UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Prise de décision séquentielle dans l'incertain : Exploiter la structure et rester dans le cadre

MÉMOIRE

pour l'obtention de l'

Habilitation à diriger des recherches

(Spécialité Informatique)

présentée et soutenue publiquement le 18 décembre 2017

par

Olivier Buffet

Composition du jury

Rapporteurs :	Patrice Perny	Professeur, Université Pierre et Marie Curie
	Cédric Pralet	Maître de recherche, ONERA, Toulouse
	Bruno Zanuttini	Maître de conférences, Université de Caen Basse-Normandie
Examinateurs :	François Charpillet	Directeur de recherche, INRIA Nancy - Grand Est
	Alain Dutech	Chargé de recherche, INRIA Nancy - Grand Est
	Ammar Oulamara	Professeur, Université de Lorraine
	Olivier Teytaud	Chercheur, Google, Zürich

Laboratoire Lorrain de Recherche en Informatique et ses Applications
UMR 7503

Loria
Laboratoire lorrain de recherche en informatique et ses applications

Mis en page avec la classe thloria.

Table des matières

À Rémi et Martin,
à Fabienne, leur maman,
à mes parents.

Avertissement

L'exposé qui suit vise à décrire mes travaux passés (et futurs) de manière compréhensible, mais sans se noyer dans les détails. Le plus souvent, on notera les variables aléatoires par des majuscules, et les ensembles par des majuscules calligraphiques. Toutefois, la diversité des thématiques abordées rendrait difficile l'usage de notations et d'un formalisme complètement homogène de bout en bout, d'où parfois un certain manque de rigueur.

L'inégalité de traitement entre les sujets dépend de différents biais tels que leurs utilités relatives dans ce document, mes compétences, et mes préférences.

Chapitre 1

Introduction

Sommaire

Préambule

Ce premier chapitre vise à situer mes travaux de recherche et introduire le plan de ce document. Il me permettra aussi de mettre en avant quelques éléments qui expliquent en partie ma motivation pour les thématiques abordées. J'ai toutefois bien conscience que, pour expliquer ma motivation et justifier le chemin parcouru, il faudrait entre autres tenir compte de bien des hasards et faire un travail de l'ordre de la psychanalyse.

Un certain nombre de termes vont être ici définis. Bien souvent ces définitions sont des choix qui pourraient prêter à des discussions sur lesquelles je préfère ne pas m'attarder.

1.1 Intelligence et science

« *Intelligence* » vient du latin *intellegentia* (la faculté de comprendre), de *intellegere* (comprendre), lui-même décomposable en (i) *inter* (entre) et (ii) *legere* (choisir, cueillir). [1] L'intelligence serait donc à l'origine la faculté de faire un choix entre des choses, à distinguer une chose d'une autre, par exemple le vrai du faux. On retrouve assez bien ici l'idée que l'intelligence est la faculté de comprendre, de faire le lien entre les choses, de résoudre des problèmes. On notera que la mise en œuvre de cette faculté à souvent comme finalité d'agir, par exemple dans la résolution de problème.

La *science*, elle, peut être définie comme l'activité consistant à chercher à comprendre le monde qui nous entoure (plus que des situations particulières). Il s'agit donc de faire appel à l'intelligence pour aborder des sujets universels plus que pour répondre à des besoins propres, même si l'un n'empêche pas l'autre. Le lien entre intelligence et action est comparable au lien entre science et technique. On le retrouve dans le *Discours de la méthode* de Descartes (1637) quand ce dernier écrit que la science doit nous permettre de « *nous rendre comme maîtres et possesseurs de la nature* ». En outre, non seulement la science a un objet universel, mais elle est aussi souvent une activité collective, les scientifiques partageant leurs avancées entre eux.

