Chapitre 3

Processus décisionnels de Markov partiellement observables

Si les systèmes rencontrés dans la réalité peuvent souvent être modélisés comme des processus markoviens, l'agent chargé de les contrôler ou d'apprendre à les contrôler a rarement accès à une information suffisante pour connaître l'état du processus. L'agent observe le processus mais ne le connaît pas. Le formalisme des Processus Décisionnels de Markov Partiellement Observable (ou POMDP) permet justement de modéliser ce genre de situation où un agent n'a accès qu'à des informations souvent partielles sur le processus à contrôler.

EXEMPLE.– Dans l'exemple de l'entretien de la voiture des chapitres précédents (voir les paragraphes 1.1 et 2.1), nous avons en fait implicitement supposé que nous connaissions l'état de la voiture. Bien souvent ce n'est pas le cas, car personne ne va constamment mesurer le taux d'étanchéité du joint de culasse ou l'usure des plaquettes de frein. En nous contentant d'un rapide coup d'oeil, nous n'avons pas accès à l'état de la voiture au sens formel mais simplement à une observation, souvent incomplète et imprécise, de cet état. Le formalisme des POMDP permet de modéliser ce genre de problèmes et ses solutions indiquent comment choisir optimalement les actions à réaliser en ne s'appuyant que sur ces observations imparfaites. Pourtant, il faut encore une fois faire l'hypothèse que nous connaissons la dynamique du processus, c'est-à-dire les conséquences des actions (les transitions), leur coût (les récompenses) mais aussi la probabilité avec lesquelles un état donné de la voiture produit telle ou telle observation (la fonction d'observation).

Chapitre rédigé par Alain DUTECH et Bruno SCHERRER.

Cette problématique liée au réalisme des problèmes pouvant être abordés par les méthodes de la programmation dynamique est apparue très tôt. Dès 1965, Aström pose les bases théoriques des POMDP et de leur résolution en utilisant les états de croyance [AST 65]. Il faut attendre les travaux de Smallwood et Sondik pour voir les premiers algorithmes de résolution qui sont très peu efficaces [SMA 73, SON 71, SON 78]. On peut dire que c'est avec l'algorithme WITNESS [CAS 94] puis ITERATIVE PRUNNING [ZAN 96] que la recherche dans le domaine a vraiment pris son essor et, depuis, de nombreux algorithmes exacts ou approchés ont vu le jour.

Ce chapitre est consacré aux POMDP. Dans un premier temps (section 3.1), nous détaillons et explicitons le formalisme en introduisant la notion d'état d'information. Puis, nous montrons que l'application directe des méthodes vues aux chapitres précédents (ce qui revient à essayer de contrôler un POMDP comme si c'était un MDP) est en général vouée à l'échec (section 3.2). Il existe cependant des principes généraux permettant des méthodes exactes de résolution des POMDP (section 3.3) ce qui se traduit par des algorithmes de type « itération sur les valeurs » (3.4) ou « itération sur les politiques » (3.5).

3.1. Définition formelle des POMDP

3.1.1. *Définition d'un POMDP*

Un processus décisionnel de Markov partiellement observable est un MDP dans lequel l'agent ne connaît pas l'état réel du processus : l'agent n'a accès qu'à une observation partielle de cet état [CAS 98]. On le définit, de manière analogue à un MDP (voir paragraphe 1.2.1), par : $(\mathcal{S}, \mathcal{A}, \Omega, T, p, O, r, b_0)$ où :

– $\mathcal{S}$ est l'espace d'état ;

– $\mathcal{A}$ est l'espace des actions ;

– Ω est l'espace des observations ;

– T est l'axe temporel ;

– $p()$ sont les probabilités de transition entre états ;

– $O()$ sont les probabilités d'observation sur les états ;

– $r()$ est une fonction de récompense sur les transitions d'états ;

– b_0 est la distribution de probabilité initiale sur les états.

Les seules nouveautés par rapport à un MDP sont donc la distribution initiale des états du processus b_0, l'espace des observations Ω et la fonction d'observation associée $O()$.

Le principe général, illustré figure 3.1 est que, à chaque instant t de T, l'agent ne connaît pas l'état courant $s_t \in \mathcal{S}$ mais peut seulement le percevoir partiellement

sous forme d'une observation $o_t \in \Omega$. Cette observation est donnée par la fonction d'observation O. Quand il applique une action $a_t \in \mathcal{A}$ sur le processus, cela modifie alors aléatoirement l'état du processus selon $p()$ pour l'amener dans l'état s_{t+1} alors que l'agent ne perçoit que l'observation o_{t+1} qui dépend de la fonction $O()$. Enfin, comme dans un MDP, l'agent reçoit une récompense $r \in \mathbb{R}$.

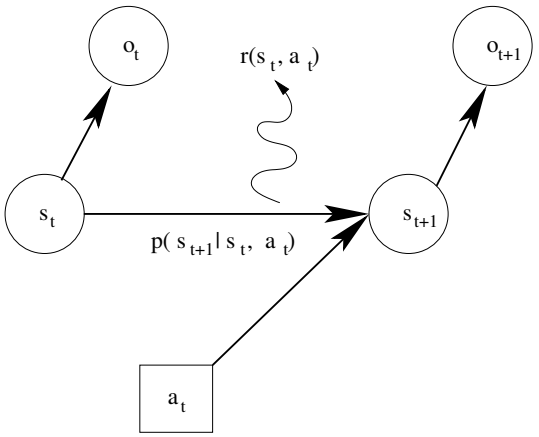

Figure 3.1. *Vue générale d'un* POMDP *sous forme d'un diagramme d'influence*

Tout comme $\mathcal{S}$ et $\mathcal{A}$, l'espace des observations Ω est supposé fini. Si, dans le cas le plus général, l'observation de l'état courant peut dépendre de la dernière transition en date du processus, le cas le plus classique suppose que cette observation ne dépend que de l'état courant. A un instant t, la probabilité pour l'agent d'observer o dans l'état s est donnée par $\Pr(o|s) = O_t(o|s)$. On impose que $\forall t$, $\forall s$, $\sum_o O_t(o|s) = 1$.

Il est à noter que, même quand la fonction d'observation $O()$ est déterministe, c'est-à-dire qu'à chaque état est associé une et une seule observation, l'agent peut ne pas connaître l'état : deux états peuvent être associés à la même observation. C'est d'ailleurs quand il y a ambiguïté sur l'état qu'il est intéressant de parler de POMDP, car sinon le problème que l'agent doit résoudre possède la structure d'un MDP classique.

C'est donc dans le cadre plus large où les ambiguïtés perceptives peuvent se produire que nous allons nous placer dans la suite de ce chapitre. Nous ferons aussi l'hypothèse que le processus est stationnaire ($O()$ et $p()$ ne dépendent pas du temps).

3.1.2. *Critères de performance*

Tout comme dans le cadre des MDP, la résolution d'un POMDP se fait en cherchant à maximiser un critère de performance donné. Ces critères sont les mêmes que ceux utilisés pour les MDP (voir paragraphe 1.2.3).

Si les critères existent toujours, il n'en est pas forcément de même des fonctions de valeur que nous avons détaillées pour les MDP. En effet, les fonctions de valeur existent bien quand elles sont définies sur l'espace $\mathcal{S}$ des états, or cet espace n'est pas accessible à l'agent. Il est donc temps de parler des états d'informations et donc des espaces associés qui, comme nous le verrons, conditionnent l'existence de ces fonctions de valeur et donc la résolution des POMDP.

3.1.3. *Etat d'information*

L'agent, qui n'a pas accès à l'état, doit choisir ses actions en fonction des informations qui lui sont disponibles, c'est pourquoi nous parlerons d'*état d'information*[1] pour parler des espaces de définition des politiques (ou des fonctions de valeur) permettant effectivement à l'agent de contrôler le POMDP de façon optimale.

3.1.3.1. *Définition*

Appelons I_t l'ensemble des informations accessibles à un agent à un instant t. Après avoir effectué l'action a_t, l'agent perçoit le processus sous la forme de l'observation o_{t+1}, ce qui lui permet d'augmenter l'ensemble des informations dont il dispose, par exemple par :

$$I_{t+1} = \tau(I_0, I_1, \ldots, I_t, o_{t+1}, a_t). \tag{3.1}$$

DÉFINITION 3.1.– *Etat d'information d'un* POMDP
I forme un ensemble d'états d'information si la séquence $(I)_t$ définit une chaîne de Markov contrôlée (par les actions), c'est-à-dire si :

$$\forall t,\ \Pr(I_{t+1}|I_0, I_1, \ldots, I_t, o_{t+1}, a_t) = \Pr(I_{t+1}|I_t, o_{t+1}, a_t).$$

Attention, si certains espaces de définition des politiques peuvent paraître naturels (comme par exemple l'ensemble Ω des observations), dans le cas général ces espaces ne sont pas des espaces d'information. La section 3.2 revient plus en détails sur ce problème crucial des POMDP.

On peut alors définir une fonction de transition pour ce processus :

$$\phi(I, a, o, I') \quad = \quad \Pr(I_t = I'|I_{t-1} = I, a, o). \tag{3.2}$$

1. *Information state* en anglais.

La récompense associée à ces états d'information s'écrit alors :

$$\rho(I,a) \quad = \quad \sum_{s\in\mathcal{S}} r(s,a)\Pr(s|I). \tag{3.3}$$

En principe, un POMDP peut toujours être transformé en un MDP défini sur un ensemble d'états d'information. Le problème est de trouver une représentation intéressante et utilisable en pratique de ces états d'information.

3.1.3.2. *Etat d'information complet*

La façon la plus simple et la plus naïve de représenter les états d'information est d'utiliser toute l'information disponible sur le processus depuis le début (depuis l'instant $t = 0$). Dans ce cas, l'état d'information est constitué des historiques complets des observations et actions passées.

Plus formellement, l'*état d'information complet* (noté I_t^C) au temps t est constitué de :

– la distribution de probabilité initiale sur les états b_0,

– l'historique des observations passées et de l'observation présente $(o_0, \cdots, o_t)$,

– l'historique des actions passées $(a_0, \cdots, a_{t-1})$.

Il est clair que les états d'information complets satisfont à la propriété de Markov énoncée précédemment. Le principal problème de cette représentation est que la taille d'un état d'information grossit à chaque pas de temps. Cette représentation n'est donc pas facilement manipulable, surtout quand on considère des processus à horizon infini.

3.1.3.3. *Statistiques suffisantes*

On peut représenter plus efficacement les états d'information en utilisant des statistiques suffisantes (ou exhaustives) vis-à-vis du contrôle du processus [BER 95].

DÉFINITION 3.2.– *Information suffisante*
Une séquence d'états d'information $(I)_t$ définit un processus d'information suffisant quand on a :

– $I_t = \tau(I_{t-1}, o_t, a_{t-1})$,

– $\Pr(s_t|I_t) = \Pr(s_t|I_t^C)$,

– $\Pr(o_t|I_{t-1}, a_{t-1}) = \Pr(o_t|I_{t-1}^C, a_{t-1})$,

où I_{t-1}^C et I_t^C sont des états d'information complets. On peut aussi voir un état d'information suffisant comme un état qui permet de prédire le comportement du processus,

ce qui permet ensuite de le contrôler. Ces méta-informations sur le processus satisfont la propriété de Markov et préservent les informations contenues dans les états d'information complets qui sont suffisantes pour contrôler le processus. L'avantage principal des états d'information suffisants est qu'ils peuvent se représenter de façon plus compacte que les états d'information complets. Leur taille n'augmente pas avec le temps, par exemple. Le désavantage vient du fait que leur mise à jour est un peu plus complexe et que leur espace de définition peut être continu.

3.1.3.4. *Etats de croyance*

Des états d'information suffisants couramment utilisés sont les états de croyance[2].

DÉFINITION 3.3.– *Etat de croyance*
Un état de croyance b_t à l'instant t est défini par :

$$b_t(s) = \Pr(s_t = s | I_t^C).$$

Un état de croyance est une distribution de probabilité sur l'ensemble des états. On note $\mathcal{B}$ l'ensemble des états de croyance, c'est l'espace de toutes les distributions de probabilité sur $\mathcal{S}$.

Comme nous l'avons dit, un état d'information suffisant doit être mis à jour après chaque action. Ainsi, si b est un état de croyance, nous pouvons exprimer l'état de croyance b_o^a après une transition du processus, c'est-à-dire après que l'agent a effectué une action a et reçu une observation o :

$$\begin{aligned}
b_o^a(s') &= \Pr(s'|b,a,o) \\
&= \frac{\Pr(s',b,a,o)}{\Pr(b,a,o)} \\
&= \frac{\Pr(o|s',b,a)\Pr(s'|b,a)\Pr(b,a)}{\Pr(o|b,a)\Pr(b,a)} \\
&= \frac{\Pr(o|s',b,a)\Pr(s'|b,a)}{\sum_{s''\in\mathcal{S}}\Pr(s''|b,a)\Pr(o|s'',a)}
\end{aligned}$$

$$\begin{aligned}
&= \frac{O(o|s')\sum_{s\in\mathcal{S}}\Pr(s'|a,s)\Pr(s)}{\sum_{s\in\mathcal{S}}\sum_{s''\in\mathcal{S}}O(o|s'')\Pr(s''|a,s)\Pr(s)} \\
&= \frac{O(o|s')\sum_{s\in\mathcal{S}}p(s'|s,a)b(s)}{\sum_{s\in\mathcal{S}}\sum_{s''\in\mathcal{S}}O(o|s'')p(s''|s,a)b(s)}.
\end{aligned}$$

2. Pour l'anglais *belief states*.

On obtient alors :

$$b_o^a(s') = \frac{O(o|s') \sum_{s \in \mathcal{S}} p(s'|s,a) b(s)}{\sum_{s \in \mathcal{S}} \sum_{s'' \in \mathcal{S}} O(o|s'') p(s''|s,a) b(s)}. \tag{3.4}$$

Puisque les états de croyance forment un processus markovien, nous pouvons expliciter ce processus, notamment au niveau des fonctions de transition et de récompense. Pour cela, nous allons définir la probabilité conditionnelle d'une observation par :

$$\begin{aligned} \omega(b,a,o) &= \Pr(o|b,a) \\ &= \sum_{s \in \mathcal{S}} \sum_{s'' \in \mathcal{S}} O(o|s'',a) p(s''|s,a) b(s). \end{aligned}$$

On peut alors calculer la fonction de transition entre deux états de croyance. Etant donné un état de croyance b et une action a, chaque observation o peut donner lieu à un état de croyance successeur b_o^a différent. La probabilité de transiter vers un état de croyance b' donné est alors égal à la somme de toutes les probabilités de transiter vers des b_o^a égaux à b'. Ce qui s'écrit :

$$\phi(b,a,b') = \Pr(b'|b,a) = \sum_{o \in \Omega} \omega(b,a,o) \delta(b', b_o^a),$$

$$\text{avec} \qquad \delta(x,y) = \begin{cases} 0 & \text{si } x = y, \\ 1 & \text{sinon.} \end{cases}$$

Quant à la récompense associée à un état de croyance, on l'obtient aisément par :

$$\rho(b,a) = \sum_{s \in \mathcal{S}} r(s,a) b(s).$$

Le processus de Markov ainsi défini sur les états de croyance est difficile à résoudre car il est défini sur un espace d'état continu, celui des distributions de probabilité sur l'espace d'état $\mathcal{S}$. De plus, sauf pour certains POMDP bien particuliers qu'on appelle

« transitoires »[3] (voir paragraphe 3.3.2.2), la suite des états de croyance générée par une politique donnée comporte un nombre infini d'états de croyance différents. Néanmoins, nous verrons plus loin (paragraphe 3.3) qu'il est parfois possible d'utiliser les propriétés particulières des fonctions de valeurs de ce processus pour proposer des algorithmes exacts ou quasi exacts.

