } \sigma && (4.44)\\ &\quad \textit{sujet à } \forall z_i' \neq z_i, V^{z_i'}(b_i) + \sigma \leq V^{z_i}(b_i)\\ &\quad\quad \textit{avec } \sum_i \sum_{z_{-i} \in \mathcal{Z}^{-i}} b_i(s, z_{-i}) = 1\\ &\quad\quad\quad b_i(s, z_{-i}) \geq 0 \qquad\qquad \forall s \in \mathcal{S}, \forall z_{-i} \in \mathcal{Z}^{-i} \end{aligned}$$

La politique z_i est ϵ-dominée si et seulement si $\sigma < \epsilon$.

Algorithme 4.3 : Élagage exact

Données : Ensembles des arbres locaux $\forall i, \mathcal{Z}_t^i$ (et leurs vecteurs α)
Résultat : Ensembles des arbres locaux $\forall i, \mathcal{Z}_t^i$ élagués (et leurs vecteurs α)

```
1 flag ← true
2 tant que flag faire
3     flag ← false
4     pour chaque i ∈ I, z_i ∈ Z_t^i faire
5         si q_i est dominé alors
6             Z_t^i ← Z_i^t \ {z_i}
7             flag ← true
```

L'opérateur de programmation dynamique est donc généralement décomposé en deux phases (algorithme 4.4) : mise à jour (ligne 1) et élagage (ligne 2) comme pour le cas POMDP. Dans le cas général, le nombre d'arbres de politique reste cependant du même ordre de grandeur que sans élagage.

Algorithme 4.4 : Opérateur de Bellman par mise à jour et élagage

Données : Ensembles des arbres locaux $\forall i, \mathcal{Z}_{t+1}^i$ (et leurs vecteurs α)
Résultat : Ensembles des arbres locaux $\forall i, \mathcal{Z}_t^i$ (et leurs vecteurs α)

```
1 (Z̄_t^i)_{i∈I} ← miseÀJour((Z_{t+1}^i)_{i∈I})
2 (Z_t^i)_{i∈I} ← élagage((Z̄_t^i)_{i∈I})
```

4.3.4 IPG

La génération incrémentale de politique (IPG, *Incremental Policy Generation*) [Amato *et al.*, 2009] est une méthode qui peut-être utilisée pour accélérer les méthodes de planification à base de programmation dynamique (algorithmes exacts, PBDP [Szer et Charpillet, 2006a], MBDP [Seuken et Zilberstein, 2007b], IMBDP [Seuken et Zilberstein, 2007a], MBDP-OC [Carlin et Zilberstein, 2008a], PBIP [Dibangoye *et al.*, 2008]) : une analyse d'accessibilité à un pas de temps est utilisée pour élaguer les ensembles d'arbres de politiques locaux ou pour restreindre l'espace de recherche d'arbres de politique. Si un agent a effectué une action a_i puis observé une observation o_i, il

peut en général restreindre l'espace des états $s' \in \mathcal{S}_i(a_i, o_i)$ dans lequel il peut se trouver :

$$\mathcal{S}_i(a_i, o_i) = \{s' \in \mathcal{S} | \exists (s, a_{-i}, o_{-i}) \in \mathcal{S} \times \mathcal{A}_{-i} \times \Omega_{-i}, \mathcal{J}(s', \langle o_i, o_{-i} \rangle | s, \langle a_i, a_{-i} \rangle) \neq 0\} \quad (4.45)$$

Cette information peut être utilisée lors de la construction des arbres de politique par programmation dynamique. Un sous-arbre z'_i n'a besoin d'être considéré comme sous-arbre d'observation o_i d'un arbre z_i d'action racine a_i que s'il existe une distribution de probabilité $b'_i \in \Delta(\mathcal{S}(a_i, o_i) \times \mathcal{Z}^{-i}_{t+1})$ sur l'ensemble de croyances restreint telle que

$$V^{z'_i}(b'_i) \geq \max_{\bar{z}'_i \in \mathcal{Z}^i_{t+1} \setminus \{z'_i\}} V^{\bar{z}'_i}(b'_i) \quad (4.46)$$

On définit donc l'ensemble $\mathcal{Z}^i_{t+1}(a_i, o_i)$ des sous-arbres utiles pour une action racine a_i et la branche d'observation o_i comme :

$$\mathcal{Z}^i_{t+1}(a_i, o_i) = \left\{ z'_i \in \mathcal{Z}^i_{t+1} | \exists b'_i \in \Delta(\mathcal{S}_i(a_i, o_i) \times \mathcal{Z}^i_{t+1}), V^{z'_i}(b_i) \geq \max_{\bar{z}'_i \in \mathcal{Z}^i_{t+1} \setminus \{z'_i\}} V^{\bar{z}'_i}(b_i) \right\} \quad (4.47)$$

Cet ensemble peut être obtenue en faisant un élagage exact de $\mathcal{Z}^i_{t+1}$ (section 4.3.3) en restreignant les croyances sur $\mathcal{S}_i(a_i, o_i)$.

Utilisation de la croyance initiale Si l'ensemble des états accessibles à l'instant t peut être restreint à un sous-ensemble $\mathcal{S}_t(\mathcal{S}_t, a_i, o_i)$, on peut restreindre encore plus les états accessibles ç l'instant $t+1$ ainsi que les ensembles $\mathcal{Z}^i_{t+1}(a_i, o_i)$:

$$\mathcal{S}_i(\mathcal{S}_t, a_i, o_i) = \left\{ s' \in \mathcal{S} | \exists (s, a_{-i}, o_{-i}) \in \mathcal{S}_t \times \mathcal{A}_{-i} \times \Omega_{-i}, \mathcal{J}(s', \langle o_i, o_{-i} \rangle | s, \langle a_i, a_{-i} \rangle) \neq 0 \right\} \quad (4.48)$$

$$\mathcal{Z}^i_{t+1}(a_i, o_i) = \left\{ z'_i \in \mathcal{Z}^i_{t+1} | \exists b'_i \in \Delta(\mathcal{S}_i(\mathcal{S}_t, a_i, o_i) \times \mathcal{Z}^i_{t+1}), V^{z'_i}(b_i) \geq \max_{\bar{z}'_i \in \mathcal{Z}^i_{t+1} \setminus \{z'_i\}} V^{\bar{z}'_i}(b_i) \right\} \quad (4.49)$$

Les ensembles $\mathcal{S}_t$ des états accessibles aux différents instants t peuvent être déterminés par programmation dynamique *top-down* :

$$\mathcal{S}_0 = \{s \in \mathcal{S} | b_0(s) \neq 0\} \quad (4.50)$$

$$\mathcal{S}_{t+1} = \left\{ s' \in \mathcal{S} | \exists (s, a) \in \mathcal{S}_t \times \mathcal{A}, \mathcal{T}(s' | s, a) \neq 0 \right\} \quad (4.51)$$

4.4 Programmation dynamique à base de points

Le nombre d'arbres de politiques locales et donc le nombre de vecteur α à considérer dans les approches par programmation exacte explose très rapidement ce qui limite son utilisation à des problèmes de très petite taille et pour des horizons très faibles. Comme dans le cadre POMDP, des méthodes approchées à base de points peuvent être utilisées. En particulier, les algorithmes de type MBDP (MBDP, IMBDP, MBDP-OC, PBIP, PBIP-IPG) permettent de traiter des problèmes significativement plus grands que les approches précédentes : pour cela, elles bornent le nombre d'arbres de politique locaux conservés dans chaque ensemble d'arbres locaux, $|\mathcal{Z}^i_t| \leq maxTrees, \forall i, \forall t$, et les choisissent en utilisant une information heuristique sur le problème. L'approche que nous proposons dans cette thèse cherche le même type de politique mais propose d'utiliser plus d'information heuristique sur le problème.

Les algorithmes MBDP et PBIP sont importants pour la suite de la thèse. La description des algorithmes IMBDP, MBDP-OC, PBIP-IPG et DecRSPI peut être sautée dans une première lecture.

4.4.1 PBDP

PBDP (*Point Based Dynamic Programming* – Programmation dynamique à base de points) [Szer et Charpillet, 2006a, Szer et Charpillet, 2006b] propose d'exploiter l'accessibilité des croyances locales pour effectuer l'élagage. Ceci permet de réduire le nombre d'arbres conservés en excluant les régions de l'espace des croyances locales qui ne sont pas accessibles.

Lors de l'élagage des ensembles $\bar{\mathcal{Z}}_t^i$, les politiques pour les temps $t+1$ à T sont connues. Déterminer l'accessibilité des croyances locales B_t^i à l'instant t nécessiterait cependant de connaître les politiques globales (de 0 à T) que l'on cherche à déterminer : l'accessibilité des croyances locales à l'instant t dépend de ce que les agents font avant l'instant t, ce que l'on ne connaît pas à ce moment de la planification.

Algorithme exhaustif PBDP considère toutes les croyances accessibles à un agent i en considérant toutes les politiques jointes z^0 possibles dont les sous-politiques locales z_i^t à l'instant t sont dans $\bar{\mathcal{Z}}_t^i$ (algorithme 4.5). Les arbres conservés sont les arbres qui sont optimaux pour au moins une croyance accessible. Il s'agit d'un algorithme théorique : considérer toutes les croyances accessibles nécessite de considérer toutes les politiques jointes ce qui a la même complexité que faire la planification sans faire d'élagage.

Algorithme 4.5 : Élagage à base de points (PBDP)

Données : Ensembles des arbres locaux $\forall i \in \mathcal{I}, \bar{\mathcal{Z}}_t^i$ à élaguer
Résultat : Ensembles des arbres locaux $\forall i \in \mathcal{I}, \mathcal{Z}_t^i$ élagués

```
1 pour chaque agent i ∈ I faire
2     Z_t^i ← ∅
3     pour chaque arbre de politique z joints d'horizon T terminant par des arbres de Z_t^i faire
4         pour chaque historique local h_i^t faire
5             Déterminer la croyance b_i
6             Z_t^i ← Z_i^t ∪ {arg max_{z_i ∈ Z̄_t^i} V^{z_i}(b_i)}
```

Algorithme réel Cet algorithme théorique n'est cependant pas réalisable dans la pratique : les deux boucles "pour tous" (lignes 3 et 4) ne sont pas réalisables dans la pratique : la première consiste à faire la mise à jour exhaustive ce qui rend inutile l'élagage ; la deuxième nécessite d'énumérer un ensemble de taille exponentielle par rapport au temps. Dans la pratique, l'algorithme pratique approche l'algorithme théorique en échantillonnant des valeurs dans ces deux boucles :

- il ne sélectionne qu'un sous-ensemble des arbres joints possibles ;
- il ne considère qu'un sous-ensemble des historiques.

Cette approche est intéressante parce qu'elle utilise des croyances issues de politiques heuristiques pour sélectionner les arbres de politiques locales à conserver dans les ensembles $\mathcal{Z}_t^i$.

4.4.2 MBDP

MBDP (*Memory Bounded Dynamic Programming* - Programmation dynamique à mémoire bornée) [Seuken et Zilberstein, 2007b] a été proposé pour résoudre des Dec-POMDPS avec des horizons de grande taille (algorithme 4.6). Pour cela, seul un nombre borné d'arbres de politiques par agent est conservé pour chaque horizon de planification. MBDP remplace la phase d'élagage exact par un élagage à base de points qui ne conserve qu'un nombre borné *maxTrees* d'arbres de

politiques par agents :

$$|\mathcal{Z}_t^i| \leq maxTrees \qquad \forall t, \forall i \in \mathcal{I} \tag{4.52}$$

La politique prend la forme d'un treillis de largeur *maxTrees* (figure 4.5). Ceci permet de borner la complexité aussi bien en terme de temps que d'espace mémoire : la complexité est linéaire avec l'horizon et exponentielle avec le nombre d'observations.

Sélection Pour cela, *maxTrees* points de croyance (distributions de probabilités sur les états) sont générés en utilisant une ou plusieurs heuristiques. Pour chaque point de croyance b, la meilleure politique jointe, $z = (z_1, \ldots, z_n)$, est sélectionnée (ligne 3) et les arbres locaux correspondant z_i sont déplacés de $\bar{\mathcal{Z}}_i^t$ vers $\mathcal{Z}_i^t$ (lignes 5 et 6). Alors que PBDP génère des croyances multi-agent b_i de l'agent i pour sélectionner des arbres z_i de l'agent i, MBDP génère des distributions de probabilités b sur les états, que l'on peut assimiler à des croyances centralisées, sélectionner des arbres joints z.

Remarque: Comme le nombre d'arbres joints *maxTrees* est relativement faible, on veut éviter une situation où les mêmes arbres sont sélectionnés plusieurs fois pour des points de croyance b différents. Pour cette raison, les arbres locaux sélectionnés sont supprimés de l'ensemble $\bar{\mathcal{Z}}_i^t$. Ceci permet d'obtenir la bonne propriété que quand le paramètre *maxTrees* tend vers l'infini, MBDP effectue une planification exacte. Une limite de cette approche est qu'il est possible qu'aucun arbre joint z optimal pour b ne soit sélectionné après l'opération de *lookahead*. Ceci se produit si tous les arbres joints optimaux z pour b ont certain de leurs arbres locaux z_i sélectionnés et certains non sélectionnés :

$$\forall z \in \arg\max V^z(b), \exists K \subsetneq \mathcal{I}, K \neq \emptyset \text{ t.q. } \begin{cases} \forall i \in K, z_i \in \mathcal{Z}_i^t \\ \forall i \in \mathcal{I} \setminus K, z_i \in \bar{\mathcal{Z}}_i^t \end{cases} \tag{4.53}$$

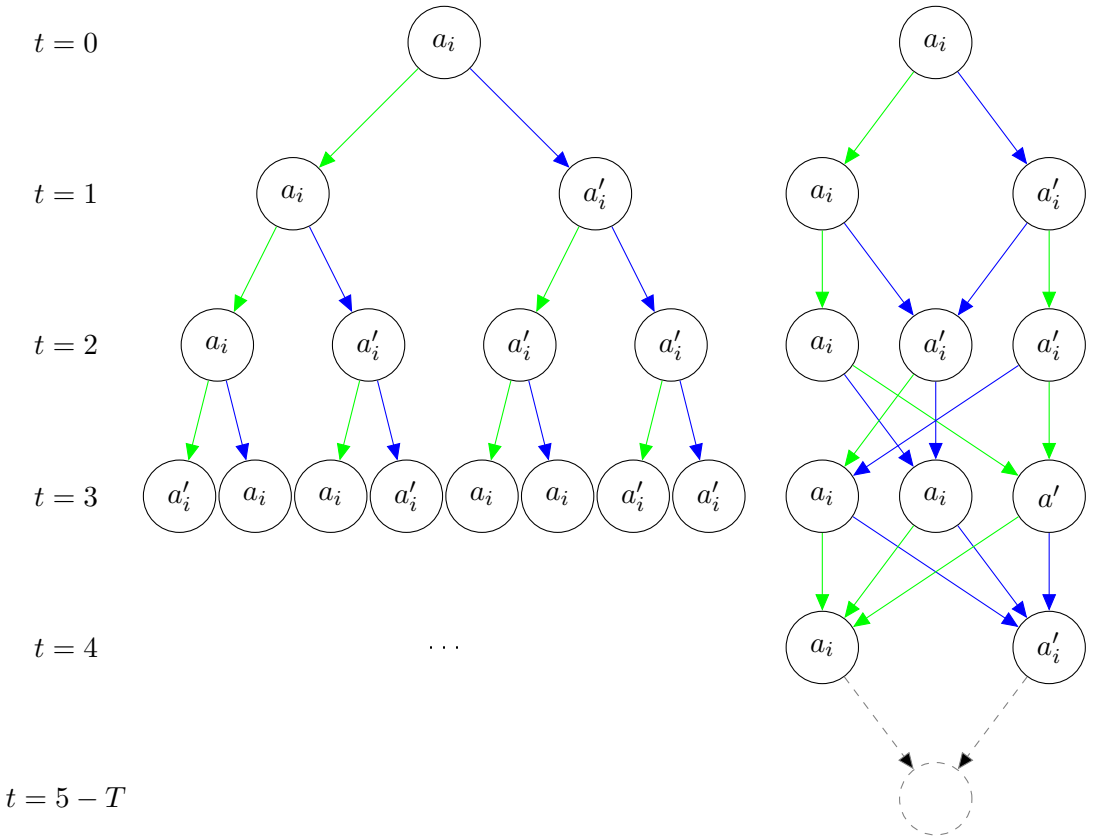
Heuristiques Pour générer les points b de croyance considérés, MBDP utilise un certain nombre d'heuristiques. Les points de croyance utilisés sont des estimations de la probabilité sur les états étant donné des politiques heuristiques $\tilde{\pi}$: politique optimale MDP, politique aléatoire, etc. On définit, les distributions p_t de probabilité heuristiques *a priori* sur les états S_t :

$$p_t(s) = \Pr(S_t = s | B_0 = b_0, \tilde{\pi}) \qquad \forall s \in \mathcal{S} \tag{4.54}$$

Plusieurs points de croyance p_t peuvent être obtenus en utilisant plusieurs politiques heuristiques. Si plus de points de croyance sont nécessaires, d'autres points de croyance b à l'instant t sont des estimations par échantillonnages p_t :

$$b = \frac{1}{M} \sum_{k=0}^{M} \delta_{x_k} \text{ avec } \forall k, x_k \sim p_t \tag{4.55}$$

Complexité Chaque mise à jour exhaustive génère au plus $|\mathcal{A}_i| maxTrees^{|\Omega_i|}$ arbres de politiques pour chaque agent i et au plus $|\mathcal{A}| maxTrees^{\sum_i |\Omega_i|}$ arbres jointes. Pour chaque arbre joint z, le vecteur α^z correspondant doit être calculé : le calcul d'un vecteur α^z est en $|\mathcal{S}|^2|\Omega|$. Pour chaque point de croyance, chercher le meilleur arbre joint est en $|\mathcal{S}||\mathcal{A}| maxTrees^{\sum_i |\Omega_i|}$. La complexité globale est en $T|\mathcal{S}|^2|\mathcal{A}||\Omega| maxTrees^{|\mathcal{I}|\omega}$ avec $\omega = max_i|\Omega_i|$. La complexité est constante par rapport à l'horizon de planification. Cependant, elle reste exponentielle par rapport



Politique non-contrainte Politique à largeur bornée ($maxTrees = 3$)

$H = 5$, $\mathcal{A}_i = \{a_i, a'_i\}$, $\Omega_i = \{o_i, o'_i\}$

FIGURE 4.5 – Politique locale générale et politique à largeur bornée

Algorithme 4.6 : Élagage à base de points (MBDP)

Données : Points de croyance $\mathcal{B}^t$ avec $|\mathcal{B}^t| = maxTrees$;
Ensembles des arbres locaux $\forall i, \bar{\mathcal{Z}}_t^i$ à élaguer (et leurs vecteurs α)
Résultat : Ensembles des arbres locaux $\forall i, \mathcal{Z}_t^i$ élagués (et leurs vecteurs α) avec $|\mathcal{Z}_t^i| \leq maxTrees$

1 **pour chaque** $i \in \mathcal{I}$ **faire** $\mathcal{Z}_t^i \leftarrow \emptyset$
2 **pour chaque** $b \in \mathcal{B}^t$ **faire**
`/* Lookahead */`
3 $(q_i)_{i \in \mathcal{I}} \leftarrow \arg\max_{q \in \prod_i \bar{\mathcal{Z}}_t^i} V_q(b)$
4 **pour chaque** $i \in \mathcal{I}$ **faire**
`/* Ajouter` z_i `à la sélection */`
5 $\mathcal{Z}_i^t \leftarrow \mathcal{Z}_i^t \cup \{q_i\}$
6 $\bar{\mathcal{Z}}_t^i \leftarrow \bar{\mathcal{Z}}_t^i \setminus \{q_i\}$

au nombre d'observations du fait de la phase de mise à jour exhaustive : la fin de ce chapitre présente des méthodes qui cherchent à résoudre ce problème, soit en utilisant une mise à jour partielle (IMBDP et MBDP-OC), soit en évitant l'étape de mise à jour (PBIP).

L'idée principale de MBDP est qu'en bornant le nombre d'arbres de politiques par agent, on borne en même temps le nombre d'arbres construits par mise à jour et donc la complexité de l'algorithme. Cependant, le nombre *maxTrees* maximal d'arbres de politiques reste faible car la complexité croît très fortement avec ce paramètre, en particulier quand le nombre d'observations est important. Le nombre d'arbres locaux construits par mise à jour exhaustive est en $|\mathcal{A}_i| maxTrees^{|\Omega_i|}$. Le nombre de points de croyance utilisés est le même que la borne *maxTrees* et est donc très petit et représente mal l'ensemble des situations possibles en particulier quand le nombre d'état est important car l'espace des croyances est un espace continu de dimension $|\mathcal{S}|-1$. De plus, les heuristiques utilisées sont passablement *ad hoc* : elles utilisent plusieurs estimations b différentes *d'une même distribution* de probabilité p_t heuristique *à priori* de l'état S_t. Dans la seconde partie de cette thèse, nous proposons d'utiliser comme points les croyances centralisées, c'est-à-dire les distributions de probabilité *a posteri* de l'état S_t étant donné différentes valeurs de l'historique joint H_t

$$b(s) = \Pr(S_t = s | H_t) \qquad \forall s \in \mathcal{S} \tag{4.56}$$

Notre formulation permet d'utiliser plus d'informations heuristiques (plus de points de croyance) ce qui permet de sélectionner des politiques de meilleure qualité.

4.4.3 Mise à jour partielle

Les techniques de mise à jour partielle limitent la complexité liée à la mise à jour en ne générant qu'un sous-ensemble des arbres construit à partir des arbres $\mathcal{Z}_t^i$ afin de passer à l'échelle par rapport au nombre d'observations :

$$\mathcal{Z}_{t-1}^i \subset \mathcal{A}_i \times (\mathcal{Z}_t^i)^{\Omega_i} \tag{4.57}$$

4.4.3.1 IMBDP

IMBDP (*Improved Memory Bounded Dynamic Programming* – Programmation dynamique à mémoire bornée améliorée) [Seuken et Zilberstein, 2007a] est une modification de MBDP qui

remplace la phase de mise à jour exhaustive par une phase de mise à jour partielle (algorithme 4.7) : elle fait la mise à jour un sous-ensemble $\bar{\Omega}_i$ de l'ensemble Ω_i des observations locales.

Ensemble restreint d'observations L'algorithme identifie un ensemble $\bar{\Omega}_i$ de *maxObs* observations locales probables à l'instant t : pour cela, il génère un point de croyance b heuristique de la même manière que PBDP et en déduit une action a prescrite par la politique heuristique $\tilde{\pi}$. L'ensemble $\bar{\Omega}_i$ est alors défini comme l'ensemble des *maxObs* observations les plus probables selon

$$\Pr(O_t^i = o_i | S_{t-1} \sim b, A_t = a) \tag{4.58}$$

Mise à jour partielle Au lieu de générer tous les arbres de politiques possibles en assignant à chaque observation locale tous les sous-arbres possibles, ceci n'est fait que pour les *maxObs* observations de $\bar{\Omega}_i$. Les $|\mathcal{A}_i| maxTrees^{maxObs}$ arbres ainsi générés sont complétés de manière aléatoire pour les autres observations locales $o_i \in \Omega_i \setminus \bar{\Omega}_i$. Pour les observations locales $o_i \in \Omega_i \setminus \bar{\Omega}_i$, les arbres sont ensuite optimisés par *hill-climbing* pour b.

Algorithme 4.7 : Mise à jour partielle (IMBDP)

Données : Ensmbles des arbres locaux $\forall i \in \mathcal{I}, \mathcal{Z}_{t+1}^i$ (et leurs vecteurs α) ;
Point de croyance b et action jointe a donnée par la politique heuristique $\tilde{\pi}$
Résultat : Ensembles des arbres locaux $\forall i \in \mathcal{I}, \bar{\mathcal{Z}}_t^i$ (et leurs vecteurs α)

1 Générer une croyance b selon $\tilde{\pi}$ et a l'action jointe associée
2 **pour chaque** $i \in \mathcal{I}$ **faire**
`/* Mise à jour partielle pour l'agent` i `sur l'ensemle` $\bar{\Omega}_i$ `*/`
3 $\bar{\Omega}_i \leftarrow$ *maxObs* meilleurs éléments de Ω_i selon $\Pr(o_i|b,a)$
4 $\bar{\mathcal{Z}}_t^i \leftarrow \{z_i | \varphi_i(z_i) \in \mathcal{A}_i; \forall o_i \in \bar{\Omega}_i, \psi_i(z_i, o_i) \in \mathcal{Z}_{t+1}^i; \forall o_i \notin \bar{\Omega}_i, \psi_i(z_i, o_i) =$ `aléatoire`$(\mathcal{Z}_{t+1}^i)\}$
5 Optimisation des arbres par *hill-climbing* pour b

IMBDP permet de passer à l'échelle par rapport au nombre d'observations. Cependant, la méthode utilisée pour faire la mise à jour partielle est assez *ad hoc* et ne se base que sur un échantillon de l'exécution.

4.4.3.2 MBDP-OC

L'approche IMBDP utilise uniquement la probabilité des observations locales et ignore leur valeur. MBDP-OC (*Memory Bounded Dynamic Programming with Observations Compression* – Programmation dynamique à mémoire bornée avec compression d'observations) [Carlin et Zilberstein, 2008b] modifie IMBDP afin de prendre en compte la valeur des observations : MBDP-OC tente de minimiser les erreurs de qualité de solution dues à la mise à jour partielle.

Mise à jour partielle L'idée de la compression des observations est de fusionner certaines observations en groupes d'observations $\bar{o}_i$. Lors de la phase de mise à jour, les arbres générés ont le même sous-arbre pour toutes les observations $o_i \in \bar{o}_i$ d'un même groupe (algorithme 4.8) :

$$\mathcal{Z}_t^i(a_i) = \left\{ z_i | \varphi_i(z_i) = a_i; \forall \bar{o}_i \in \bar{\Omega}_i(a_i), \exists z_i' \in \mathcal{Z}_{t+1}^i, \forall o_i \in \bar{o}_i, \psi_i(z_i, o_i) = z_i' \right\} \tag{4.59}$$

Ceci est fait de manière indépendante pour chaque action locale $a_i \in \mathcal{A}$:

$$\mathcal{Z}_t^i = \bigcup_{a_i \in \mathcal{A}_i} \mathcal{Z}_t^i(a_i) \tag{4.60}$$

Algorithme 4.8 : Mise à jour partielle (MBDP-OC)

Données : Ensembles des arbres locaux $\forall i \in \mathcal{I}, \mathcal{Z}_t^i$ (et leurs vecteurs α) ; politique heuristique $\tilde{\pi}$
Résultat : Ensembles des arbres locaux $\forall i \in \mathcal{I}, \bar{\mathcal{Z}}_i^t$ (et leur vecteurs α)

Générer une croyance b selon $\tilde{\pi}$
pour chaque $i \in \mathcal{I}$ **faire**
  `/* Mise à jour partielle pour l'agent` i `*/`
  $\mathcal{Z}_t^i \leftarrow \emptyset$
  pour chaque $a_i \in \mathcal{A}_i$ **faire**
    `/* Mise à jour partielle pour l'action locale` a_i `*/`
    $\bar{\Omega}_i \leftarrow$ `compressionObservations`$(i, b, a_i, (\mathcal{Z}_{t+1}^i)_{i\in\mathcal{I}})$
    $\mathcal{Z}_t^i \leftarrow \mathcal{Z}_i^t \cup \{z_i | \varphi_i(z_i) = a_i, \forall \bar{o}_i \in \bar{\Omega}_i, \exists z_i' \in \mathcal{Z}_{t+1}^i, \forall o_i \in \bar{o}_i, \psi_i(z_i, o_i) = z_i'\}$

Compression des observations La procédure de compression d'observations génère pour chaque agent i une partition $\bar{\Omega}_i$ de l'ensemble Ω_i des observations locales. Au départ les groupes $\bar{o}_i \in \bar{\Omega}_i$ d'observations sont les singletons, $\bar{\Omega}_i = \{\{o_i\} | o_i \in \Omega_i\}$. Chaque groupe $\bar{\Omega}_i$ est associé à une erreur $\epsilon(\bar{\Omega}_i)$. Les groupes sont fusionnés de manière à minimiser une erreur d'agrégation $\epsilon(\bar{\Omega}_i \cup \bar{\Omega}_i')$ (algorithme 4.9).