1. `https://fr.wiktionary.org/wiki/intelligence` / `http://www.cnrtl.fr/etymologie/intelligence`

1.2 Sciences cognitives et intelligence artificielle

Au sein de la classification courante des sciences, les sciences cognitives ont une place particulière puisqu'elles ont pour objet les mécanismes de la pensée humaine, animale ou artificielle, donc l'intelligence elle-même. Il y a une évidente mise en abîme puisqu'on a là des activités consistant à faire œuvre d'intelligence pour comprendre l'intelligence (comme illustré de manière abstraite par la figure 1.1, page 4). À titre informatif, Miller (2003) cite les six disciplines suivantes comme clairement identifiées au sein des sciences cognitives en 1978 : la psychologie, la philosophie, la linguistique, l'anthropologie, les neurosciences, et l'intelligence artificielle. C'est à cette dernière, l'intelligence artificielle (IA), que nous allons désormais nous intéresser plus particulièrement.

Ici, on restreindra l'intelligence artificielle au problème de concevoir des systèmes capables le mieux possible de comprendre des choses (de construire des connaissances) ou de résoudre des problèmes. En un sens, il s'agit de mettre la question de l'intelligence en équations, sous la forme d'un problème mathématique. On met ainsi sciemment de côté — sans nier leur pertinence — les travaux visant

— à reproduire le fonctionnement de l'intelligence naturelle (laquelle peut toutefois être source d'inspiration) ; ou

— à accomplir des tâches comparables à celles rencontrées dans la nature ; idéalement on voudrait accomplir tout type de tâches, et peut-être un jour voir un éléphant artificiel jouer aux échecs (Brooks, 1990).

Ce choix nous éloigne de l'intelligence artificielle telle qu'évaluée par exemple dans le test de Turing (1950). Ce point de vue soulève évidemment des interrogations. Est-ce que toute activité intelligente peut être ainsi abordée ? Toute intelligence naturelle est-elle le produit d'une machine ? [2] Sinon, quelles sont les limites de cette mise en équation ? Qu'y a-t-il de propre à l'intelligence naturelle qui ne peut être retrouvé dans l'intelligence artificielle ?

Le plus souvent, plutôt que de travailler sur des machines en général (des combinaisons de composants mécaniques, électroniques ou optiques), on préfère travailler avec des algorithmes, ce qui permet de considérer tout calcul possible à l'aide de machines de Turing universelles (Turing, 1936). Mais ces algorithmes vont être exécutés sur des supports physiques et donc subir les contraintes inhérentes à ceux-ci : par exemple, vitesse d'exécution, mémoire et capacités perceptives d'une machine, comme d'un être vivant, sont limitées. L'intelligence artificielle va donc avoir un soucis de réalisme par rapport à ces contraintes, entrant alors dans le domaine de la *rationalité limitée* (Russell, 1997; Russell et Norvig, 2010).

En aparté, on notera qu'un algorithme n'est jamais qu'une procédure, une méthode à appliquer de manière automatique/systématique. Or, si l'on peut dire qu'une méthode est intelligente ou astucieuse, son application « bête » n'est pas ce que l'on appellerait couramment un signe d'intelligence. Peut-être se trompe-t-on donc en parlant d'intelligence artificielle dans ce cas. [3] Ce paradoxe est connu sous le nom d'« effet IA », [4] et est parfois aussi formulé en disant que, dès lors qu'un algorithme résout un problème, celui-ci n'est plus un problème d'intelligence artificielle. À titre de comparaison, on pourrait dire qu'un tour de magie n'en est plus un dès lors que l'on connaît son « truc ». Mais l'intelligence et la magie disparaissent-ils quand on sait les expliquer ? [5]

"It's still magic even if you know how it's done." – Terry Pratchett, *A Hat Full of Sky*

Mais revenons à ce que cherche l'intelligence artificielle. Pour concevoir un système réalisant des tâches intelligentes, nous allons avoir besoin de spécifier ces tâches. S'il s'agit d'apprendre quelque chose sur la base d'exemples, il va falloir fournir en entrée les exemples, et préciser ce que l'on entend par « apprendre ». Apprendre peut signifier approcher au mieux (au sens d'un critère particulier) une fonction f à l'aide d'exemples de la forme $(x, f(x))$. S'il s'agit de trouver une solution satisfaisante à un problème, il peut être question d'un problème dont les solutions sont exprimées à l'aide d'un vecteur de nombres entiers ou réels, et qui s'exprime par des égalités et inégalités qui doivent être satisfaites par les vecteurs solutions.

2. Cela rejoindrait le mécanisme en philosophie, c'est-à-dire la perception de la plupart des phénomènes comme suivant le modèle des liens de cause à effet.