3.1.4. *Politique*

Les espaces sur lesquels il est possible de définir une politique sont plus divers que dans le cas des MDP. Il est inutile de calculer des politiques définies sur les états ou les historiques d'états puisque ces derniers sont inaccessibles à l'agent. Il est envisageable de définir des politiques sur l'espace des observations ou l'historique des observations. Comme nous le verrons à la section 3.2, il n'est pas possible de garantir l'optimalité de telles politiques mais elles peuvent néanmoins constituer une option viable dans certains cas.

Il est plus intéressant de travailler sur les espaces d'information que nous venons d'aborder dans la partie précédente. Dans l'absolu, il existe une multitude de types d'états d'information possibles, mais qui peuvent tous plus ou moins se ramener à des historiques d'action et d'observation sur le processus. C'est notamment le cas des états d'information complets. Nous allons nous intéresser à la manière de représenter les politiques dans ce cas. Cette démarche peut aisément s'adapter à d'autres politiques définies sur d'autres espaces (comme par exemple des historiques d'observation).

Pour un état d'information complet initial I, la politique optimale d'horizon N peut être représentée par un arbre, elle correspond à une sorte de plan conditionnel. En effet, pour un état d'information I, la politique optimale d'horizon 1 est l'action optimale associée à cet état, notons la $a^1(I)$. Intéressons-nous maintenant à la politique d'horizon 2 à partir de I, elle commence par une action a. Après avoir choisi cette action, l'agent obtient une observation o et se trouve donc dans le nouvel état d'information I_o^a. S'il veut alors agir optimalement, il choisira alors $a^1(I_o^a)$. On voit que la deuxième action de l'agent dépend de l'observation o perçue, c'est une sorte de plan conditionnel.

Ainsi, une politique d'horizon N pour un état d'information I peut se représenter sous la forme d'un arbre comme celui de la figure 3.2. L'exécution d'un plan conditionnel ou d'une politique d'ordre N à partir d'un état d'information I consiste essentiellement à traverser cet arbre en partant de la racine en suivant, tout à tour, des nœuds « action » et des branches « observation ». Quand on arrive à un nœud, on exécute l'action associée à ce nœud, puis en fonction de l'observation reçue, on navigue vers le nœud suivant le long de la branche associée à l'observation.

---

3. De l'anglais « *transient* ».

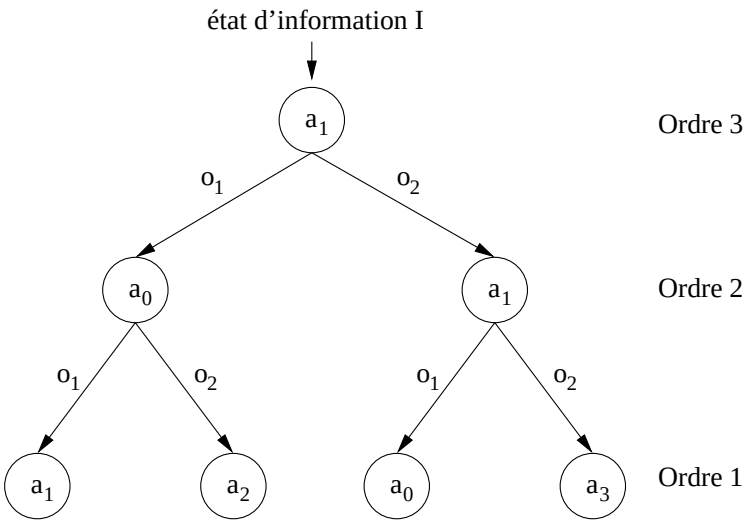

Figure 3.2. Plan conditionnel. *Cet arbre représente une politique d'horizon* 3 *pour un état d'information* I *donné. C'est aussi une sorte de plan conditionnel. A chaque étape du plan, un nœud contient l'action à effectuer et, en fonction de l'observation reçue, on se déplace vers un nœud d'ordre inférieur en suivant la branche appropriée.*

Les arbres de politiques permettent donc de représenter les politiques d'horizon fini. En revanche, la représentation sous forme d'arbre n'est pas pratique dans le cas des problèmes à horizon infini car les arbres grandissent de manière exponentielle. Une alternative, qui est utilisable dans les cas où les politiques sont *cycliques*, est de représenter les politiques en utilisant des automates à états finis qui permettent de représenter des plans conditionnels qui bouclent. Un exemple d'automate est décrit à la figure 3.3 qui s'exécute comme précédemment : on applique l'action dans le nœud actuel avant de transiter vers le nœud suivant en fonction de l'observation reçue. Le nœud de départ pour appliquer la politique dépend de l'état d'information de départ.

3.1.5. *Fonctions de valeur*

De la même manière que pour les MDP, on peut définir une fonction de valeur pour les POMDP définis sur les états d'information. Nous nous plaçons ici, à titre d'exemple, dans le cadre du critère actualisé, mais les développements faits sur les différents critères pour les MDP en section 1.5 sont utilisables de la même manière.

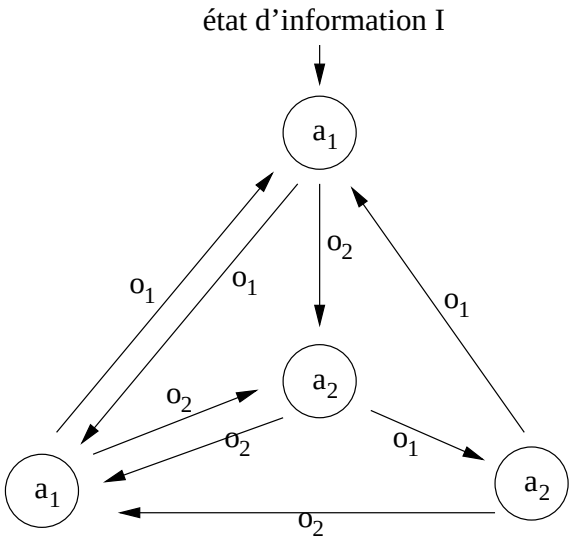

Figure 3.3. Automate à états fini pour politique cyclique infinie. *L'automate permet de représenter des politiques infinies qui présentent des cycles. Comme dans le cas des arbres, on exécute une politique en appliquant l'action dans le nœud actuel avant de transiter vers le nœud suivant en fonction de l'observation reçue.*

Ainsi, l'opérateur d'itération de la fonction de valeur pour une politique π_t définie sur l'espace des états d'information $\mathcal{I}$ s'écrit :

$$V_n^{\pi_t}(I) \quad = \quad \rho(I, \pi_t(I)) + \gamma \sum_{I' \in \mathcal{I}} \phi(I, \pi_t(I), I') V_{n-1}^{\pi_t}(I'). \tag{3.5}$$

On peut, de la même manière, écrire l'équation de Bellman définissant la fonction de valeur optimale du MDP défini sur les états d'information :

$$\mathcal{V}^*(I) \quad = \quad \max_{a \in \mathcal{A}} \left[\rho(I, a) + \gamma \sum_{I' \in \mathcal{I}(I,a)} \phi(I, a, I') \mathcal{V}^*(I') \right], \tag{3.6}$$

où $\mathcal{I}(I, a)$ est l'ensemble des successeurs de l'état d'information I. Nous avons vu précédemment que ces successeurs se calculaient à partir de I par $I' = \tau(I, o, a)$,

on peut donc en déduire qu'il y a au maximum $|\Omega|$ successeurs à un état d'information et écrire cette équation en sommant cette fois-ci sur l'ensemble des observations possibles, ce qui donne :

$$\mathcal{V}^*(I) = \max_{a \in \mathcal{A}} \left[\rho(I,a) + \gamma \sum_{o \in \Omega} \Pr(o|I,a) \mathcal{V}^*(\tau(I,o,a)) \right]. \quad (3.7)$$

On peut aussi écrire cette équation en utilisant directement les paramètres du POMDP. On obtient alors :

$$\mathcal{V}^*(I) = \max_{a \in \mathcal{A}} \left[\sum_{s \in \mathcal{S}} r(s,a) \Pr(s|I) \right. \quad (3.8)$$

$$\left. + \gamma \sum_{s_i n \mathcal{S}} \sum_{s' \in \mathcal{S}} \sum_{o \in \Omega} \Pr(s|I) p(s'|s,a) O(o|s') \mathcal{V}^*(\tau(I,o,a)) \right]. \quad (3.9)$$

Comme dans le cas des MDP, l'opérateur associé à cette équation est une contraction pour la norme $\max$ et pour $0 \leq \gamma < 1$ et les mêmes démonstrations assurent l'existence et l'unicité de la solution (voir théorèmes 1.4, 1.5 et 1.6).

3.2. Problèmes non markoviens (information incomplète)

Comme nous l'avons signalé, pour un POMDP les observations ne sont pas des états d'information. Autrement dit, un agent ne peut pas contrôler de façon optimale un POMDP en ne se basant que sur l'observation courante du processus pour décider de son action. Dans cette section, nous allons étudier plus en détail les politiques définies seulement sur l'observation courante dans le but de mieux comprendre les différences entre les POMDP et les MDP. Cette compréhension permettra aussi de mieux aborder ensuite les problèmes liés à la résolution des POMDP.

3.2.1. *Politiques adaptées*

Dans cette partie, nous allons uniquement considérer des politiques π de la forme :

$$\pi : (\Omega \times T) \longrightarrow \Pi(\mathcal{A}),$$

où $\Pi(\mathcal{A})$ est l'ensemble des distributions de probabilité sur $\mathcal{A}$.

Si, pour essayer de résoudre un POMDP, on adapte les algorithmes définis pour les MDP en assimilant naïvement les observations à des états, on cherche en fait des politiques de la forme que nous venons de rappeler. Nous appelons cette famille de politique des *politiques adaptées*. Et nous allons voir qu'en fait, elles *ne sont pas adaptées* au contrôle optimal d'un POMDP, même si les résultats pratiques peuvent s'avérer satisfaisants.

3.2.2. *Critère pondéré*

3.2.2.1. *Politique adaptée stochastique*

Plaçons-nous dans le cas d'un critère actualisé. Il est alors facile de montrer que, contrairement aux MDP, il n'existe pas de politique adaptée déterministe optimale. L'exemple de la figure 3.4 permet de montrer facilement que :

PROPOSITION 3.1.– *Il existe des* POMDP *où la meilleure politique stochastique adaptée peut être arbitrairement meilleure que la meilleure politique déterministe*.

PREUVE.– La figure 3.4 montre un POMDP avec deux états (1a et 1b) et une seule observation (1). Il n'existe que deux politiques adaptées déterministes (soit tout le temps faire « A », soit tout le temps faire « B »). Au mieux, ces politiques amènent une récompense de $+R$ suivie d'une séquence infinie de $-R$, soit une valeur de $\frac{(1-2\gamma)R}{1-\gamma}$. La politique *stochastique* qui choisit « A » avec une probabilité de 0.5 et « B » avec une probabilité de 0.5 reçoit en moyenne une récompense de 0 à chaque instant. Dès lors, il suffit d'augmenter R et de choisir $\gamma > 0.5$ pour que la différence de valeur entre la politique stochastique et la meilleure politique déterministe soit aussi grande que l'on veut. □

De plus, il est possible, à l'aide d'exemples aussi simples, de montrer que (voir [JAA 95]) :

PROPOSITION 3.2.– *Il existe des* POMDP *où la meilleure politique stochastique adaptée peut être arbitrairement plus mauvaise que la politique optimale du* MDP *sous-jacent.*

PROPOSITION 3.3.– *Il existe des* POMDP *où la meilleure politique adaptée peut être non stationnaire.*

3.2.2.2. *Fonction de valeur adaptée*

Les faits précédents amènent à se poser la question de l'existence d'une politique optimale, même stochastique. Pour cela, on peut se poser la question de l'existence d'une fonction de valeur optimale.

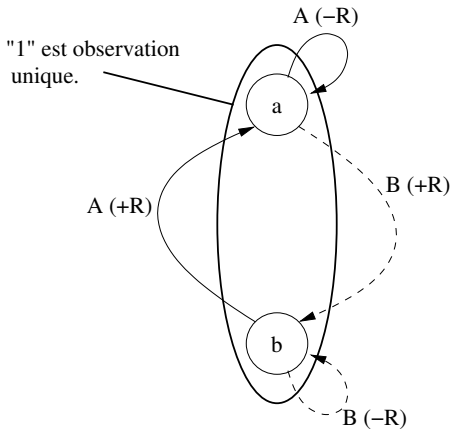

Figure 3.4. Besoin de politiques adaptées stochastiques. *Le* POMDP *de cette figure est constitué de deux états (1a et 1b) qui génèrent tous les deux la même observation « 1 », ce que nous symbolisons par une ellipse. Face à cette observation, l'agent peut choisir entre les actions « A » ou « B » qui, selon l'état sous-jacent, sont associées à des récompenses positives ($+R$) ou négatives ($-R$). On ne peut trouver de politique déterministe optimale.*

Partons d'une politique adaptée $\pi : \Omega \longrightarrow \Pi(\mathcal{A})$, il est possible d'en déduire une politique π' définie sur les états par :

$$\overset{\pi'}{\Pr}(a|s) = \sum_{o \in \Omega} \Pr(o|s) \overset{\pi}{\Pr}(a|o) = \sum_{o \in \Omega} O(o|s)\pi(a|s). \tag{3.10}$$

Connaissant les états sous-jacents du POMDP, on peut appliquer cette politique π' et en déduire une fonction de valeur $\mathcal{V}_{\pi'}$ qui vérifie :

$$\mathcal{V}_{\pi'}(s) \quad = \quad \sum_{a \in \mathcal{A}} Pr(a|\pi', s) \left[r(s) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a)\mathcal{V}_{\pi'}(s') \right].$$

Cette propriété utilise le fait que, pour la politique π', les états sont issus d'un processus qui vérifie la propriété de Markov. Or, ce n'est pas le cas quand on utilise π sur les observations, car les observations ne sont pas des états d'information. On ne peut alors définir la fonction de valeur d'une politique adaptée, c'est-à-dire définie sur les seules observations.