Algorithme 4.9 : Compression d'observations

Données : Agent i ;
  Croyance b ;
  Action locale a_i ;
  Ensembles des arbres locaux $\forall j, \mathcal{Z}_t^j$ de politiques locales
Résultat : Ensemble $\bar{\Omega}_i$ d'au plus *maxObs* groupes d'observations

`/* Initialisation : chaque observation a son propre groupe */`
$\bar{\Omega}_i \leftarrow \{\{o_i\} | o_i \in \Omega_i\}$
tant que $|\bar{\Omega}_i| > maxObs$ **faire**
  `/* Recherche de deux ensembles` $\bar{o}_i$ `et` $\bar{o}_i'$ `à fusionner */`
  $(\bar{o}_i, \bar{o}_i') \leftarrow \arg\min_{\{\bar{o}_i, \bar{o}"_i\} \subset \bar{\Omega}_i} \epsilon(\bar{o}_i \cup \bar{o}_i')$
  `/* Fusionne les deux ensembles` $\bar{o}_i$ `et` $\bar{o}_i'$ `*/`
  $\bar{\Omega}_i \leftarrow \bar{\Omega}_i \setminus \{\bar{o}_i, \bar{o}_i'\}$
  $\bar{\Omega}_i \leftarrow \bar{\Omega}_i \cup \{\bar{o}_i \cup \bar{o}_i'\}$

Erreur d'agrégation L'erreur d'agrégation est définie comme :

$$\epsilon(\bar{o}_i) = \min_{z_i \in \mathcal{Z}_t^i} \sum_{o_i \in \bar{o}_i} \max_{\substack{a_{-i} \in \mathcal{A}_{-i} \\ a = \langle a_i, a_{-i} \rangle}} \sum_{\substack{o_{-i} \in \Omega_{-i} \\ o = \langle o_i, o_{-i} \rangle}} \Pr(o|b,a) \max_{\substack{z_{-i} \in \mathcal{Z}_t^{-i} \\ z = \langle z_i, z_{-i} \rangle}} \underbrace{\left[\max_{\substack{\bar{z}_i \in \mathcal{Z}_t^i \\ \bar{z} = \langle \bar{z}_i, z_{-i} \rangle}} V^{\bar{z}}(b,a,o) - V^z(b,a,o) \right]}_{L_i(z_i|b,a,o,z_{-i})} \tag{4.61}$$

où $L_i(z_i|b,a,o,z_{-i})$ est la perte associée au fait d'effectuer z_i au lieu de la meilleure politique locale disponible après avoir effectué a et reçu o depuis la croyance b sachant que les autres agents font z_{-i}.

4.4.4 PBIP

PBIP (*Point Based Incremental Pruning* – Élagage incrémental à base de points) [Dibangoye *et al.*, 2009] utilise une autre approche pour résoudre le problème de passage à l'échelle par rapport au nombre d'observations de la mise à jour exhaustive. Alors que IMBDP et MBDP-OC remplacent la mise à jour exhaustive par une mise à jour partielle, PBIP évite la génération exhaustive des politiques jointes en utilisant une seule phase de recherche *branch-and-bound* au lieu des deux phases de mise à jour et élagage (algorithme 4.10).

Algorithme 4.10 : Opérateur de Bellman (PBIP)

Données : Ensemble $\mathcal{B}^t$ des *maxTrees* points de croyance ;
Ensemble des arbes locaux $\forall i, \mathcal{Z}^i_{t+1}$ (et leurs vecteurs α)
Résultat : Ensemble des arbres locaux $\forall i, \mathcal{Z}^i_t$ (et leurs vecteurs α), avec $|\mathcal{Z}^i_t| \leq maxTrees$

1 $blacklist \leftarrow \emptyset$
2 Soit l'espace de recherche $K = \prod_i \mathcal{A}_i \times (\mathcal{Z}^i_{t+1})^{\Omega_i}$
3 **pour chaque** $i \in \mathcal{I}$ **faire** $\mathcal{Z}^i_t \leftarrow \emptyset$
4 **pour chaque** $b \in \mathcal{B}^t$ **faire**
5 Recherche *branch-and-bound* de $z = (z_i)_{i\in\mathcal{I}} = \arg\max_{z\in K\setminus blacklist} V^z(b)$
6 $blacklist \leftarrow blacklist \cup \{z\}$
7 **pour chaque** $i \in \mathcal{I}$ **faire** $\forall i \in \mathcal{I}, \mathcal{Z}^i_t \leftarrow \mathcal{Z}^t_i \cup \{z_i\}$

Dans le cas POMDP, pour chaque b donné, il était possible de rechercher par *lookahead* le meilleur α ou de manière équivalente le meilleur arbre de politique z sans énumérer l'ensemble des possibilités (section 3.4.3) : ceci était pouvait être fait de manière efficace car pour une action a donnée le choix du vecteur α' ou de manière équivalente du sous-arbre z (section 3.5) à associer aux observations o est indépendant pour chaque observation o ; l'espace de recherche réellement parcouru est donc de taille $|\mathcal{A}||\Omega|maxTrees$ plutôt que $|\mathcal{A}|maxTrees^{|\Omega|}$. Dans le cas Dec-POMDP, du fait de la contrainte de décentralisation (4.33) le choix des sous-arbre joints $\psi(z, o)$ n'est plus indépendant : le choix d'un sous-arbre joint $\psi(z, o)$ pour un o contraint le choix des sous-arbres $\psi(z, o')$ pour tout o' qui possède un composant o_i en commun avec o : $\exists i \in \mathcal{I}, o_i = o'_i$. Pour chaque point de croyance b, le problème de trouver le meilleur arbre joint z est un problème d'optimisation combinatoire : l'algorithme MBDP utilise ujne énumération exhaustive des solutions pour résoudre ce problème ; le contribution de PBIP consiste à utiliser la recherche *branch-and-bound* pour résoudre ce problème d'optimisation combinatoire.

4.4.4.1 Opération *branch*

Un arbre de politique jointe q est définie par :
- son action racine $\varphi(z) \in \mathcal{A}$;
- pour chaque observation jointe o, un sous-arbre de politique $\psi(z, o) \in \mathcal{Z}_{t+1}$.

Un nœud $\tilde{z}$ de l'arbre de recherche représente un sous-ensemble de l'espace de recherche, c'est-à-dire de l'ensemble des arbres de politique jointe, défini par un ensemble de contraintes. Il peut être considéré comme un arbre de politique potentiellement incomplet c'est-à-dire dont certaines propriétés sont encore des inconnues.

Chaque nœud fils $\tilde{z}'$ de l'arbre de recherche est un sous-ensemble de son nœud parent $\tilde{z}$ obtenu en ajoutant une contrainte sur l'arbre de politique joint z (figure 4.6) :
- soit en fixant l'action jointe racine $\varphi(z) = a \in \mathcal{A}$: $\tilde{z}' = \{z \in \tilde{z} | \varphi(z) = a\}$;

– soit en fixant un sous-arbre de politique joint $\psi(z, o) = z' \in \mathcal{Z}_{t+1}$ pour une observation jointe $o \in \Omega$ donnée : $\tilde{z}' = \{z \in \tilde{z} | \psi(z, o) = z'\}$.

Les nœuds feuilles de l'arbre de recherche représentent des arbres de politiques joints : descendre dans la structure de l'arbre de recherche revient à ajouter des contraintes sur l'arbre de politique joint jusqu'à atteindre un nœud feuille valide (qui ne correspond qu'à une seule solution) ou invalide (qui ne correspond à aucune solution). La recherche consiste à parcourir cet arbre de recherche pour trouver un nœud feuille qui représente une solution optimale.

Remarque: Nous considérons ici deux types complètement différents d'arbres qu'il ne faut pas confondre (figure 4.7) :

– les *arbres de politiques* z dont les nœuds sont des actions et les branches des observations ;
– l'*arbre de recherche* d'un arbre de politique joint dont les nœuds $\tilde{z}$ sont des ensemble d'arbres de politiques jointes ou de manière équivalente des arbres de politiques partiels (définis par un ensemble de contraintes) et dont les branches représentent des relations de type *sous-ensemble*.

Chaque nœud de l'arbre de recherche représente (contient) un arbre de politique incomplet.

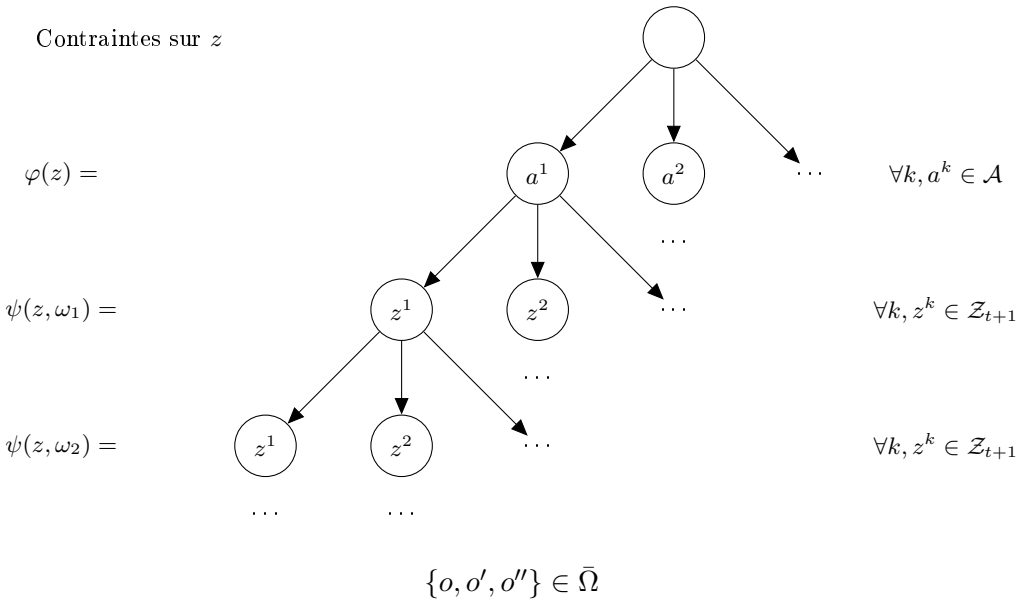


FIGURE 4.6 – Arbre de recherche d'un arbre de politique jointe z

Exemple 4.4.1*:* La figure 4.7 représente la descente dans un arbre de recherche : pour chaque profondeur de l'arbre de recherche, un nœud de l'arbre de recherche (mis en couleur dans la partie gauche de la figure) est mis en correspondance avec l'arbre de politique partiel qu'il représente (partie de droite de la figure) ; cet arbre de politique partiel représente un ensemble d'arbres de politiques complets que l'on peut obtenir en complétant les propriétés non définies de l'arbre de recherche. Le nœud racine représente un arbre complètement non spécifié c'est-à-dire l'espace de recherche complet que l'on notera $\bar{\mathcal{Z}}_t$. Le premier nœud de la deuxième couche de l'arbre de

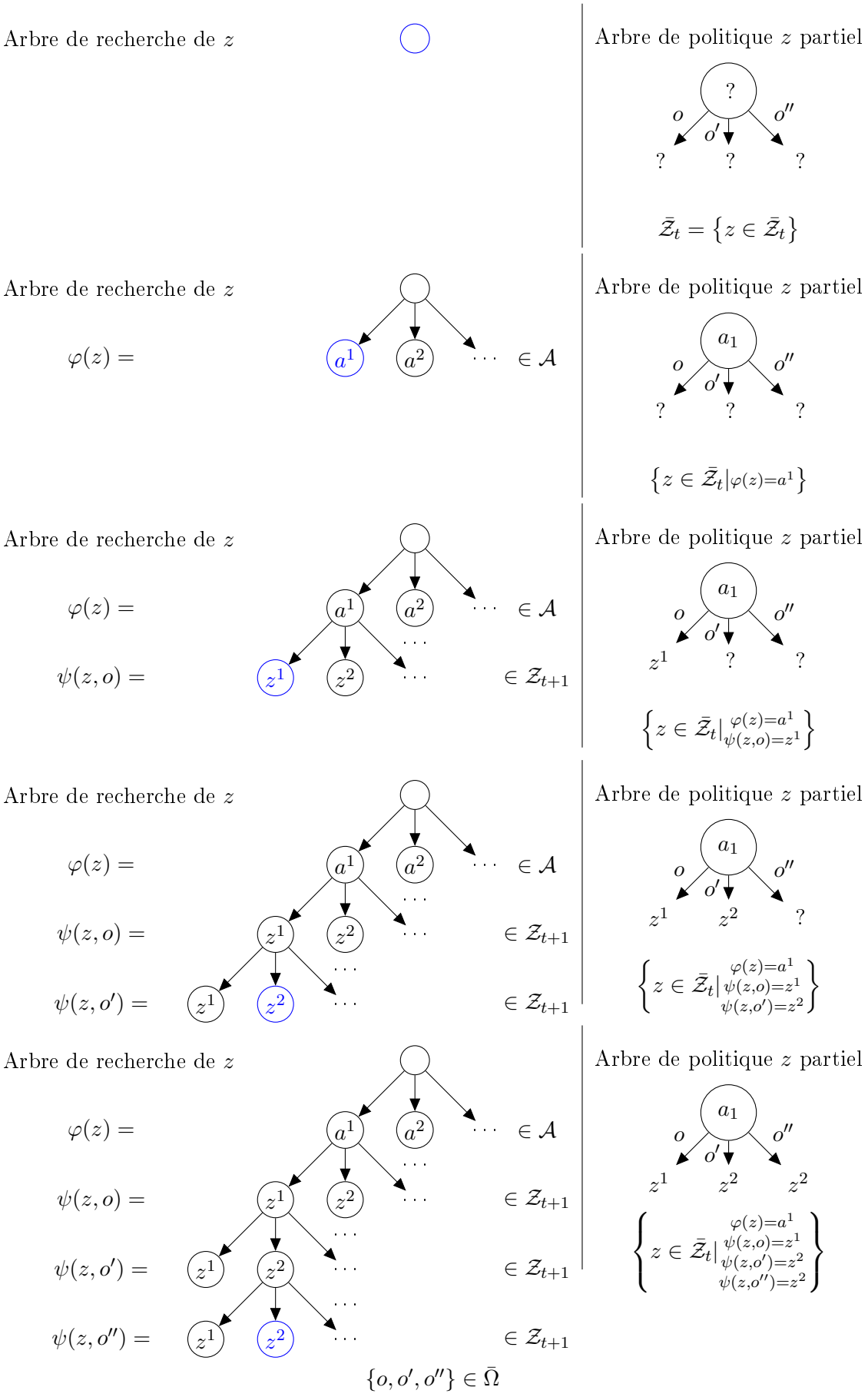


FIGURE 4.7 – Correspondance entre arbre de politique et arbre de recherche

recherche ajoute $\varphi(z) = a_1$ à l'arbre et représente le sous-ensemble de recherche

$$\left\{z \in \bar{\mathcal{Z}}_t | \varphi(z) = a_1\right\} \tag{4.62}$$

Les premier nœud de la troisième couche ajoutent la contrainte $\psi(z, o) = z'$ à l'arbre de recherche et représente un sous-ensemble de son nœud parent :

$$\left\{z \in \bar{\mathcal{Z}}_t | \varphi(z) = a_1, \psi(z, o) = z'_1\right\} \tag{4.63}$$

À la dernière couche, l'ensemble des contraintes d'un nœud donné et défini un arbre complet et représente un ensemble singleton. Le nœud coloré représente par exemple :

$$\left\{z | \varphi(z) = a_1, \psi(z, o) = z'_1, \psi(z, o') = z'_2, \psi(z, o'') = z'_2\right\} \tag{4.64}$$

4.4.4.2 Contrainte de décentralisation

L'arbre de politique joint z doit être décentralisable, c'est-à-dire qu'il doit être possible de diviser l'arbre de politique joint en $|\mathcal{I}|$ arbres de politiques locaux z_i qui vérifient l'équation (4.33).

Exemple 4.4.2*:* Par exemple, choisir le sous-arbre joint $\psi(z, (o_1, o_2))$ ajoute une contrainte sur l'arbre joint $\psi(z, (o_1, o'_2))$ parce que l'arbre local $\psi_1(z_1, o_1)$ est fixé :

$$\left.\begin{array}{l}\psi(z, (o_1, o_2)) = (\psi_1(z_1, o_1), \psi_2(z_2, o_2)) \\ \psi(z, (o_1, o'_2)) = (\psi_1(z_1, o_1), \psi_2(z'_2, o_2))\end{array}\right\} \text{ d'où } \psi(z, (o_1, o_2))_1 = \psi(z, (o_1, o'_2))_1 \tag{4.65}$$

Ainsi, après avoir choisi $\psi(z, (o_1, o_2))$ et $\psi(z, (o'_1, o'_2))$ les sous-arbres locaux $\psi_1(z_1, o_1)$, $\psi_1(z_1, o'_1)$, $\psi_2(z_2, o_2)$, $\psi_2(z_2, o'_2)$ sont définis et quatre sous-arbres joints sont complètement définis :

- le choix de $\psi(z, (o_1, o_2))$ définit les sous-arbres locaux $\psi_1(z_1, o_1)$ et $\psi_2(z_2, o_2)$;
- le choix de $\psi(z, (o'_1, o'_2))$ définit les sous-arbres locaux $\psi_1(z_1, o'_1)$ et $\psi_2(z_2, o'_2)$;
- $\psi(z, (o'_1, o_2)) = (\psi_1(z_1, o'_1), \psi_2(z_2, o_2))$;
- $\psi(z, (o_1, o'_2)) = (\psi_1(z_1, o_1), \psi_2(z_2, o'_2))$.

Pour un nœud donné $\tilde{z}$ de l'arbre de recherche, l'ensemble Ω des observations jointes peut être divisé en deux ensembles :

- l'ensemble Ω^1 des observations jointes o pour lesquelles le sous-arbre joint $\psi(z, o)$ est déjà fixé par les contraintes ;
- l'ensemble Ω^2 des observations jointes pour lesquelles le sous-arbre joint $\psi(z, o)$ n'est pas encore complètement fixé par les contraintes [3].

Ces deux ensembles partitionnent l'ensemble des observations : $\Omega = \Omega^1 \cup \Omega^2$. Après avoir assigné un sous-arbre joint à une observation jointe o, la politique jointe partielle peut être invalide : c'est le cas si elle ne vérifie pas la condition de décentralisation (4.33).

Il est uniquement nécessaire d'assigner un sous-arbre de politique joint pour $\max_i |\Omega_i|$ observations jointes judicieusement choisies et les contraintes de décentralisation permettent de

3. Ce sous-arbre z joint peut cependant déjà être partiellement contraint par des contraintes de type $\psi_i(z_i, o_i) = z'_i$ avec $i \in \mathcal{I}$, $o_i \in \Omega_i$, $z'_i \in \mathcal{Z}_t^i$.

définir les sous-arbres pour les observations restantes. Une telle séquence $\bar{\Omega} = (\omega_1, \ldots, \omega_{max_i|\Omega_i|})$ d'observations est appelée *base d'observations*. La profondeur de l'arbre de recherche est donc $1 + \max_i |\Omega_i|$ au lieu de $1 + |\Omega|$. Il est possible d'éviter de vérifier les contraintes de décentralisation en fixant des observations jointes qui sont différentes composante par composante : ceci est possible pour les $min_i|\Omega_i|$ observations de la base d'observations $\bar{\Omega}$.

En résumé, le problème d'optimisation combinatoire utilise $1 + |\bar{\Omega}|$ variables :

$$\varphi(z) \in \mathcal{A} \tag{4.66}$$

$$\psi(z, o) \in \mathcal{Z}_{t+1} \qquad \forall o \in \bar{\Omega} \tag{4.67}$$

La contrainte de décentralisation (4.33) lie les $\psi(z, o)$ pour les $\max_i |\Omega_i| - \min_i |\Omega_i|$ dernières observations de $\bar{\Omega}$ via les $\psi_i(z_i, o_i)$.

4.4.4.3 Opération *bound*

Pour guider la recherche *branch-and-bound*, une valeur heuristique $f(\tilde{z}) = V^{\tilde{z}}(b)$ de tout nœud de l'arbre de recherche est introduite : il doit s'agir d'un majorant de la valeur $V^z(b)$ du meilleur arbre de politique $z \in \tilde{z}$ vérifiant les contraintes $\tilde{z}$: $f(\tilde{z}) \geq \max_{z \in \tilde{z}} V^z(b)$.

Une telle valeur heuristique est typiquement introduite par relaxation des contraintes du problème. Le problème est relâché en ignorant les contraintes de décentralisation pour les observations de Ω^2. Pour ces observations, la contribution $k(a, o, z')$ du meilleur sous-arbre joint est utilisée :

$$f(\tilde{z}) = \mathcal{R}(b, a) \quad + \gamma \sum_{o \in \Omega^1} \mathcal{O}(o|b, a) \left[\alpha_{\psi(\tilde{z}, o)} \cdot \tau(b, a, o)\right] \quad + \gamma \sum_{o \in \Omega^2} \mathcal{O}(o|b, a) \max_{z' \in \mathcal{Z}_{t+1}} \left[\alpha_{z'} \cdot \tau(b, a, o)\right] \tag{4.68}$$

$$= \mathcal{R}(b, a) \quad + \gamma \sum_{o \in \Omega^1} k(a, o, \psi(\tilde{z}, o)) \quad + \gamma \sum_{o \in \Omega^2} \max_{z' \in \mathcal{Z}_{t+1}} k(a, o, z') \tag{4.69}$$

$$= g(\tilde{z}) \quad + h(\tilde{z}) \tag{4.70}$$

où $\mathcal{O}(o|b, a) = \Pr(O_{t+1} = o | B_t = b, A_t = a)$. Il s'agit d'une heuristique de type A* composée d'une somme d'une partie exacte $g(\tilde{z})$ et d'une partie admissible $h(\tilde{z})$.

L'algorithme *backtrack* quand cette heuristique est inférieure à la valeur du meilleur arbre de politique joint trouvé jusqu'alors.

Les $k(a, o, z')$ sont les contributions des sous-arbres pour b :

$$k(a, o, z') = \alpha^{z'} \cdot \tau(b, a, o) \tag{4.71}$$

Elles peuvent être précalculées au début du *lookahead* (algorithme 4.11) en même temps que les $k(a, o) = \max_{z' \in \mathcal{Z}_{t+1}} k(a, o, z')$.

4.4.4.4 Détails de l'algorithme

Nous présentons ici un certain nombre de détails de l'algorithme qui peuvent être ignorés en première lecture.

Doublons Comme MBDP, PBIP évite de sélectionner plusieurs fois les mêmes arbres (voir section 4.4.2). Pour cela, PBIP maintient un ensemble *blacklist* des arbres *joints* déjà sélectionnés : cet ensemble est exclu de l'ensemble de recherche. La condition $z \notin$ *blacklist* est vérifiée dans les nœuds terminaux de l'arbre de recherche (les arbres de politiques complets). Comme pour

Algorithme 4.11 : Calcul des contributions

Données : Croyance initiale b ;
Ensemble des sous-arbres de politique joints $\mathcal{Z}_{t+1}$ (et leurs vecteurs α) ;
Résultat : Contributions $k(a, o, z')$ et meilleures contributions $k(a, o)$

```
pour chaque a ∈ A, o ∈ Ω faire
    b' ← τ(b, a, o)
    pour chaque z' ∈ Z_{t+1} faire
        k(a, o, z') ← α^{z'} · b'
    k(a, o) ← max_{z'∈Z_{t+1}} k(a, o, z')
```

MBDP, quand le paramètre *maxTrees* tend vers l'infini, on obtient une planification exacte. Cette méthode évite le défaut de la méthode utilisée par MBDP : dans MBDP, il est possible qu'aucun arbre joint z optimal pour un des points de croyance b ne soit sélectionné (si (4.53) est vérifié). Cependant, cette approche peut sélectionner moins que *maxTrees* arbres locaux pour certains agents.

Stratégie d'exploration L'exploration proposée est en profondeur d'abord avec certains aspects de meilleur d'abord : les enfants d'un nœud donné de l'arbre de recherche sont développés dans l'ordre de leur valeur heuristique. La recherche en meilleur d'abord a besoin de conserver un nombre d'arbres de recherche exponentiel avec le nombre d'observations, ce qui est évité par la recherche en profondeur d'abord.

Stratégie d'exploration alternative D'autres stratégies de recherche peuvent être envisagées. En particulier, la recherche en meilleur d'abord devrait conduire à des temps de calcul inférieurs pour une consommation mémoire supérieure. Nous avons essayé d'utiliser la recherche en meilleur d'abord : nous appellerons cette méthode PBIP/BeFS (pour *Best First Search*). Nous avons testé cette méthode sur un certain nombre de *benchmarks* : la consommation mémoire reste faible et les temps de calcul sont bien meilleurs (voir le chapitre 6).

4.4.4.5 Analyse

Pour une croyance donnée, les $|\mathcal{A}||\Omega|maxTrees^{|\mathcal{I}|}$ contributions sont calculées (pour chaque tuple constitué d'une action, d'une observation et d'un sous-arbre de politique) : chaque contribution est calculée en $|\mathcal{S}|^2$. Puis une recherche de politique est faite : le nombre de nœuds feuilles de l'arbre de recherche est $|\mathcal{A}|maxTrees^{|\mathcal{I}|\omega}$ avec $\omega = |\bar{\Omega}| = max_i|\Omega_i|$. L'opération *bound* d'un nœud de politique est en $|\Omega|$. La complexité dans le pire cas de l'opérateur DP est en :

$$maxTrees|\mathcal{A}||\Omega|\left[maxTrees^{|\mathcal{I}|}|\mathcal{S}|^2 + maxTrees^{|\mathcal{I}|\omega}\right] \tag{4.72}$$

Contrairement aux approches IMBDP et MBDP-OC, la complexité théorique reste exponentielle avec $\max_i |\Omega_i|$ mais en pratique une grande partie de l'arbre de recherche est élaguée. Dans les *benchmarks*, PBIP résout des problèmes avec un plus grand nombre d'observations que MBDP et trouve des solutions de meilleure qualité que les approches utilisant une mise à jour partielle [Dibangoye *et al.*, 2009].

Cette approche est intéressant car elle montre comment on peut généraliser l'opération de *lookahead*. La complexité du *lookahead* est supérieure que dans le cas POMDP. Les méthodes d'optimisation combinatoire peuvent être utilisées pour trouver une solution, exacte ou approchée, au problème de *lookahead* pour une complexité inférieure. Ce problème d'optimisation combinatoire

peut par exemple être résolu avec les techniques de résolution de problèmes de satisfaction de contraintes pondérées [Kumar et Zilberstein, 2010]. Dans le chapitre 6, nous proposons de chercher une solution approchée de ce problème en utilisant les techniques de recherche d'équilibre de Nash.

De plus, le principe de la recherche d'arbre de politique par *branch-and-bound* est importante pour la suite dans la thèse. Dans notre contribution principale (chapitre 7), nous utilisons une approche similaire pour chercher des arbres z_i de politiques *locales* par recherche *branch-and-bound*.