3. Peut-être devrait-on réserver le terme d'intelligence artificielle à un système automatique capable d'analyser une situation nouvelle, un problème nouveau, pour proposer une méthode de résolution adaptée. Mais n'est-ce pas déplacer (inutilement) le problème que de raisonner ainsi ?

4. `https://en.wikipedia.org/wiki/AI_effect`

5. Et un phénomène naturel est-il moins admirable parce que la science sait l'expliquer ?

On notera qu'on est ici déjà en train de restreindre l'intelligence à des tâches particulières. On ne cherchera pas ici à concevoir un algorithme capable de tout faire. Par contre on cherchera à concevoir un algorithme capable d'effectuer un type de tâche en général (de résoudre une famille de problèmes). Concevoir un algorithme capable de trouver une trajectoire pour un robot mobile (quand il en existe une), quels que soient l'environnement et les points de départ et d'arrivée, n'est pas trouver une trajectoire pour un unique environnement et d'uniques points de départ et d'arrivée (auquel cas on peut pré-calculer la solution).

Pour concevoir un algorithme intelligent, on va donc définir (formellement) une famille de problèmes, l'analyser, et chercher un algorithme dont on est capable de montrer qu'il résout ce problème. On va aussi analyser cet algorithme pour identifier ses propriétés utiles : sa complexité (besoins de mémoire et de temps en fonction des caractéristiques du problème), son comportement au cours du temps s'il converge progressivement vers une solution, ... On va pouvoir se demander par ailleurs s'il peut être transformé pour modifier ses propriétés, en faisant par exemple en sorte que sa complexité computationnelle soit globalement meilleure ou simplement différente (c'est-à-dire de manière à ce que l'algorithme soit meilleur dans différentes parties de l'espace des problèmes considérés).[6] Dans certains cas, la recherche d'un algorithme satisfaisant amènera à établir le meilleur compromis entre les bonnes propriétés souhaitées d'une part, et l'acceptabilité des hypothèses faites sur le problème d'autre part. De manière réciproque, on peut se demander ce qui pourrait se passer si un algorithme est appliqué alors que certaines de ses hypothèses ne sont pas vérifiées (une tentation naturelle dans bien des cas). Quelles propriétés sont perdues ou préservées ?

1.3 Mes travaux sur la prise de décision séquentielle

1.3.1 De quoi s'agit-il ?

Comme on l'a évoqué, quand l'intelligence n'est pas spéculative, elle agit. Mes travaux s'intéressent ainsi à la conception d'intelligences agissantes, aux prises avec leur monde extérieur. On parle alors d'*agent rationnel* en interaction avec son environnement, c'est-à-dire capable de le percevoir et d'agir sur celui-ci. Plus précisément, mes travaux considèrent des cadres dynamiques, dans lesquels l'agent doit prendre une séquence de décisions pour contrôler l'état du système qu'il constitue avec l'environnement. On parle alors de *prise de décision séquentielle*. Cela recouvre à la fois la *planification d'actions* dans un environnement dont on connaît la dynamique (les règles de fonctionnement), mais aussi l'*apprentissage par renforcement*, c'est-à-dire l'apprentissage du comportement à adopter par essai-erreur.

Souvent, il faut tenir compte d'incertitudes pour traiter de tels problèmes. Il peut s'agir d'incertitudes inhérentes à la dynamique du système (à supposer qu'elle est par nature stochastique et non déterministe). Il peut s'agir d'incertitudes liées aux capacités perceptives limitées, l'agent n'ayant à chaque instant qu'une vue partielle et bruitée de l'état réel de l'environnement. Il peut s'agir d'incertitudes introduites dans le modèle pour en réduire la complexité.