Mais on peut se servir de $\mathcal{V}_{\pi'}$ pour définir la fonction de valeur d'une observation comme étant la valeur moyenne des états sous-jacents à cette observation si on avait utilisé π'. Nous appelons cette nouvelle fonction la *fonction de valeur adaptée*.

DÉFINITION 3.4.– *Fonction de valeur adaptée*
Pour une politique adaptée π, il existe une politique π' définie sur les états par l'équation (3.10) qui permet de définir la fonction de valeur adaptée de π de la manière suivante :

$$\vartheta^{\pi}(o) \quad = \quad \sum_{s \in \mathcal{S}} \Pr^{\pi}(s|o) \mathcal{V}_{\pi'}(s), \tag{3.11}$$

où $\Pr^{\pi}(s|o)$ est la distribution asymptotique de probabilité sur les états, c'est-à-dire la probabilité que l'état du MDP *sous-jacent soit s quand on observe o quand t tend vers l'infini et $\mathcal{V}_{\pi'}$ la fonction de valeur de π'.*

Cette définition n'est qu'une définition et ne permet pas à l'agent de calculer la fonction de valeur d'une observation puisqu'il n'a pas accès à s. Néanmoins, munis de cette définition, nous pouvons montrer que, contrairement à un MDP, pour un POMDP, il n'existe pas forcément de politique stationnaire qui maximise la valeur de toutes les observations simultanément. Pour un MDP, la politique optimale maximisait simultanément la valeur de tous les états.

La preuve de cette affirmation peut se faire au travers de la figure 3.5. Dans ce POMDP à quatre états (1, 2a, 2b et 3) et trois observations (1, 2 et 3), le seul choix possible d'action se fait pour l'observation « 1 ». Si l'on y augmente la probabilité de choisir l'action « A », on augmente la valeur de l'observation « 1 » et on diminue la valeur de l'observation « 2 ». L'effet inverse se produit si on augmente la probabilité de choisir l'action « B ». On ne peut donc maximiser la valeur de ces deux observations simultanément.

Ainsi, la fonction de valeur adaptée n'a pas du tout les mêmes propriétés que la fonction de valeur d'un MDP. En particulier, on ne peut pas définir de fonction de valeur adaptée optimale, ni donc de politique adaptée optimale. Du moins, en utilisant le critère γ-pondéré.

3.2.2.3. *Convergence des algorithmes adaptés*

Puisqu'il n'existe ni fonction de valeur adaptée optimale, ni politique adaptée optimale, on peut légitimement se demander si les algorithmes définis pour les MDP sont d'une utilité quelconque si on les adapte aux POMDP. En fait, il n'est même pas sûr que ces algorithmes convergent, puisque leur convergence était assurée par l'existence d'une solution optimale dans le cadre des MDP.

Si l'on y regarde de plus près, il y a deux facteurs à prendre en compte. D'une part, la fonction de valeur d'une observation dépend de la probabilité d'occupation des états sous-jacents. On peut alors se dire que les algorithmes qui ne modifient pas cette probabilité sous-jacente ont des chances de converger. D'autre part, nous avons

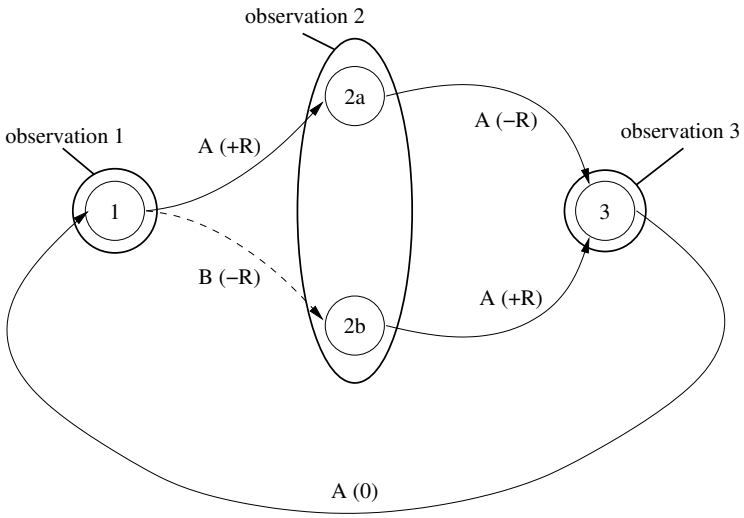

Figure 3.5. Pas de politiques adaptées optimales. *Le* POMDP *de cette figure est constitué de quatre états (1, 2a, 2b et 3) et de trois observations (1, 2 et 3). La seule décision concernant la politique est faite pour l'observation « 1 ». Y augmenter la probabilité de choisir « A » augmente la valeur de « 1 » et diminue celle de « 2 ». C'est un exercice très facile de montrer qu'on ne peut donc trouver de politique maximisant la valeur de toutes les observations.*

vu qu'il n'était pas possible, dans le cas général, de maximiser la fonction de valeur de toutes les observations simultanément. Dès lors, il paraît difficile de prévoir le comportement d'algorithmes qui essaient justement de trouver cette fonction de valeur dominante (comme par exemple l'itération sur les valeurs ou l'itération sur les politiques car l'étape non linéaire de maximisation effectuée à chaque itération est une source d'instabilité potentielle). On peut néanmoins penser que des algorithmes qui explorent de manière stationnaire l'environnement et s'appuient sur une approximation stochastique de la fonction de valeur peuvent avoir une chance de converger. Les résultats démontrés dans la littérature semblent confirmer ces intuitions, puisque la convergence de deux algorithmes entrant dans le cadre énoncé plus haut a été prouvée (voir [SIN 94]).

TD(0) : dans un POMDP, sous les conditions classiques de convergence, l'algorithme TD(0) converge avec une probabilité de 1 vers la solution du système d'équations ci-dessous :

$$\vartheta^{\pi}(o) \quad = \quad \sum_{s \in \mathcal{S}} \overset{\pi}{\Pr}(s|o) \left[r(s) + \gamma \sum_{o' \in \Omega} \overset{\pi}{\Pr}(s, o') \vartheta^{\pi}(o') \right], \tag{3.12}$$

où $\Pr^{\pi}(s, o') = \sum_{s' \in \mathcal{S}} \Pr^{\pi}(s'|s) O(o'|s')$.

Il est à noter que la fonction de valeur vers laquelle on converge n'est pas forcément la fonction de valeur définie à l'équation (3.11). Il est d'ailleurs possible de montrer sur des exemples assez simples que cela n'est pas le cas.

Q-Learning : dans un POMDP, un algorithme de Q-Learning qui utilise une politique d'exploration stationnaire $\pi_{\exp}$ converge vers la solution du système d'équations suivant avec une probabilité de 1 (sous les conditions classiques de convergence).

$$Q(o,a) = \sum_{s \in \mathcal{S}} \Pr^{\pi_{\exp}}(s|o,a) \left[r(s,a) + \gamma \sum_{o' \in \Omega} \overset{a}{\Pr}(s,o') max_{a' \in \mathcal{A}} Q(o',a') \right],$$

où $\Pr^{\pi_{\exp}}(s|o,a)$ est la probabilité asymptotique d'occupation de s sous la politique d'exploration $\pi_{\exp}$ sachant qu'on observe o après avoir effectué l'action a et où $\Pr^a(s,o') = \sum_{s' \in \mathcal{S}} p(s'|s,a) O(o'|s')$.

Il est important de noter que le Q-Learning ne converge que si on explore l'environnement avec une politique stationnaire (pas d'algorithme ϵ-greedy par exemple). En outre, le Q-Learning est limité par le fait qu'il converge vers une politique déterministe dont nous avons vu qu'elle pouvait être clairement sous-optimale.

Ces limitations posent avec insistance le problème de l'apprentissage dans un POMDP car, comme nous le verrons, la recherche de solutions optimales quand on connaît le modèle est en partie résolu.

3.2.3. *Algorithmes adaptés et critère moyen adapté*

Nous venons de voir qu'il n'est pas possible de définir une fonction de valeur qui soit optimale sur toutes les observations à la fois. Afin de comparer deux politiques adaptées entre elles et donc de définir une politique adaptée optimale, l'idée est de transformer cette fonction de valeur sur les observations en un scalaire, par exemple en écrivant que l'on cherche à maximiser $\hat{\vartheta} = \sum_{o \in \Omega} P_\Omega(o) \vartheta^\pi(o)$ où P_Ω est une distribution de probabilité sur Ω. Parmi les choix possibles pour cette distribution, on peut naturellement penser à la distribution initiale sur les observations, ou à la probabilité asymptotique des observations. Cette dernière solution est en fait équivalente à utiliser le *critère moyen* et donc le gain associé U^π (voir paragraphe 1.5.4), comme l'ont montré les auteurs de [SIN 94].

En effet, si on pose $\Pr^\pi(o)$ comme étant la probabilité asymptotique d'une observation en suivant une politique π, on peut écrire le critère scalaire ci-dessus comme :

$$\sum_{o \in \Omega} \overset{\pi}{\Pr}(o) \vartheta^\pi(o) = \sum_{o \in \Omega} \overset{\pi}{\Pr}(o) \sum_{s \in \mathcal{S}} \overset{\pi}{\Pr}(s|o) V^\pi(s)$$

$$
\begin{aligned}
&= \sum_{s\in\mathcal{S}}\sum_{o\in\Omega} \overset{\pi}{\Pr}(o)\overset{\pi}{\Pr}(s|o)V^\pi(s) \\
&= \sum_{s\in\mathcal{S}} \overset{\pi}{\Pr}(s)V^\pi(s) \\
&= \sum_{s\in\mathcal{S}} \overset{\pi}{\Pr}(s)r(s,\pi(s)) + \gamma\sum_{s\in\mathcal{S}} \overset{\pi}{\Pr}(s)\sum_{s'\in\mathcal{S}} p(s'|s,\pi(s))V^\pi(s') \\
&= U^\pi + \gamma\sum_{s'\in\mathcal{S}} \overset{\pi}{\Pr}(s')V^\pi(s') \\
&= U^\pi + \gamma\sum_{o\in\Omega} \overset{\pi}{\Pr}(o)V^\pi(o)
\end{aligned}
$$

et donc $\sum_{o\in\Omega} \Pr^\pi(o)V^\pi(o) = \frac{U^\pi}{1-\gamma}$.

Il est alors tentant de chercher une politique adaptée stochastique optimale au sens de ce critère scalaire en cherchant tout simplement la politique adaptée stochastique qui optimise la récompense moyenne. A notre connaissance, le seul algorithme pour trouver une politique de ce type a été proposé par Jaakkola, Singh et Littman [JAA 95]. Cet algorithme de type « itération sur les politiques » s'appuie sur une méthode de Monte-Carlo pour évaluer la récompense moyenne d'une politique, ce qui permet de calculer des *Q-valeurs* et d'améliorer la politique courante. Le problème majeur de cette méthode, outre le fait qu'elle n'a jamais été testée en pratique, est qu'elle ne converge au mieux que vers un maximum local de la fonction de valeur scalaire.

Le chapitre sur les méthodes de gradient pour la recherche de politique optimale (chapitre 3 du volume 2) exposera d'autres méthodes approchées pour résoudre les POMDP, notamment au paragraphe 3.2.5.

3.3. Calculer une politique exacte sur les états d'information

Par l'intermédiaire des états d'information, nous savons transformer un POMDP en un MDP. En théorie, les outils de résolution des MDP peuvent être utilisés pour trouver une politique optimale. En pratique, le problème est plus délicat, notamment à cause de la taille et de la nature de l'espace des états d'information. En fait, la résolution d'un POMDP ayant un seul état initial, même en horizon fini, est un problème PSPACE-dur [PAP 87]. Ce sont des problèmes dont la complexité en espace mémoire et donc en temps, est au moins polynomiale en la taille du problème. Si l'horizon N est plus petit que $|\mathcal{S}|$, le POMDP est PSPACE-complet.

NOTE.– Dans la suite de ce chapitre, nous allons travailler avec le critère actualisé et la fonction de valeur associée V.

3.3.1. *Cas général*

3.3.1.1. *Horizon fini*

En théorie, pour un horizon fini N, il suffit d'utiliser le principe de la programmation dynamique pour, d'une manière similaire à l'algorithme 1.1, trouver la politique et la fonction de valeur optimales. Ainsi, partant de la fonction de valeur $V_0^* = \max_{a \in \mathcal{A}} \rho(I, a)$ au dernier pas de temps, on applique récursivement l'opérateur de programmation dynamique (équation (3.7), section 3.1.5) N fois de la manière suivante :

$$V_n^*(I) \quad = \quad \max_{a \in \mathcal{A}} \left[\rho(I,a) + \gamma \sum_{o \in \Omega} \Pr(o|I,a) V_{n-1}^*(\tau(I,o,a)) \right]. \tag{3.13}$$

L'équation (3.13) définit aussi un opérateur que nous noterons L. On a alors $V_n^* = LV_{n-1}^*$.

Une fois que l'on connaît la fonction de valeur optimale, la politique optimale μ_n^* s'en déduit par :

$$\mu_n^*(I) \quad = \quad \operatorname*{argmax}_{a \in \mathcal{A}} \left[\rho(I,a) + \gamma \sum_{o \in \Omega} \Pr(o|I,a) V_{n-1}^*(\tau(I,o,a)) \right].$$

3.3.1.2. *Horizon infini*

Plaçons-nous maintenant dans le cas d'un horizon infini. L'opérateur L que nous venons de définir possède les mêmes propriétés que l'opérateur utilisé dans les MDP au paragraphe 1.5.2, c'est une contraction (pour toutes fonctions U et V, on a $\|LU - LV\| \leq \gamma \|U - V\|$ où $\|V\| = \max_{\mathcal{I}} V(I)$).