4.4.5 PBIP-IPG

La méthode IPG (section 4.3.4) peut être utilisée avec l'ensemble des méthodes à base de points [Amato *et al.*, 2009]. L'utilisation des points de croyance permet d'inclure la connaissance de l'état initial dans l'analyse d'accessibilité de manière immédiate. Si certains états S_t ne sont pas réalisables pour le point de croyance b considéré, cette informations d'accessibilité peut-être utilisée pour limiter les états $S_{t+1} \in \mathcal{S}_i(b, a_i, o_i)$ accessibles pour le sous-arbre d'observation o_i de l'arbre z_i d'action racine a_i :

$$\mathcal{S}_i(b, a_i, o_i) = \left\{ s' \in \mathcal{S} | \exists (s, a_{-i}, o_{-i}) \in \mathcal{S} \times \mathcal{A}_{-i} \times \Omega_{-i}, \mathcal{J}(s', \langle o_i, o_{-i} \rangle | s, \langle a_i, a_{-i} \rangle) b(s) \neq 0 \right\} \tag{4.73}$$

$$\mathcal{Z}_i^{t+1}(a_i, o_i) = \left\{ z'_i \in \mathcal{Z}_i^{t+1} | \exists b'_i \in \Delta(\mathcal{S}_i(b, a_i, o_i) \times \mathcal{Z}_i^{t+1}), V^{z'_i}(b_i) \geq \max_{\bar{z}'_i \in \mathcal{Z}_i^{t+1} \setminus \{z'_i\}} V^{\bar{z}'_i}(b_i) \right\} \tag{4.74}$$

En particulier, l'algorithme PBIP-IPG est obtenu en utilisant PBIP. Le sous-arbre d'observation jointe o de l'arbre d'action racine a est recherché uniquement dans le sous-ensemble

$$\mathcal{Z}_{t+1}(b, a, o) = \prod_{i \in \mathcal{I}} \mathcal{Z}_i^{t+1}(b, a_i, o_i) \tag{4.75}$$

4.5 Conclusion

Nous avons présenté la planification multi-agent en nous concentrant sur les approches à base de points. Ces approches fixent une borne (*maxTrees*) au nombre d'arbres de politiques locales que chaque agent doit conserver à chaque étape de la planification dynamique. Elles échantillonnent l'espace des distributions sur les états et, pour chaque point de croyance b échantillonné, cherchent le meilleur arbre joint z. La contrainte de décentralisation sur la politique (4.13) rend la recherche du meilleur arbre joint z beaucoup plus difficile que pour la planification mono-agent : il s'agit d'un problème d'optimisation combinatoire.

L'algorithme PBIP est particulièrement intéressant. Il résout ce problème d'optimisation combinatoire par recherche *branch-and-bound* [Dibangoye *et al.*, 2009] ce qui ouvre la voie à l'utilisation d'autres méthodes exactes ou approchées [Kumar et Zilberstein, 2010] d'optimisation combinatoire.

Dans le chapitre 7, nous revenons sur ces limites et proposons une nouvelle approche pour les lever :

- en considérant les points de croyance définis par (5.4) ;
- en sélectionnant tous les arbres locaux à un instant t au sein d'un même problème d'optimisation ce qui permet de sélectionner des arbres complémentaires ;

– en levant la limite sur le nombre de points de croyance considérés qui peux être arbitrairement plus grand que la limite *maxTrees* sur le nombre d'arbres ce qui permet de mieux couvrir l'ensemble des croyances utiles.

Deuxième partie

Contributions

Chapitre 5

Introduction

Sommaire

Nous avons présenté les approches de planification par programmation dynamique de l'état de l'art. Dans le cadre multi-agent, ces approches calculent des ensembles $\mathcal{Z}_t^i$ d'arbres de politiques à partir des ensembles $\mathcal{Z}_{t+1}^i$ de sous-arbres.

Les algorithmes de type MBDP (MBDP, IMBDP, MBDP-OC et PBIP) sont similaires (algorithme 5.1). Pour un ensemble $\mathcal{B}_t$ de points de croyance b à l'instant t fixé, ils sélectionnent, pour chaque agent i, au plus *maxTrees* arbres locaux z_i en cherchant les arbres joints z optimaux pour chaque point de croyance b. Leur comportement diffère en deux points :

- la manière dont le meilleur arbre joint z (ou une version approchée) pour un point de croyance b donné est recherché (opération de *lookahead*, ligne 4) ;
- l'ensemble *blacklist* des solutions exclues de l'ensemble de recherche (construit à la ligne 5).

Algorithme 5.1 : Opérateur de Bellman à base de points générique

Données : Ensemble $\mathcal{B}_t$ de points de croyance ;
Ensemble des arbres locaux $\forall i, \mathcal{Z}_{t+1}^i$ (et leurs vecteurs α)
Résultat : $\forall i, \mathcal{Z}_t^i$ (et leurs vecteurs α)

```
1 blacklist ← ∅
2 pour chaque i ∈ I faire  Z_t^i ← ∅
3 pour chaque b ∈ B_t faire
4     z = (z_i)_{i∈I} ← lookahead(b, (Z_{t+1}^i)_{i∈I}, blacklist)
5     blacklist ← blacklist ∪ filtre(z)
6     pour chaque i ∈ I faire  Z_t^i ← Z_i^t ∪ {z_i}
```

Remarque: L'utilisation d'un ensemble *blacklist* de solutions exclues permet d'éviter de sélection-

ner plusieurs fois les mêmes arbres de politiques pour différents points de croyance b : l'idée est qu'il vaut mieux ajouter des arbres sous-optimaux qu'utiliser un petit nombre d'arbres de politique. Pour MBDP, tous les arbres z_i locaux déjà sélectionnés sont exclus de la recherche [4]

$$blacklist \leftarrow blacklist \cup \left\{ \langle z_i, z'_{-i} \rangle \middle| i \in \mathcal{I}, z'_{-i} \in \mathcal{A}_i \times (\mathcal{Z}^i_{t+1})^{\Omega_i} \right\} \tag{5.1}$$

Pour PBIP, seuls les arbres joints z déjà sélectionnés sont exclus de la recherche

$$blacklist \leftarrow blacklist \cup \{z\} \tag{5.2}$$

La différence entre ces différentes approches est principalement un détail d'implémentation. (voir sections 4.4.2 et 4.4.4) pour une discussion sur ce point). Une autre solution, utilisée par PBPG, pour éviter de sélectionner un nombre d'arbres de politiques locales inférieur à *maxTrees* consiste à générer de nouveaux échantillons si nécessaire [Wu *et al.*, 2010a].

5.1 Limites

5.1.1 Complexité du *lookahead*

Une première limite de ces approches est la complexité du *lookahead*. Le *lookahead* exact nécessite de faire une recherche dans un ensemble de taille exponentielle par rapport au nombre d'observations. De plus, évaluer les vecteurs α pour tous les arbres joints $z \in \mathcal{Z}_t = \prod_{i \in \mathcal{I}} \mathcal{Z}^i_t$ sélectionnés a une complexité exponentielle par rapport au nombre d'agents : $|\mathcal{Z}_t| \leq maxTrees^{|\mathcal{I}|}$. Ces deux facteurs limitent fortement la valeur du paramètre *maxTrees*.

5.1.2 Échantillonnage et sélection

Une seconde limite de ce type d'approche réside dans les points de croyance échantillonnés et la sélection des arbres de politique [5].

Les points de croyance b considérés par MBDP et ses dérivés ainsi que DecRSPI sont des estimations *indépendantes* d'une même distribution p_t de probabilité de l'état S_t *a priori*, c'est-à-dire vue depuis l'instant $t = 0$ quand les agents suivent une politique heuristique $\tilde{\pi}$ donnée :

$$b(s) \approx p_t(s) = \Pr(S_t = s | \tilde{\pi}, B_0 = b_0) \tag{5.3}$$

S'il s'agit effectivement de croyances de l'agent sur l'état du système S_t, il ne s'agit pas réellement de croyances telles que nous les avons définies (ni de leurs approximations) c'est-à-dire de distributions de probabilité de l'état S_t *a posteriori* quand l'agent suivrait une politique heuristique, c'est-à-dire vue depuis l'instant t étant donné l'historique joint H_t comme dans le cas des approches à base de points POMDP :

$$b(s) = B_t(s) = \Pr(S_t = s | B_0 = b_0, \tilde{\pi}, H_t) \tag{5.4}$$

Comme il n'existe qu'une seule distribution p_t *a priori* pour une politique heuristique $\tilde{\pi}$ donnée, des approximations indépendantes par échantillonnage de p_t sont utilisées pour obtenir

4. Plus précisément tous les arbres joints z, dont un arbre local z_i a déjà été sélectionné sont exclus de l'ensemble de recherche.

5. Ces points sont discutés plus en détail dans le chapitre 7.

des points de croyance b différents :

$$b = \frac{1}{M} \sum_{k=1}^{M} \delta_{x_k} \text{ avec } \forall k, x_k \sim p_t \tag{5.5}$$

où δ_x est la distribution de Dirac en x, définie intuitivement [6] par :

$$\delta_x(x') = \begin{cases} +\infty, \text{ si } x = x' \\ 0, \text{ sinon} \end{cases}$$

avec

$$\int_{-\infty}^{+\infty} \delta_x(x')\, dx' = 1$$

Ces points b estiment bien p_t et l'erreur d'estimation tend vers 0 en probabilité Cette méthode est cependant assez artificielle : ces points de croyance b approchent la même distribution p_t et utilisent le fait que le nombre N d'échantillons est faible et que ces estimations sont donc de mauvaise qualité ; quand N est grand les points b de croyance convergent vers p_t presque sûrement.

De plus, comme chaque arbre joint est sélectionné par *lookahead* :

- le nombre de points de croyance b considérés, égal à *maxTrees*, est une valeur assez faible et ces points de croyance couvrent mal l'espace $\Delta\mathcal{S}$ des distributions sur les états ;
- les arbres joints sont sélectionnés de manière indépendante alors qu'une complémentarité entre les différents arbres pourrait être exploitée.

Ces points seront développés au chapitre 7 où une méthode pour les résoudre sera proposée.

5.2 Propositions

Nous présentons deux directions opposées pour modifier les algorithmes de type MBDP qui s'attaquent respectivement aux deux limites que nous venons d'évoquer.

5.2.1 *Lookahead* approché

D'une part, dans le chapitre 6, nous essayons de planifier *plus rapidement* mais éventuellement en trouvant de *moins bonnes politiques* en réduisant la complexité de l'opération de *lookahead*. Cette dernière peut être vue comme un problème d'optimisation combinatoire qui a généralement été résolu par énumération exhaustive des solutions. L'algorithme PBIP 4.4.4 a utilisé la recherche *branch-and-bound* pour le résoudre et d'autres méthodes ont été proposées qui utilisent des problèmes de satisfaction de contraintes [Kumar et Zilberstein, 2010]. Nous proposons d'utiliser des méthodes méta-heuristiques pour trouver des solutions approchées à ce problème et ainsi limiter la complexité du *lookahead*. Nous illustrons ce propos en proposant une méthode qui cherche un optimum local au problème de *lookahead* : il s'agit plus précisément d'un équilibre de Nash et la méthode obtenue est assez similaire à PBPG [Wu *et al.*, 2010a] et DecRSPI [Wu *et al.*, 2010b].

6. Plus formellement, $\delta_x(\phi) = \phi(x), \forall \phi$.

5.2.2 Nouveau critère de sélection

D'autre part, dans le chapitre 7 qui constitue la contribution principale de la thèse, nous essayons de trouver des solutions de *meilleure qualité* pour un *temps de calcul éventuellement supérieur* en nous attaquant au problème lié à l'échantillonnage des points de croyance et à la sélection des arbres de politique. Nous proposons d'utiliser un critère de sélection plus complexe pour obtenir une solution de meilleure qualité que les approches de type MBDP. Ce critère permet d'utiliser plus de points de croyance et de sélectionner des arbres de politiques complémentaires.

Ces deux approches sont fondamentalement opposées. Cependant le *lookahead* approché peut être utilisé dans la deuxième approche.

Chapitre 6

Lookahead approché

Sommaire

Le cœur des algorithmes de type MBDP consiste en une opération de *lookahead* qui cherche le meilleur arbre de politique joint z pour un point b donné (section 4.4.4). Ce problème a généralement été résolu par énumération exhaustive [Seuken et Zilberstein, 2007b]. Des techniques classiques d'optimisation combinatoire comme la recherche *branch-and-bound* [Dibangoye *et al.*, 2008]. ont été employées pour résoudre ce problème. Cependant, la taille de ces problèmes d'optimisation combinatoire est exponentielle par rapport au nombre d'observations. Pour des problèmes avec un grand nombre d'observations, l'espace de recherche est trop important pour effectuer une recherche exacte.

Dans ce chapitre, nous proposons d'utiliser les techniques méta-heuristiques de résolution des problèmes d'optimisation combinatoire pour trouver une bonne solution approchée au problème de *lookahead* pour des temps de calculs bien inférieures à la résolution exacte *branch-and-bound*. En particulier, nous proposons une méthode de *lookahead* approché par optimisation locale : l'arbre joint est recherché comme un équilibre de Nash.

6.1 *Lookahead* approché

L'opération de *lookahead* multi-agent cherche à déterminer, pour une croyance b donnée, le meilleur arbre joint construit à partir des sous-arbres des ensembles $\mathcal{Z}^i_{t+1}$:

$$\underset{z \in \bar{\mathcal{Z}}^i_t \setminus blacklist}{\arg\max} \; V^z(b) \text{ avec } \bar{\mathcal{Z}}^i_t = \prod_{i \in \mathcal{I}} \mathcal{A}_i \times (\mathcal{Z}^i_{t+1})^{\Omega_i} \tag{6.1}$$

Le paramètre *maxTrees* influe en deux endroits sur la complexité :

- pour la résolution des instances du problème (6.1), l'espace de recherche est de taille $\prod_{i \in \mathcal{I}} |\mathcal{A}| \times maxTrees^{|\Omega_i|}$;
- pour le calcul des $maxTrees^{|\mathcal{I}|}$ vecteurs α.

Le premier point est celui qui limite le plus la valeur de *maxTrees* et a l'impact le plus important sur les temps de calcul.

La première méthode qui a été proposée pour résoudre ce problème, MBDP, utilise l'énumération exhaustive des solutions [Seuken et Zilberstein, 2007b]. Cependant, comme l'espace de recherche est de taille exponentielle, cette approche ne permet de traiter que des problèmes avec un faible nombre d'observations. Des variantes, IMBDP et MBDP-OC, ont ensuite été proposées, qui restreignaient la recherche à un sous-ensemble de l'espace de recherche afin de réduire la complexité de la recherche mais l'utilisation d'un espace de recherche restreint se traduit par une baisse importante de la qualité des solutions trouvées.

Afin de résoudre des problèmes avec un nombre d'observations supérieur, Dibangoye a reconnu en ce problème un problème d'optimisation combinatoire [Dibangoye *et al.*, 2008] et a proposé d'utiliser la recherche *branch-and-bound* pour le résoudre. Cependant, la complexité reste de manière générale exponentielle par rapport au nombre d'observations.

Nous proposons d'utiliser les méthodes approchées de résolution des problèmes d'optimisation combinatoire. L'ensemble des méthodes heuristiques de résolution approchée des problèmes d'optimisation combinatoire peut être utilisé pour réduire les temps de planification et même la complexité algorithmique. La réduction de la complexité permet d'augmenter le paramètre *maxTrees* et de trouver des solutions de meilleure qualité. Dans la suite de ce chapitre, nous justifions cette approche en utilisant une méthode méta-heuristique très simple, la recherche locale, mais des méthodes plus évoluées pourraient être expérimentées ultérieurement.

6.2 Équilibre de Nash

Nous proposons d'utiliser une méthode de recherche locale pour trouver un maximum local au problème de *lookahead*. Plus précisément, pour une croyance b donnée, nous recherchons l'arbre joint z comme un équilibre de Nash de manière similaire à DecRSPI et PBPG.

6.2.1 Principe

Le problème de *lookahead* (6.1) pour une croyance b donnée peut être vu comme un jeu coopératif. Chaque agent i est un joueur qui doit choisir un arbre local. Les agents doivent effectuer leur choix de manière coordonnée afin de maximiser l'espérance des récompenses à venir : en effet, celle-ci dépend du choix de l'ensemble des agents.

Plus formellement, le problème de *lookahead* peut être représenté comme un jeu coopératif en forme normale $\langle \mathcal{I}, (\bar{\mathcal{A}}_i)_{i \in \mathcal{I}}, (u_i)_{i \in \mathcal{I}} \rangle$ (voir chapitre 9 en annexe) :

- les joueurs sont les agents $i \in \mathcal{I}$;

– les actions possibles de l'agent i sont les arbres de politiques locales

$$\bar{z}_i \in \bar{\mathcal{A}}_i = \mathcal{A}_i \times (\mathcal{Z}^i_{t+1})^{\Omega_i} \tag{6.2}$$

dont les sous-arbres sont dans $\mathcal{Z}^i_{t+1}$;
– l'utilité $u(q)$ commune des agents est la valeur de l'arbre q de politique jointe pour la croyance b donnée,

$$u(z) = V^z(b) \tag{6.3}$$

Un arbre joint optimal pour la croyance b est un équilibre Pareto optimal de ce jeu bayésien.

Nous proposons de chercher un équilibre de Nash plutôt qu'un équilibre Pareto optimal : ceci simplifie grandement la procédure de recherche. Un équilibre de Nash est recherché (algorithme 6.1, voir annexe 9). en recherchant de manière itérative la meilleure réponse (ligne 13) d'un agent i donné jusqu'à convergence. Cette procédure est répétée un certain nombre M de fois pour différents arbres joints initiaux aléatoires sélectionnés de manière aléatoire et la meilleure solution qui n'est pas dans l'ensemble *blacklist* est retenue (ligne 19). Nous appelons cette méthode MBDP/NE (pour Nash Equilibrium – Équilibre de Nash).

Algorithme 6.1 : *Lookahead* (MBDP/NE)

Données : Point b ;
Ensembles $\forall i, \mathcal{Z}^i_{t+1}$ de sous-arbres ;
Ensemble *blacklist* des arbres joints exclus de la recherche ;
Nombre M de recherches indépendantes ;
Nombre K maximum de passes de la boucle d'amélioration.
Résultat : z arbre joint en équilibre de Nash

1 $(z, v) \leftarrow (\text{null}, -\infty)$
2 **pour** $k = 1$ **à** M **faire**
`/* Générer un arbre joint initial` z' `aléatoire */`
3 **pour chaque** $i \in \mathcal{I}$ **faire**
4 $\varphi(z'_i) \leftarrow$ `aléatoire`$(\mathcal{A}_i)$
5 **pour chaque** $o_i \in \Omega_i$ **faire**
6 $\psi(z'_i, o_i) \leftarrow$ `aléatoire`$(\mathcal{Z}^i_{t+1})$
7 $v' \leftarrow V^{z'}(b)$
`/* Chercher un équilibre de Nash */`
8 *flag* $\leftarrow$ true
9 $k \leftarrow 0$
10 **tant que** *flag et* $k < K$ **faire**
11 *flag* $\leftarrow$ false
12 **pour chaque** $i \in \mathcal{I}$ **faire**
13 $(z''_i, v'') \leftarrow$ `meilleureRéponse`$(i, z'_{-i}, b, \mathcal{Z}^i_{t+1})$
14 **si** $v'' > v'$ **alors**
15 *flag* $\leftarrow$ true
16 $z'_i \leftarrow z''_i$
17 $v' \leftarrow v''$
18 $k \leftarrow k + 1$
19 **si** $z' \neq$ null *et* $z' \notin$ *blacklist et* $v' > v$ **alors** $(z, v) \leftarrow (z', v')$

Ce type d'approche a aussi été proposée indépendamment par Kumar *et al.* [Kumar et Zilberstein, 2010] et Wu *et al.* [Wu *et al.*, 2010b, Wu *et al.*, 2010a].

6.2.2 Meilleure réponse

Le cœur de l'algorithme est donc de trouver l'arbre de politique z_i local optimal étant donné une croyance b et les arbres joints z_{-i} des autres agents :

$$\max_{\substack{z_i \in \mathcal{A}_i \times (\Gamma_{t+1}^i)^{\Omega_i} \\ a=\varphi(\langle z_i, z_{-i}\rangle)}} \left[\mathcal{R}(b,a) + \gamma \sum_{o \in \Omega} \mathcal{O}(o|b,a) V^{\psi(z,o)}(\tau(b,a,o)) \right] \tag{6.4}$$

Comme dans le cas POMDP, il n'est pas nécessaire de parcourir l'ensemble $\mathcal{A}_i \times (\Gamma_{t+1}^i)^{\Omega_i}$ pour trouver le meilleur z_i : pour une action locale a_i donnée, les choix des sous-arbres $\psi(z_i, o_i)$ est indépendant pour chaque observation locale o_i ce qui permet de trouver la solution de manière efficace.

Étant donné une croyance b et l'état interne conjoint z_{-i}, on peut intégrer la connaissance de l'arbre conjoint z_{-i} dans la croyance b pour former une croyance subjective b_i définie par :

$$b_i(s, \bar{z}_{-i}) = \begin{cases} b(s) \text{ si } z_{-i} = \bar{z}_{-i} \\ 0 \text{ sinon} \end{cases} \tag{6.5}$$

Trouver le meilleur arbre local z_i correspond au problème de *lookahead* pour la croyance b_i de l'agent i dans les POMDP subjectifs de l'agent i (équation 3.73, algorithme 3.3) :

$$\max_{\substack{a_i \in \mathcal{A}_i \\ a=\langle a_i, \varphi_{-i}(z_{-i})\rangle}} \Big[\mathcal{R}(b,a) + \gamma \sum_{o_i \in \Omega_i} \max_{z_i' \in \mathcal{Z}_{t+1}^i} \overbrace{\sum_{\substack{o_{-i} \in \Omega_{-i} \\ o=\langle o_i, o_{-i}\rangle \\ z'_{-i}=\psi_{-i}(z_{-i}, o_{-i}) \\ z'=\langle z_i', z'_{-i}\rangle}} \sum_{s' \in \mathcal{S}} \mathcal{J}(s', o|b,a) \alpha^{z'}(s)}^{\texttt{contribution}_i(b,a,o_i,z_i')} \Big] \tag{6.6}$$

Ce calcul a une complexité en $|\mathcal{A}_i||\Omega||\mathcal{Z}_{t+1}^i||\mathcal{S}|^2$ soit $|\mathcal{A}_i||\Omega|maxTrees|\mathcal{S}|^2$ pour une approche à largeur bornée par *maxTrees*.

Algorithme 6.2 : Meilleure réponse d'un agent (MBDP/NE)

Données : Agent i ;
Arbre de politique conjoint z_i ;
Point b ;
Ensembles $\forall i, \mathcal{Z}_{t+1}^i$ des sous-arbres de politiques locaux
Résultat : Meilleure réponse z_i ;
Valeur $v = V^z(b)$ associée

```
(z_i, v) ← (null, −∞)
pour chaque a_i ∈ A faire
    (z̄_i, v̄) ← meilleureRéponsePourUneAction(i, z_{-i}, b, Z^i_{t+1}, a_i)
    si v > v̄ alors (z_i, v) ← (z̄_i, v̄)
```

Algorithme 6.3 : Meilleure réponse d'un agent pour une action donnée (MBDP/NE)

Données : Agent i ;
Arbres de politique conjoint z_i ;
Point b ;
Ensembles $\forall i, \mathcal{Z}^i_{t+1}$ des sous-arbres de politique de l'agent i ; Action locale a_i
Résultat : Meilleur réponse z_i avec $\varphi_i(z_i = a_i)$;
Valeur $v = V^z(b)$ associée

$\varphi_i(z_i) \leftarrow a_i$
$v \leftarrow \mathcal{R}(b, \langle a_i, \varphi_{-i}(z_{-i})\rangle)$
pour chaque $o_i \in \Omega_i$ **faire**
  `/* Chercher le meilleur sous-arbre pour` o_i `*/`
  $c \leftarrow -\infty$
  $\psi_i(z_i, o_i) \leftarrow$ null
  pour chaque $z'_i \in \mathcal{Z}^i_{t+1}$ **faire**
    $c' \leftarrow \texttt{contribution}_i(b, a, o_i, z'_i)$
    si $c' > c$ **alors**
      $c \leftarrow c'$
      $\psi_i(z_i, o_i) \leftarrow z'_i$
  `/* Contribution du sous-arbre pour` $\psi_i(z_i, o_i)$ `*/`
  $v \leftarrow v + \gamma c$

6.2.3 Variantes de l'état de l'art

Nous présentons ici des approches similaires qui utilisent une recherche d'équilibre de Nash pour effectuer un *lookahead* approché. Ces différentes méthodes ont été proposées indépendemment.

La première est très similaire à notre approche mais parcourt exhaustivement l'espace des actions jointes. PBPG (*Point Based Policy Generation*) recherche des politiques stochastiques et cherche un maximum local au problème [Wu *et al.*, 2010a]. DecRSPI [Wu *et al.*, 2010b] se base sur cette approche pour calculer une politique sans utiliser de représentation explicite des lois du problème mais uniquement un modèle génératif capable de générer des transitions et observations.

6.2.3.1 Énumération exhaustive des actions

Une variante de cette approche qui parcourt explicitement l'espace des actions jointes, comme dans PBPG, a été proposée indépendamment [Kumar et Zilberstein, 2010] (algorithme 6.4). Pour chaque action jointe $a \in \mathcal{A}$, elle cherche les meilleures règles de décision locales $\delta_i \in (\mathcal{A}_i)^{\Omega_i}$ par équilibre de Nash. La meilleure solution pour toutes les actions jointes est retenue.

Cette approche ressemble fortement à PBPG mais travaille dans l'espace des politiques déterministes : comparé à notre approche, elle décompose le jeu en $|\mathcal{A}|$ sous jeux pour chaque action jointe $a \in \mathcal{A}$. Ceci réduit l'espace de recherche de chaque jeu mais augmente le nombre de jeux à résoudre.

Notre approche présente l'avantage de ne pas parcourir l'ensemble des actions jointes de taille exponentielle par rapport au nombre d'agents ainsi que de pouvoir changer d'action locale a_i lors du calcul de la meilleure réponse de l'agent i. En contrepartie, certaines actions jointes peuvent

Algorithme 6.4 : *Lookahead* (Kumar2010)

Données : Point b ;
Ensembles de sous-arbres $\forall i, \mathcal{Z}^i_{t+1}$;
Nombre M de recherches indépendantes ;
Ensemble des arbres joints exclus de la recherche *blacklist*.
Résultat : Arbre joint z en équilibre de Nash

$(z, v) \leftarrow (\text{null}, -\infty)$
pour $k = 1$ **à** M **faire**
  pour chaque $a' \in \mathcal{A}$ **faire**
    `/* Générer une règle de décision` δ' `aléatoire */`
    pour chaque $i \in \mathcal{I}$ **faire**
      pour chaque $o_i \in \Omega_i$ **faire**
        $\delta'(o_i) \leftarrow$ `aléatoire`$(\mathcal{Z}^i_{t+1})$
    $v' \leftarrow V^{(a',\delta')}(b)$
    `/* Chercher un équilibre de Nash */`
    flag $\leftarrow$ true
    tant que *flag* **faire**
      flag $\leftarrow$ false
      pour chaque $i \in \mathcal{I}$ **faire**
        $(z''_i, v'') \leftarrow$ `meilleureRéponse`$(i, a', z'_{-i}, b, \mathcal{Z}^i_{t+1})$
        si $v'' > v'$ **alors**
          flag $\leftarrow$ true
          $\delta'_i \leftarrow \delta''_i$
          $v' \leftarrow v''$
    $z' \leftarrow (a', \delta')$
    si $z' \neq$ null *et* $z' \notin$ *blacklist et* $v' > v$ **alors** $(z, v) \leftarrow (z', v')$

ne pas être visitées.