Mes travaux s'intéressent ainsi principalement à la prise de décision séquentielle dans l'incertain, problématique rencontrée dès le DEA et à laquelle je reste attaché. Elle est à mon sens assez large et permet d'aborder de nombreux problèmes scientifiques et pratiques. En outre, une autre raison de mon intérêt pour la prise de décision dans l'incertain est la difficulté que j'éprouve moi-même à prendre des décisions dans l'incertain.[7]

1.3.2 Orientation de mes travaux de recherche

Comme on va le voir dans la présentation des fondements scientifiques de mes travaux (chapitre 2), un formalisme, celui des *processus de décision markoviens* (MDP), permet assez simplement (en théorie

6. À ce sujet, le « no free lunch theorem » (selon lequel il n'existe pas, pour une classe de problème, un algorithme qui soit meilleur que les autres pour toutes sous-classes) nous garantit en un sens qu'il y a toujours moyen de spécialiser un algorithme en s'intéressant plus particulièrement à une sous-classe restreinte.

7. En cela je rejoins la logique du corsaire Robert Surcouf, auquel un capitaine d'un bâtiment de la Royale Navy s'adressa ainsi : *« Vous, Français, vous vous battez pour l'argent. Tandis que nous, Anglais, nous nous battons pour l'honneur ! »*, Surcouf lui répliquant : *« Chacun se bat pour ce qui lui manque. »*

FIGURE 1.1 – « Exposition d'estampes », lithographie de M.C. Escher, 1956. La position de la signature répond astucieusement à la question de ce qui se passe en ce point particulier de l'image. Une réponse plus convaincante a été apportée par de Smit et Lenstra Jr. (2003).

au moins) de modéliser et résoudre un grand nombre de problèmes de prise de décision séquentielle dans l'incertain. Comme on le verra ensuite, des difficultés apparaissent

1. parce que nombre de problèmes intéressants, s'ils entrent dans le cadre des MDP, requièrent trop de ressources computationnelles pour être résolus à l'aide des algorithmes « élémentaires », et
2. parce que nombre de problèmes intéressants sortent du cadre des MDP, posant *a priori* de nouvelles questions.

S'il y a une certaine diversité dans mes travaux, on peut y voir une grande tendance, celle de chercher (1) si le problème à résoudre a des caractéristiques (/une structure) particulière, et (2), le cas échéant, si on peut exploiter ces caractéristiques pour résoudre ce problème plus efficacement. Je me suis d'abord intéressé à cette problématique générale dans le cadre de « simples » MDP.

Plus récemment, j'ai étudié des situations sortant du cadre des MDP, mais qui peuvent y être replongées (comme illustré de manière abstraite par la figure 1.1). Dans ce cas, on cherche typiquement un plongement accompagné de propriétés facilitant (encore une fois) la résolution.

Ces travaux conduisent à proposer une vision généralisée des MDP et de leurs extensions. Une démarche pour résoudre une variété de problèmes sortant *a priori* du cadre des MDP simples apparaît tout en exploitant leurs caractéristiques propres.

1.4 Organisation du mémoire

Comme déjà annoncé plus haut, ce mémoire va d'abord présenter au chapitre 2 les fondements scientifiques sur lesquels reposent la majeure partie de mes travaux : le formalisme des processus de décision markoviens et certaines de ses extensions. Mes propres contributions sont ensuite organisées en trois ensembles :

- le chapitre 3, sur la recherche et l'exploitation de structure dans les MDP, mais aussi sur la résolution de MDP par optimisation directe d'un contrôleur (par opposition à l'emploi de la programmation dynamique) ;
- le chapitre 4, sur la résolution de POMDP par optimisation directe d'un contrôleur, sur la résolution d'extensions de POMDP en se ramenant à un POMDP, et sur une approche optimiste pour l'apprentissage par renforcement bayésien ; et
- le chapitre 5, sur des travaux s'éloignant des MDP, mais mettant en jeu des systèmes dynamiques, des algorithmes de recherche, ou la théorie de l'utilité, c'est-à-dire trois sujets récurrents dans mes travaux.

Vient ensuite le chapitre 6, dans lequel je présente des directions pour mes futures recherches.

Aparté : Dans l'inventaire à la Prévert (1946) de travaux qui suit, je me permets d'attirer l'attention en particulier vers les sections 3.3, 4.3.2 et 4.3.3.

Chapitre 2

Fondements scientifiques

Sommaire

Ce chapitre présente les fondements scientifiques de la majeure partie de mes travaux de recherche, c'est-à-dire le cadre des processus de décision markoviens (MDP) et ses extensions. Il aborde les bases théoriques des MDP simples en section 2.1, des approches possibles quand les MDP grossissent trop en section 2.2, et enfin des travaux portant sur des modèles étendant les MDP en section 2.3.