Il est alors possible de trouver une fonction de valeur ϵ-optimale en utilisant une méthode de type itération sur les valeurs dont le pas d'itération est :

$$V_{i+1} \quad = \quad LV_i. \tag{3.14}$$

Puisqu'il existe une fonction de valeur optimale V^* et que L est une contraction, on peut utiliser le théorème de Banach pour prouver que cette méthode d'itération sur les valeurs converge vers V^*. Dès lors, il suffit d'un nombre fini d'itérations de la méthode pour atteindre une solution ϵ-optimale d'où il est possible de déduire une politique optimale.

3.3.2. *Etats de croyance et fonction de valeur linéaire par morceaux*

En pratique, les deux schémas de calcul évoqués dans la partie précédente sont difficilement utilisables. Le problème vient du fait que les espaces d'états d'information sont continus ou de taille conséquente et il peut alors être impossible de calculer ou de représenter les fonctions de valeur et les politiques optimales.

En travaillant avec les états de croyance et les POMDP qui admettent de tels états d'information, il est possible d'exploiter les propriétés particulières des fonctions de valeur pour proposer des algorithmes plus efficaces.

La fonction de valeur optimale pour un problème à horizon fini est linéaire par morceaux et convexe (LPMC) [SON 71]. C'est une propriété très importante car elle permet de représenter la fonction de valeur avec un nombre fini de paramètres, ainsi que le montre la figure 3.6.

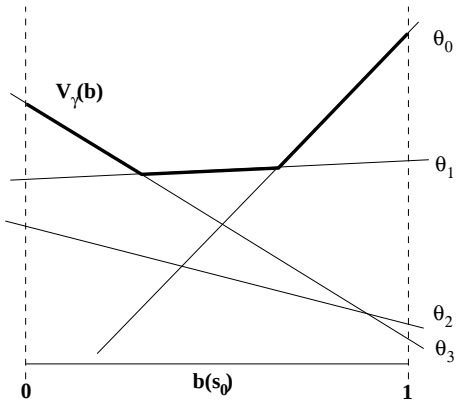

Figure 3.6. Fonction de valeur convexe linéaire par morceaux. *La fonction de valeur V d'un* POMDP *avec deux états (s_0 et s_1) est représentée à l'aide de 4 vecteurs de paramètres θ_i, chacun de dimension 2. On trouve l'espace des états d'information le long de l'axe des abscisses et les valeurs sont sur l'axe des ordonnées. Une seule probabilité $b(s_0)$ permet de décrire l'état de croyance car $b(s_1) = 1 - b(s_0)$. Sur cette figure, chaque segment linéaire de la fonction de valeur est tracé avec une ligne fine tandis que la fonction de valeur elle-même est indiquée en gras.*

Comme la fonction de valeur est définie sur l'espace $\mathcal{B}$ qui est de dimension $|\mathcal{S}|-1$, chaque segment linéaire utilisé dans sa représentation est de dimension $|\mathcal{S}|$ et peut donc être représenté par un vecteur θ avec $|\mathcal{S}|$ coefficients. Pour ces vecteurs, $\theta(s)$ sera la s-ième composante de ce vecteur. L'ensemble des vecteurs permettant de représenter la fonction de valeur LPMC est noté Θ. On dit que Θ représente V, ce qui

se traduit par :

$$V(b) = \max_{\theta \in \Theta} \sum_{s \in \mathcal{S}} b(s)\theta(s) \quad (3.15)$$

$$= \max_{\theta \in \Theta} b.\theta. \quad (3.16)$$

Il reste maintenant à démontrer que la fonction de valeur optimale est bien linéaire par morceaux et convexe. Cette démonstration s'appuie sur le théorème suivant démontré par Smallwood.

THÉORÈME 3.1.– *(Fonction de valeur linéaire par morceaux et convexe). ([SMA 73]) Soit L l'opérateur de Bellman défini à l'équation (3.13) et soit V_{init} une fonction de valeur initiale qui est linéaire par morceaux et convexe, définie sur l'espace $\mathcal{B}$ des états de croyance. Alors, pour un* POMDP *admettant des états de croyance, on a :*

– $V_n = L^n V_{\text{init}}$, obtenue après n applications de l'opérateur L sur V_{init}, est elle aussi linéaire par morceaux et convexe sur $\mathcal{B}$;

– V_n peut être représentée par un ensemble fini $\Theta = \{\theta\}$ de vecteurs de taille $|\mathcal{S}|$ par $V_n(b) = \max_{\theta \in \Theta} b.\theta$.

PREUVE.– Nous allons montrer que si, après $i-1$ applications de l'opérateur L, la fonction de valeur V_{i-1} est linéaire par morceaux et convexe, alors la fonction de valeur V_i obtenue après une nouvelle application de l'opérateur est elle aussi LPMC.

Supposons donc que V_{i-1} est LPMC, il existe alors Θ_{i-1} tel que :

$$V_{i-1} = \max_{\theta_{i-1} \in \Theta_{i-1}} \sum_{s' \in \mathcal{S}} b_{i-1}(s')\theta_{i-1}(s').$$

Si a est l'action effectuée alors qu'il restait i actions à choisir et o l'observation reçue après. Alors, on peut écrire l'état de croyance b_{i-1} comme étant :

$$b_{i-1} = \Pr(s'|b_i, a, o),$$

ce qui nous permet d'obtenir :

$$V_{i-1} = \max_{\theta_{i-1} \in \Theta_{i-1}} \sum_{s' \in \mathcal{S}} \Pr(s'|b_i, a, o)\theta_{i-1}(s').$$

On peut alors substituer V_{i-1} dans l'équation (3.13) définissant l'opérateur L, ce qui donne :

$$V_i(b_i) = \max_{a \in \mathcal{A}} \left[\rho(b_i, a) + \gamma \sum_{o \in \Omega} \Pr(o|b_i, a) \max_{\theta_{i-1} \in \Theta_{i-1}} \sum_{s' \in \mathcal{S}} \Pr(s'|b_i, a, o) \theta_{i-1}(s') \right],$$

que l'on peut réécrire en :

$$\begin{aligned}
V_i(b_i) = & \\
& \max_{a \in \mathcal{A}} \left[\sum_{s \in \mathcal{S}} r(s, a) b_i(s) + \gamma \sum_{o \in \Omega} \Pr(o|b_i, a) \max_{\theta_{i-1} \in \Theta_{i-1}} \sum_{s' \in \mathcal{S}} \Pr(s'|b_i, a, o) \theta_{i-1}(s') \right] \\
= & \max_{a \in \mathcal{A}} \left[\sum_{s \in \mathcal{S}} r(s, a) b_i(s) + \gamma \sum_{o \in \Omega} \max_{\theta_{i-1} \in \Theta_{i-1}} \sum_{s' \in \mathcal{S}} \Pr(o|b_i, a) \Pr(s'|b_i, a, o) \theta_{i-1}(s') \right] \\
= & \max_{a \in \mathcal{A}} \left[\sum_{s \in \mathcal{S}} r(s, a) b_i(s) + \gamma \sum_{o \in \Omega} \max_{\theta_{i-1} \in \Theta_{i-1}} \sum_{s' \in \mathcal{S}} \Pr(s', o|b_i, a) \theta_{i-1}(s') \right] \\
= & \max_{a \in \mathcal{A}} \left[\sum_{s \in \mathcal{S}} r(s, a) b_i(s) + \gamma \sum_{o \in \Omega} \max_{\theta \in \Theta_{i-1}} \sum_{s' \in \mathcal{S}} \left[\sum_{s \in \mathcal{S}} \Pr(s', o|s, a) b_i(s) \right] \theta(s') \right].
\end{aligned}$$

Soit $\theta_{i-1}^{a,o}(b)$ l'élément optimal de Θ_{i-1} pour b, a et o donnés, c'est-à-dire $\theta_{i-1}^{a,o}(b) = \operatorname{argmax}_{\theta_{i-1} \in \Theta_{i-1}} \sum_{s \in \mathcal{S}} \Pr(s|b_i, a, o) \theta_{i-1}(s)$. Cela revient à chercher le segment de droite qui définit la fonction de valeur pour b, a et o donnés. Alors, nous pouvons écrire :

$$\begin{aligned}
V_i(b_i) &= \max_{a \in \mathcal{A}} \left[\sum_{s \in \mathcal{S}} r(s, a) b_i(s) + \gamma \sum_{o \in \Omega} \sum_{s' \in \mathcal{S}} \left[\sum_{s \in \mathcal{S}} \Pr(s', o|s, a) b_i(s) \right] \theta_{i-1}^{a,o}(b_i, s') \right] \\
&= \max_{a \in \mathcal{A}} \left[\sum_{s \in \mathcal{S}} b_i(s) \left[r(s, a) + \gamma \sum_{o \in \Omega} \sum_{s' \in \mathcal{S}} \Pr(s', o|s, a) \theta_{i-1}^{a,o}(b_i, s') \right] \right].
\end{aligned}$$

L'expression entre les crochets intérieurs de l'équation précédente peut se représenter par $|\mathcal{A}||\Theta_{i-1}|^{|\Omega|}$ différents vecteur de taille $|\mathcal{S}|$: il faut en effet au maximum un vecteur par choix de a et d'une séquence de $|\Omega|$ vecteurs de Θ_{i-1}. On peut dire que ces vecteurs forment alors l'ensemble Θ_i, ce qui permet d'écrire $V_i(b_i)$ comme étant :

$$V_i(b_i) \quad = \quad \max_{\theta_i \in \Theta_i} \sum_{s \in \mathcal{S}} b_i(s) \theta_i(s).$$

Cela signifie que V_i est bien une fonction linéaire par morceaux et convexe et qu'elle peut se définir par un ensemble fini de vecteur θ_i de Θ_i. Chacun de ces vecteurs θ_i est

de la forme :

$$\theta_i(b,s) \quad = \quad r(s,a) + \gamma \sum_{o \in \Omega} \sum_{s' \in \mathcal{S}} \Pr(s', o|s, a) \theta_{i-1}^{a,o}(b, s'). \tag{3.17}$$

Comme la fonction de valeur initiale V_{init} est linéaire par morceaux et convexe, on en déduit aisément qu'après tout nombre fini d'applications de l'opérateur L on obtient bien une fonction de valeur qui est elle aussi LPMC, ce qui conclut la démonstration. □

Pour montrer que la fonction de valeur est LPMC en utilisant le théorème précédent, il ne nous reste plus qu'à montrer que toute fonction de valeur optimale pour un horizon de taille 1 est LPMC. Quand il ne reste plus qu'une action a à choisir, seule la récompense immédiate intervient dans la définition de la fonction de valeur. On a alors immédiatement :

$$\begin{aligned} V_1^*(b) &= \max_{a \in \mathcal{A}} \sum_{s \in \mathcal{S}} b(s) r(s,a) \\ &= \max_{a \in \mathcal{A}} b.r(a). \end{aligned}$$

V_1^* est donc trivialement LPMC et peut être représentée par au plus $|\mathcal{A}|$ vecteurs.

Ainsi, en utilisant le théorème précédent, on obtient alors immédiatement que toute fonction de valeur optimale pour un horizon fini est linéaire par morceaux et convexe. De plus, la démonstration du théorème étant constructive, on sait aussi que :

– on peut toujours calculer la fonction de valeur optimale pour un horizon fini en un nombre fini d'opérations ;

– la fonction de valeur optimale pour un horizon fini peut être représentée par un ensemble fini de vecteurs de taille $|\mathcal{S}|$;

– la politique optimale est donc calculable elle aussi en un temps fini.

3.3.2.1. *Choix des vecteurs θ*

Une fonction de valeur optimale pour un horizon fini peut être représentée par un nombre fini de vecteurs. Nous avons vu dans la démonstration du théorème 3.1 qu'il fallait au maximum $|\mathcal{A}||\Theta_{i-1}|^{|\Omega|}$ vecteurs pour représenter V_i. En fait, on peut encore réduire ce nombre en ne considérant que l'ensemble Ω^{poss} des observations possibles après avoir exécuté une action dans un état de croyance donné. La borne devient donc $|\mathcal{A}||\Theta_{i-1}|^{|\Omega|^{\text{poss}}}$. En pratique, il faut beaucoup moins de vecteurs pour représenter la fonction de valeur. Certains vecteurs (comme par exemple le vecteur θ_2 dans la

figure 3.6) sont en effet dominés par les autres et leur omission n'influence pas la fonction de valeur. On appelle ce type de vecteur un *vecteur dominé*. Inversement, un vecteur qui permet de calculer la valeur optimale pour au moins un point de l'espace des états de croyance est appelé un *vecteur utile*.

On comprend aisément que, pour faciliter le calcul des fonctions de valeur, il faut minimiser le nombre de vecteurs utilisés pour représenter les fonctions. Pour un état de croyance I donné, il faut utiliser chaque vecteur de la représentation pour trouver la fonction de valeur. Il est donc très intéressant d'éliminer tous les vecteurs dominés d'une représentation, ce qui est un problème très difficile et [LIT 95] a montré que cela ne pouvait être fait efficacement que si RP=NP, c'est-à-dire si tous les algorithmes de décision non déterministes ont une probabilité significative de succès.

La recherche des fonctions de valeur doit donc faire face à une possible explosion du nombre de vecteurs (taille exponentielle en $|\Omega|$), mais aussi à la difficulté de trouver les vecteurs utiles.

Les algorithmes de recherche de fonction optimale que nous allons détailler au paragraphe 3.4 explorent plusieurs méthodes pour rendre efficace cette recherche des vecteurs utiles à la représentation.

3.3.2.2. *Horizon infini*

Bien que chaque fonction de valeur V_n^* soit linéaire par morceaux et que :

$$\lim_{n\to\infty} \|V_n^* - V^*\| \quad = \quad 0,$$

rien ne dit que la fonction de valeur optimale pour un horizon infini soit elle aussi linéaire par morceaux. [SON 71] a montré qu'il existe une classe de POMDP où V^* est LPMC, il l'appelle la classe des POMDP *transitoires*. Quand le POMDP n'est pas transitoire, on peut approcher aussi précisément que l'on veut la fonction de valeur optimale ainsi que la politique optimale en horizon infini par une politique transitoire admettant une fonction de valeur optimale. On parle de solution ϵ-optimale.