6.2.3.2 PBPG

Dans PBPG [Wu *et al.*, 2010a], les politiques locales cherchées sont à largeur bornée *maxTrees* mais sont stochastiques : à chaque instant t, l'état Z^i_t de l'agent i appartient à un ensemble $\mathcal{Z}^i_t$ de taille maximum *maxTrees*, $Z^i_t \in \mathcal{Z}^i_t$ avec $|\mathcal{Z}^t_i| \leq maxTrees$. Chaque nœud $z_i \in \mathcal{Z}^i_t$ est associé à une action locale $a_i = \varphi_i(z_i)$ et à une distribution de probabilité sur le nœud successeur définie par $\psi_i(a'_i|z_i, o_i)$:

$$A^i_{t+1} = \varphi(Z^i_t) \tag{6.7}$$

$$Z^i_{t+1} \sim \psi_i(Z^i_t, O^i_{t+1}) \tag{6.8}$$

L'opérateur de *lookahead* cherche un état joint z pour chaque action jointe a et retient le meilleur de ces arbres. Pour un a donné, z est donné par les probabilité de transitions d'état interne que l'on notera x_i :

$$x_i(z'_i|o_i) = \psi_i(a'_i|z_i, o_i) \tag{6.9}$$

Algorithme 6.5 : Lookahead (PBPG)

Données : Point de croyance b ;
Ensembles des états $\mathcal{Z}^i_{t+1}$ à l'instant $t+1$;
État locaux $\forall i, z_i$ à l'instant t pour b
Résultat : États locaux $\forall i, z_i$.

```
pour chaque a ∈ A faire
    /* Calculer x comme un équilibre de Nash pour a                    */
    pour chaque i ∈ I faire  x_i ← aléatoire((ΔZ^i_{t+1})^{Ω_i})
    répéter
        pour chaque i ∈ I faire
            Calculer x_i à partir de x_{-i} avec le programme linéaire (6.10)
    jusqu'à convergence
    /* Construire l'arbre joint z_a                                    */
    z_a ← (a, x)
    v_a ← V^z(b)
/* Conserver la meilleure solution                                     */
a ← arg max_{a∈A} v_a
z ← z_a
```

L'état joint z pour une jointe action a donnée est cherché comme un équilibre de Nash. La meilleure réponse x_i d'un agent i étant donné x_{-i} est donnée par le programme linéaire :

$$\begin{aligned} &\text{maximiser } \epsilon && (6.10)\\ &\text{s.t. } V^\pi_t(b) + \epsilon \leq \sum_{s' \in \mathcal{S}, o \in \Omega} \mathcal{J}(s', o|b, a) \sum_z x_i(z'_i|o_i) x_{-i}(z'_{-i}|o_{-i}) V^{z'}_{t+1}(s') \\ &\quad x_i \in (\Delta \mathcal{Z}^i_{t+1})^{\Omega_i} && \forall i \in \mathcal{I} \end{aligned}$$

6.2.3.3 DecRSPI

DecRSPI (*Decentralized Rollout Sampling Policy Iteration*) [Wu *et al.*, 2010b] fonctionne de manière similaire que PBPG mais ne suppose pas de représentation explicite des lois du système mais uniquement la capacité à simuler le processus stochastique. Ceci est particulièrement utile quand le système est trop grand pour utiliser une représentation explicite des probabilités du système.

Lookahead L'opération de *lookahead* cherche un état interne joint $z = (z_1, \ldots, z_n)$ comme un équilibre de Nash (voir l'annexe sur théorie des jeux, section 9.3). Contrairement à l'approche PBPG, cette recherche n'est pas effectuée de manière indépendante pour chaque observation jointe $a \in \mathcal{A}$.

Points de croyance Comme dans MBPD les distributions d'états b sont des estimations par échantillonnage de la probabilité *a priori* de S_t pour la politique heuristique $\tilde{\pi}$ (algorithme 6.7) :

$$b(s) \approx \Pr(S_t = s | B_0 = b_0, \tilde{\pi}) \tag{6.11}$$

et peuvent être représentées comme une suite finie $(s_k)_{1 \leq k \leq N}$ d'états :

$$b = \frac{1}{N} \sum_{n=1}^{N} \delta_{s_k} \tag{6.12}$$

Algorithme 6.6 : Opérateur de *lookahead* (DecRSPI)

Données : Point de croyance b ;
Ensembles des états locaux $\forall i, \mathcal{Z}^i_{t+1}$ à l'instant $t+1$
Résultat : États locaux z à l'instant t
`/* Choisir` z `aléatoiremment */`
pour chaque $i \in \mathcal{I}$ **faire** $z_i \leftarrow$ `aléatoire`$(\mathcal{A}_i \times (\Delta\mathcal{Z}^i_{t+1})^{\Omega_i})$
`/* Optimisation locale */`
répéter
  pour chaque $i \in \mathcal{I}$ **faire**
    $z_i \leftarrow$ `meilleurRéponse`$(b, i, z, \mathcal{Z}^i_{t+1})$
jusqu'à *convergence*

Algorithme 6.7 : Échantillonnage de distributions (DecRSPI)

Données : Croyance initiale b_0 ;
Paramètres *maxTrees* ;
Politique heuristique $\tilde{\pi} = (\nu, \psi, \varphi)$
Résultat : Ensembles $\forall t, B_t$ d'échantillons
pour $t = 1$ **à** T **faire** $B_t \leftarrow \emptyset$
pour $k = 1$ **à** *maxTrees* **faire**
  pour $t = 1$ **à** T **faire** $b_t \leftarrow 0$
  `/* Générer un point de croyance pour chaque instant` t `*/`
  pour chaque $n = 0$ **à** N **faire**
    `/* Initialiser le processus */`
    Échantillonner s selon b_0
    $z \leftarrow \nu(b_0)$
    `/* Simuler le processus */`
    pour $t = 1$ **à** T **faire**
      `/* Une étape du processus */`
      $a \leftarrow \varphi(z)$
      Échantillonner $('s, o)$ par simulation depuis (s, a)
      Tirer le nouveau z selon $\psi(\cdot|z, o)$
      $s \leftarrow s'$
      `/* Ajouter l'état` s `à l'estimation` b_t `*/`
      $b_t \leftarrow b_t + \frac{1}{N}\delta_s$
  `/* Ajouter les points de croyance générés */`
  pour $t = 1$ **à** T **faire** $B_t \leftarrow B_t \cup \{b_t\}$

Meilleure réponse L'algorithme utilisé pour trouver la meilleure réponse de l'agent i étant donné celle des autres agents commence par calculer pour chaque action locale $a_i \in \mathcal{A}_i$ la meilleure règle de sélection de sous-nœuds définie par $\forall z'_i, o_i, x_i(z'_i|o_i) = \pi(z'_i|z_i, o_i)$ (algorithme

6.8) Celle-ci est donnée par le programme linéaire :

$$\text{maximise} \sum_{o_i \in \Omega_i} \sum_{z'_i \in \mathcal{Z}^i_{t+1}} \Phi_i(o_i, z'_i) x_i(z'_i|o_i) \tag{6.13}$$

$$\text{sujet à } x_i \in (\Delta \mathcal{Z}^i_{t+1})^{\Omega_i} \tag{6.14}$$

où $\Phi(o_i, z'_i)$ représente la contribution de l'état interne local z'_i de l'agent i à la valeur $V^z(b)$ si z'_i est utilisé après l'observation locale o_i

$$\Phi_i(o_i, z'_i) = \sum_{s' \in \mathcal{S}} \sum_{\substack{o_{-i} \in \Omega_{-i} \\ o = \langle o_i, o_{-i} \rangle}} \sum_{z'_{-i} \in \mathcal{Z}^{-i}_{t+1}} \mathcal{J}(s', o|b, a) \psi_{-i}(z'_{-i}|z_{-i}, o_{-i}) V^{\langle z'_i, z'_{-i} \rangle}(s') \tag{6.15}$$

$$V^z(s) = \sum_{o_i \in \Omega} \sum_{z'_i \in \mathcal{Z}^i_{t+1}} \psi_i(z'_i|z_i, o_i) \Phi_i(o_i, z'_i) \tag{6.16}$$

Algorithme 6.8 : Meilleure réponse d'un agent (DecRSPI)

Données : Croyance centralisée b ;
Agent i ;
État interne joint z ;
Ensembles $\mathcal{Z}^i_{t+1}$ des états internes locaux l'agent i à l'instant suivant.
Résultat : z_i meilleure réponse estimée de l'agent i

```
1 Z̄_t^i ← ∅
2 pour chaque a_i ∈ A_i faire
3     Φ̂_i ← estimation(b, a_i, z_{-i})
4     x_i ← résoudreLP (6.13)
5     Z̄_t^i ← Z̄_t^i ∪ {(a_i, x_i)}
6 z_i ← arg max_{z_i ∈ Z̄_t^i} Rollout(b, ⟨z_i, z_{-i}⟩)
```

Comme les lois du problème ne sont pas connues explicitement, les paramètres $\Phi(o_i, z'_i)$ sont estimés pas échantillonnage (algorithme 6.9). De même, la valeur $V^z(b)$ est estimée par échantillonnage de Monte-Carlo [Doucet *et al.*, 2001] (algorithme Rollout).

Outre le fait que DecRSPI utilise un modèle génératif du problème, une différence avec PBPG est le résultat de complexité. La majorité des algorithmes existant ont une complexité au moins exponentielle par rapport au nombre d'agents : en particulier la complexité des approches de type MBDP est exponentielle par rapport au nombre d'agents par exemple parce qu'elle calculent un vecteur α pour chaque arbre joint et parcourent généralement l'ensemble des actions jointes. L'approche DecRSPI est très intéressante parce que sa complexité est quadratique [7] par rapport au nombre $|\mathcal{I}|$ d'agents. Ceci résulte de plusieurs propriétés de l'algorithme :

- tous les arbres joints $z \in \mathcal{Z}^i_t$ induits par les ensembles $\mathcal{Z}^i_t$ d'arbres locaux ne sont pas considérés et pas évalués ;
- l'ensemble des actions jointes $a \in \mathcal{A}$ n'est pas parcouru contrairement à PBPG ;
- les vecteurs α ne sont pas calculés et l'ensemble des observations jointes $o \in \Omega$ n'est donc pas parcouru.

7. L'article original affirme que la complexité est linéaire par rapport au nombre d'agents mais elle est en réalité quadratique.

Algorithme 6.9 : Estimation de Φ (DecRSPI)

Données : Point de croyance b ;
Agent i ;
Action locale a_i ;
État conjoint z_{-i}
Résultat : $\hat{\Phi}_i$ estimation de Φ_i

$a_{-i} \leftarrow \varphi_{-i}(z'_{-i})$
Tirer N échantillons de (s', o) sachant (s, a)
En déduire une estimation $\mathcal{P}$ de $\Pr(s', o_{-i}|o_i), \forall o_i$
pour chaque $(o_i, z'_i) \in \Omega_i \times \mathcal{Z}^i_{t+1}$ **faire**
    $\hat{\Phi}_i(o_i, z'_i) \leftarrow 0$
    pour chaque $n = 1$ **à** N **faire**
        Tirer (s', o_{-i}) selon $\mathcal{P}(s', o_{-i}|o_i)$
        Tirer z_{-i} selon $\psi_{-i}(z_{-i}|z_i, o_i)$
        $\hat{\Phi}_i(o_i, z'_i) \leftarrow \hat{\Phi}_i(o_i, z'_i) + \text{Rollout}(s', z')$
    $\hat{\Phi}_i(o_i, z'_i) \leftarrow \frac{1}{N}\hat{\Phi}_i(o_i, z'_i)$

Algorithme 6.10 : Évaluation d'un état interne joint (Rollout)

Données : État s à évaluer ;
Instant t ;
État interne joint z
Résultat : Valeur estimée v

$v \leftarrow 0$
pour $k = 1$ **à** N **faire**
    Simuler le processus depuis (s, z) : soit v' les récompenses cumulées
    $v \leftarrow v + v'$
$v \leftarrow \frac{1}{N}v$

Ces trois ensembles $\mathcal{Z}_t$, $\mathcal{A}$ et Ω sont de taille exponentielle par rapport au nombre d'agents.

Cependant, la complexité est quadratique par rapport à l'horizon de planification (plutôt que linéaire dans le cas de MBDP) du fait de l'échantillonnage utilisé pour évaluer les état internes joints z.

6.2.3.4 Synthèse

Ces approches sont très proches de celle que nous proposons.

Par rapport à DecRSPI et PBPG, l'utilisation de politiques déterministes permet de remplacer l'utilisation d'un programme linéaire pour le calcul de la meilleure réponse d'un agent donné par un opération très similaire au *lookahead* POMDP et qui peut être effectué de manière très efficace.

Cependant, DecRSPI est intéressante car sa complexité par rapport au nombre d'agents est quadratique. Les approximations qui permettent ceci peuvent être adaptées pour la planification d'arbres de politiques afin de combiner les avantages des deux approches. Si tous les arbres joints de $\mathcal{Z}_t = \prod_{i \in \mathcal{I}} \mathcal{Z}^i_t$ obtenus par les arbres locaux $\mathcal{Z}^i_t$ sélectionnés ne sont pas considérés, l'algorithme

peut avoir une complexité qui n'est plus exponentielle par rapport au nombre d'agents, comme la majorité des approches, mais quadratique [Wu *et al.*, 2010b]. Des propositions d'adaptation de ces approximations pour notre approche sont présentées plus loin (section 6.2.5).

6.2.4 Résultats

Les tableaux 6.1 et 6.2 comparent les résultats expérimentaux de cette approche avec les algorithmes de l'état de l'art. PBIP/BeFS désigne une variante de PBIP avec recherche en meilleur d'abord que nous avons proposé dans la section 4.4.4. MBDP/NE est l'algorithme que nous proposons sans les modifications de la section 6.2.5.

La colonne VEM (Valeur Espérées Moyenne) représente la moyenne sur toutes les exécutions de la valeur espérée de la politique obtenue. La colonne σ représente l'écart type de la valeur espérée.

6.2.4.1 Problème du tigre décentralisé

La table 6.1 montre les résultats du problème du tigre décentralisé [Nair *et al.*, 2003] en horizon 100 : il s'agit d'une variante du problème POMDP classique du tigre [Kaelbling *et al.*, 1995]. Les résultats sont des moyennes sur 50 exécutions.

Il s'agit d'un problème de petite taille et le gain de temps obtenu avec un *lookahead* approché est assez faible par rapport à PBIP/BeFs. Le *lookahead* par recherche d'équilibre de Nash donne des politiques de qualité comparable au *lookahead* exact pour des temps de calculs inférieurs aux approches de recherche *branch-and-bound* de type PBIP. Il est intéressant de remarquer que dans certains cas, le *lookahead* par recherche d'équilibres de Nash produit des solutions de meilleure qualité que le *lookahead* exact.

Algorithme	*maxTrees*	M	VEM	σ	Temps (s)
PBIP	20	-	139.66	4.99	16.59
PBIP/BeFs	20	-	138.99	2.17	8.90
MBDP/NE	20	10	114.36	8.94	1.28
	20	30	147.36	4.78	3.66
	20	50	152.96	3.67	7.48
Kumar2010	20	1	154.38	9.43	0.50

TABLE 6.1 – Problème du tigre décentralisé

Pour ce problème, la méthode MBDP/NE2 trouve de très bon résultats avc $M = 1$. Pour des problèmes de petites taille, un parcours plus exhaustif de l'espace de recherche est préférable. Cependant, quand le nombre d'actions jointes est important, en particulier quand le nombre d'agents est élevé, on peut s'attendre à ce que l'approche que nous proposons soit plus efficace car elle évite de parcourir l'ensemble des actions jointes.

6.2.4.2 *Cooperative Box Pushing*

Le problème *Cooperative Box Pushing* [Seuken et Zilberstein, 2007a] est un autre problème avec un grand nombre d'observations. Il a été proposé comme *benchmark* pour des algorithmes conçus pour traiter un grand nombre d'observations (IMBDP) : $|\mathcal{I}| = 2$, $|\mathcal{A}_i| = 4$, $|\Omega_i| = 5$. Les résultats expérimentaux présentés dans la table 6.2 sont des moyennes sur 10 exécutions. Pour

maxTrees = 7, l'espace recherche d'un *lookahead* est de taille :

$$\left|\prod_{i\in\mathcal{I}} \mathcal{A}_i \times (\mathcal{Z}_{t+1}^i)^{\Omega_i}\right| = (4 \times 20^7)^2 \approx 4.52 \times 10^9$$

Les temps de calcul sont comparables avec PBIP/BeFs. Pour *maxTrees* = 20, l'espace recherche d'un *lookahead* est beaucoup plus grand :

$$\left|\prod_{i\in\mathcal{I}} \mathcal{A}_i \times (\mathcal{Z}_{t+1}^i)^{\Omega_i}\right| = (4 \times 20^5)^2 \approx 1.64 \times 10^{14}$$

PBIP/BeFS ne termine pas dans le temps imparti alors que l'approche par recherche d'équilibre de Nash trouve une solution assez rapidement.

Algorithme	*maxTrees*	*M*	AEV	σ	Temps (s)
PBIP/BeFS	7	-	423.4	14.0	101
MBDP/NE	7	10	405.3	12.0	56
MBDP/NE	7	30	423.0	9.6	83
Kumar2010	7	1	420.5	5.5	62
PBIP/BeFS	20	-	?	?	>24h
MBDP/NE	20	10	431.8	8.1	498
MBDP/NE	20	30	445.2	7.2	822
Kumar2010	20	1	441.7	7.4	565

TABLE 6.2 – *Cooperative Box Pushing*, $T = 20$

6.2.4.3 Résumé

L'utilisation de *lookahead* approché permet de trouver des solutions plus rapidement qu'avec le *lookahead* exact. Le gain semble particulièrement significatif quand l'espace de recherche est important : ce type d'approche devrait permettre de traiter des problèmes avec un plus grand nombre d'observations ou de trouver des solutions avec un nombre d'arbres de politique *maxTrees* plus important. L'utilisation de *lookahead* approché n'affecte pas la qualité de la politique trouvée dans les problèmes que nous avons testés.

6.2.5 Perspectives

MBDP/NE est proche de PBPG et DecRSPI mais travaille dans l'espace des politiques déterministes ce qui simplifie grandement les calculs. Le calcul de la meilleure réponse ne nécessite pas de résoudre un programme linéaire mais est une opération beaucoup plus simple similaire au *lookahead* POMDP. Cependant, DecRSPI utilise un certain nombre de techniques qui rendent la complexité quadratique par rapport au nombre d'agents. Nous montrons ici quels sont les points qui rendent la complexité exponentielle par rapport au nombre d'agents et comment on peut adapter les méthodes utilisées par DecRSPI pour passer à l'échelle par rapport au nombre d'agents.

Trois points rendent la complexité de MBDP/NE exponentielle par rapport au nombre d'agents :

- la possibilité d'appliquer l'opération de meilleure réponse un nombre de fois exponentiel par rapport au nombre d'agents ;

- calculer l'ensemble des vecteurs α^z pour tous les arbres joints z de $\mathcal{Z}_t = \prod_{i \in \mathcal{I}} \mathcal{Z}_t^i$;
- parcourir l'ensemble des observations jointes $o \in \Omega$ pour calculer un vecteur α (équation 4.37) ou des observations conjointes $o_{-i} \in \Omega_{-i}$ pour choisir un sous-arbre z_i' pour une observation locale o_i (equation 6.6).

Limitation du nombre d'itération En ce qui concerne le premier point, dans la pratique le nombre d'itérations avant d'atteindre un équilibre est très faible mais certains cas pathologiques pourraient parcourir une grande partie de l'espace des solutions avant d'atteindre un équilibre de Nash. Si cela pose problème, on peut fixer un nombre K de passes maximal (ligne 2 de l'algorithme 6.1) de la boucle principale de l'algorithme de recherche d'un équilibre de Nash ou le nombre d'utilisation de l'algorithme de meilleure réponse pour limiter le nombre d'arbres joints considérés. Il est alors possible que l'opération de *lookahead* ne produise pas un équilibre de Nash.

Remarque: Ce point n'est pas abordé dans DecRSPI.

Évaluation paresseuse des vecteurs α En ce qui concerne le second point, dans une approche par recherche locale, généralement, seul un petit nombre des vecteurs α sont réellement utilisés car seul un petit nombre des arbres joint z' sont réellement considérés pour construire les arbres joints z. Ceci est particulièrement vrai si on borne le nombre de passes K de la boucle d'optimisation locale. N'évaluer les vecteurs α que quand ils sont effectivement utilisés permet d'éviter de calculer l'ensemble exponentiel de vecteurs α.

Remarque: Dans DecRSPI, les politiques à l'instant t sont évaluées en simulant le processus à partir de l'instant t. Un défaut de cette approche est que la complexité est quadratique par rapport à l'horizon alors qu'elle est linéaire dans la plupart des approches. Mémoriser les vecteurs α permet d'éviter cette complexité quadratique.

Échantillonnage des observations En ce qui concerne le dernier point, une adaptation de l'approche de DecRSPI consiste à calculer une approximation du calcul des vecteurs α (équation (4.37)) par échantillonnage des observations :

$$\hat{\alpha}^z(s) = \mathcal{R}_{\varphi(z)}(s) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s'|s, \varphi(z)) \frac{1}{N} \sum_{k=1}^{N} \alpha^{\psi(z, o_{s,s',k})}(s') \tag{6.17}$$

où les $o_{s,s',k}$ sont des observations jointes indépendantes échantillonnées selon $\mathcal{O}(o|s,a,s')$. Conserver en mémoire les évaluations α^z des arbres joints permet d'obtenir une complexité linéaire par rapport à l'horizon plutôt que quadratique comme dans DecRSPI. On peut faire de même pour calculer une valeur approchée de l'équation (6.6), en échantillonnant (s', o) selon les probabilités $\mathcal{J}(s', o|b, a)$.

Si ces trois modifications sont utilisées, l'algorithme obtenu n'a plus une complexité exponentielle mais quadratique par rapport au nombre d'agents.

6.3 Conclusion

Les techniques méta-heuristiques des problèmes d'optimisation peuvent être utilisées pour accélérer grandement les temps de planification des algorithmes de type MBDP sans réduction de la qualité de la planification. Dans certains cas, la politique trouvée est même de meilleure qualité qu'en utilisant un *lookahead* exact. De plus, des approximations supplémentaires permettent de réduire la complexité des algorithmes de type MBDP. Nous avons concentré notre approche sur la recherche d'un maximum local, ici un équilibre de Nash, par recherche locale mais des méta-heuristiques plus poussées pourraient être testées : entropie croisée [De Boer *et al.*, 2002], etc.

Dans le prochain chapitre, nous étudions une direction opposée qui consiste à utiliser un critère de sélection plus complexe pour trouver des solutions de meilleure qualité. La méthode de résolution de ce critère utilise cependant un *lookahead* et les techniques de *lookahead* approché comme celle que nous venons de présenter peuvent être utilisées pour accélérer la planification dans ce type d'approches.

Chapitre 7

PSMBDP

Sommaire

Les approches de type MBDP utilisent *maxTrees lookaheads* en *maxTrees* points de croyance b pour déterminer les ensembles $\mathcal{Z}_t^i$ d'au plus *maxTrees* arbres de politiques locaux à chaque instant t. Contrairement au cas mono-agent [8], le nombre *maxTrees* d'arbres à retenir a un impact important sur le temps de calcul nécessaire et dans la pratique ce paramètre est fixé à une valeur assez faible. Un problème avec ce type d'approche et que chaque arbre z de politique joint n'est sélectionné qu'en utilisant un seul point de croyance b. Les arbres sont sélectionnés pour être bons en ces points de croyance. Dans le cas mono-agent, un grand nombre de points de croyance b peuvent être utilisés ce qui permet d'obtenir de bonnes politiques pour un grand nombre de situations. Cependant, quand le nombre de points de croyance est fortement limité, comme dans le cas multi-agent, ce n'est plus nécessairement le cas.

Nous revenons d'abord sur deux approches de planification de l'état de l'art : MBDP et ses dérivés (pour la planification Dec-POMDP) d'une part et PSDP (pour la planification MDP) d'autre part. Afin de les comparer, nous les présentons sous l'angle de la planification POMDP. La première approche constitue l'état de l'art de la planification décentralisée et c'est cette approche que nous cherchons à améliorer : chaque opérateur de Bellman est effectué par *maxTrees* choix *indépendants* d'arbres joints (opération de *lookahead*) en *maxTrees* points de l'espace des croyances. La deuxième approche effectue l'opérateur de Bellman en effectuant un choix *reposant sur un critère global* sur l'ensemble des croyances plutôt qu'une succession de plusieurs choix locaux indépendants.

Nous proposons une approche par programmation dynamique PSMBDP (*Policy Search Memory Bounded Dynamic Programming*), [Corona et Charpillet, 2009, Corona et Charpillet, 2010b] que l'on pourrait qualifier d'hybride entre ces deux types d'approches. Comme les approches basées sur MBDP, elle borne à un paramètre *maxTrees* le nombre d'arbres de politique par agent à chaque étape de la programmation dynamique afin de limiter la complexité. À chaque instant, le problème est de choisir *maxTrees* arbres locaux par agent. Nous introduisons donc une information heuristique sur le problème sous la forme d'une distribution de probabilités sur les croyances centralisées, c'est-à-dire les croyances du POMDP sous-jacent : cette distribution est utilisée pour introduire un critère heuristique de qualité d'une sélection des *maxTrees* arbres locaux par agent. Ce critère heuristique permet de formuler le problème de sélection de *l'ensemble* des *maxTrees* arbres de politiques locaux par agent $i \in \mathcal{I}$ comme un problème d'optimisation combinatoire. Notre approche utilise donc un *critère global* pour la sélection de *l'ensemble des arbres* alors que les approches de planification par programmation dynamique multi-agent approchée de l'état de l'art utilisent une série de *lookaheads* indépendants. Ce mode de sélection est intéressant car il permet de tirer au mieux profit du nombre limité d'arbres que l'on peut sélectionner. Ce problème est cependant d'une taille trop importante pour être résolu directement et nous proposons une heuristique simple pour trouver une bonne solution à ce problème. Nous présentons enfin les résultats expérimentaux de cette approche sur un certain nombre de *benchmarks* de la communauté.

Afin de faciliter la compréhension, nous présentons tout d'abord l'approche dans le cas POMDP avant de généraliser au cas Dec-POMDP qui est réellement intéressant.

8. Dans le cas mono-agent, la complexité est quadratique par rapport au nombre de points de croyance b, d'arbres z, ou de vecteurs α utilisés.

7.1 Motivation

Nous revenons ici sur certains algorithmes présentés précédemment, MBDP pour les Dec-POMDP et PSDP pour les MDP, que nous analysons dans le cadre POMDP. En effet, un POMDP étant un MDP sur les croyances et un POMDP étant un Dec-POMDP à un seul agent, on peut appliquer les deux méthodes dans le cadre POMDP. Nous comparons ensuite leurs comportements afin de motiver notre approche.

7.1.1 MBDP

Appliqués à un seul agent, MBDP et ses dérivés peuvent être vus comme des algorithmes d'approximation de la fonction de valeur à base de points. Le paramètre *maxTrees* est alors le nombre de vecteurs α à conserver dans chaque ensemble Γ_t. Pour chacune des *maxTrees* croyances b à l'instant t, le meilleur vecteur α_b est ajouté à l'ensemble Γ_t. Dans le cas mono-agent, ce vecteur peut être construit directement sans faire l'énumération exhaustive comme expliqué dans la section 3.4.3. La fonction de valeur V_{t+1} induit une fonction de décision π_t à l'instant t :

$$A_{t+1} = \pi_t(B_t) = \arg\max_{a \in \mathcal{A}} \left[\mathcal{R}(B_t, a) + \gamma \sum_{o \in \Omega} \Pr(o|B_t, a) V_{t+1}(\tau(B_t, a, o)) \right] \tag{7.1}$$

Dans le cadre POMDP, la recherche du meilleur vecteur α pour une croyance b donnée a une complexité polynomiale (linéaire par rapport au nombre d'observations et au nombre de vecteurs α de l'étape précédente) car chaque sous-arbre peut être choisi indépendamment : la complexité de l'opérateur DP est quadratique par rapport au nombre maximum d'arbres de politique *maxTrees* et ce paramètre peut donc prendre une valeur importante.