2.1 Les « petits » MDP

Nous allons voir ici brièvement ce que sont les MDP, et quelles principales approches exactes de résolution existent.

2.1.1 Qu'est-ce qu'un MDP ?

On souhaite contrôler des systèmes (à temps discret) dont la dynamique est incertaine, mais sur laquelle on a une influence (un moyen de contrôle), ce qui amène naturellement à voir ce système comme une chaîne de Markov dont la dynamique dépend du choix d'une action. En outre, pour spécifier un objectif, on va associer une utilité aux états visités et aux actions effectuées.

Tout cela conduit à employer le cadre des *processus de décision markoviens* (MDP) (Bellman, 1957; Puterman, 1994; Bertsekas et Tsitsiklis, 1996). Un MDP se formalise comme un tuple $\langle \mathcal{S}, \mathcal{A}, P, \mathcal{R} \rangle$ où

— $\mathcal{S}$ est un ensemble (fini) d'états du système ;
— $\mathcal{A}$ est un ensemble (fini) d'actions possibles ;
— $P_{\cdot,\cdot}$ est la fonction de transition du système ; $P_{s,s'}(a) = Pr(s'|s,a)$ est la probabilité que le système transite vers l'état $s' \in \mathcal{S}$ quand l'action $a \in \mathcal{A}$ est effectuée depuis l'état $s \in \mathcal{S}$; et
— $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ est une fonction de récompense, laquelle associe à toute transition un scalaire représentant l'utilité immédiate de cette transition.

La figure 2.1 montre un modèle graphique représentant un MDP.

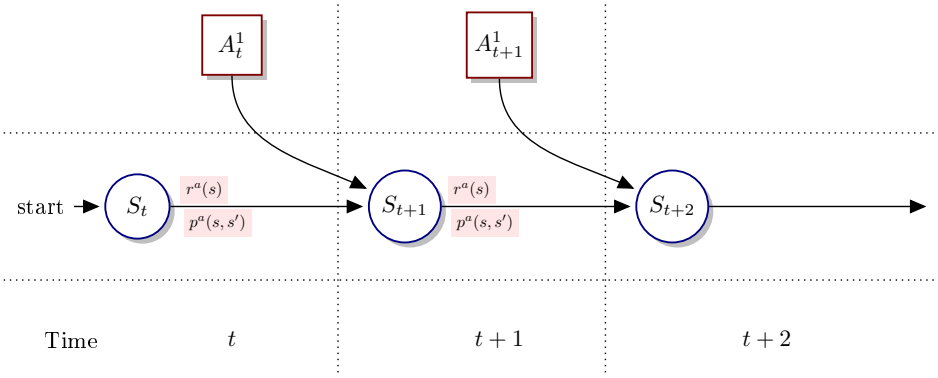


FIGURE 2.1 – Modèle graphique d'un MDP

Le plus souvent, et sans perte de généralité, on considère que la fonction de récompense ne dépend pas de l'état d'arrivée. En outre, une récompense peut être négative pour représenter un coût.

L'incertitude des transitions peut être liée à la nature du système considéré, mais aussi à la simplification de son modèle (par manque de connaissances ou pour le rendre accessible). Dans tous les cas, on suppose que la dynamique est markovienne, c'est-à-dire que la connaissance de l'état courant et de l'action courante sont suffisants pour prédire au mieux le prochain état :

$$\forall s_0, a_0, \ldots, s_t, a_t, s_{t+1},\ Pr(s_{t+1}|s_t, a_t) = Pr(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_0, a_0).$$

Par ailleurs, la définition fournie ici ne décrit pas un problème à résoudre. On n'a spécifié ni l'objectif, ni la forme des solutions. Il y a en fait une variété de *problèmes de décision markoviens.*[8] Les sections suivantes présentent les instances les plus usuelles et leur résolution.