En revanche, toutes les fonctions de valeur en horizon infini, même celles qui ne sont pas transitoires, sont convexes.

3.4. Algorithmes exacts d'itération sur les valeurs

3.4.1. *Etapes de l'opérateur de programmation dynamique*

Le problème crucial de l'application de l'opérateur de la programmation dynamique est la construction de l'ensemble de vecteurs Θ_n à partir de Θ_{n-1}. Cette opération est résumée (on peut aussi dire cachée) dans l'équation (3.17). Nous allons

détailler le principe de cette construction et en donner un exemple qui essayera d'expliciter les principales notions utilisées dans les algorithmes que nous allons étudier par la suite.

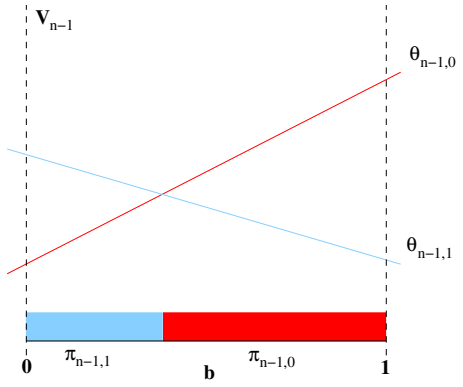

Figure 3.7. ***Fonction de valeur à l'étape*** $n-1$. *Ce* POMDP *à deux états sert d'exemple pour expliciter la construction de l'ensemble* Θ_n. *A l'étape* $n-1$, *la fonction de valeur est représentée par deux vecteurs, qui sont chacun associés à une politique. Les régions de dominance sont démarquées par les barres au bas de la figure. Les politiques associées à ces régions sont indiquées sous les barres.*

Partons donc d'un ensemble Θ_{n-1} de vecteurs décrivant la fonction de valeur optimale pour un horizon de taille $n-1$. Chacun des vecteurs θ_{n-1} domine les autres dans une région de l'espace $\mathcal{B}$ des états de croyance et, fait important, représente aussi la meilleure politique à suivre dans cette région. On peut associer chaque vecteur θ_{n-1} à une politique π_{n-1} d'horizon $n-1$. La figure 3.7 illustre le cas qui nous servira d'exemple, avec un ensemble de 2 vecteurs ($\theta_{n-1,0}$ et $\theta_{n-1,1}$). Les régions où dominent les vecteurs sont représentées par la barre grisée le long de l'axe des abscisses.

La construction de l'ensemble Θ_n se comprend plus facilement en la décomposant et c'est aussi la démarche suivie par les deux algorithmes que nous allons présenter. Supposons que nous voulons calculer $V_n^{a1,o1}$ après avoir effectué l'action $a1$ et observé $o1$. On sait alors que la fonction de valeur sera représentée par les vecteurs $\{\theta_n^{a1,o1}\}$ donnés par l'équation (3.17) adaptée à ce cas précis, c'est-à-dire :

$$\theta_n^{a1,o1}(b,s) \quad = \quad \frac{r(s,a1)}{|\Omega|} + \gamma \sum_{s' \in \mathcal{S}} p(s,a1,s') O(s',o1) \theta_{n-1}^{a1,o1}(b^{a1,o1},s) \tag{3.18}$$

Cette fonction de valeur sera représentée avec au plus $|\Theta_{n-1}|$ vecteurs. La figure 3.8 illustre cette transformation de la fonction de valeur. Il est à noter que cette fonction

de valeur est un peu particulière puisqu'elle prend en compte la probabilité que $o1$ soit observée, ce qui se traduit par le terme $\frac{r(s,a)}{|\Omega|}$ puisque nous avons fait le choix ici de distribuer uniformément la récompense immédiate sur les observations.

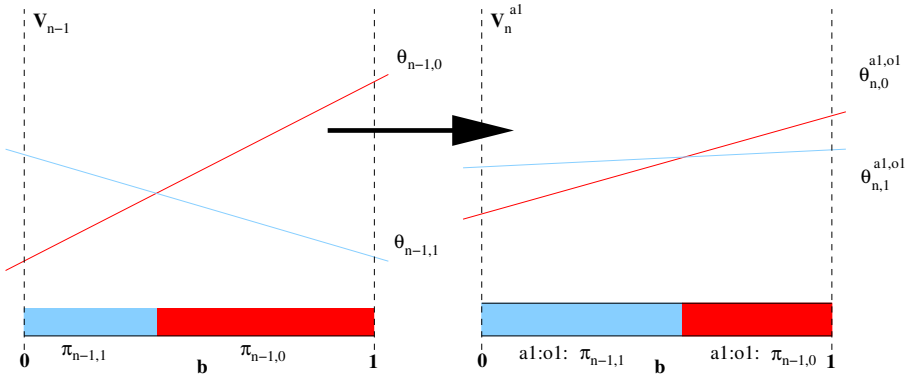

Figure 3.8. Fonction de valeur à l'étape n pour $a1$ et $o1$. *Pour une action et une observation données, la nouvelle fonction de valeur se représente avec, au plus, le même nombre de vecteur. Ces vecteurs définissent des nouvelles régions où les politiques d'horizon n commencent toutes par $a1 : o1$ avant d'utiliser la meilleure politique d'horizon $n-1$.*

Avec $a1$ toujours fixé, on peut ainsi calculer la fonction de valeur pour chaque observation o possible. L'idée est maintenant d'utiliser ces fonctions de valeur pour calculer la fonction de valeur V_n^a reçue après avoir effectué l'action $a1$. C'est aussi une fonction LPMC. C'est en fait la moyenne des fonctions de valeur associées à chaque couple $(a1, o)$. Etant donné que les fonctions $V_n^{a1,o1}$ prennent déjà en compte les probabilités d'observation, il est trivial de montrer que, pour chaque état de croyance b on a :

$$\theta_n^{a1}(b) \quad = \quad \sum_{o \in \Omega} \theta_n^{a1,o}(b). \tag{3.19}$$

De cette manière, nous générons en fait au plus $|\Theta_{n-1}|^{|\Omega^{\text{succ}}(a1)|}$ vecteurs. Pour chaque vecteur, nous aurons un ensemble de politiques qui commencent toutes par l'action $a1$ et qui dépendent ensuite de l'observation reçue. Cette étape est illustrée par la figure 3.9.

Il reste enfin à combiner les fonctions de valeur calculées pour chaque action afin de déterminer la fonction de valeur V_n pour un horizon n. Pour chaque état de

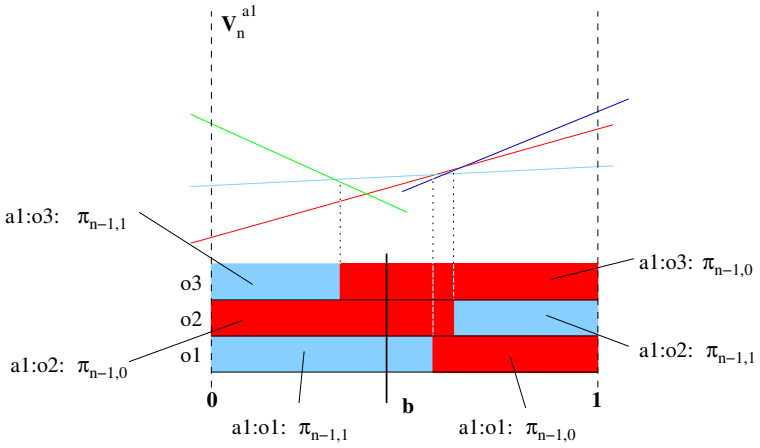

Figure 3.9. Fonction de valeur à l'étape n pour $a1$. *Pour une action donnée, la nouvelle fonction de valeur se représente avec des vecteurs qui sont sommes des vecteurs $\theta_n^{a1,o}$. Ces vecteurs définissent des nouvelles régions où les politiques d'horizon n commencent toutes par $a1$ avant d'utiliser la meilleure politique d'horizon $n-1$ en fonction de l'observation qui sera perçue. Par exemple, pour l'état de croyance b indiqué, après avoir effectué $a1$, il faudra utiliser $\pi_{n-1,1}$ si $o = o1$ et $\pi_{n-1,0}$ sinon.*

croyance, la fonction est représentée par le meilleur vecteur V_n^a et on a :

$$\theta_n(b) \quad = \quad \max_{a \in \mathcal{A}} \theta_n^a(b). \tag{3.20}$$

C'est à cette étape qu'il est possible d'éliminer le plus de vecteurs qui ne sont pas utiles pour représenter la fonction de valeur. La figure 3.10 explicite cette étape en combinant les fonctions de valeur V_n^{a1} et V_n^{a2} de deux actions, en éliminant les vecteurs inutiles (indiqué en pointillés).

3.4.2. *Obtenir une représentation parcimonieuse de V*

Nous définissons ici la notion de représentation parcimonieuse de V et présentons quelques techniques pour obtenir cette représentation parcimonieuse en élaguant les vecteurs dominés.

3.4.2.1. *Région*

Un ensemble Θ donné définit une partition sur l'espace des vecteurs estimés $\mathcal{B}$. Chaque partie de cette partition est associée à un vecteur θ de Θ. Par exemple, sur la

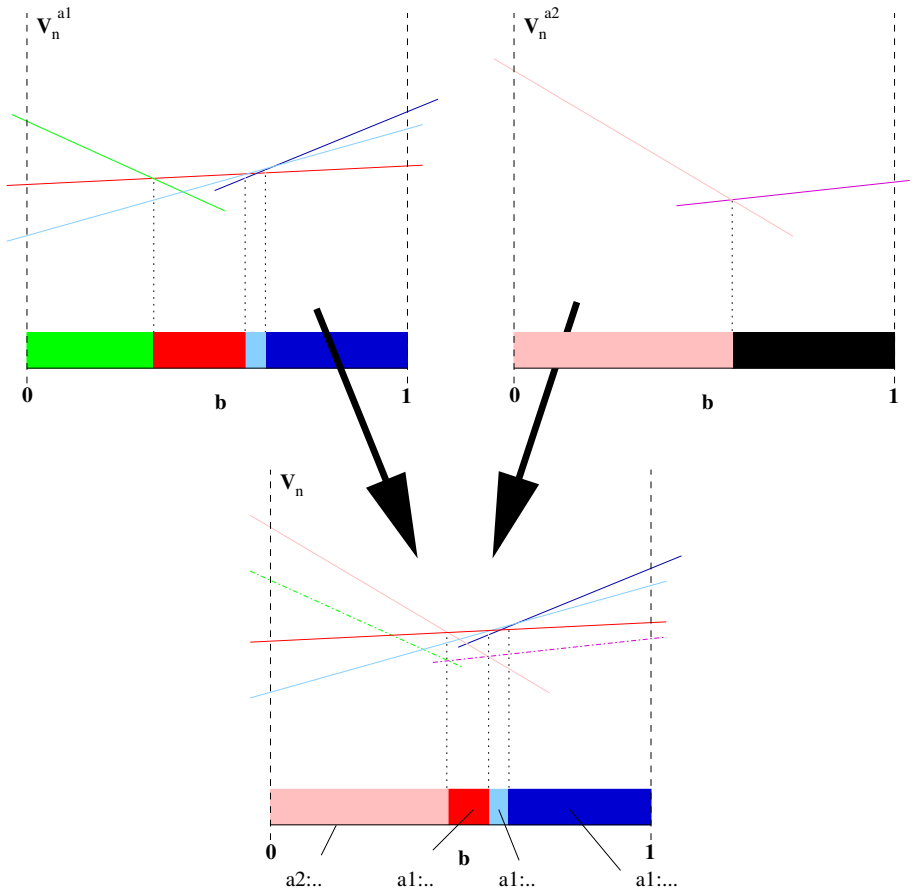

Figure 3.10. Fonction de valeur à l'étape n. *Il faut maintenant chercher la meilleure action, ce qui revient à chercher, pour chaque état de croyance, le meilleur vecteur pour le représenter. Nous avons ici combiné les fonctions de valeur de deux actions ($a1$ et $a2$), ce qui permet en outre d'éliminer des vecteurs inutiles (indiqués en pointillés). Pour chacune des régions déterminées, nous avons indiqué le début de la politique optimale d'horizon n.*

figure 3.7, la partition est constituée de deux parties. La région $R(\theta, \Theta)$ associée à un vecteur est la portion de $\mathcal{B}$ où ce vecteur domine les autres.

DÉFINITION 3.5.– *Soient un espace d'états de croyance $\mathcal{B}$ et une représentation d'une fonction de valeur Θ, la région $R(\theta, \Theta)$ associée à un vecteur θ de Θ est définie par :*

$$R(\theta, \Theta) \quad = \quad \{b \mid b.\theta > b.\theta',\ \theta' \in \Theta - \{\theta\},\ b \in \mathcal{B}\}. \tag{3.21}$$

A cause de l'inégalité stricte, les régions ne définissent pas exactement une partition de $\mathcal{B}$ et les points exclus sont des points où plusieurs vecteurs donnent une même valeur à la fonction de valeur. Ces points sont assez particuliers et peuvent poser problème, comme nous le verrons au paragraphe 3.4.2.5.

La notion de région est très importante. Elle est utilisée par de nombreux algorithmes pour calculer la fonction de valeur. L'algorithme `TrouveVectDansRegion` (algorithme 3.1) décrit la procédure qui détermine si la région d'un vecteur est vide (c'est alors un vecteur inutile) et, dans le cas contraire, retourne un vecteur particulier de cette région. Il faut pour cela utiliser une méthode de programmation linéaire pour optimiser une fonction sous contraintes qui est définie par la méthode `PrepareProgLin` explicitée dans l'algorithme 3.2.

Algorithme 3.1 : TrouveVectDansRegion(θ, Θ)

Entrées : Une représentation Θ, un vecteur $\theta \in \Theta$
Sorties : Un point de cette région ou **null**

LP $\leftarrow$ `PrepareProgLin` (θ, Θ)
`ResoudProgLin` (LP)
si `SansSolution` *(*LP*)* **alors**
  retourner null
si *value(*LP*)* ≤ 0 **alors**
  retourner null
retourner `Solution` *(*LP*)*

Algorithme 3.2 : PrepareProgLin(θ, Θ)

Entrées : Une représentation Θ, un vecteur $\theta \in \Theta$
Sorties : Un problème de programmation linéaire
résoudre
  $\max_{\mathbb{R}} \epsilon$
  avec
    $x.(\theta - \tilde{\theta}) \geq \epsilon,\ \forall \tilde{\theta} \in \Theta,\ \tilde{\theta} \neq \theta$
      $x \in \Pi(\mathcal{S})$

3.4.2.2. *Représentation parcimonieuse*

Dans un ensemble Θ quelconque, certains vecteurs ne sont pas nécessaires à la représentation de la fonction de valeur. C'est le cas des vecteurs dominés.