Dans le cadre Dec-POMDP, la recherche du meilleur arbre joint z pour un b donné nécessite une recherche dans l'ensemble $\prod_{i \in \mathcal{I}} \mathcal{A}_i \times (\mathcal{Z}^i_{t+1})^{\Omega_i}$ de taille $|\mathcal{A}| maxTrees^{\sum_{i \in \mathcal{I}} |\Omega_i|}$. Ceci limite la valeur du paramètre *maxTrees* qui doit être relativement faible. Un problème est que, comme ce paramètre *maxTrees* est aussi le nombre de points de croyance b utilisés pour la sélection des arbres, le choix des arbres est basé sur un très petit nombre d'échantillons b par rapport à la taille de l'ensemble $\Delta\mathcal{S}$ des croyances centralisées.

Remarque: L'ensemble des croyances centralisées est un ensemble continu de dimensions $|\mathcal{S}| - 1$. Si on fixe la croyance initiale $B_0 = b_0$, le nombre de croyances b réellement accessibles à un instant t est fini [9] :

- au plus $(|\mathcal{A}||\Omega|)^t$ pour une politique stochastique ;
- au plus $|\Omega|^t$ pour une politique déterministe donnée.

Les arbres de politiques sont sélectionnés de manière à être bons dans le voisinage de ces points de croyance : dans le cas POMDP, le fait que le nombre de points de croyance est important permet d'obtenir une fonction de bonne qualité. Dans le cas Dec-POMDP, il semble préférable d'utiliser une technique plus complexe de sélection pour choisir au mieux les *maxTrees* arbres qui sont en nombre plus limités.

9. Nous considérons un problème avec des ensembles d'observations et d'actions finis.

Notre approche cherche à résoudre ces deux problèmes. Nous formulons chaque étape de la planification comme le problème de choix des *maxTrees* arbres locaux par agent étant donné un nombre N de points de croyances. Ainsi chaque arbre local sélectionné n'est plus choisi un utilisant un seul point de croyance mais un grand nombre N de points. De plus, le nombre de points N est décorrélé du paramètre *maxTrees* et peut être arbitrairement plus grand. Ces points de croyances N sont vus comme représentant une distribution de probabilité.

Remarque: Le cas limite quand N tend vers l'infini permet de considérer n'importe quelle distribution, éventuellement continue, sur les croyances centralisées.

7.1.2 PSDP

PSDP (*Policy Search Dynamic Programming* – Programmation dynamique par recherche de politique) est un algorithme de recherche d'une politique markovienne d'un MDP dans un sous-espace donné (voir section 2.3.2) [Bagnell *et al.*, 2003]. Comme nous l'avons vu, un POMDP est MDP dont les états sont les croyances de l'agent. On peut donc envisager d'appliquer cette méthode aux POMDP [10].

Une politique globale π est cherchée dans un sous-ensemble $\hat{\Pi} = \hat{\Xi}^T$ où les règles de décision $\pi_t \in \hat{\Xi}$ sont des fonctions des croyances vers les actions. Les distributions heuristiques μ_t sont des distributions sur les croyances et les π_t sont cherchés par programmation dynamique *bottom-up* exacte,

$$\pi_t \in \arg\max_{\xi \in \Xi} \mathop{\mathrm{E}}_{b \sim \mu_t} V^{\xi, \pi_{t+1}, \ldots, \pi_{T-1}}(b) \tag{7.2}$$

ou de manière ϵ-approchée,

$$\mathop{\mathrm{E}}_{b \sim \mu_t} [V^{\pi_t, \ldots, \pi_{T-1}}(b)] \geq \max_{\xi \in \Xi} \mathop{\mathrm{E}}_{b \sim \mu_t} \left[V^{\xi, \ldots, \pi_{T-1}}(b) \right] - \epsilon \tag{7.3}$$

Remarque: Le théorème 2.3.1 indique une borne de la différence entre la valeur de la politique π obtenue et toute autre politique $\pi' \in \hat{\Pi}$,

$$\mathop{\mathrm{E}}_{b_0 \sim \mu_0} V^{\pi}(b_0) \geq \mathop{\mathrm{E}}_{b_0 \sim \mu_0} V^{\pi'}(b_0) - T(R_{\max} - R_{\min}) d(\mu, \mu') - T\epsilon \tag{7.4}$$

avec

$$\mu'_t(b) = \Pr(B_t = b | B_0 = b_0, \pi') \tag{7.5}$$

$$d(\mu, \mu') = \|\mu - \mu'\|_1 = \sum_{t=0}^{T-1} \mathop{\mathrm{E}}_{b \sim \mu} |(\mu_t - \mu'_t)(b)| \tag{7.6}$$

Cette garantie théorique de qualité n'est pas très intéressante pour les POMDP. Si le nombre d'observations est fini et que μ_0 est une distribution de support fini, les μ_t sont des distributions finies mais leur support est de manière générale de taille exponentielle avec t (en $|\Omega|^t$) ce qui empêche leur évaluation exacte et la recherche d'une solution en garantissant une borne d'erreur

10. Nous présentons cette approche dans le but de la comparer à l'approche précédente. L'application pratique de l'algorithme PSDP au cadre POMDP pose un certain nombre de problèmes que nous n'essayons pas, ici de résoudre.

ϵ donnée. De plus, les supports des distributions μ_t et μ'_t ont dans le cas général une intersection faible ou nulle. Dans ce cas, les distances $d(\mu_t, \mu'_t)$ sont proches de 1 ce qui limite la portée de la borne (2.60).

7.1.3 Analyse

Ces deux approches, PSDP et MBDP, sont similaires. Une heuristique *top-down* est utilisée pour choisir les fonctions de décision π_t. La fonction de décision est dans les deux cas cherchée dans un sous-ensemble restreint des fonctions de décision possibles :

- dans PSDP, un ensemble restreint $\hat{\Xi}$ est fixé explicitement ;
- dans MBDP, le paramètre *maxTrees* contraint l'espace de recherche à un sous-ensemble des fonctions de valeurs V_{t+1}, et donc des règles de décision π_t, possibles.

Celles-ci sont choisies par programmation dynamique *bottom-up*. Une différence entre les deux approches réside dans la manière dont l'information heuristique est utilisée pour le choix de la politique.

Dans MBDP et ses dérivés, chaque arbre est choisi de manière à être optimal pour une croyance donnée. Le choix global des *maxTrees* arbres ne repose pas sur un critère global d'optimisation sur l'espace $\Delta\mathcal{S}$ de la fonction de valeur mais résulte de *maxTrees* optimisationss locales de celle-ci en *maxTrees* points de croyance $b \in \Delta\mathcal{S}$: la fonction de valeur est choisie de manière à être optimale pour les points de croyance b considérés.

Dans PSDP, la règle de décision π_t est choisie de manière à être bonne *en moyenne* pour toutes les croyances. La fonction de valeur obtenue est donc une meilleure approximation, en moyenne de la fonction de valeur que l'on obtiendrait en utilisant une mise à jour exhaustive.

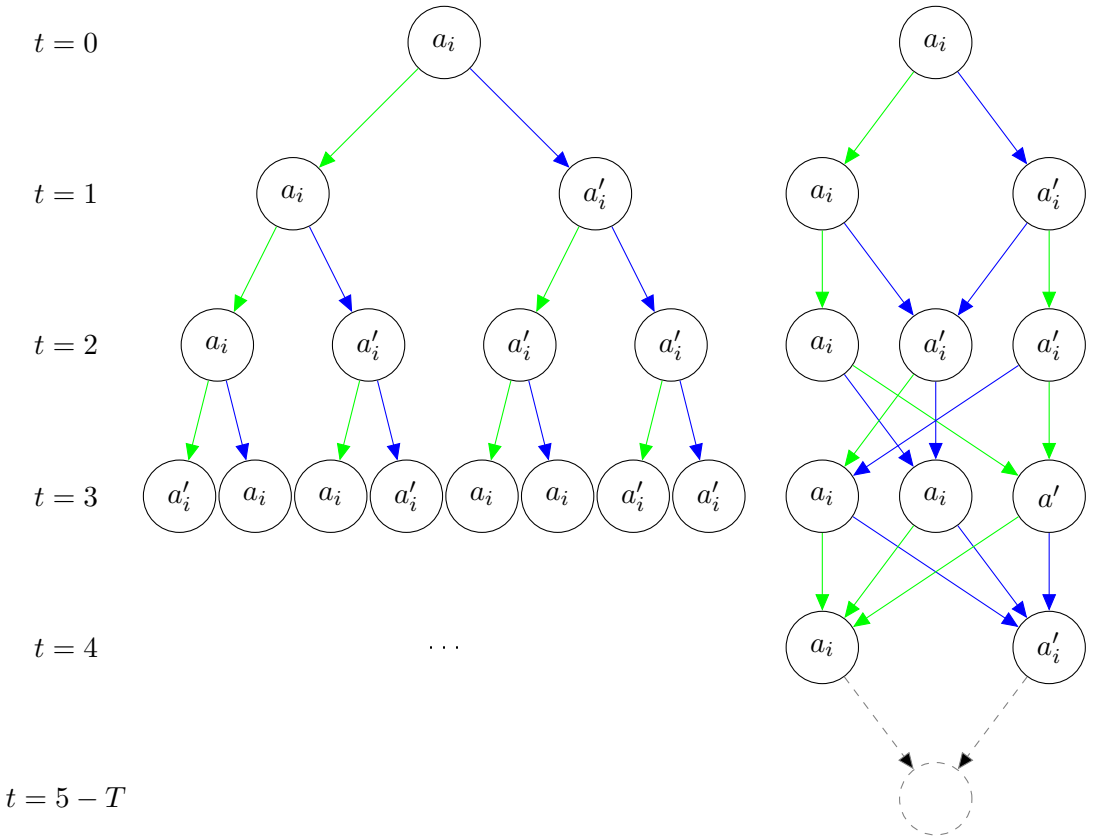
Dans le cas mono-agent, le nombre d'arbres *maxTrees* peut être important et on peut se contenter de faire des optimisation locales par *lookahead* de la fonction de valeur optimale : comme le nombre de points de croyance est important l'approximation peut être de bonne qualité. L'approche que nous proposons est basée sur l'idée que, pour une planification où le nombre d'arbres *maxTrees* est faible comme la planification multi-agent, choisir des arbres localement optimaux ne conduit pas à une bonne fonction de valeur globale car le nombre d'échantillons (de points de croyance) est faible : il est alors préférable d'optimiser directement la fonction de valeur en utilisant un critère global sur l'ensemble de l'espace des croyances centralisées. Cette idée est illustrée un peu plus loin après avoir présenté le principe général (exemples 7.3.1 et 7.3.2).

7.2 Principe général

7.2.1 Espace de recherche

Nous proposons d'utiliser un critère d'optimisation global pour la planification par programmation dynamique avec politiques de taille bornée : la politique est cherchée dans l'espace des politiques de largeur bornée par *maxTrees* (figure 7.1).

Trouver une des meilleures politiques dans cet ensemble pour une croyance initiale $B_0 = b_0$ donnée peut être formulé comme un problème d'optimisation combinatoire dans un sous-espace



Politique non-contrainte Politique à largeur bornée ($maxTrees = 3$)

$H = 5$, $\mathcal{A}_i = \{a_i, a'_i\}$, $\Omega_i = \{o_i, o'_i\}$

FIGURE 7.1 – Politique locale générale et politique à largeur bornée

de l'espace des politiques :

$$\begin{aligned} &\text{maximise } V^{z_0}(b_0) && && (7.7)\\ &\quad\text{pour } \varphi_i(z_i) \in \mathcal{A}_i && \forall i \in \mathcal{I}, \forall t \in \{0,\dots,T-1\}, \forall z_i \in \mathcal{Z}_t^i\\ &\qquad \psi_i(z_i, o_i) \in \mathcal{Z}_{t+1}^i && \forall i \in \mathcal{I}, \forall t \in \{0,\dots,T-2\}, \forall z_i \in \mathcal{Z}_t^i\\ &\quad\text{avec } |\mathcal{Z}_t^i| = maxTrees && \forall i \in \mathcal{I}, \forall t \in \{1,\dots,T-1\}\\ &\qquad \mathcal{Z}_0^i = \left\{z_i^0\right\} && \forall i \in \mathcal{I} \end{aligned}$$

Le nombre de combinaisons est :

$$\prod_{i\in\mathcal{I}} |\mathcal{A}_i|^{maxTrees(T-1)+1} maxTrees^{|\Omega_i| maxTrees(T-1)} \tag{7.8}$$

L'espace de recherche n'est plus doublement exponentiel par rapport à l'horizon T comme dans le cas non borné [11] mais est simplement exponentiel par rapport à l'horizon.

7.2.2 Programmation dynamique

Comme MBDP et ses dérivés, nous utilisons une approche par programmation dynamique *bottom-up* pour trouver une politique de cette forme, ce qui permet de rendre la complexité linéaire par rapport à l'horizon T. À chaque instant t, un ensemble $\mathcal{Z}_t^i$ d'au plus *maxTrees* arbres de politiques est construit pour chaque agent $i \in \mathcal{I}$ à partir de l'ensemble $\mathcal{Z}_{t+1}^i$ des sous-arbres :

$$\mathcal{Z}_T^i = \{\perp\} \text{ (état terminal)} \qquad \forall i \in \mathcal{I} \tag{7.9}$$

$$\mathcal{Z}_{t-1}^i \in \left\{ X \subseteq \mathcal{A}_i \times \left(\mathcal{Z}_t^i\right)^{\Omega_i} \middle| 1 \leq |X| \leq maxTrees \right\} \qquad \forall i \in \mathcal{I} \tag{7.10}$$

Les algorithmes basés sur MBDP résolvent ce problème en sélectionnant *maxTrees* arbres joints z indépendamment. Plutôt que de choisir chaque arbre localement pour un b donné, nous proposons de choisir globalement les ensembles $\mathcal{Z}_{t-1}^i$ des *maxTrees* arbres de politiques à l'instant $t-1$ de chaque agent i afin de maximiser un critère global : le problème d'optimisation combinatoire (7.7) est alors résolu de manière heuristique par une séquence de problèmes d'optimisation combinatoire plus petits.

7.3 Cas POMDP

Nous présentons ici une version mono-agent de PSMBDP. Ceci permet de justifier l'approche ainsi que d'expliquer le principe dans un cas plus simple que le cas Dec-POMDP. L'algorithme n'est cependant pas réellement intéressant dans le cadre POMDP.

Le problème d'optimisation combinatoire (7.7) prend la forme :

$$\begin{aligned} &\text{maximise } V^{z}(b_0) && && (7.11)\\ &\quad\text{pour } \varphi(z) \in \mathcal{A} && \forall t \in \{0,\dots,T-1\}, \forall z \in \mathcal{Z}_t\\ &\qquad \psi(z, o) \in \mathcal{Z}_{t+1} && \forall t \in \{0,\dots,T-2\}, \forall z \in \mathcal{Z}_t\\ &\quad\text{avec } |\mathcal{Z}_t| = maxTrees && \forall t \neq 0\\ &\qquad |\mathcal{Z}_0| = 1 \end{aligned}$$

11. Le nombre de nœuds d'un arbre de politique local d'un agent i est en $|\Omega_i|^T$ et le nombre d'arbres possibles est donc en $|\mathcal{A}_i|^{|\Omega_i|^T}$.

À chaque instant t donné, l'agent ne peut être que dans *maxTrees* états internes possibles.

Une solution heuristique à ce problème est cherchée par programmation dynamique à mémoire bornée. On cherche un ensemble $\mathcal{Z}_{t-1}$ d'au plus *maxTrees* arbres de politiques à l'instant $t-1$ construits à partir de l'ensemble $\mathcal{Z}_t$ des *maxTrees* sous-arbres à l'instant t :

$$\mathcal{Z}_T = \{\perp\} \text{ (état terminal)} \tag{7.12}$$

$$\mathcal{Z}_{t-1} \in \left\{ X \subseteq \mathcal{A} \times (\mathcal{Z}_t)^\Omega \middle| 1 \leq |X| \leq maxTrees \right\} \tag{7.13}$$

Le problème est donc de choisir un tel ensemble. Nous proposons d'introduire un critère explicite d'optimisation sur le choix de cet ensemble $\mathcal{Z}_{t-1}$ qui est alors formulé comme un problème d'optimisation combinatoire.

Remarque: À l'inverse, les approches de la famille MBDP considèrent ce problème comme une série de *maxTrees* choix *indépendants* c'est-à-dire *maxTrees* problèmes d'optimisation combinatoire en *maxTrees* points de croyance b.

7.3.1 Critère moyen

Une première approche pour la sélection des arbres de politique à l'instant t consiste à maximiser la moyenne de $V^t(b)$ (figure 7.2) :

$$\begin{aligned} \text{maximise } & \int V_t(b)\, db \\ \text{avec } & V_t(b) = \arg\max_{z \in \mathcal{Z}_t} V^z(b) \\ \text{sujet à } & \mathcal{Z}_t \subseteq \bar{\mathcal{Z}}_t = \mathcal{A} \times (\mathcal{Z}_{t+1})^\Omega \\ & 1 \leq |\mathcal{Z}_t| \leq maxTrees \end{aligned} \tag{7.14}$$

Le critère cherche une approximation avec un nombre borné *maxTrees* de vecteurs α de la fonction de valeur obtenue par mise à jour exhaustive. Le choix de l'ensemble $\mathcal{Z}_t$ d'au plus *maxTrees* arbres parmi les $|\mathcal{A}|maxTrees^{|\Omega|}$ arbres de $\bar{\mathcal{Z}}_t$ générés par mise à jour exhaustive est alors un problème d'optimisation combinatoire.

7.3.2 Critère espéré

L'accessibilité des croyances peut être utilisée afin de concentrer la planification sur les parties pertinentes de l'espace des croyances [Szer et Charpillet, 2006a]. De manière plus générale, certaines croyances peuvent être plus probables. Nous introduisons une distribution heuristique de probabilité de la croyance B_t à l'instant t dans le critère (figure 7.3) :

$$\begin{aligned} \text{maximise } & \int V_t(b)\mu_t(b)\, db = \mathop{\mathrm{E}}_{b \sim \mu_t} [V_t(b)] \\ \text{avec } & V_t(b) = \arg\max_{z \in \mathcal{Z}_t} V^z(b) \\ \text{sujet à } & \mathcal{Z}_t \subseteq \bar{\mathcal{Z}}_t = \mathcal{A} \times (\mathcal{Z}_{t+1})^\Omega \\ & 1 \leq |\mathcal{Z}_t \leq maxTrees \end{aligned} \tag{7.15}$$

Remarque: La variable aléatoire B_t est une fonction de l'historique H_t.

Remarque: $B_t \in \Delta\mathcal{S}$ est elle-même une distribution de probabilité *a posteriori* de l'état S_t et $\mu_t \in \Delta(\Delta\mathcal{S})$ est une « méta-distribution » de probabilité, une distribution de probabilité sur les distributions de probabilité sur les états.

La distribution de probabilité *a priori* de B_t ne peut pas être déterminée avant la planification parce qu'elle dépend de la politique. En pratique, nous utilisons une distribution heuristique μ_t : un problème est de choisir une bonne distribution heuristique. Nous introduisons une politique $\tilde{\pi}$ obtenue de manière heuristique : ce peut être une politique du MDP ou du POMDP sous-jacent obtenue par planification exacte ou optimale dans ces problèmes plus simples à résoudre, ou encore une politique Dec-POMDP approchée. Nous utilisons cette politique heuristique comme approximation de la politique recherchée et nous utilisons comme distributions heuristiques μ_t les distributions *a priori* des croyances de B_t obtenues en suivant la politique heuristique $\tilde{\pi}$ depuis la distribution heuristique μ_0 :

$$\mu_t(b) = \Pr(B_t = b | B_0 \sim \mu_0, \tilde{\pi}) \approx \Pr(B_t = b | B_0 \sim \mu_0, \pi) \tag{7.16}$$

Généralement, on considère une croyance initiale $B_0 = b_0$ connue, $\mu_0(b_0) = \Pr(B_0 = b_0) = 1$, et la distribution heuristique μ_t est une prédiction de B_t depuis l'instant 0 pour une politique heuristique $\tilde{\pi}$ donnée.

L'exemple 7.3.1 illustre l'idée que nous avions déjà évoquée que lorsque le nombre d'arbres *maxTrees* est faible ce type de critère de sélection conduit à de meilleures politiques que l'optimisation locale en *maxTrees* points de croyance b.

Exemple 7.3.1*:* Considérons les trois vecteurs $\alpha_1, \alpha_2, \alpha_3$ donnés dans 7.4 et la distribution heuristique μ_t. Si *maxTrees* $= 1$, utiliser MBDP conduit à générer une seule croyance selon μ_t et donc à sélectionner soit α_1 soit α_2 : dans les deux cas l'espérance des récompenses si l'état suit effectivement μ_t est de l'ordre de $\big(0.5, 0.5\big)^T \cdot \alpha_1 = \big(0.5, 0.5\big)^T \cdot \alpha_2 = -60$. Utiliser le critère d'optimisation global conduit à sélectionner α_3 qui est meilleur en moyenne pour les croyances de μ_t : $\big(0.5, 0.5\big)^T \cdot \alpha_3 = 2$.

Exemple 7.3.2*:* La figure 7.5 montre le problème lié au choix indépendant des arbres de politiques. Nous considérons ici une distribution heuristique uniforme. Pour *maxTrees* $= 2$, un algorithme tel que PBIP a une chance sur 2 de sélectionner les arbres correspondant à α_1 et α_2 (comme représenté sur la figure) ou α_3 et α_4 : ceci se produit si les deux points de croyance échantillonnés sont sélectionnés dans la même moitié de l'espace $\Delta\mathcal{S}$. Ajouter α_2 à la sélection alors que α_1 a déjà été sélectionné ou inversement, n'est cependant pas très intéressant car il n'apporte qu'un gain assez faible par rapport à α_1. Inversement, ajouter α_4 est plus intéressant car le gain par rapport à α_1 est, en moyenne, bien supérieur. Dans une approche de type POMDP, ce problème est pallié

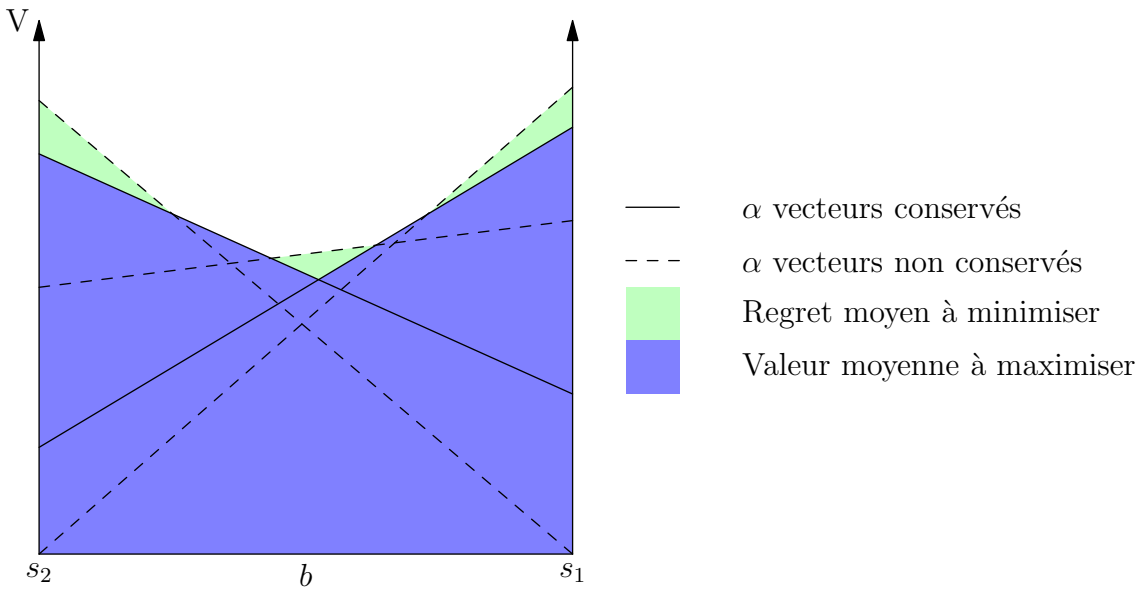


FIGURE 7.2 – Critère moyen

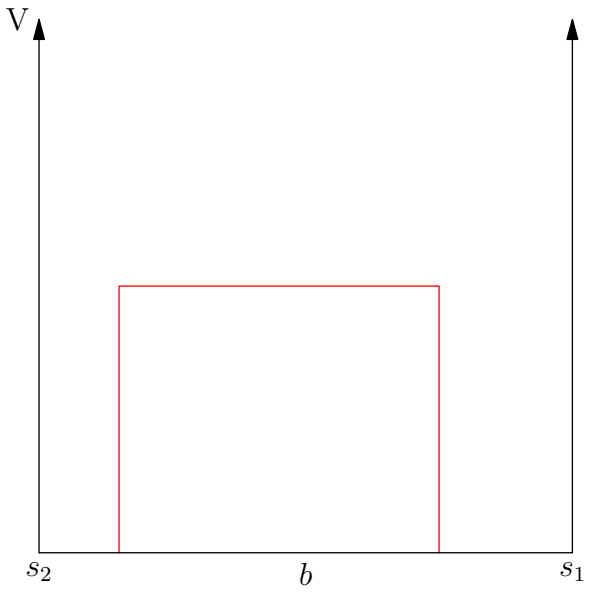


FIGURE 7.3 – Exemple de distribution heuristique

par le fait que le nombre *maxTrees* d'arbres sélectionnés est important. Dans le cadre multi-agent, l'utilisation du critère d'optimisation global que nous proposons conduit à sélectionner des vecteurs α qui ne sont pas similaires, par exemple α_1 et α_4 mais qui, conjointement, forment une bonne fonction de valeur.

7.3.3 Échantillonnage

En horizon fini, pour un nombre fini d'observations et pour une croyance initiale $B_0 = b^0$, la distribution μ_t est une distribution finie et l'intégration est une somme finie :

$$\mathop{\mathrm{E}}_{b\sim\mu_t}\left[V^t(b)\right] = \sum_b \Pr(B_t = b|B_0, \pi)V^t(b) \tag{7.17}$$

Cependant, l'ensemble des croyances atteignables est de manière générale de taille exponentielle avec l'instant t ce qui empêche l'évaluation exacte.

Dans le cas contraire (horizon infini, nombre d'observations infinies, distribution μ_0 infinie), le nombre de croyances atteignables est de manière générale infini. Plus particulièrement, si on utilise une distribution initiale μ_0 sur la croyance initiale $B_0 \sim \mu_0$ ou que l'ensemble d'observations n'est pas discret, l'ensemble des croyances atteignables n'est généralement pas discret.

Dans tous les cas, l'échantillonnage de Monte-Carlo[Doucet *et al.*, 2001] sur la distribution de probabilité heuristique de B_t peut être utilisée pour approcher le critère (figure 7.6),

$$\mathop{\mathrm{E}}_{b\sim\mu_t}\left[V^t(b)\right] \approx \frac{1}{N}\sum_{k=1}^{N} V^t(b_k) \tag{7.18}$$

où les b_k sont des échantillons indépendants de μ_t et N est le nombre d'échantillons.

Si les distributions heuristiques μ_t considérées sont les distributions *a priori* des croyances B_t suivant une politique heuristique $\tilde{\pi}$, les échantillons peuvent être obtenus en simulant le processus stochastique avec la politique heuristique $\tilde{\pi}$ jusqu'à l'instant t (algorithme 7.1).