DÉFINITION 3.6.– *Soit Θ représentant une fonction de valeur. Un vecteur θ de Θ est dominé si $\forall b \in \mathcal{B}$ on a :* $b.\theta \leq \max_{\theta' \in \Theta} b.\theta'$.

Si θ est dominé, on montre trivialement que Θ et $\Theta - \{\theta\}$ représentent la même fonction de valeur : un vecteur dominé peut être enlevé de Θ sans danger. En fait, on peut montrer (voir par exemple [LIT 96]) qu'une fonction de valeur LPMC possède une unique représentation minimale où aucun vecteur n'est dominé. On dit que cette représentation est parcimonieuse. Sur l'exemple de la figure 3.11 on voit bien que les vecteurs θ_2 et θ_4 ne sont pas utiles pour représenter V. Dans ce cas, la représentation parcimonieuse est $\Theta = \{\theta_0, \theta_1, \theta_3\}$.

DÉFINITION 3.7.– *Une représentation Θ parcimonieuse d'une fonction de valeur LPMC est telle que, pour tout $\theta \in \Theta$, la région $R(\theta, \Theta)$ n'est pas vide.*

Il reste maintenant à construire cette représentation parcimonieuse à partir d'une représentation donnée.

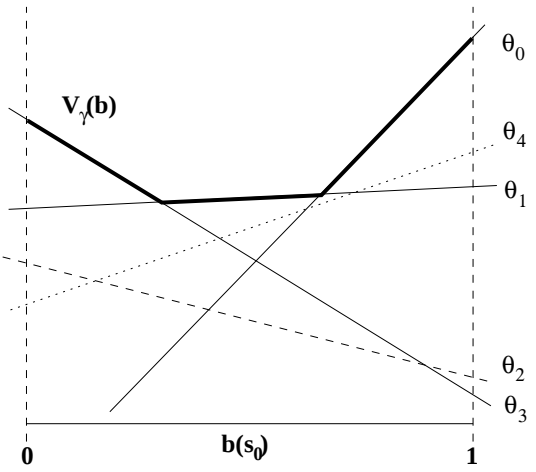

Figure 3.11. Représentation parcimonieuse de V. *Tous les vecteurs de Θ ne sont pas utiles pour représenter V. Le vecteur θ_2, qui est entièrement dominé par θ_1, sera enlevé par la procédure `VerifDomination`. Quant au vecteur θ_4, il faut la procédure plus complexe `Elagage` pour l'éliminer.*

3.4.2.3. *Elimination des vecteurs dominés*

La *recherche de domination simple* est une procédure très simple pour éliminer des vecteurs inutiles d'une représentation Θ. On recherche les vecteurs qui sont entièrement dominés par un seul autre vecteur. Ce sont des vecteurs θ tels qu'il existe un autre vecteur $\tilde{\theta} \in \Theta$ tel que $\forall s' \in \mathcal{S}$, $\theta(s) \leq \tilde{\theta}(s')$. Bien que cette procédure ne garantisse pas de diminuer Θ ou d'éliminer tous les vecteurs inutiles (voir figure 3.11), elle est très efficace d'un point de vue computationnel.

L'algorithme `VerifDomination` (algorithme 3.3) détaille la procédure qui permet de nettoyer une représentation des vecteurs entièrement dominés. On y utilise la procédure `EnleveElement` qui retire un élément d'un ensemble.

Algorithme 3.3 : VerifDomination(Θ)

Entrées : Une représentation Θ
Sorties : Une représentation sans vecteur dominé

si $|\Theta| < 2$ **alors**
  retourner Θ
$\tilde{\Theta} \leftarrow \emptyset$
répéter
  $\theta \leftarrow$ `EnleveElement`(Θ)
  si $\nexists \theta' \in \tilde{\Theta}$ *t.q.* $\theta' \geq \theta$ **alors**
    $\tilde{\Theta} \leftarrow \{\theta' | \theta' \in \tilde{\Theta},\ \theta \not\geq \theta'\}$
    $\tilde{\Theta} \leftarrow \tilde{\Theta} \cup \{\theta\}$
jusqu'à $\Theta = \emptyset$
retourner $\tilde{\Theta}$

3.4.2.4. *Elagage*

La méthode la plus simple pour rendre une représentation parcimonieuse consiste à regarder, pour chacun de ses vecteurs, si la région associée est vide. C'est le principe de l'algorithme proposé par Monahan [MON 82], mais c'est une méthode plutôt inefficace. Il existe une méthode d'élagage beaucoup plus efficace proposée par Lark et White [WHI 91] et détaillée dans l'algorithme `Elagage` (algorithme 3.4).

Le principe de `Elagage` est de construire incrémentalement la représentation parcimonieuse en ayant, à tout moment, un sous-ensemble $\hat{\Theta} \subset \Theta$ de cette représentation parcimonieuse. Prenant dans $\tilde{\Theta}$ un nouveau vecteur candidat θ, le fait de chercher si sa région associée dans $\hat{\Theta}$ n'est pas vide est rapide (car $\hat{\Theta}$ est petit), par contre une réponse positive n'assure pas que c'est un vecteur dominant (on ne connaît pas encore la vraie représentation) mais simplement que $\hat{\Theta}$ n'est pas encore complète. Il faut encore y ajouter au moins un vecteur dominant et on utilise pour cela la routine `MeilleurVecteur` alors que θ a été remis dans $\tilde{\Theta}$. La routine `MeilleurVecteur` recherche, pour l'état de croyance retourné par `TrouveVectDansRegion`, le meilleur vecteur restant dans $\hat{\Theta}$ qui sera alors ajouté à $\hat{\Theta}$. Il y a une petite subtilité dans le choix de ce meilleur vecteur, comme nous allons le voir maintenant.

3.4.2.5. *Choix d'un vecteur en un point*

Les diverses versions de l'équation (3.17) que nous avons explicitées dans le paragraphe 3.4.1 détaillant une application de l'opérateur de la programmation dynamique

Algorithme 3.4 : Elagage($\tilde{\Theta}$)

Entrées : Une représentation $\tilde{\Theta}$ de V
Sorties : Une représentation parcimonieuse Θ de V

$\hat{\Theta} \leftarrow \emptyset$
tant que $\tilde{\Theta} \neq \emptyset$ **faire**
  $\theta \leftarrow$ `EnleveElement`$(\tilde{\Theta})$
  $b \leftarrow$ `TrouveVectDansRegion`$(\theta, \hat{\Theta})$
  si $b \neq$ **null alors**
    $\tilde{\Theta} \leftarrow \tilde{\Theta} \cup \{\theta\}$
    $\theta^* \leftarrow$ `MeilleurVecteur`$(\tilde{\Theta}, b)$
    $\tilde{\Theta} \leftarrow \tilde{\Theta} - \{\theta\}$
    $\hat{\Theta} \leftarrow \hat{\Theta} \cup \{\theta^*\}$
$\Theta \leftarrow \hat{\Theta}$
retourner Θ

permettent de calculer le vecteur dominant en un point b quelconque de l'espace des états de croyance. Il subsiste un problème potentiel quand plusieurs vecteurs sont candidats en un point (illustré par la figure 3.12) : lequel est le vecteur dominant en un point b donné ?

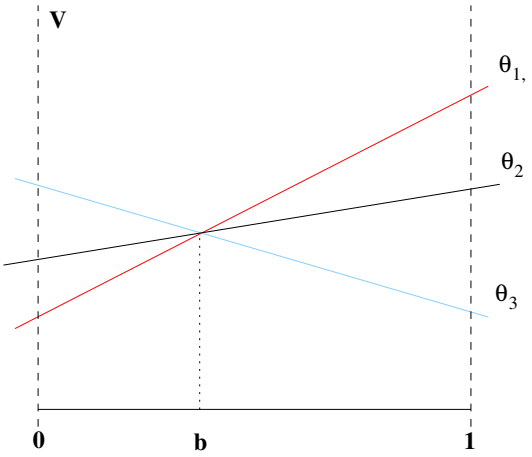

Figure 3.12. Vecteur dominant. *Quel est le vecteur dominant au point b ?*

Il faudrait en fait pouvoir tester la validité de ces vecteurs dans un voisinage de b, ce qui n'est pas simple à mettre en place de manière exacte. Une solution plus subtile

consiste à doter les vecteurs d'un ordre lexicographique. Pour cela il faut doter les états d'un ordre fixe et arbitraire sur $\mathcal{S}$, on le note $s \prec s'$.

DÉFINITION 3.8.– *Le vecteur θ est lexicographiquement plus grand que θ' (noté $\theta \overset{L}{>} \theta'$) s'il existe un état s tel que $\theta(s) > \theta'(s)$ et $\theta(s') = \theta'(s')$ pour tout $s' \prec s$.*

Alors, sans entrer dans les détails de la démonstration, Littman a prouvé (théorème 3.2) que, en cas de doute, le vecteur maximal au sens de cet ordre lexicographique sera bien un vecteur qui fait partie de la représentation parcimonieuse de la fonction de valeur. En effet, la région de ce vecteur sera non vide, et c'est bien sur le vecteur qui est retourné par l'algorithme `MeilleurVecteur` (algorithme 3.6), en s'aidant de la procédure `MaximumLexicographique` (algorithme 3.5) qui retourne le maximum lexicographique de deux vecteurs.

THÉORÈME 3.2 [LIT 96].– *Soient Θ une fonction de valeur, b un point de l'espace des états de croyance et Λ l'ensemble des vecteurs donnant une valeur maximum de la fonction de valeur en b ($\Lambda = \{\mathrm{argmax}_{\theta \in \Theta}\, b.\theta\}$). Alors, s'il existe un $\lambda^* \subset \Lambda$ tel que $\lambda^* \overset{L}{>} \lambda$ pour tous les autres $\lambda \in \Lambda$, alors $R(\lambda^*, \Theta)$ est non vide.*

Algorithme 3.5 : MaximumLexicographique(θ, $\tilde{\theta}$)

Entrées : Deux vecteurs θ et $\tilde{\theta}$ de Θ
Sorties : Le maximum lexicographique des deux vecteurs

```
pour chaque s ∈ S faire
    si θ(s) > θ̃(s) alors
        retourner θ
    si θ(s) < θ̃(s) alors
        retourner θ̃
retourner θ
```

3.4.3. *L'algorithme* WITNESS

L'algorithme WITNESS a été proposé par Cassandra, Littman et Kaelbling [CAS 94] en 1994. Il a ensuite été étudié plus formellement afin de prouver son optimalité [LIT 96]. Pour chaque action a de $\mathcal{A}$, l'algorithme calcule une représentation parcimonieuse de Θ_n^a à partir de Θ_{n-1} en explorant un nombre fini de régions de $\mathcal{B}$. C'est dans la façon de choisir ces régions que réside toute l'intelligence de l'algorithme, comme nous allons le voir. Il est à noter que d'autres méthodes s'appuient aussi sur l'exploration de régions, mais pour construire Θ_n directement (voir [SON 71, SMA 73, CHE 88]).

Algorithme 3.6 : MeilleurVecteur(Θ, b)

Entrées : Une représentation Θ, un état de croyance b
Sorties : Le meilleur vecteur de Θ pour cet état

$v^* \leftarrow -\infty$
pour chaque $\theta \in \Theta$ **faire**
  $v \leftarrow b.\theta$
  si $v = v^*$ **alors**
    $v^* \leftarrow$ `MaximumLexicographique`(θ^*, θ)
  si $v > v^*$ **alors**
    $v^* \leftarrow v$
    $\theta^* \leftarrow \theta$
retourner θ^*

3.4.3.1. *Voisinage d'un vecteur*

La notion de *voisinage* d'un vecteur θ_n^a de V_n^a est cruciale pour l'algorithme WITNESS puisque c'est en vérifiant les voisins d'un vecteur que l'on peut savoir si la construction courante de la représentation parcimonieuse de V_n^a est complète ou pas.

On peut réécrire l'équation (3.18) sous la forme :

$$\theta_n^{a,o} = \frac{r(a)}{|\Omega|} + \gamma P^{a,o}\theta_{n-1}^{a,o}(b^{a,o}), \tag{3.22}$$

où $\theta_{n-1}^{a,o}(b^{a,o})$ est le vecteur de Θ_{n-1} qui est le meilleur pour l'état de croyance $b^{a,o}$. Mais si on enlève les références aux états de croyance, on peut aussi construire toute une famille de vecteurs :

$$\widetilde{\theta^{a,o}} = \frac{r(a)}{|\Omega|} + \gamma P^{a,o}\theta_{n-1}, \tag{3.23}$$

où θ_{n-1} est simplement un vecteur de Θ_{n-1}. On construit ainsi l'ensemble $\widetilde{\Theta}_n^{a,o}$ et, par combinaison, l'ensemble $\widetilde{\Theta}^a$ qui contient la représentation parcimonieuse Θ_n^a de V_n^a. Il y $|\Theta_{n-1}|^{|\Omega|}$ vecteurs possibles dans $\widetilde{\Theta}_n^a$. On peut définir la notion de voisinage pour ces vecteurs comme suit :

DÉFINITION 3.9.– *Un vecteur ν de $\widetilde{\Theta}_n^a$ est un voisin du vecteur $\theta_n^a = \sum_{o\in\Omega} \theta_n^{a,o}$ (qui lui est aussi dans Θ_n^a) si :*

$$\nu = \tilde{\theta}_n^{a,o'} + \sum_{o\neq o'} \theta_n^{a,o},$$

où $o' \in \Omega$, $\tilde{\theta}_n^{a,o'} \in \widetilde{\Theta}_n^{a,o}$ *et* $\tilde{\theta}_n^{a,o'} \neq \theta_n^{a,o'}$.

Un vecteur $\tilde{\theta}$ possède $|O|(|\Theta_{n-1}| - 1)$ voisins, on note $\mathcal{N}(\tilde{\theta})$ l'ensemble de ses voisins. Tout l'intérêt des voisins vient du théorème suivant qui dit que, s'il existe un point de $\mathcal{B}$ où il existe un meilleur vecteur, alors c'est vrai aussi pour un des voisins de ce vecteur (voir [CAS 98]).