Remarque: Les lois des politiques sont étendues pour pouvoir représenter des politiques heuristiques MDP, POMDP et Dec-POMDP :

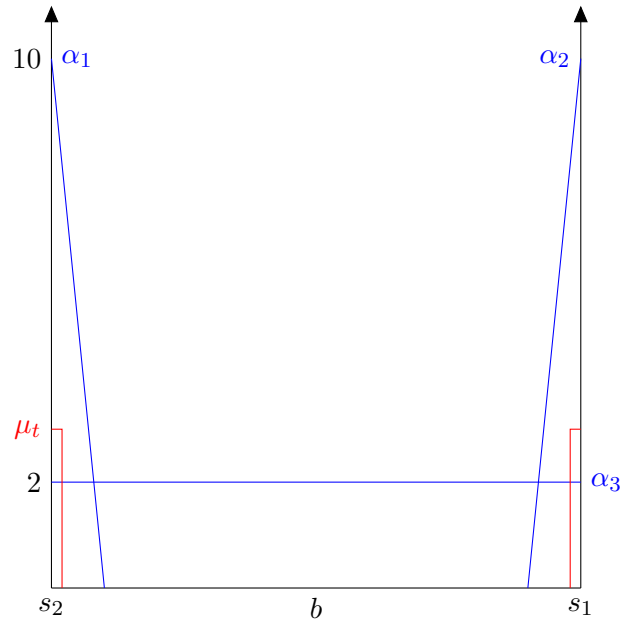
$$\nu(z|s, b) = \Pr(Z_0 = z|S_0 = s, B_0 = b) \tag{7.19}$$

$$\psi(z'|z, a, s', o) = \Pr(Z_{t+1} = z'|Z_t = z, A_{t+1} = a, O_{t+1} = o, S_{t+1} = s') \tag{7.20}$$

7.4 Passage aux Dec-POMDP

7.4.1 Critère exact

Le principe est généralisé au cas Dec-POMDP. En utilisant la distribution de probabilité *a priori* heuristique μ_t des croyances centralisées B_t, PSMBDP cherche pour chaque agent i un ensemble $\mathcal{Z}_t^i$ d'au plus *maxTrees* arbres de politiques parmi les $|\mathcal{A}_i|maxTrees^{|\Omega_i|}$ générés par mise



$\alpha_1 = \begin{pmatrix} -70 & 10 \end{pmatrix}^T$, $\alpha_2 = \begin{pmatrix} 10 & -70 \end{pmatrix}^T$, $\alpha_3 = \begin{pmatrix} 2 & 2 \end{pmatrix}^T$

FIGURE 7.4 – Intérêt de l'optimisation globale

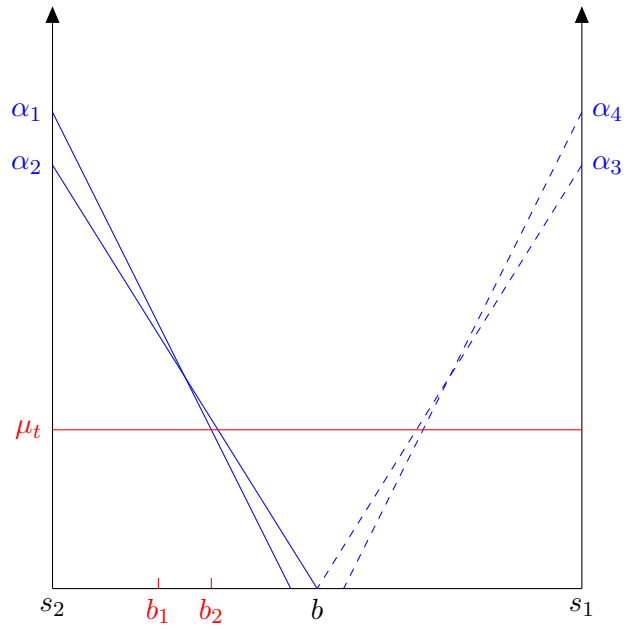


FIGURE 7.5 – Choix indépendant ou dépendant des arbres

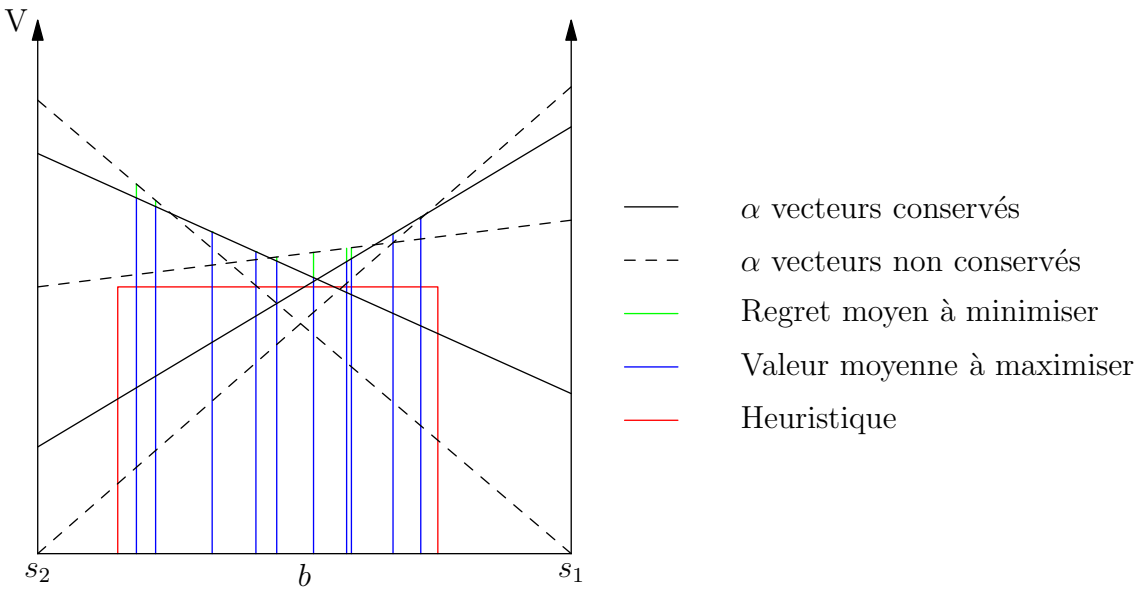


FIGURE 7.6 – Critère échantillonné

Algorithme 7.1 : Échantillonnage des croyances

Données : Politique (Dec-POMDP, POMDP ou MDP) heuristique $\tilde{\pi} = (\tilde{\nu}, \tilde{\varphi}, \tilde{\psi})$;
Distribution μ_0 de la croyance initiale B_0

Résultat : Estimations de μ_t de Monte-Carlo de $\Pr(B_t | B_0 \sim \mu_0, \tilde{\pi})$ utilisant N échantillons $b_{t,k}$ pour tout t

pour $k = 1$ **à** N **faire**
  `/* Initialiser le processus */`
  Échantillonner la croyance initiale $b_{0,k}$ selon μ_0
  Échantillonner l'état initial s selon $b_{0,k}$
  Échantillonner l'état interne joint initial z depuis $\tilde{\nu}(s, b_{0,k})$
  `/* Simuler le processus */`
  pour $t = 1$ **à** H **faire**
    `/* Mettre à jour le processus */`
    Échantillonner l'action a selon $\varphi(z)$
    Échantillonner le nouveau état s' et l'observation jointe o selon $\mathcal{J}(s, a)$
    Échantillonner le nouvel état interne joint z' selon $\psi(z, a, s', o)$
    Mise à jour de la croyance centralisée : $b_{t,k} \leftarrow \tau(b_{t-1,k}, a, o)$
    $s \leftarrow s'$
    $z' \leftarrow z$

à jour exhaustive :

$$\begin{aligned} \text{maximise } & \underset{b\sim\mu_t}{\mathrm{E}}\left[V_t(b)\right] && (7.21)\\ \text{avec } & V_t(b) = \max_{z\in\prod_{i\in\mathcal{I}}\mathcal{Z}_t^i} V^z(b) \\ \text{sujet à } & \mathcal{Z}_i^t \subseteq \bar{\mathcal{Z}}_i^t = \mathcal{A}_i \times (\mathcal{Z}_i^{t+1})^{\Omega_i} && \forall i \in \mathcal{I} \\ & 1 \leq |\mathcal{Z}_i^t| \leq maxTrees && \forall i \in \mathcal{I} \end{aligned}$$

Remarque: Contrairement au cas POMDP, les agents n'ont pas accès à la croyance centralisée B_t à chaque instant. On n'a donc pas la garantie que les agents seront effectivement capables de choisir le meilleur arbre joint de $\mathcal{Z}^t$ à chaque instant. Cependant, on s'assure que pour un maximum de croyances centralisées, il existe un bon arbre de politique. C'est l'étape suivante de mise à jour, qui, en choisissant des arbres de l'ensemble $\mathcal{Z}^{t-1}$ à partir de ceux de l'ensemble $\mathcal{Z}^t$, fait en sorte que les agents choisissent de préférence de bons arbres joints à l'instant t.

7.4.2 Échantillonnage

Le critère est évalué par échantillonnage de Monte-Carlo sur μ_t. Le critère (7.18) est généralisé au cas Dec-POMDP :

$$\begin{aligned} \text{maximise } & \sum_{k=1}^{N} V^t(b_k^t) && (7.22)\\ \text{avec } & V_t(b) = \max_{z\in\prod_{i\in\mathcal{I}}\mathcal{Z}_t^i} V^z(b) \\ \text{sujet à } & \mathcal{Z}_t^i \subseteq \bar{\mathcal{Z}}_t^i = \mathcal{A}_i \times (\mathcal{Z}_{t+1}^i)^{\Omega_i} && \forall i \in \mathcal{I} \\ & 1 \leq |\mathcal{Z}_t^i| \leq maxTrees && \forall i \in \mathcal{I} \end{aligned}$$

Comme précédemment, les distributions heuristiques μ_t induites par une politique heuristique $\tilde{\pi}$ peuvent être utilisées. Des échantillons de cette distribution sont obtenus en simulant le processus avec la politique heuristique $\tilde{\pi}$ jusqu'à l'instant t (algorithme 7.1).

La politique heuristique que nous avons utilisée est la politique $\tilde{\pi}$ POMDP gloutonne par rapport à la fonction de valeur optimale MDP $V_t^{\star,\mathrm{MDP}}$:

$$A_{t+1} = \tilde{\pi}_t(B_t) = \arg\max_{a\in\mathcal{A}} \left[H_a V_t^{\star,\mathrm{MDP}}\right](B_t) \qquad (7.23)$$

Les échantillons de cette distribution sont obtenus en simulant le processus (algorithme 7.1). Dans la pratique, nous avons utilisé la politique MDP optimale comme politique heuristique car elle est très simple à calculer.

Remarque: Une autre solution consiste à fixer directement la forme des distributions heuristiques μ_t utilisées. Une solution simpliste consiste à utiliser la distribution uniforme : on se ramène alors au cas du critère moyen. Nous avons testé cette approche mais elle ne fournissait pas de résultats satisfaisants pour les problèmes envisagés. D'autres solutions pourraient être envisagées (distribution de Dirichlet, Dirichlet généralisée, distribution logistique normale) mais se pose alors le problème du choix de leurs paramètres : cette approche n'a pas été développée dans cette thèse.

7.4.3 Comparaison avec MBDP

Une différence principale avec les approches basées sur MBDP est que, dans celles-ci, chaque arbre de politique joint est sélectionné en ne se basant que sur un seul point de croyance b. Dans PSMBDP, les arbres de politiques sont sélectionnés tous ensembles en se basant sur la distribution μ_t avec le critère (7.21). Ce critère permet de prendre en compte la complémentarité entre les différents arbres (voir exemple 7.3.2).

Dans la pratique, cette distribution est approchée par échantillonnage de Monte-Carlo et le nombre N d'échantillons peut être arbitrairement plus grand que le paramètre *maxTrees*. Ceci permet de mieux couvrir l'espace des croyances qui est un espace continu de dimensions $|\mathcal{S}| - 1$.

Dans le cas où le nombre N de points de croyance est inférieur ou égal au paramètre *maxTrees*, résoudre le problème d'optimisation combinatoire revient à choisir le meilleur arbre joint local pour chaque point de croyance b généré. Le comportement de PSMBDP revient alors à une approche à base *lookahead* comme MBDP et PBIP.

7.5 Résolution gloutonne

Le problème d'optimisation combinatoire (7.22) est de taille trop importante pour être résolu de manière exacte. Nous proposons une méthode simple de résolution qui construit les ensembles $\mathcal{Z}_t^i$ incrémentalement en ajoutant des arbres locaux z_i. Nous présentons tout d'abord le principe général de notre approche et nous introduisons ensuite une méthode qui permet de trouver les arbres locaux z_i à ajouter à la sélection par recherche *branch-and-bound* ce qui permet d'éviter l'énumération exhaustive de l'ensemble des arbres locaux possibles.

7.5.1 Principe

Afin de limiter la complexité de la recherche, une méta-heuristique gloutonne peut être utilisée : les ensembles $\mathcal{Z}_t^i$ des arbres de politique locaux sont construits de manière incrémentale en ajoutant le meilleur arbre z_i à l'ensemble $\mathcal{Z}_t^i$ pour un i donné jusqu'à atteindre la limite *maxTrees* (algorithme 7.2).

Initialisation L'arbre joint z qui maximise le critère est tout d'abord sélectionné (ligne 2). Il s'agit d'une opération de *lookahead* pour la croyance moyenne

$$\bar{b} = \frac{1}{N} \sum_{k=1}^{N} b_{t,k} \approx \underset{b \sim \mu_t}{\mathrm{E}} [b] \tag{7.24}$$

On peut chercher une solution exacte avec le *lookahead* de PBIP ou PBIP-IPG (section 4.4.4 et 4.4.5) ou une solution approchée par recherche d'équilibre de Nash comme dans MBDP/NE (section 6.2). Les arbres locaux z_i trouvés sont placés dans les ensembles $\mathcal{Z}_t^i$.

Ajout d'arbre Ensuite, l'algorithme choisit un agent i et cherche le meilleur arbre local (ligne 11) à ajouter pour optimiser le critère (de variable z_i) :

$$\begin{aligned} \text{maximise } & \sum_{k=1}^{N} V_t(b_{t,k}) \\ \text{avec } & V_t(b) = \max_{z' \in X} V^{z'}(b) \\ \text{où } & X = \left\{ (z'_j)_{j \in \mathcal{I}} \middle| z'_i \in \mathcal{Z}_t^i \cup \{z_i\}, \forall j \neq i, z'_j \in \mathcal{Z}_t^j \right\} \\ \text{sujet à } & z_i \in \bar{\mathcal{Z}}_t^i = \mathcal{A}_i \times (\mathcal{Z}_i^{t+1})^{\Omega_i} \end{aligned} \tag{7.25}$$

Cette étape est répétée jusqu'à ce que le nombre maximum *maxTrees* d'arbres soit atteint pour chaque agent ou jusqu'à ce que le critère ne puisse plus être amélioré en ajoutant de nouveaux arbres.

Le problème d'optimisation combinatoire (7.22) est décomposé en une série de problèmes d'optimisation combinatoires plus petits (7.25) et donc de complexité inférieure.

Dans une première approche, les sous-problèmes peuvent être résolus par énumération exhaustive avec un opérateur de programmation dynamique composé d'une phase de mise à jour exhaustive et une phase d'élagage (algorithme 7.2). Cependant, ces problèmes peuvent être résolus en utilisant les techniques d'optimisation combinatoire, en particulier la recherche *branch-and-bound*.

7.5.2 Recherche *branch-and-bound*

L'énumération exhaustive ne passe pas à l'échelle par rapport au nombre d'observations parce que la phase de mise à jour exhaustive construit un nombre d'arbres de politiques exponentiel par rapport au nombre d'observations. Afin de passer à l'échelle avec le nombre d'observations, nous voulons résoudre le problème d'optimisation combinatoire (7.25) sans faire l'énumération exhaustive de tous les arbres de politiques possibles : l'opérateur de programmation dynamique en deux phases est remplacé par un opérateur avec une seule phase d'optimisation globale qui utilise une recherche *branch-and-bound* pour énumérer implicitement les arbres de politique [Corona et Charpillet, 2010a].

L'algorithme 7.2 de sélection gloutonne peut être implémenté sans construire et énumérer explicitement l'ensemble $\bar{\mathcal{Z}}_t^i = \mathcal{A}_i \times (\mathcal{Z}_i^{t+1})^{\Omega_i}$ des arbres de politiques locales. Ce problème d'optimisation combinatoire peut être résolu par recherche *branch-and-bound* de manière similaire à ce qui est fait dans PBIP [Dibangoye *et al.*, 2009].

7.5.2.1 *Branch*

Un arbre de politique local z_i pour l'agent i à l'instant t est défini par :

- son action racine locale $\varphi_i(z_i) \in \mathcal{A}_i$,
- pour chaque observation o_i, un sous-arbre de politique local $\psi_i(z_i, o_i) \in \mathcal{Z}_{t+1}^i$.

Un nœud $\tilde{z}_i$ de l'arbre de recherche représente un *sous-ensemble de l'ensemble des politiques locales* z_i possibles défini par un ensemble de contraintes. Il peut être vu comme une politique locale incomplète dont certaines des propriétés ne sont pas encore définies.

Chaque nœud fils $\tilde{z}_i'$ est un sous-ensemble de son nœud parent $\tilde{z}_i$ obtenu en ajoutant une contrainte sur l'arbre de politique local z_i (figure 7.7) :

- soit en fixant l'action locale $\varphi_i(z_i) \in \mathcal{A}_i$: $\tilde{z}_i' = \{z_i \in \tilde{z}_i | \varphi_i(z_i) = a\}$;
- soit en fixant un sous-arbre de politique local $\psi_i(z_i, o_i) \in \mathcal{Z}_{t+1}^i$ pour une observation locale o_i donnée : $\tilde{z}_i' = \{z_i \in \tilde{z}_i | \psi_i(z_i, o_i) = z_i'\}$.

Une différence avec PBIP est que l'on cherche un comportement pour un agent i donné plutôt que pour l'ensemble des agents : la recherche est effectuée dans l'espace des arbres de politiques locales au lieu de l'espace des arbres de politiques jointes. Cet espace de recherche est plus petit, $|\mathcal{A}_i| maxTrees^{|\Omega_i|}$ au lieu de $|\mathcal{A}|(maxTrees^{|\mathcal{I}|})^{\max_{i \in \mathcal{I}} |\Omega_i|}$ pour PBIP et chaque nœud de l'arbre de recherche est valide. De plus, comme la recherche ne concerne qu'un seul agent i, il n'y a pas de contrainte de décentralisation à vérifier.

Algorithme 7.2 : Élagage avec optimisation incrémentale gloutonne

Données : N points de croyance $b_{t,k}$;
Ensembles des arbres locaux $\forall i, \bar{\mathcal{Z}}_t^i$ à élaguer (et leurs vecteurs α)
Résultat : Ensembles des arbres locaux $\forall i, \mathcal{Z}_t^i$ élaguée (et leurs vecteurs α)

Soit le critère $\forall \mathcal{Z}, I(\mathcal{Z}) = \sum_{k=1}^{N} \max_{z \in \mathcal{Z}} V_z(b_{t,k})$
`/* Lookahead initial (recherche exacte ou approchée) */`
$(z_i)_{i \in \mathcal{I}} \leftarrow$ `lookahead`$(\frac{1}{N} \sum b_{t,k}, (\mathcal{Z}_{t+1}^i)_{i \in \mathcal{I}})$
pour chaque $i \in \mathcal{I}$ **faire**
  $\mathcal{Z}_t^i \leftarrow \{z_i\}$
  $\bar{\mathcal{Z}}^{it} \leftarrow \bar{\mathcal{Z}}_i^t \setminus \{z_i\}$
`/* Construction incrémentale */`
flag $\leftarrow$ true
tant que $\exists i, |\mathcal{Z}_t^i| \neq maxTrees \wedge flag$ **faire**
  flag $\leftarrow$ false
  pour chaque $i \in \mathcal{I}$ **faire si** $|\mathcal{Z}_t^i| \neq maxTrees$ **alors**
    `/* Recherche d'un arbre local` z_i `*/`
    Let $\forall z_i, K(z_i) = \left(\mathcal{Z}_t^i \cup \{z_i\}\right) \times \prod_{j \neq i} \mathcal{Z}_j^t$
    $z_i \leftarrow \arg\max_{\bar{z}_i \in \bar{\mathcal{Z}}_t^i} I(K(\bar{z}_i))$
    si *amélioration de* $I(\mathcal{Z}_t)$ *si on ajoute* z_i **alors** `// Ajouter` z_i
      $\mathcal{Z}_t^i \leftarrow \mathcal{Z}_i^t \cup \{z_i\}$
      $\bar{\mathcal{Z}}_t^i \leftarrow \bar{\mathcal{Z}}_i^t \setminus \{z_i\}$
      flag $\leftarrow$ true

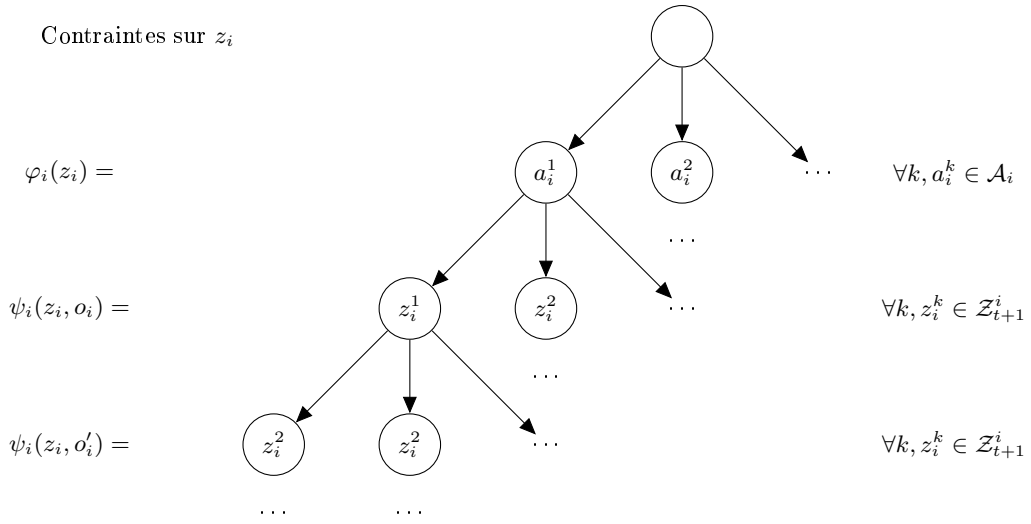


FIGURE 7.7 – Arbre de recherche d'un arbre de politique local z_i

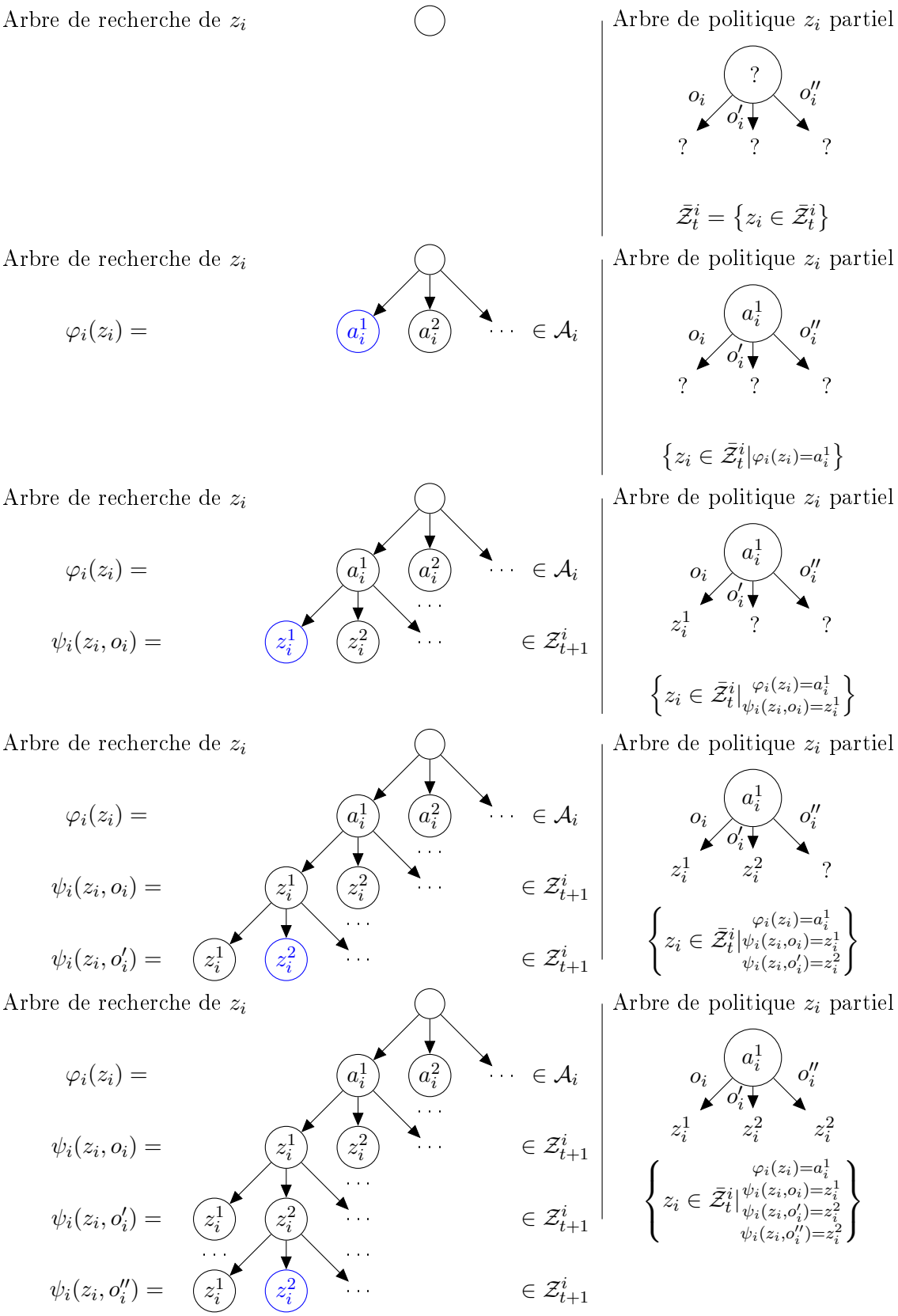


FIGURE 7.8 – Correspondance entre arbre de politique et arbre de recherche

7.5.2.2 *Bound*

On peut obtenir une heuristique admissible $f(\tilde{z}_i)$ en remplaçant chaque $V_t(b)$ dans (7.25) par un majorant. On sépare la partie connue de la partie inconnue :

$$V_t(b) = \max(k(b), V'(b)) \tag{7.26}$$
$$k(b) = \max_{z \in \mathcal{Z}_t} V^z(b) \tag{7.27}$$
$$V'(b) = \max_{z_{-i} \in \mathcal{Z}^t_{-i}} V^{\langle z_i, z_{-i} \rangle}(b) \tag{7.28}$$

Le terme $k(b)$ est connu et constant pour une recherche donnée. On peut obtenir un majorant $f(\tilde{z})$ de $f(z)$ en majorant les $V^{\langle z_i, z_{-i} \rangle}$:

$$\forall \tilde{z}, \forall b, V^{\tilde{z}}(b) \geq \max_{z \in \tilde{z}} V^z(b) \Rightarrow f(\tilde{z}_i) \geq \max_{z_i \in \tilde{z}_i} f(z_i) \tag{7.29}$$

avec le majorant

$$f(\tilde{z}_i) = \sum_{k=1}^{N} \max \left[k(b_{t,k}), \max_{z_{-i} \in \mathcal{Z}_t^{-i}} V^{\langle \tilde{z}_i, z_{-i} \rangle}(b_{t,k}) \right] \tag{7.30}$$

Remarque: On désigne par $\tilde{z} = \langle \tilde{z}_i, z_{-i} \rangle$ l'ensemble des politiques jointes $z = \langle z_i, z_{-i} \rangle$ formées de z_i et de z_{-i},

$$\tilde{z} = \langle \tilde{z}_i, z_{-i} \rangle = \{ \langle z_i, z_{-i} \rangle | z_i \in \tilde{z}_i \} \tag{7.31}$$

ou de manière équivalente l'arbre de politique joint partiellement défini formé de z_{-i} et de l'arbre local partiellement défini $\tilde{z}_i$.

On cherche donc un majorant $V^{\langle \tilde{z}_i, z_{-i} \rangle}(b)$ de $V^z(b)$:

$$V^{\langle \tilde{z}_i, z_{-i} \rangle}(b) = \max_{z \in \langle \tilde{z}_i, z_{-i} \rangle} V^z(b) \tag{7.32}$$

La difficulté de ce problème d'optimisation combinatoire est que le meilleur sous-arbre de politique locale $z_i = \psi_i(z_i, o_i)$ pour une observation locale o_i peut être différent pour différents points de croyance b. Un majorant est obtenu en relâchant cette contrainte c'est-à-dire en permettant de choisir un sous-arbre $z_i = \psi_i(z_i, o_i)$ différent pour chaque croyance b. On divise l'ensemble des observations locales Ω_i en deux ensembles : l'ensemble Ω_i^1 des observations jointes pour lesquelles z_i est défini et l'ensemble Ω_i^2 des observations jointes pour lesquelles z_i ne l'est pas encore. On pose de même les ensembles d'observations jointes Ω^1 et Ω^2 correspondants :

$$\Omega^1 = \left\{ \langle o_i, o_{-i} \rangle | o_i \in \Omega_i^1, o_{-i} \in \Omega_{-i} \right\} \tag{7.33}$$
$$\Omega^2 = \left\{ \langle o_i, o_{-i} \rangle | o_i \in \Omega_i^2, o_{-i} \in \Omega_{-i} \right\} \tag{7.34}$$

Le majorant est obtenu en choisissant le meilleur sous-arbre $z_i = \psi_i(z_i, o_i)$ pour la croyance b

courante pour chaque observation locale $o_i \in \Omega_i^2$:

$$V^{\tilde{z}}(b) = \mathcal{R}(b,a) \tag{7.35}$$
$$+ \sum_{\substack{o\in\Omega^1 \\ z'=\psi(\tilde{z},o)}} \mathcal{O}(o|b,a)\left[\alpha^{z'}\cdot\tau(b,a,o)\right] \tag{7.36}$$
$$+ \sum_{o_i\in\Omega_i^2} \max_{z'_i\in\mathcal{Z}_{t+1}^i} \sum_{\substack{o_{-i}\in\Omega_{-i} \\ o=\langle o_i,o_{-i}\rangle \\ z'_{-i}=\psi_{-i}(z_{-i},o_{-i}) \\ z'=\langle z'_i,z'_{-i}\rangle}} \mathcal{O}(o|b,a)\left[\alpha^{z'}\cdot\tau(b,a,o)\right] \tag{7.37}$$

avec $a = \varphi(\tilde{z})$. Ce choix est fait de manière indépendante pour chaque point de croyance b.

Remarque: Dans un précédent article, nous avions proposé un critère plus grossier où un sous-arbre différent est choisi pour chaque observation jointe $o \in \Omega^2$ plutôt que pour chaque observation locale $o_i \in \Omega_i^2$:

$$V^{\tilde{z}}(b) = \mathcal{R}(b,a) \tag{7.38}$$
$$+ \sum_{\substack{o\in\Omega^1 \\ z'=\psi(\tilde{z},o)}} \mathcal{O}(o|b,a)\left[\alpha^{z'}\cdot\tau(b,a,o)\right] \tag{7.39}$$
$$+ \sum_{\substack{o\in\Omega^2 \\ z'_{-i}=\psi_{-i}(z_{-i},o_{-i})}} \mathcal{O}(o|b,a) \max_{\substack{z'_i\in\mathcal{Z}_{t+1}^i \\ z'=\langle z'_i,z'_{-i}\rangle}} \left[\alpha^{z'}\cdot\tau(b,a,o)\right] \tag{7.40}$$

Dans la pratique, sur les *benchmarks* utilisés, on ne constate pas une différence significative de temps de calcul entre les deux approches. Les résultats présentés plus bas utilisent cette dernière approche.

7.5.3 Complexité

Le tableau 7.1 résume les propriétés de complexité des opérateurs DP de PBIP et PSMBDP. Nous faisons l'hypothèse que $\forall(i,j) \in \mathcal{I}^2$, $|\mathcal{A}_i| = |\mathcal{A}_j|$, $|\Omega_i| = |\Omega_j|$.

Pour PBIP, l'opérateur DP effectue *maxTrees* recherche par *lookahead* pour *maxTrees* points de croyance b. Pour chaque b, les $|\mathcal{A}||\Omega||\mathcal{Z}_{t+1}|$ contributions $k(a,o,z')$ sont précalculées (algorithme 4.11) avec une complexité en

$$|\mathcal{S}|^2 + maxTrees^{|\mathcal{I}|}|\mathcal{S}| \tag{7.41}$$

L'arbre cherché est défini par un choix de $\varphi(z) \in \mathcal{A}$ et $\omega = |\bar{\Omega}| = \max_{i\in\mathcal{I}}|\Omega_i|$ choix de $\psi(z,o) \in \mathcal{Z}_{t+1}$ pour $o \in \bar{\Omega}$ et l'espace de recherche est donc de taille

$$|\mathcal{A}| maxTrees^{|\mathcal{I}|\omega} \tag{7.42}$$

Une opération *bound* est une somme sur les observations jointes. La complexité d'un opérateur de Bellman approché de PBIP est :

$$maxTrees\left[|\mathcal{A}|\Omega|\left(\mathcal{S}|^2 + maxTrees^{|\mathcal{I}|}|\mathcal{S}|\right) + |\mathcal{A}| maxTrees^{|\mathcal{I}|\omega}|\Omega|\right] \approx |\Omega||\mathcal{S}|^2|\mathcal{A}| maxTrees^{|\mathcal{I}|\omega} \tag{7.43}$$

avec $\omega = \max_{i \in \mathcal{I}} |\Omega_i|$.

Pour PSMBDP, l'opérateur de Bellman effectue une recherche d'un arbre joint de manière similaire, par exemple avec le *lookahead* de PBIP, et une séquence de recherches d'arbres de politique locaux. Dans le pire des cas, le nombre de ces recherches est en $(maxTrees - 1)|\mathcal{I}|^2$ (si chaque passe sur tous les agents n'ajoute un arbre que pour un seul agent) mais dans la pratique, il est généralement de l'ordre de $maxTrees|\mathcal{I}|$.

L'arbre cherché est défini par un choix de $\varphi_i(z_i) \in \mathcal{A}_i$ et $|\Omega_i|$ choix de $\psi_i(z_i, o_i) \in \mathcal{Z}^i_{t+1}$ pour $o_i \in \Omega_i$ et l'espace de recherche est donc de taille $|\mathcal{A}_i| maxTrees^{|\Omega_i|}$. Chaque opération *bound* a une complexité en

$$N maxTrees^{|\mathcal{I}|-1} |\Omega| \left(|\mathcal{S}|^2 + maxTrees|\mathcal{I}||\mathcal{S}| \right) \tag{7.44}$$

La complexité de l'opérateur de Bellman est en

$$\begin{aligned}(maxTrees - 1)|\mathcal{I}|^2 |\mathcal{A}_i| maxTrees^{|\Omega_i|} N maxTrees^{|\mathcal{I}|-1} |\Omega| \left(|\mathcal{S}|^2 + maxTrees|\mathcal{S}| \right) + K \approx \\ |\Omega||\mathcal{S}|^2 |\mathcal{A}_i| maxTrees^{|\mathcal{I}|+\omega} |\mathcal{I}|^2 + K\end{aligned} \tag{7.45}$$

où K est la complexité du *lookahead* initial.

Il faut ajouter la complexité du calcul des vecteurs α qui est identique pour les deux algorithmes :

$$maxTrees^{|\mathcal{I}|} |\mathcal{S}|^2 |\Omega| \tag{7.46}$$

Quand PSMBDP est utilisé avec une recherche initiale de type PBIP, cette dernière a donc un impact prépondérant sur la complexité de l'opérateur : intuitivement, c'est parce que la taille de l'espace de recherche $\mathcal{A} \times (\mathcal{Z}')^{\bar{\Omega}}$ de PSMBDP est égale à la taille $\mathcal{A}_i \times (\mathcal{Z}'_i)^{\Omega_i}$ de l'espace de recherche élevée à la puissance du nombre d'agents :

$$|\mathcal{A} \times (\mathcal{Z}')^{\bar{\Omega}}| \approx |\mathcal{A}_i|^{|\mathcal{I}|} |\mathcal{Z}'_i|^{|\mathcal{I}||\Omega_i|}| = (|\mathcal{A}_i||\mathcal{Z}'_i|^{|\Omega_i|}|)^{|\mathcal{I}|} = |\mathcal{A}_i \times (\mathcal{Z}'_i)^{\Omega_i}|^{|\mathcal{I}|} \tag{7.47}$$

Ceci peut être résolu en utilisant une autre méthode pour sélectionner l'arbre joint initial comme la recherche d'un équilibre de Nash de MBDP/NE.

7.6 Évaluation expérimentale

7.6.1 Méthode

Nous avons testé les algorithmes proposés et les avons comparés avec MBDP et ses variantes sur les problèmes suivants : le problème du *broadcast* multi-agent, le problème du tigre décentralisée [Nair *et al.*, 2003] (table 7.2), Meeting on a grid [Bernstein, 2005] (table 7.3), le problème des pompiers [Oliehoek *et al.*, 2008] (table 7.5), une version modifiée du problème des pompiers (table 7.6), ainsi que le *Cooperative box Pushing Problem* (table 7.4). Les tables indiquent la valeur espérée moyenne (VEM), sa déviation standard (σ) et le temps moyen de calcul en secondes.

Dans certains cas, la valeur de la politique optimale du MDP sous-jacent est indiquée : il s'agit d'un majorant de la valeur de la politique optimale Dec-POMDP.

7.6.1.1 MBDP et dérivés

PBIP/BeFS est une variante de PBIP qui utilise la recherche en meilleur d'abord.

MBDP/NE est la variante de MBDP à base d'équilibre de Nash (section 6.2) et MBDP/NE2 est l'approche similaire qui parcourt l'ensemble des actions jointes [Kumar et Zilberstein, 2010] :

ces algorithmes ont été implémentés sans les approximations discutées dans la section 6.2.5 (calcul paresseux des vecteurs α et calcul approché des vecteurs α).

MBDP et ses dérivés utilisent une heuristique basée sur la distribution *a priori* sur les états : la moitié des points de croyance sont tirés de l'heuristique MDP et l'autre moitié est tirée de l'heuristique aléatoire [Seuken et Zilberstein, 2007b].

7.6.1.2 PSMBDP

Nous avons testé l'approche PSMBDP avec construction incrémentale gloutonne (section 7.5) avec mise à jour exhaustive (section 7.5.1) et avec recherche *branch-and-bound* des arbres locaux (section 7.5.2). Le premier algorithme est représenté par PSMBDP dans les tableaux.

En ce qui concerne la recherche *branch-and-bound*, nous avons testé les stratégies de recherche en profondeur d'abord et meilleur d'abord. Dans les *benchmarks* utilisés, la recherche meilleur d'abord utilise une quantité raisonnable de mémoire et les temps de calculs sont bien inférieurs. Les résultats présentés utilisent cette stratégie d'exploration. Cet algorithme est désigné par PSMBDP/BeFS (pour *Best First Search*).

La recherche d'un arbre joint initial utilise le *lookahead* de PBIP (pour PSMBDP) ou PBIP/BeFS (pour PSMBDP/BeFS).

7.6.2 Résultats

7.6.2.1 Problème du tigre décentralisé

La table 7.2 montre les résultats du problème du tigre décentralisé [Nair *et al.*, 2003] pour $T = 100$, avec 50 exécutions et, pour PSMBDP, $N = 100$.

Pour ce problème, les solutions trouvées par PSMBDP sont d'une qualité inférieure à MBDP (pour $maxTrees = 20$) mais utiliser la même heuristique que MBDP pour générer les points de croyance fournit des résultats meilleurs que les approches de type MBDP.

On peut aussi remarquer qu'il n'est pas nécessaire d'utiliser une grande valeur de *maxTrees* pour trouver une bonne solution à ce problème : seul un faible nombre (généralement 5) d'arbres de politiques à chaque instant sont généralement retenus par PSMBDP car ajouter d'autres arbres n'améliore pas le critère. Inversement, les algorithmes de type MBDP sélectionnent des arbres de politiques sous-optimaux pour les points de croyance considérés. Les temps de calculs sont donc plus faible en utilisant PSMBDP car les nombres d'arbres et de vecteurs α générés par la mise à jour exhaustive sont plus faibles.

Les résultats ne sont pas indiqués pour PSMBDP/BeFS : ils sont plus lents que la mise à jour exhaustive pour ce problème car il s'agit d'un problème de très petite taille, $|\mathcal{A}_i| = 3$ et $|\Omega_i| = 2$. L'espace des arbres locaux est de taille $3 \times 7^2 = 147$ et $3 \times 20^2 = 1200$ pour $maxTrees = 7$ et $maxTrees = 20$ respectivement.

7.6.2.2 *Meeting on a Grid*

Dans le problème *Meeting on a Grid* [Bernstein, 2005], deux agents sont placés dans une grille 2×2 et doivent passer le plus de temps possible sur la même cellule. La table 7.3 montre les résultats pour $T = 100$, $maxTrees = 7$, 100 exécutions et, pour PSMBDP, $N = 100$. Pour ce problème, les résultats sont comparables pour les approches PSMBDP/BeFS et PBIP.

TABLE 7.1 – Complexité comparée de PBIP et PSMBDP

	PBIP	PMSBDP
Nombre de recherches	$maxTrees$	$(maxTrees - 1)\lvert\mathcal{I}\rvert^2$ †
Compl. précalculs	$\lvert\mathcal{A}\rvert\lvert\Omega\rvert\left[\lvert\mathcal{S}\rvert^2 + maxTrees^{\lvert\mathcal{I}\rvert}\lvert\mathcal{S}\rvert\right]$	0
Espace de recherche	$\lvert\mathcal{A}\rvert maxTrees^{\lvert\mathcal{I}\rvert\omega}$	$\lvert\mathcal{A}_i\rvert maxTrees^{\lvert\Omega_i\rvert}$
Compl. *bound*	$\lvert\Omega\rvert$	$N maxTrees^{\lvert\mathcal{I}\rvert-1}\lvert\Omega\rvert\left(\lvert\mathcal{S}\rvert^2 + maxTrees\lvert\mathcal{I}\rvert\lvert\mathcal{S}\rvert\right)$

† : auxquelles s'ajoute une recherche centralisée d'un arbre joint initial (recherche PBIP, PBIP-IPG, équilibre de Nash ...)

Algorithme	VEM	σ	Temps (s)
	$maxTrees = 7$		
MBDP	91.59	10.03	5.92
PBIP	88.33	10.27	0.27
PSMBDP	91.20	3.23	0.63
PSMBDP †	166.27	7.06	1.67
	$maxTrees = 20$		
MBDP	138.37	3.21	540.62
PBIP	139.68	4.59	14.84
PSMBDP	Même résultats que $maxTrees = 7$		

† : distribution heuristique de MBDP

TABLE 7.2 – Problème du tigre décentralisé

Algorithm	VEM	σ	Temps (s)
MBDP	76.56	3.30	725.82
PBIP	76.56	2.88	2.10
PSMBDP/BeFS	73.25	3.87	3.57

TABLE 7.3 – *Meeting on a Grid*

7.6.2.3 *Cooperative Box Pushing*

La table 7.4 présente les résultats du problème *Cooperative Box Pushing* [Seuken et Zilberstein, 2007a]. Ce problème possède un grand nombre d'observations ($|\mathcal{I}| = 2$, $|\Omega_i| = 5$) et a été proposé comme *benchmark* pour des algorithmes conçus pour traiter un grand nombre d'observations. Les résultats sont des moyennes sur 25 exécutions.

Pour $T = 20$, PSMBDP/BeFS trouve des solutions légèrement meilleure que PBIP pour un temps de calcul comparable à PBIP/BeFS. Pour $T = 50$, la valeur moyenne est aussi comparable mais la variance est plus importante pour PMSBDP/BeFS qui trouve parfois des solution de bien meilleure qualité que les approches de type MBDP.

Les résultats de PBIP ne sont pas présents pour $H = 50$ car les temps de calculs sont très importants ($T > 20,000s$).

7.6.2.4 Problème des pompiers

La table 7.5 montre les résultats pour le problème des pompiers [Oliehoek *et al.*, 2008] avec 2 agents, 4 maisons et 3 niveaux de feu pour un horizon de 100 et $maxTrees = 7$. Les résultats sont des moyennes sur 25 exécutions. Pour PSMBDP, le nombre de points N est fixé à 100. Il s'agit d'un problème avec un nombre d'états bien plus grand que les *benchmarks* précédents. PSMBDP trouve des solutions de meilleure qualité que les approches de type MBDP comme PBIP.

7.6.2.5 Problème modifié des pompiers

Afin de tester les problèmes avec un plus grand nombre d'observations, nous utilisons une version modifiée du problème des pompiers où chaque agent observe directement l'intensité du feu dans la maison où il s'est rendu : le problème est donc en principe plus simple pour l'agent car l'agent dispose de plus d'informations mais exploiter cette information supplémentaire demande un travail de calcul supérieur.

La table 7.6 montre les résultats pour le problème modifié avec $|I| = 2$, 4 maisons, le feu allant de 0 à 3, $T = 100$, $maxTrees = 3$ et, pour PSMBDP, $N = 100$. Les résultats sont des moyennes sur 25 exécutions.

PBIP n'arrive pas à exploiter cette information supplémentaire et trouve des politiques de même qualité que pour le problème original. Inversement, PSMBDP/BeFs trouve de meilleures politiques qu'avec le problème original.

7.7 Conclusion

Nous avons présenté une nouvelle approche à base de points qui formule le problème du choix de *maxTrees* arbres de politiques locales par agents comme un problème d'optimisation combinatoire. Cette approche peut être vue comme une généralisation des approches de type MBDP où le nombre N de points de croyance peut être supérieur ou égal au paramètre *maxTrees* : il est donc possible de choisir un nombre N de points de croyance bien plus grand ce qui permet de mieux couvrir l'espace des croyances. De plus, le critère utilisé permet d'utiliser au mieux la ressource limitée que constituent les arbres de politiques en exploitant la complémentarité entre arbres de politiques et en utilisant le plus d'information heuristique possible pour leur sélection.

Le problème de sélection est formulé comme un problème d'optimisation combinatoire de taille très importante et sa résolution exacte n'est pas envisageable. Nous proposons une méthode métaheuristique pour trouver une bonne solution à ce problème qui construit une solution de manière incrémentale en ajoutant des arbres locaux à la sélection.

Algorithm	W	VEM	σ	Temps (s)
	$H = 20$			
Optimal MDP	-	511	-	-
PBIP	7	429	11.4	11 336
PBIP/BeFS	7	421	10.3	67
	9	432	10.2	181
PSMBDP/BeFS	7	441	11.2	160
	9	444	12.6	189
	$H = 50$			
Optimal MDP	-	1 306	-	-
PBIP	7	?	?	>20 000
PBIP/BeFS	7	1 063	16.2	178
	9	1 078	13.1	471
PSMBDP/BeFS	7	1 076	69.7	483
	9	1 088	31.7	590

TABLE 7.4 – *Cooperative Box Pushing*

Algorithm	VEM	σ	Temps (s)
Optimal MDP	-35.81	-	-
	$maxTrees = 3$		
PBIP	-226.94	23.68	40
PBIP ‡	-111.86	0.00	195
PBIP/BeFS	-215.95	20.21	65
PBIP/BeFS ‡	-111.86	0.00	79
PSMBDP/BeFS	-89.08	0.00	278
	$maxTrees = 7$		
PBIP	-160.25	20.84	171
PBIP ‡	-111.85	0.00	234
PBIP/BeFS	-184.08	23.40	364
PBIP/BeFS ‡	-111.86	0.00	126
PSMBDP/BeFS	-89.08	0.00	274

‡ : distribution heuristique de PSMBDP

TABLE 7.5 – Problème des pompiers

Expérimentalement, notre approche permet de trouver pour certains *benchmarks* des solutions de bien meilleure qualité que les approches à base de *lookahead*. Cette approche semble particulièrement intéressante quand l'espaces des états, et donc celui des croyances centralisées, est de grande taille (voir le problème des pompiers). La résolution approchée que nous avons proposée permet de conserver des temps de calculs comparables aux approches de type MBDP tout en améliorant nettement la qualité des solutions trouvées.

La méthode que nous avons proposée repose sur un critère explicite à optimiser : on cherche à maximiser la valeur espérée quand la croyance centralisée suit une distribution heuristique donnée. Nous avons proposé une méthode pour trouver une solution approchée à ce problème : les méthodes classiques d'optimisation combinatoire pourraient être employées pour résoudre ce problème.

En ce qui concerne l'heuristique utilisée, les résultats que nous avons obtenus se basaient sur la distribution des croyances obtenue en suivant une politique POMDP gloutonne par rapport à la politique optimale. Il serait intéressant de tester d'autres approches pour générer ces heuristiques : distribution induite par une politique obtenue par planification POMDP, distributions paramétriques, etc.

Nous avons étudié la construction incrémentale de politique avec recherche *branch-and-bound* pour la résolution du problème d'optimisation combinatoire à chaque instant de la planification. Le principe pourrait être généralisé en utilisant d'autres méta-heuristiques pour résoudre ce problème.

Algorithm	VEM	σ	Temps (s)
Optimal MDP	-35.81	-	-
	$maxTrees = 3$		
PBIP	-270.77	24.75	137
PBIP ‡	-111.84	0.00	60
PBIP/BeFS	-214.04	30.65	249
PBIP/BeFS ‡	-111.84	0.00	107
PSMBDP/BeFS	-78.43	0.00	680
	$maxTrees = 7$		
PBIP	-216.64	35.92	612
PBIP ‡	-111.85	0.00	156
PBIP/BeFS	-255.91	19.23	1191
PBIP/BeFS ‡	-111.85	0.00	438
PSMBDP/BeFS	-78.43	0.00	735

‡ : distribution heuristique de PSMBDP

TABLE 7.6 – Problème des pompiers modifié

Chapitre 8

Conclusion

Sommaire

8.1 Contributions de la thèse

La planification des problèmes de décision décentralisée formalisés dans le cadre Dec-POMDP a une complexité NEXP-difficile [Bernstein *et al.*, 2000]. La majorité des algorithmes récents de planification Dec-POMDP se focalisent sur la recherche de bonnes solutions en limitant les ressources nécessaires en calcul. Un nombre limité de sous-politiques de taille de plus en plus grande est construit par une approche de programmation dynamique *bottom-up*. De manière similaire aux algorithmes à base de points pour POMDP, ils utilisent des heuristiques pour focaliser l'effort de calcul sur un nombre *maxTrees* limité de situations (croyances) en optimisant localement pour chaque croyance considérée : chaque croyance considérée amène à la sélection de la meilleure sous-politique pour cette croyance.

8.1.1 *Lookahead* approché

Une première limite de ce type d'approches est que l'espace de recherche du *lookahead* croît fortement avec le paramètres *maxTrees* et le nombre d'observations. Nous avons proposé de remplacer la recherche exacte d'un arbre joint z optimal pour un point de croyance b donné par une recherche approchée. En particulier, chercher l'arbre joint z comme un équilibre de Nash permet de simplifier grandement la recherche en remplaçant la recherche dans l'ensemble des arbres de politique joints par une séquence de recherches dans l'ensemble des arbres de politique locaux. Nous avons testé ces approches sur des problèmes à deux agents mais le gain devrait être plus important pour des problèmes avec un plus grand nombre d'agents.

8.1.2 Optimisation globale

Une limite des approches de type MBDP est que, pour des raisons de complexité le nombre *maxTrees* de sous-politiques conservées de cette manière est relativement faible (typiquement une dizaine). Par conséquent le nombre *maxTrees* de croyances considérées (qui est le même que le nombre de sous-politiques conservées) est très faible et est peu représentatif de la variété des situations possibles (dans un espace qui est de taille exponentielle par rapport au nombre d'états du problème). De plus, chaque sous-politique retenue est choisie en ne se basant que sur une seule croyance et peut donc être fortement aléatoire.

Nous avons proposé une nouvelle approche pour remédier à ces problèmes [Corona et Charpillet, 2009, Corona et Charpillet, 2010b, Corona et Charpillet, 2010a]. Nous représentons l'information d'accessibilité sur les croyances par une distribution de probabilité qui représente l'importance estimée de chaque croyance pour le problème de planification considéré. Ainsi le problème du choix des *maxTrees* sous-politiques est formulé en un problème d'optimisation combinatoire qui cherche à maximiser l'espérance des récompenses étant donné cette distribution sur les croyances. Les *maxTrees* problèmes indépendants de choix d'une sous-politique utilisant un critère d'optimisation local pour la croyance associée sont remplacés par un problème global du choix des *maxTrees* avec un critère global d'optimisation. En particulier, le nombre de croyances considérées peut être arbitrairement plus grand que le nombre *maxTrees* de sous-politiques à construire et on peut considérer que les approches de type MBDP sont des cas particuliers de l'approche proposée quand le paramètres *maxTrees* est égal au nombre N de points de croyance échantillonnés. Le critère proposé permet de prendre en compte la complémentarité entre les arbres de politiques ainsi que de mieux couvrir l'espace des croyances.

Ce problème d'optimisation combinatoire est cependant de taille trop importante pour être résolu de manière exacte. Nous proposons une résolution approchée qui part d'un arbre joint, éventuellement trouvé par *lookahead*, et construit la solution de manière incrémentale en ajoutant des arbres de politiques locaux z_i. Encore une fois ces arbres sont cherchés dans l'espace des arbres locaux plutôt que dans l'espace des arbres joints.

Dans plusieurs *benchmarks*, cette approche permet de trouver des solutions de meilleure qualité que les approches de type MBDP.

8.2 Synthèse des approches de planification bornée

Le problème de la planification consiste à déterminer la meilleure politique pour une croyance initiale B_0 donnée (problème P_1 dans la figure 8.1). La programmation dynamique permet de décomposer ce problème P en une séquence de sous-problèmes de mise à jour pour chaque horizon : chaque mise à jour consiste à déterminer l'ensemble des arbres de politiques locales utiles à un instant t à partir des arbres de politiques locales utiles à l'instant $t+1$ (problème M).

Les algorithmes de la famille de PBDP approchent le problème de la mise à jour M par un problème M' de mise à jour bornée qui consiste à retenir au plus *maxTrees* arbres de politique locale par agent. Ce problème de mise à jour bornée M' peut être vu comme une décomposition d'un problème de planification bornée P' qui approche le problème de planification exacte P : ce problème de planification bornée P' consiste à déterminer la meilleure politique de largeur *maxTrees* pour la croyance initiale B_0.

Idéalement, on voudrait sélectionner dans le problème de mise à jour bornée M' la sélection d'arbres de politiques qui conduit à la solution optimale du problème de planification bornée P'.

On ne sait cependant pas caractériser cette sélection optimale. Dans la pratique, des solutions heuristiques sont utilisées pour sélectionner ces arbres de politiques.