THÉORÈME 3.3.– *Pour tout* $\tilde{\theta}_n^a \in \widetilde{\Theta}_n^a$, *il existe un* $b \in \mathcal{B}$ *et un* $\tilde{\theta'}_n^a \in \widetilde{\Theta}_n^a$ *tels que* $b.\tilde{\theta'}_n^a > b.\tilde{\theta}_n^a$ *si et seulement s'il existe un voisin* $\nu \in \mathcal{N}(\tilde{\theta}_n^a)$ *tel que* $b.\nu > b.\tilde{\theta}_n^a$.

PREUVE.– Nous allons procéder en deux temps.

– Prouvons que $b.\nu > b.\tilde{\theta}_n^a \implies b.\tilde{\theta'}_n^a > b.\tilde{\theta}_n^a$.
Comme $\nu \in \widetilde{\Theta}_n^a$, c'est évident.

– Prouvons que $b.\tilde{\theta'}_n^a > b.\tilde{\theta}_n^a \implies b.\nu > b.\tilde{\theta}_n^a$.
Nous allons prouver que ν existe en le construisant. Nous avons :

$$\begin{aligned} b.\tilde{\theta'}_n^a &> b.\tilde{\theta}_n^a \\ \sum_o b.\tilde{\theta'}_n^{a,o} &> \sum_o b.\tilde{\theta}_n^{a,o}. \end{aligned}$$

Comme la première somme est plus grande que la deuxième, il existe forcément une observation o' telle que :

$$b.\tilde{\theta'}_n^{a,o'} > b.\tilde{\theta}_n^{a,o'}.$$

Nous allons nous servir de ce fait pour construire ν :

$$\begin{aligned} \sum_{o \neq o'} b.\tilde{\theta}_n^{a,o} &= \sum_{o \neq o'} b.\tilde{\theta}_n^{a,o} \\ b.\tilde{\theta'}_n^{a,o'} + \sum_{o \neq o'} b.\tilde{\theta}_n^{a,o} &> b.\tilde{\theta}_n^{a,o'} + \sum_{o \neq o'} b.\tilde{\theta}_n^{a,o} \\ b.\left(\tilde{\theta'}_n^{a,o'} + \sum_{o \neq o'} \tilde{\theta}_n^{a,o}\right) &> b.\tilde{\theta}_n^a. \end{aligned}$$

Il suffit de poser $\nu = \tilde{\theta'}_n^{a,o'} + \sum_{o \neq o'} \tilde{\theta}_n^{a,o}$ pour terminer la démonstration de cette implication et donc du théorème. □

3.4.3.2. *L'algorithme*

L'algorithme WITNESS construit progressivement une représentation parcimonieuse Θ_n^a d'une fonction de valeur V_n^a pour une action a fixée. Comme décrit dans l'algorithme `Witness` (Alg. 3.7), l'ensemble $\hat{\Theta}$ grossit en stockant petit à petit les vecteurs qui composent Θ, de sorte que l'on a toujours $\hat{V}(b) \leq V(b)$.

Pour ce faire, l'algorithme choisit d'abord un vecteur b de $\mathcal{B}$ et cherche le meilleur vecteur de la région à laquelle appartient ce vecteur, ainsi que tous les voisins de ce meilleur vecteur. Ces voisins forment l'ensemble Υ qui servira, en quelque sorte, d'agenda. Un par un, les vecteurs v de l'agenda sont examinés pour :

– soit l'enlever de l'agenda si la région définie par v est vide (car v est alors un vecteur inutile pour V) ;

– soit ajouter le meilleur vecteur de la région définie par v aux vecteurs définissant V et mettre tous les voisins de ce vecteur dans l'agenda. Il est aussi important de remettre v dans l'agenda car, bien que v ne soit pas maximal pour l'instant, nous n'avons pas encore la certitude qu'il est inutile.

Algorithme 3.7 : Witness(Θ_{n-1}, a)

Entrées : Une représentation parcimonieuse Θ_{n-1} de V^*_{n-1}, une action a
Sorties : Une représentation parcimonieuse de $V^{*,a}_n$

$b \leftarrow$ un état de croyance de $\mathcal{B}$
$\hat{\Theta} \leftarrow \{\theta^a_n(b)\}$
$\Upsilon \leftarrow \mathcal{N}(\theta^a_n(b))$
tant que $\Upsilon \neq \emptyset$ **faire**
  $v \leftarrow$ `EnleveElement`(Υ)
  si $v \in \hat{\Theta}$ **alors**
    $b \leftarrow$ **null**
  sinon
    $b \leftarrow$ `TrouveVectDansRegion`$(v, \hat{\Theta})$
  si $b \neq$ **null alors**
    $\hat{\Theta} \leftarrow \hat{\Theta} \cup \{\theta^a_n(b)\}$
    $\Upsilon \leftarrow \Upsilon \cup \{v\}$
    $\Upsilon \leftarrow \Upsilon \cup \mathcal{N}(\theta^a_n(b))$
$\Theta^a_n \leftarrow \hat{\Theta}$
retourner Θ^a_n

La validité de l'algorithme repose sur le théorème 3.3. En essence, quels que soient les vecteurs actuellement présents dans $\hat{\Theta}$, s'il y a un vecteur de $\bar{\Theta}^a_n$ qui serait meilleur en un point b, alors un des voisins v du vecteur $\hat{\theta}$ de $\hat{\Theta}$ qui est actuellement le meilleur en b donne aussi une meilleure valeur en ce point b. Comme l'algorithme vérifie tous les voisins des vecteurs de $\hat{\Theta}$, nous sommes sûrs de ne manquer aucun vecteur de Θ^a_n. La preuve formelle est un peu plus complexe.

L'efficacité de l'algorithme peut être améliorée de plusieurs manières. En particulier, [CAS 98] mentionne le fait de choisir avec pertinence les vecteurs de la représentation initiale $\hat{\Theta}$ ou d'éviter de tester plusieurs fois un vecteur v de Υ ou encore

de vérifier l'utilité des voisins d'un vecteur avant de les ajouter à Υ. Il n'en reste pas moins que la portée pratique de cet algorithme est limitée car, pour des raisons de mémoire et de temps de calcul, on ne peut appliquer plus de quelques itérations (de l'ordre de 4 ou 5) à des problèmes avec une poignée d'états.

3.4.4. *Elagage itératif*

L'algorithme d'ÉLAGAGE ITÉRATIF[4] est un peu plus efficace que l'algorithme WITNESS. Nous présentons tout d'abord un algorithme d'élagage très simple, puisqu'il parcourt tous les vecteurs pour élaguer ceux qui sont dominés.

3.4.4.1. *Enumération complète*

Pour trouver une représentation parcimonieuse de Θ_n, il est possible d'énumérer tous les vecteurs possibles de cet ensemble et de les élaguer ensuite. C'est la méthode proposée par Monahan [MON 82] et nous allons la détailler un peu pour comprendre le fonctionnement de l'élagage itératif.

Posons : $\overline{\Theta}_n^{a,o} = \left\{ \frac{r(a)}{|\Omega|} + \gamma \sum_{s' \in \mathcal{S}} p(s,a,s') O(s',o) \theta_{n-1}^{a,o}(b^{a,o}) \right\}_{(a,o)}$.

C'est l'ensemble de tous les vecteurs possibles de $\Theta_n^{a,o}$. Si on cherche toutes les combinaisons possibles de ces vecteurs (en s'inspirant de l'équation (3.19)), on obtient un nouvel ensemble $\overline{\Theta}_n^a$ qui contient Θ_n^a. Nous appellerons cette opération la *somme croisée* et nous noterons :

$$\overline{\Theta}_n^a = \bigoplus_o \overline{\Theta}_n^{a,o}.$$

Ainsi, l'ensemble complet des vecteurs générant V_n s'écrit :

$$\overline{\Theta}_n = \bigcup_a \overline{\Theta}_n^a.$$

Et il ne reste plus qu'à élaguer cet ensemble pour obtenir une représentation parcimonieuse :

$$\Theta_n = \text{ELAGAGE}\left(\bigcup_a \overline{\Theta}_n^a \right).$$

L'inconvénient de cette méthode est que sa complexité est exponentielle en la taille de Ω. L'idée est alors d'effectuer l'élagage de manière incrémentale.

---

4. Pour l'anglais *Iterative pruning*.

3.4.4.2. *Enumération incrémentale*

Comme WITNESS, l'algorithme d'élagage incrémental cherche d'abord des représentations parcimonieuses de Θ_n^a pour ensuite les combiner et trouver Θ_n. Comme $\Theta_n^a = \text{ELAGAGE}(\bar{\Theta}_n^a)$, nous avons :

$$\Theta_n^a = \text{ELAGAGE}\left(\bigoplus_o \overline{\Theta}_n^{a,o}\right).$$

Puisque la procédure ELAGAGE cherche en fait les vecteurs maximaux, il est facile de montrer que ces derniers sont en fait obtenus par combinaisons de vecteurs eux-mêmes maximaux, ce qui nous permet d'écrire que :

$$\begin{aligned}\Theta_n^a &= \text{ELAGAGE}\left(\bigoplus_o \text{ELAGAGE}(\overline{\Theta}_n^{a,o}))\right) \\ &= \text{ELAGAGE}\left(\bigoplus_o \Theta_n^{a,o}\right).\end{aligned}$$

Il faut conserver l'opérateur ELAGAGE à l'extérieur de la somme croisée car, si tous les vecteurs maximaux sont des combinaisons de vecteurs maximaux, certaines de ces combinaisons sont tout de même inutiles.

En continuant, on obtient :

$$\begin{aligned}\Theta_n^a &= \text{ELAGAGE}\left(\bigoplus_o \Theta_n^{a,o}\right) \\ &= \text{ELAGAGE}\left(\Theta_n^{a,0} \oplus \Theta_n^{a,1} \oplus \Theta_n^{a,2} \oplus \cdots \oplus \Theta_n^{a,|\Omega|-1}\right) \\ &= \text{ELAGAGE}\left(\cdots \text{ELAGAGE}(\text{ELAGAGE}(\Theta_n^{a,0} \oplus \Theta_n^{a,1}) \oplus \Theta_n^{a,2}) \cdots \oplus \Theta_n^{a,|\Omega|-1}\right).\end{aligned}$$

Cette équation résume donc l'algorithme d'élagage incrémental. L'idée, comme le montre la figure 3.13, est d'élaguer les combinaisons de deux ensembles de vecteurs $\Theta_n^{a,0}$ et $\Theta_n^{a,1}$ (représentés par leur partition en régions) pour obtenir un ensemble élagué intermédiaire. Les vecteurs de ce nouvel ensemble sont ensuite combinés avec ceux de $\Theta_n^{a,2}$ et ainsi de suite.

Cette méthode est détaillée par l'algorithme 3.8. L'ensemble Ψ sert à stocker tous les vecteurs de $\Theta_n^{a,o}$ et les résultats intermédiaires obtenus par élagage de deux de ces vecteurs.

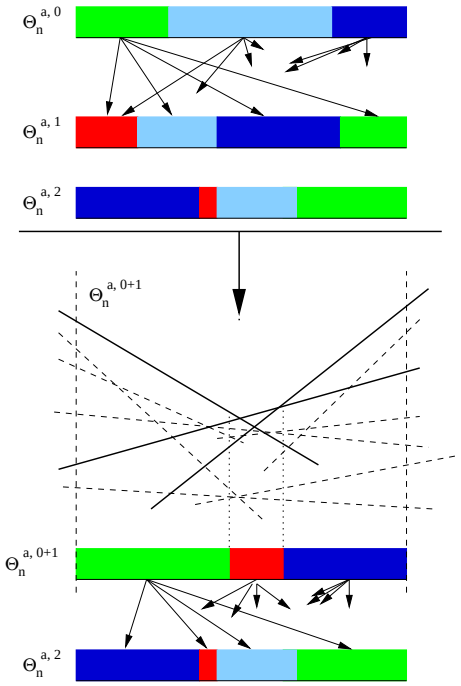

Figure 3.13. Elagage incrémental de Θ_n^a. *Pour construire une représentation parcimonieuse de Θ_n^a, l'élagage incrémental part de deux ensembles $\Theta_n^{a,0}$ et $\Theta_n^{a,1}$. L'ensemble formé par toutes les combinaisons de vecteurs issus de ces deux ensembles est ensuite élagué pour former l'ensemble parcimonieux $\Theta_n^{a,1+2}$. C'est ensuite* cet *ensemble qui est combiné avec $\Theta_n^{a,2}$ pour créer $\Theta_n^{a,0+1+2}$ et ainsi de suite pour obtenir Θ_n^a.*

Algorithme 3.8 : ElagageIncremental(Θ_{n-1}, a)

Entrées : Une représentation parcimonieuse Θ_{n-1} de V_{n-1}^*, une action a
Sorties : Une représentation parcimonieuse de $V_n^{*,a}$

$\Psi \leftarrow \bigcup_o \{\Theta_n^{a,o}\}$
tant que $|\Psi| > 1$ **faire**
  $A \leftarrow$ `EnleveElement`(Ψ)
  $B \leftarrow$ `EnleveElement`(Ψ)
  $D \leftarrow$ ELAGAGE$(A \oplus B)$
  $\Psi \leftarrow \Psi \cup \{D\}$
retourner Ψ

3.5. Algorithmes d'itération sur les politiques

A l'instar de l'algorithme d'itération sur les politiques (voir algorithme 1.5), il est possible de chercher une solution optimale à un POMDP directement dans l'espace des politiques. Reste alors à bien préciser dans quel espace se fait cette recherche.

Si on se place dans l'espace des politiques définies sur les états de croyance ($\pi : \mathcal{B} \rightarrow \mathcal{A}$), on se trouve dans un espace infini continu où il faut chercher un maximum absolu. L'algorithme proposé par Sondik (voir [SON 78]) permet, en théorie, de trouver une solution exacte pour les POMDP transitoires et une solution ϵ-optimale sinon. En revanche, son application pratique est limitée à des cas très simples au vu de la complexité de chaque itération.

Une alternative intéressante, proposée par Hansen [HAN 98b], s'appuie sur le fait que les politiques des POMDP transitoires peuvent être représentées par des automates avec un nombre fini d'états, comme celui de la figure 3.3. Cette représentation permet de lever la principale difficulté de l'algorithme de Sondik car ce dernier, qui s'appuie sur un découpage de $\mathcal{B}$ en région, doit transformer cette représentation en un contrôleur à états finis équivalent, ce qui est extrêmement coûteux. Hansen choisit de travailler directement avec des contrôleurs à états finis.

Le cœur de l'algorithme repose sur une mise à jour du contrôleur à l'aide de l'opérateur de la programmation dynamique, ce qui permet d'améliorer les performances de la politique.

Sondik a montré que la fonction de valeur d'une politique exprimée comme un contrôleur à états finis est linéaire par morceaux [SON 78]. Soit un tel contrôleur δ, appelons V^δ sa fonction de valeur, qui peut donc être décrite par un ensemble de vecteurs $\{\theta^i\}$, avec exactement un vecteur par nœud i du contrôleur. A chaque vecteur θ^i, lié au nœud i, on peut associer une action $a(i)$ et une transition vers le nœud $l(i,o)$ (ou le vecteur $\theta^{l(i,o)}$) pour chaque observation o. Cette fonction de valeur vérifie l'équation suivante pour chaque nœud i du contrôleur et chaque état s du POMDP :

$$\theta^i(s) \quad = \quad r(s,a(i)) + \gamma \sum_{s',o} \Pr(s'|s,a(i)) \Pr(o|s') \theta^{l(i,o)}. \tag{3.24}$$

En appliquant l'opérateur de la programmation dynamique (voir paragraphe 3.4.1) sur les vecteurs de V^δ, on obtient une nouvelle fonction de valeur $\hat{V}^\delta$. Il est facile de montrer que chaque vecteur $\hat{\theta}^j$ de $\hat{V}^\delta$ est associé à une action $a(j)$ et, pour chaque observation o, à une transition $\hat{l}(j,o)$ vers un vecteur $\theta^{\hat{l}(j,o)}$ de V^δ. Ces vecteurs $\hat{\theta}^j$ de $\hat{V}^\delta$ peuvent être des copies de vecteurs de δ (même action et mêmes liens). Ils peuvent aussi être de nouveaux vecteurs et vont permettre de modifier V^δ (et donc le contrôleur), de la manière suivante :

– si $\hat{\theta}^j$ domine un vecteur θ^i de V^δ, on associe au nœud i l'action et les transitions du nœud j ;

– sinon, on ajoute le nœud j à δ.

Il faut enfin enlever de δ tous les nœuds qui n'ont pas de vecteur associé dans $\hat{V}^\delta$ mais qui ne peuvent être atteints par un des autres nœuds de δ auquel est associé un vecteur de $\hat{V}^\delta$. Cette série d'opérations permet de définir un nouveau contrôleur $\hat{\delta}$. C'est le coeur de l'algorithme de Hansen (voir algorithme 3.9). Dans sa thèse [HAN 98a], Hansen prouve le théorème (voir théorème 3.4) qui assure qu'une itération améliore la politique, ce qui garantit la convergence vers une solution ϵ-optimale après un nombre fini d'itérations (ou, dans le cas des POMDP transitoires, vers la solution optimale).

Algorithme 3.9 : ItérationSurLesPolitiques(δ, ϵ)

Entrées : Un contrôleur à état fini δ et un réel positif ϵ
Sorties : Un contrôleur à état fini δ^* qui est ϵ-optimal

répéter
  Calculer V^δ à partir de δ en résolvant les équations (3.24)
  Construire $\hat{V}^\delta \leftarrow$ `OpProgrammationDynamique`(V^δ)
  $\hat{\delta} \leftarrow \emptyset$
  pour chaque $\hat{\theta^j} \in \hat{V}^\delta$ **faire**
    si *il existe un nœud i de δ associé à $\hat{\theta^j}$ avec action et liens identiques*
    alors
      ajouter i à $\hat{\delta}$
    sinon si *il existe nœud i tq $\hat{\theta}^j$ domine θ^i* **alors**
      ajouter i a $\hat{\delta}$, avec l'action et les liens de $\hat{\theta}^j$
    sinon
      ajouter un *nouveau* nœud à $\hat{\delta}$ avec action et liens de $\hat{\theta}^j$
  Ajoute à $\hat{\delta}$ tous les autres nœuds de δ qui sont atteignables depuis $\hat{\delta}$
  $\delta \leftarrow \hat{\delta}$
jusqu'à $\|\hat{V}^\delta - V^\delta\| \leq \epsilon(1-\gamma)/\gamma$
retourner $\hat{\delta}$

THÉORÈME 3.4.– *Si un contrôleur à états fini δ n'est pas optimal, une itération de l'algorithme `ItérationSurLesPolitiques` le transforme en un contrôleur $\hat{\delta}$ dont la fonction de valeur est au moins aussi bonne pour tous les états de croyance et meilleure pour quelques états de croyance.*

3.6. Conclusion et perspectives

Les processus décisionnels de Markov partiellement observables (POMDP) permettent de modéliser et contrôler les systèmes dynamiques incertains dont l'état n'est que partiellement connu. Le contrôleur n'a pas accès à l'état du processus mais doit se contenter d'observations imparfaites de cet état. En général, il n'est pas possible de contrôler optimalement un POMDP en utilisant uniquement l'observation courante du processus. Il faut en effet avoir accumulé suffisamment d'informations, comme par exemple l'historique de toutes les observations passées, pour définir une politique optimale. Ainsi, les méthodes classiques s'appuient sur les états de croyance qui sont des résumés suffisants de l'information contenue dans les historiques d'observations. En fait, les méthodes de la programmation dynamique (itération sur les valeurs et itération sur les politiques) peuvent être appliquées sur ces états de croyance.

En pratique, les algorithmes classiques comme WITNESS ou ELAGAGE INCRÉMENTAL utilisent le fait que la fonction de valeur est linéaire par morceau et cherchent à la représenter efficacement. Cependant, ces algorithmes exacts ne peuvent être appliqués à des problèmes comportant plus d'une dizaine d'états à cause de l'accroissement potentiellement exponentiel du nombre d'éléments nécessaires pour représenter la fonction de valeur. Il en va de même d'autres algorithmes classiques que nous n'avons pas détaillés ici, comme par exemple les algorithmes de Monahan [MON 82] ou Cheng [CHE 88]. Une autre alternative est de s'intéresser à la factorisation des états et des observations, comme le font [GUE 01, HAN 00, POU 05]. Il se peut aussi que des travaux beaucoup plus récents (voir [KOL 94, ARA 07]), s'appuyant sur la programmation linéaire pour représenter une politique par l'ensemble des trajectoires qu'elle peut générer, puissent mener à des algorithmes permettant de résoudre de manière exacte des problèmes plus complexes. Pour l'instant, les gains en complexités sont relativement restreints.

Ces travaux théoriques à l'aspect pratique limité ont inspiré des algorithmes pour trouver des solutions approchées aux POMDP. Il est possible de ne chercher la fonction de valeur que pour certains points de l'espace des états de croyance en espérant que cette fonction de valeur approchée sera proche de l'optimale [PIN 03, SPA 05]. D'autres optent pour une recherche « avant » à partir d'un unique état estimé de départ pour ne calculer la fonction de valeur que sur les états de croyance atteignables (voir par exemple [BON 00]). D'autres travaux combinent la programmation dynamique avec une recherche heuristique pour, à chaque étape de l'itération sur la fonction de valeur, restreindre fortement l'ensemble des états estimé sur lesquels la fonction de valeur est calculée. Il est ainsi possible de s'attaquer à des problèmes beaucoup plus complexes [SEU 07].

Le problème le plus crucial avec les POMDP reste celui de l'apprentissage. Les méthodes d'apprentissage indirectes (apprendre d'abord un modèle avant de planifier) ne sont pas très performantes car il est difficile d'apprendre la structure cachée

d'un POMDP. Actuellement, il faut utiliser des méthodes d'apprentissage qui cherche des solutions approchées. Les travaux de [MCC 95, DUT 00] font l'hypothèse que le POMDP peut s'appréhender comme un MDP d'ordre k. D'autres travaux apprennent directement dans l'espace des politiques paramétriques en utilisant des montées de gradient pour en trouver les meilleurs paramètres [BAX 00].

Enfin, il faut mentionner les travaux récents autour de l'utilisation des *représentations par états prédictifs*[5] . [LIT 02] ayant montré qu'on peut efficacement représenter un POMDP avec ces états prédictifs, qui sont en fait les probabilités que certaines trajectoires se réalisent dans le futur, des algorithmes d'apprentissage commencent à voir le jour, aussi bien pour découvrir ces états prédictifs que pour apprendre leurs probabilités de réalisation [SIN 03, ABE 07].

3.7. Bibliographie

[ABE 07] ABERDEEN D., BUFFET O., THOMAS O., « Policy-Gradient for PSRs and POMDPs », *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS'07)*, 2007.

[ARA 07] ARAS R., DUTECH A., CHARPILLET F., « Mixed integer linear programming for exact finite-horizon planning in decentralized POMDPs », *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS'07)*, 2007.

[AST 65] ASTRÖM K., « Optimal control of Markov decision processes with incomplete state estimation », *Journal of Mathematical Analysis and Applications*, vol. 10, p. 174–205, 1965.

[BAX 00] BAXTER J., BARTLETT P., « Reinforcement Learning in POMDP's via Direct Gradient Ascent », *In Proceedings of the 17th International Conference on Machine Learning (ICML'00)*, 2000.

[BER 95] BERTSEKAS D., *Dynamic Programming and Optimal Control*, Athena Scientific, Belmont, MA, 1995.

[BON 00] BONET B., GEFFNER H., « Planning with Incomplete Information as Heuristic Search in Belief Space », *Proceedings of the 5th International Conference on Artificial Intelligence Planning Systems (AIPS'00)*, 2000.

[CAS 94] CASSANDRA A., KAELBLING L., LITTMAN M., « Acting Optimally in Partially Observable Stochastic Domains », *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI'94)*, 1994.

[CAS 98] CASSANDRA A., Exact and Approximate Algorithms for Partially Observable Markov Decision Processes, thèse de doctorat, Brown University, 1998.

[CHE 88] CHENG H.-T., Algorithms for Partially Observable Markov Decision Processes, thèse de doctorat, University of British Columbia, Canada, 1988.

5. De l'anglais *Predictive State Representation* (PSR).

[DUT 00] DUTECH A., « Solving POMDP using Selected Past-Events », *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI'00)*, 2000.

[GUE 01] GUESTRIN C., KOLLER D., PARR R., « Solving factored POMDPs with linear value functions », *Proceedings of the IJCAI-01 Workshop on Planning under Uncertainty and Incomplete Information*, 2001.

[HAN 98a] HANSEN E., Finite-Memory Control of Partially Observable Systems, thèse de doctorat, Department of Computer Science, University of Massachussets at Amherst, 1998.

[HAN 98b] HANSEN E., « An Improved Policy Iteration Algorithm for Partially Observable MDPs », *Advances in Neural Information Processing Systems 10 (NIPS'97)*, 1998.

[HAN 00] HANSEN E., FENG Z., « Dynamic Programming for POMDPs using a Factored State Representation », *Proceedings of the International Conference on Artificial Intelligence Planning and Scheduling (AIPS'00)*, 2000.

[JAA 95] JAAKKOLA T., SINGH S., JORDAN M., « Reinforcement learning algorithm for partially observable Markov decision problems. », *Advances in Neural Information Processing Systems 7 (NIPS'94)*, MIT Press, Cambridge, MA, 1995.

[KOL 94] KOLLER D., MEGIDDO N., VON STENGEL B., « Fast Algorithms for Finding Randomized Strategies in Game Trees », *Proceedings of the 26th ACM Symposium on Theory of Computing (STOC '94)*, p. 750–759, 1994.

[LIT 95] LITTMAN M., CASSANDRA A., KAELBLING L., Efficient Dynamic Programming Updates in Partially Observable Markov Decision Processes, Rapport n°CS-95-19, Brown University, 1995.

[LIT 96] LITTMAN M. L., Algorithms for Sequential Decision Making, thèse de doctorat, Computer Science Department, Brown University, 1996.

[LIT 02] LITTMAN M., SUTTON R., SINGH S., « Predictive Representations of State », *Advances in Neural Information Processing Systems 14 (NIPS'01)*, p. 1555–1561, 2002.

[MCC 95] MCCALLUM A., Reinforcement Learning with Selective Perception and Hidden State, thèse de doctorat, Department of Computer Science, University of Rochester, Rochester, NY, 1995.

[MON 82] MONAHAN G. E., « A Survey of Partially Observable Markov Decison Processes : Theory, Models and Algorithms », *Management Science*, vol. 28, n°1, p. 1–16, 1982.

[PAP 87] PAPADIMITRIOU C. H., TSITSIKLIS J. N., « The Complexity of Markov Decision Processes », *Journal of Mathematics of Operations Research*, vol. 12, n°3, p. 441–450, 1987.

[PIN 03] PINEAU J., GORDON G., THRUN S., « Point-based value iteration : An anytime algorithm for POMDPs », *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'03)*, p. 1025–1032, 2003.

[POU 05] POUPART P., Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes, thèse de doctorat, University of Toronto, 2005.

[SEU 07] SEUKEN S., ZILBERSTEIN S., « Memory-Bounded Dynamic Programming for DEC-POMDPs », *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, 2007.

[SIN 94] SINGH S., JAAKKOLA T., JORDAN M., « Learning without State Estimation in Partially Observable Markovian Decision Processes », *Proceedings of the 11th International Conference on Machine Learning (ICML'94)*, 1994.

[SIN 03] SINGH S., LITTMAN M., JONG N., PARDOE D., STONE P., « Learning Predictive State Representations », *Proceedings of the 20th International Conference of Machine Learning (ICML'03)*, 2003.

[SMA 73] SMALLWOOD R. D., SONDIK E. J., « The Optimal Control of Partially Observable Markov Processes over a Finite Horizon », *Operations Research*, vol. 21, p. 1071–1088, 1973.

[SON 71] SONDIK E., The Optimal Control of Partially Observable Markov Decision Processes, thèse de doctorat, Stanford University, California, 1971.

[SON 78] SONDIK E., « The Optimal Control of Partially Observable Markov Processes over the Infinite Horizon : Discounted Costs », *Operations Research*, vol. 26, n°2, p. 282–304, 1978.

[SPA 05] SPAAN M., VLASSIS N., « Perseus : Randomized Point-based Value Iteration for POMDPs », *Journal of Artificial Intelligence Research*, vol. 24, p. 195–220, 2005.

[WHI 91] WHITE C. C., « Partially Observed Markov Decision Processes: A Survey », *Annals of Operational Research*, vol. 32, 1991.

[ZAN 96] ZANG N., LIO W., Planning in Stochastic Domains : Problem Characteristics and Approximation, Rapport n°HKUST-CS96-31, Honk-Kong University of Science and Technology, 1996.