Les algorithmes de la famille de MBDP décomposent le problème de mise à jour bornée M' en *maxTrees lookaheads* pour *maxTrees* croyances B_t à l'instant sélectionnées de manière heuristique. Ce problème de *lookahead* (problème L) peut être résolu de nombreuses manières : énumération exhaustive (MBDP), énumération exhaustive d'un sous-ensemble (IMBDP, MBDP-OC), recherche *branch-and-bound* (PBIP, PBIP/BeFS). Dans le chapitre 6, nous avons soutenu l'idée que les méthodes de recherche méta-heuristiques peuvent être employées pour trouver rapidement une solution approchée au problème L de *lookahead* : nous avons plus particulièrement présenté une méthode (MBDP/NE) qui utilise la recherche locale (recherche d'un équilibre de Nash). Cette première direction consiste à faire une approximation supplémentaire par rapport aux autres méthodes de type MBDP afin de réduire les temps de calcul.

Cependant, nous avons soutenu l'idée qu'il était préférable de formuler explicitement le problème de mise à jour bornée comme un problème d'optimisation combinatoire en introduisant un critère explicite, heuristique, à optimiser (problème M'') : le critère heuristique que nous avons considéré est la maximisation de l'espérance des récompenses étant donné la distribution de probabilité μ_t de la croyance B_t à l'instant t étant donné une politique heuristique donnée. De manière générale cette distribution heuristique μ_t est trop complexe à déterminer et le critère est donc approché par échantillonnage de Monte-Carlo (problème M'''). Ce problème d'optimisation combinatoire est cependant de taille trop importante pour être résolu de manière exacte et des méthodes méta-heuristiques doivent être employées. Nous avons proposé une méthode gloutonne simple qui décompose ce problème en sous-problèmes de sélection du meilleur arbre de politique locale pour un agent donné à ajouter à la sélection (problème I). Ce sous-problème peut être résolu de manière exhaustive (PSMBDP) ou par recherche *branch-and-bound* (PSMBDP/BeFs, PSMBDP/DFS). Contrairement à notre première contribution qui est une version approchée des algorithmes de type MBDP, cette approche cherche à obtenir des solutions de meilleure qualité que les algorithmes de type MBDP en représentant explicitement le problème de mise à jour bornée comme un problème d'optimisation combinatoire.

8.3 Perspectives

8.3.1 Méta-heuristiques

Les deux approches que nous avons présenté, approche de type MBDP d'une part et approche PSMBDP d'autre part, se basent sur la résolution de problèmes d'optimisation combinatoire :

- recherche de l'arbre joint optimal pour une croyance donnée dans le cas des approches de type MBDP ;
- recherche de la sélection d'arbres locaux pour chaque agent pour une distribution heuristique donnée sur les croyances dans le cas de PSMBDP.

Trois types d'approches ont été utilisées pour résoudre ces problèmes :

- énumération explicite d'un ensemble de solutions (MBDP, IMBDP) ;
- recherche heuristique *branch-and-bound* (PBIP, PSMBDP glouton par recherche heuristique) ;
- méta-heuristiques :
  - approximation par construction incrémentale gloutonne (PSMBDP glouton) ;
  - recherche locale par équilibre de Nash (MBDP/NE).

D'autres méta-heuristiques pourraient être essayées pour résoudre ce type de problème afin de chercher des solutions convenables quand l'espace de recherche devient trop grand ou que le critère

de sélection est trop complexe et ne permet pas un élagage efficace. En particulier, les approches à base d'entropie croisée ont été utilisées avec succès dans de nombreuses autres approches de planification dans l'incertain et pourraient peut-être fournir des solutions approchées de bonne qualité.

8.3.2 Heuristique

Les résultats de PSMBDP dépendent fortement de l'heuristique employée. Un problème est donc de savoir comment déterminer une bonne heuristique pour un problème donné. En particulier, le choix de l'heuristique peut être un moyen d'introduire une connaissance experte sur le problème. Des formes paramétriques de distributions sur les croyances pourraient être employées à cet effet (distribution de Dirichlet, Dirichlet généralisée, distribution logistique normale) et le problème est alors de savoir comment fixer leurs paramètres. En particulier, une distribution heuristique fixée *a priori* pourrait être utilisée pour introduire une connaissance à priori sur le problème ou la politique.

8.3.3 Autres politiques

Nous nous sommes intéressés à la planification en horizon fini avec des politiques représentées par des arbres de politiques. Il serait intéressant de généraliser le principe de PSMBDP à d'autres formes de politiques, en particulier les politiques en horizon infini et les politiques paramétrées.

8.4 Conclusion

Nous avons présenté une nouvelle approche pour la planification approchée qui choisit à chaque instant un nombre limité d'arbres de politiques par agent. Contrairement aux approches de l'état de l'art qui sélectionnent chaque arbre de politique joint de manière indépendante (les approches à base de point), nous considérons ce problème comme un choix unique de l'ensemble des arbres de politiques locaux associés à un critère d'optimisation explicite. Nous avançons que cette méthode est plus intéressante que la formulation classique de l'état de l'art : elle permet d'utiliser plus d'information heuristique (plus de points) que les approches à base de points et elle permet de prendre en compte la complémentarité entre les différents arbres de politique. Dans la pratique, l'approche permet de trouver des politiques de meilleure qualité que les approches à base de points pour un même nombre d'arbres de politiques.

Le principe central de notre méthode est de formuler le choix d'un nombre borné d'arbres de politiques comme un problème d'optimisation combinatoire. Ce principe pourrait être employé pour construire d'autres méthodes en particulier en modifiant la procédure utilisée pour résoudre ce problème d'optimisation combinatoire. Le choix d'autres méta-heuristiques pour résoudre ce problème pourrait conduire à des solutions de meilleur qualité, à réduire les temps de calcul ou à traîter des problèmes de taille plus importante.

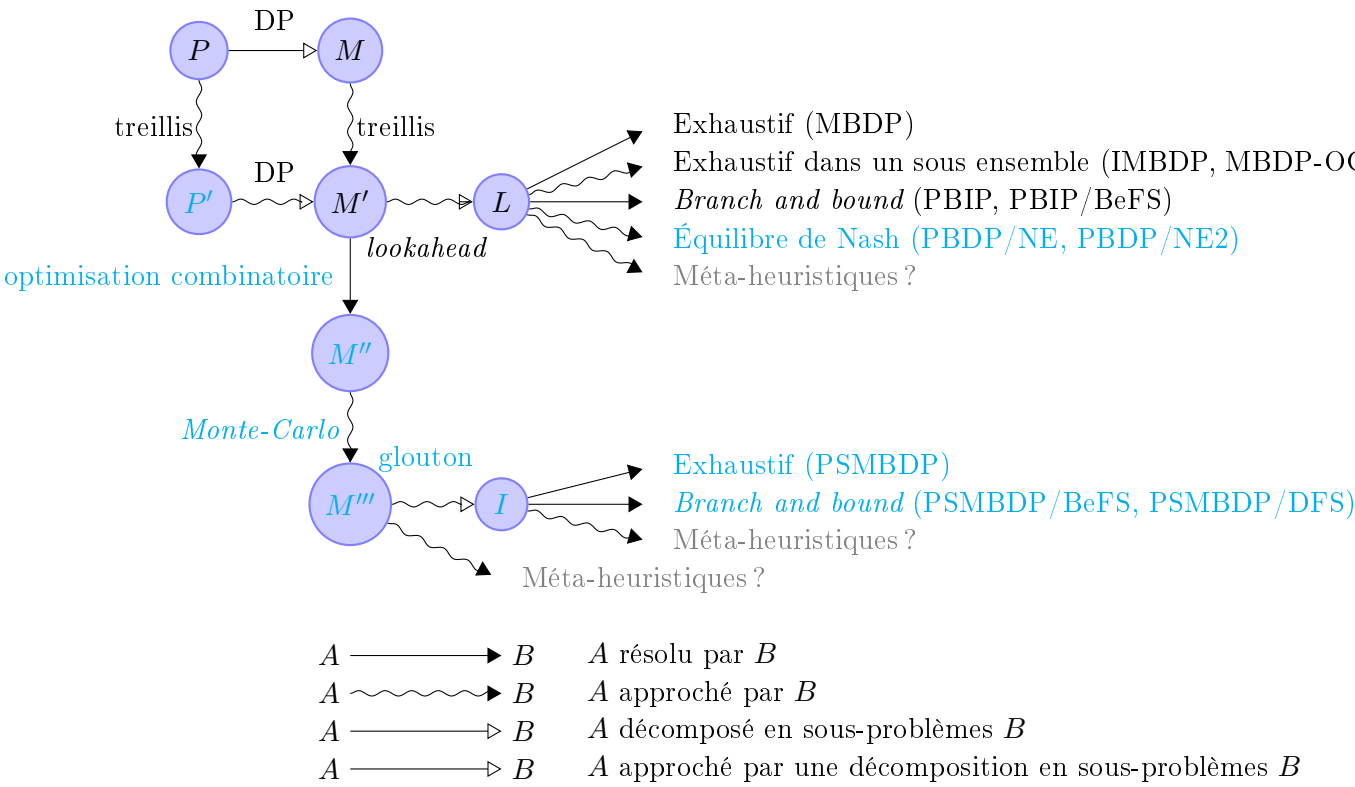


FIGURE 8.1 – Synthèse des approches de planification bornée

Bibliographie

[Amato *et al.*, 2009] Amato, C., Dibangoye, J. S. et Zilberstein, S. (2009). Incremental Policy Generation for Finite-Horizon Dec-POMDPs. *In Proceedings of the 19th International Conference on Automated Planning and Scheduling*, pages 2–9.

[Amato et Zilberstein, 2009] Amato, C. et Zilberstein, S. (2009). Achieving goals in decentralized pomdps. *In In Proc. of the Eighth Int. Joint Conf. on Autonomous Agents and Multiagent Systems*.

[Bagnell *et al.*, 2003] Bagnell, J. D., Kakade, S., Ng, A. et Schneider, J. (2003). Policy Search by Dynamic Programming. *In Neural Information Processing Systems*, volume 16. MIT Press.

[Bellifemine *et al.*, 2001] Bellifemine, F., Poggi, A. et Rimassa, G. (2001). Developing multi-agent systems with jade. *In* Castelfranchi, C. et Lespérance, Y., éditeurs : *Intelligent Agents VII Agent Theories Architectures and Languages*, volume 1986 de *Lecture Notes in Computer Science*, pages 42–47. Springer Berlin / Heidelberg.

[Bellman, 1957] Bellman, E. R. (1957). *Dynamic Programming*. Princeton University Press.

[Bellman, 1953] Bellman, R. E. (1953). The Theory of Dynamic Programming.

[Bernstein, 2005] Bernstein, D. S. (2005). Bounded Policy Iteration for Decentralized POMDPs. *In Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 1287–1292.

[Bernstein *et al.*, 2000] Bernstein, D. S., Zilberstein, S. et Immerman, N. (2000). The Complexity of Decentralized Control of Markov Decision Processes. *In Mathematics of Operations Research*, page 2002.

[Bertsekas et Tsitsiklis, 2006] Bertsekas, D. et Tsitsiklis, J. (2006). Neuro-Dynamic Programming : An Overview.

[Bertsekas, 1987] Bertsekas, D. P. (1987). *Dynamic Programming : Deterministic and Stochastic Models*. Rentice-Hall.

[Brooks, 1991] Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence*.

[Brooks et Stein, 1994] Brooks, R. A. et Stein, L. A. (1994). Building Brains for Bodies. *Autonomous Robots*, 1:7–25.

[Carlin et Zilberstein, 2008a] Carlin, A. et Zilberstein, S. (2008a). POMDP and DEC-POMDP Point-Based Observation Aggregation.

[Carlin *et al.*, 2010] Carlin, A. S., Schurr, N. et Marecki, J. (2010). ALARMS : Alerting an Reasoning Management System for Next Generation aircraft hazards. *In Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*.

[Carlin et Zilberstein, 2008b] Carlin, A. S. et Zilberstein, S. (2008b). Value-based observation compression for DEC-POMDPs. *In Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 501–508, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

[Cassandra *et al.*, 1997] Cassandra, A., Littman, M. L. et Zhang, N. L. (1997). Incremental pruning : A simple, fast, exact method for partially observable markov decision processes. *In Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 54–61. Morgan Kaufmann Publishers.

[Cassandra, 2008] Cassandra, A. R. (2008). A survey of pomdp applications.

[Cheng, 1988] Cheng, H.-T. (1988). *Algorithms for Partially Observable Markov Decision Processes*. Thèse de doctorat, University of British Columbia.

[Colorni *et al.*, 1991] Colorni, A., Dorigo, M. et Maniezzo, V. (1991). Distributed optimization by ant colonies. *In European Conference on Artificial Life*, pages 134–142.

[Corona et Charpillet, 2009] Corona, G. et Charpillet, F. (2009). Programmation dynamique à mémoire bornée avec distribution sur les croyances pour les DEC-POMDP. *Journées Francophones Planification Décision Apprentissage*.

[Corona et Charpillet, 2010a] Corona, G. et Charpillet, F. (2010a). Distribution over Beliefs for Dec-POMDP planning. *In Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*.

[Corona et Charpillet, 2010b] Corona, G. et Charpillet, F. (2010b). Distribution sur les croyances pour la planification de Dec-POMDP. *Revue d'Intelligence Artificielle*, 24(4):525–544.

[De Boer *et al.*, 2002] De Boer, P. T., Kroese, D. P., Mannor, S. et Rubinstein, R. (2002). A Tutorial on the Cross-Entropy Method. *Annals of Operations Research*, 134.

[Dibangoye *et al.*, 2008] Dibangoye, J. S., Mouaddib, A.-I. et Chaib-draa, B. (2008). Recherche incrémentale à base de points pour la résolution des DEC-POMDPs. *In Actes des quinzièmes JFSMA*, Brest, France.

[Dibangoye *et al.*, 2009] Dibangoye, J. S., Mouaddib, A.-I. et Chaib-draa, B. (2009). Point-based incremental pruning heuristic for solving finite-horizon dec-pomdps. *In Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, pages 569–576, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

[Dorigo et Caro, 1999] Dorigo, M. et Caro, G. D. (1999). The ant colony optimization meta-heuristic.

[Doucet *et al.*, 2001] Doucet, A., De Freitas, N. et Gordon, N., éditeurs (2001). *Sequential Monte Carlo methods in practice*. Springer.

[Dutch et Scherrer, 2008] Dutch, A. et Scherrer, B. (2008). Processus Décisionnels de Markov partiellement observables. *In Processus décisionnels de Markov en intelligence artificielle*. Hermes Science Publications / Lavoisier.

[Dutech *et al.*, 2008] Dutech, A., Scherrer, B. et Thiéry, C. (2008). La carotte et le bâton ... et tetris. *Interstices*.

[Feng et Zilberstein, 2004] Feng, Z. et Zilberstein, S. (2004). Region-based incremental pruning for pomdps. *In Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence (UAI2004*, pages 146–153.

[Garcia, 2008] Garcia, F. (2008). Processus décisionnels de markov. *In Processus décisionnels de Markov en intelligence artificielle*. Hermes Science Publications / Lavoisier.

[Geist, 2009] Geist, M. (2009). *Optimisation des chaînes de production dans l'industrie sidérurgique : une approche statistique de l'apprentissage par renforcement*. Phd thesis in mathematics, Université Paul Verlaine de Metz (en collaboration avec Supélec, ArcelorMittal et l'INRIA).

[Gordon, 1999] Gordon, G. J. (1999). Approximate solutions to markov decision processes. Rapport technique.

[Goto *et al.*, 2001] Goto, J. H., Lewis, M. E. et Puterman, M. L. (2001). Coffee, tea, or ... ? : A markov decision process model for airline meal provisioning.

[Hansen *et al.*, 2004] Hansen, E., Bernstein, D. et Zilberstein, S. (2004). Dynamic programming for partially observable stochastic games. *In Proceedings of the Nineteenth National Conference on Artificial Intelligence*, pages 709–715. AAAI Press.

[Hauskrecht, 1997] Hauskrecht, M. (1997). Planning and control in stochastic domains with imperfect information. Rapport technique, Massachusetts Institute of Technology.

[Kaelbling *et al.*, 1995] Kaelbling, L. P., Littman, M. L. et Cassandra, A. R. (1995). Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, 101:99–134.

[Kakade, 2003] Kakade, S. M. (2003). *On the Sample complexity of Reinforcement Learning*. Thèse de doctorat, University College London.

[Kumar et Zilberstein, 2010] Kumar, A. et Zilberstein, S. (2010). Point-based backup for decentralized pomdps : Complexity and new algorithms. *In Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems*, pages 1315–1322, Toronto, Canada.

[Littman, 1994] Littman, M. L. (1994). The witness algorithm : Solving partially observable markov decision processes. Rapport technique.

[Littman *et al.*, 1995] Littman, M. L., Dean, T. L. et Kaelbling, L. P. (1995). On the complexity of solving markov decision problems. *In In Proc. of the Eleventh International Conference on Uncertainty in Artificial Intelligence*, pages 394–402.

[Lovejoy, 1991] Lovejoy, W. S. (1991). Computationally feasible bounds for partially observed markov decision processes. *In Operations Research archive*, pages 162–175.

[Mundhenk *et al.*, 2000] Mundhenk, M., Goldsmith, J., Lusena, C. et Allender, E. (2000). Complexity of finite-horizon markov decision process problems. *J. ACM*, 47:681–720.

[Murphy, 2002] Murphy, K. P. (2002). *Dynamic Bayesian Networks : Representation, Inference and Learning*. Thèse de doctorat, University of California.

[Nair *et al.*, 2003] Nair, R., Tambe, M., Yokoo, M., Pynadath, D. et Marsella, S. (2003). Taming Decentralized POMDPs : Towards Efficient Policy Computation for Multiagent Settings. *In Proceedings of the Twentieth International Joint Conference on Artificial Intelligences (IJCAI-03)*, pages 705–711. IJCAI.

[Oliehoek *et al.*, 2008] Oliehoek, F. A., Spaan, M. T. J. et Vlassis, N. (2008). Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research*, 32(1):289–353.

[Ooi et Wornell, 1996] Ooi, J. et Wornell, G. (1996). Decentralized control of a multiple access broadcast channel : performance bounds. *In Proceedings of the 35th IEEE*, volume 1, pages 293–296.

[Pineau *et al.*, 2003] Pineau, J., Gordon, G. et Thrun, S. (2003). Point-based value iteration : An anytime algorithm for POMDPs. *In International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 1025 – 1032.

[Poon, 2001] Poon, K.-M. (2001). A fast heuristic algorithm for decision-theoretic planning. Mémoire de D.E.A., The Hong-Kong University of Science and Technology.

[Poslad *et al.*,] Poslad, S., Buckle, P. et Hadingham, R. [the fipaos agent platform : Open source for open standards.

[Puterman, 1994] Puterman, M. L. (1994). *Markov Decision Processes : Discrete Stochastic Dynamic Programming*. Wiley-Interscience.

[Rao et Georgeff, 1991] Rao, A. S. et Georgeff, M. P. (1991). Modeling rational agents within a BDI-architecture. *In* Allen, J., Fikes, R. et Sandewall, E., éditeurs : *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 473–484. Morgan Kaufmann publishers Inc. : San Mateo, CA, USA.

[Remon *et al.*, 2010] Remon, S., Conrad, J., Gomes, C. et Selman, B. (2010). Playing hames against nature : optimal policies for renewable resource allocation. *In Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*.

[Rummery et Niranjan, 1994] Rummery, G. A. et Niranjan, M. (1994). On-line q-learning using connectionist systems. Rapport technique.

[Seuken et Zilberstein, 2007a] Seuken, S. et Zilberstein, S. (2007a). Improved Memory-Bounded Dynamic Programming for Decentralized POMDPs. *In Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligences (UAI-07)*, Vancouver, BC, Canada.

[Seuken et Zilberstein, 2007b] Seuken, S. et Zilberstein, S. (2007b). Memory-Bounded Dynamic Programming for DEC-POMDPs. *In Proceedings of the Twentieth International Joint Conference on Artificial Intelligences (IJCAI-07)*. IJCAI.

[Shani *et al.*, 2007] Shani, G., Brafman, R. I. et Shimony, S. E. (2007). Forward search value iteration for pomdps. *In Proceedings of the Twentieth International Joint Conference on Artificial Intelligences (IJCAI-07)*. IJCAI.

[Singh *et al.*, 1994] Singh, S., Jaakkola, T. et Jordan, M. (1994). Learning without state estimation in partially observable markovian decision processes. *In Proceedings of the Eleventh International Conference on Machine Learning*.

[Smallwood et Sondik, 1973] Smallwood, R. D. et Sondik, E. J. (1973). The optimal control of partially observable markov processes over a finite horizon. *Operations Research*, 21(8):1071–1088.

[Smith et Simmons, 2004] Smith, T. et Simmons, R. (2004). Heuristic search value iteration for pomdps.

[Sondik, 1971] Sondik, E. J. (1971). *The optimal control of partially observable Markov processes*. Thèse de doctorat, Stanford University.

[Spaan et Vlassis, 2005] Spaan, M. T. J. et Vlassis, N. (2005). Perseus : Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220.

[Sutton et Barto, 1998] Sutton, R. S. et Barto, A. G. (1998). *Reinforcement Learning : An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press.

[Szer et Charpillet, 2006a] Szer, D. et Charpillet, F. (2006a). Point-based dynamic programming for DEC-POMDPs. *In Proceedings of the Twenty-First AAAI National Conference on Artificial Intelligence*.

[Szer et Charpillet, 2006b] Szer, D. et Charpillet, F. (2006b). Point-based Dynamic Programming for DEC-POMDPs. *In Proceedings of the 21st National Conference on Artificial Intelligence.*

[Todorov, 2006] Todorov, E. (2006). Optimal control theory. *In* at al, D. K., éditeur : *Bayesian Brain : Probabilistic Approaches to Neural Coding*, chapitre 12, pages 269–298. MIT Press.

[Varakantham *et al.*, 2005] Varakantham, P., Maheswaran, R. et Tambe, M. (2005). Exploiting Belief Bounds : Practical POMDPs for Personal Assistant Agents. *In Proceedings og the 4th International Joint Conference on Autonomous Agents and Multiagent Systems.*

[Virin *et al.*, 2007] Virin, Y., Shani, G., Shimony, S. E. et Ronen, B. (2007). Scaling up : Solving POMDPs through Value Based Clustering. *In Proceedings of the Twenty-Second Conference on Artificial Intelligence*, pages 1910–1911.

[Watkins, 1989] Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards.* Thèse de doctorat, King's College of Cambridge, UK.

[Wu *et al.*, 2010a] Wu, F., Zilberstein, S. et Chen, X. (2010a). Point-Based Policy Generation for Decentralized POMDPs. *In Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems*, pages 1307–1314, Toronto, Canada.

[Wu *et al.*, 2010b] Wu, F., Zilberstein, S. et Chen, X. (2010b). Rollout Sampling Policy Iteration for Decentralized POMDPs. *In Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence.*

[Zhang et Liu, 1996] Zhang, N. L. et Liu, W. (1996). Planning in Stochastic Domains : Problem Characteristics and Approximations (Version II). Rapport technique.

Troisième partie

Annexes

Chapitre 9

Théorie des jeux

Sommaire

9.1 Jeu en forme normale

Un jeu en forme normale permet de représenter le problème stratégique d'un ensemble d'agents dont les actions peuvent éventuellement entrer en compétition. Chaque agent i cherche une stratégie π_i afin de maximiser un critère $u_i(\pi)$ qui lui est propre où $\pi = (\pi_1, \dots \pi_n)$ est la stratégie de l'ensemble des agents. Ainsi, quand l'agent i choisit sa stratégie π_i, il doit prévoir le comportement des autres agents. Chaque agent i a, de manière générale, son propre critère u_i.

Définition 9.1.1. Un jeu en forme normale est un tuple $\langle \mathcal{I}, (\mathcal{A}_i)_{i\in\mathcal{I}}, (u_i)_{i\in\mathcal{I}} \rangle$ où :
- $\mathcal{I}$ est l'ensemble des joueurs (agents) i ;
- $\mathcal{A}_i$ est l'ensemble des actions possibles de l'agent i ;
- u_i est la fonction qui a toute action jointe $a \in \mathcal{A} = \prod_{i\in\mathcal{I}} \mathcal{A}_i$ associe l'utilité $u_i(a)$ pour l'agent i.

9.2 Stratégie

Définition 9.2.1. Une stratégie locale π_i pour l'agent i est une distribution de probabilité sur ses actions $\mathcal{A}_i$:

$$\pi_i(a_i) = \Pr(a_i) \tag{9.1}$$

Une stratégie jointe π est un tuple de stratégies jointes, elle définie les probabilités des actions jointes :

$$\pi(a_1, \dots a_n) = \Pr(a_1, \dots a_n) = \prod_{i\in\mathcal{A}} \pi_i(a_i) \tag{9.2}$$

Une stratégie (locale ou jointe) est pure si elle concentrée en une seule action : $\exists a_i \in \mathcal{A}_i, \pi_i(a_i) = 1$ ou $\exists a \in \mathcal{A}, \pi(a) = 1$.

Définition 9.2.2. L'utilité d'une u_i d'une stratégie jointe π pour l'agent i est définie comme l'espérance de l'utilité :

$$u_i(\pi) = \mathop{\mathrm{E}}_{a\sim\pi} u_i(a) \tag{9.3}$$

9.3 Équilibre de Nash

La théorie des jeux cherche à déterminer une stratégie rationnelle pour chaque agent : une politique jointe $\pi = (\pi_1, \ldots, \pi_n)$ est rationnelle si aucun agent i n'a intérêt à prendre une autre politique π'_i. Ceci est formalisé par le concept d'équilibre de Nash.

Définition 9.3.1. Un équilibre de Nash est une stratégie jointe $\pi = (\pi_1, \ldots, \pi_n)$ qui est un maximum local :

$$\forall i \in \mathcal{I}, \forall \pi'_i, u_i(\pi) \geq u_i(\pi_1, \ldots, \pi'_i, \ldots, \pi_n) \tag{9.4}$$

On peut trouver un équilibre de Nash en partant d'une stratégie jointe quelconque en remplaçant la stratégie π_i d'un agent i par la la stratégie optimale de cet agent étant donné les stratégies π_{-i} des autres agents jusqu'à convergence vers un équilibre local (algorithme 9.1).

Algorithme 9.1 : Chercher un équilibre de Nash dans un jeu en forme normale

Données : π_i stratégie quelconque pour chaque agent $i \in \mathcal{I}$
Résultat : π_i stratégie pour chaque agent $i \in \mathcal{I}$, en équilibre de Nash

```
répéter
    pour chaque i ∈ I faire
        π_i ← arg max_{π'_i ∈ Π_i} u_i(⟨π'_i, π_{-i}⟩)
jusqu'à convergence
```

9.4 Équilibre Pareto optimal

Certains équilibres de Nash sont meilleurs que d'autres pour tous les agents et devraient donc être pris de manière préférentielle. Ce sont les équilibre Pareto optimaux. Tout jeu fini, comporte au moins un équilibre de Nash Pareto optimal. Il peut exister plusieurs équilibres Pareto optimaux mais il n'existe pas nécessairement d'équilibre de Nash Pareto optimal qui soit aussi bon que tous les autres pour tous les agents.

Définition 9.4.1. Un équilibre de Nash est Pareto optimal quand il n'y a pas d'équilibre de Nash π' tel que :

$$\forall i \in \mathcal{I}, u_i(\pi') \geq u_i(\pi) \tag{9.5}$$

$$\exists i \in \mathcal{I}, u_i(\pi') > u_i(\pi) \tag{9.6}$$

9.5 Jeux coopératifs

Dans le cas des jeux coopératifs, tous les agents partagent la même utilité que nous noterons u par commodité :

$$\forall (i,j) \in \mathcal{I}^2, u_i = u_j = u \tag{9.7}$$

Dans le cas des jeux coopératifs, les équilibres Pareto optimaux sont équivalents.