2.1.2 Le principe d'optimalité de Bellman et son application à horizon temporel fini

Considérons d'abord un MDP auquel on ajoute la contrainte que seul un *horizon temporel* fini T est considéré. L'objectif est alors de contrôler ce système entre les instants $t = 0$ et $t = T$, quel que soit l'état de départ s_0, typiquement de manière à maximiser l'espérance de la somme des récompenses cumulées :

$$\mathbb{E}[R_0 + R_1 + \cdots + R_{T-1}|s_0],$$

où les R_t sont les variables aléatoires associées à la fonction de récompense.

On a là un problème d'optimisation bien posé. La solution cherchée doit indiquer comment choisir une action a à effectuer quel que soit non seulement l'état atteint s, mais aussi le pas de temps t (parce que les possibilités de récompenses à long terme ne sont pas les mêmes selon le temps restant). On peut prouver que, sans perte d'optimalité, ce choix peut être effectué de manière déterministe, sans échantillonner une action. Une solution, appelée *politique* $\pi_{0:T-1}$, est alors la donnée d'une fonction $\pi_t : \mathcal{S} \to \mathcal{A}$ pour chaque $t \in \{0, \ldots, T-1\}$.

La fonction donnant l'espérance des récompenses futures cumulées à partir d'un état s à un instant t et en suivant une politique $\pi_{0:T-1}$ est appelée *valeur* de la politique. Pour l'instant t, on la notera ici $V_{\pi_{t:T-1}}(s)$ parce qu'elle ne dépend que des décisions prises à partir de cet instant. Pour $t \in \{0, \ldots, T-1\}$, et en définissant $V_{\pi_{T:T-1}}(s) \stackrel{\text{def}}{=} 0$, on a en effet :

$$\begin{aligned}
V_{\pi_{t:T-1}}(s) &\stackrel{\text{def}}{=} \mathbb{E}_{\pi_{0:T-1}}[R_t + R_{t+1} + \ldots + R_{T-1}|S_t = s] \\
&= \mathbb{E}_{\pi_{t:T-1}}[R_t + R_{t+1} + \ldots + R_{T-1}|S_t = s] \\
&= \mathbb{E}_{\pi_{t:T-1}}[R_t|S_t = s] + \mathbb{E}_{\pi_{t:T-1}}[R_{t+1} + \ldots + R_{T-1}|S_t = s] \\
&= \mathbb{E}_{\pi_{t:t}}[R_t|S_t = s] + \sum_{s' \in \mathcal{S}} P_{s,s'}(\pi_t(s))\mathbb{E}_{\pi_{t:T-1}}[R_{t+1} + \ldots + R_{T-1}|S_{t+1} = s'] \\
&= r(s, \pi_t(s)) + \sum_{s' \in \mathcal{S}} P_{s,s'}(\pi_t(s))V_{\pi_{t+1:T-1}}(s').
\end{aligned}$$

On pourra écrire ce calcul récursif sous une forme vectorielle en introduisant l'opérateur L_π :

$$\begin{aligned}
V_{\pi_{t:T-1}} &= \vec{r}_{\pi_t} + P_{\pi_t} V_{\pi_{t+1:T-1}} \\
&= L_{\pi_t} V_{\pi_{t+1:T-1}},
\end{aligned}$$

où $\vec{r}_{\pi_t}$ est le vecteur des récompenses pour chaque état quand la politique instantanée π_t est appliquée, et P_{π_t} est la matrice des probabilités de transition pour cette même politique.

Chercher une politique optimale pour un état de départ s en particulier consiste à trouver une politique

8. On confondra souvent les notions de processus de décision markovien et de problème de décision markovien.

maximisant $V_{\pi_{0:T-1}}(s)$, d'où :

$$\begin{aligned}
V_0^*(s) &\stackrel{\text{def}}{=} \max_{\pi_{0:T-1}} V_{\pi_{0:T-1}}(s) \\
&= \max_{\pi_{0:T-1}} \left[r(s, \pi_0(s)) + \sum_{s' \in \mathcal{S}} P_{s,s'}(\pi_0(s)) V_{\pi_{1:T-1}}(s') \right] \\
&= \max_{\pi_0} \max_{\pi_{1:T-1}} \left[r(s, \pi_0(s)) + \sum_{s' \in \mathcal{S}} P_{s,s'}(\pi_0(s)) V_{\pi_{1:T-1}}(s') \right] \\
&= \max_{\pi_0} \left[r(s, \pi_0(s)) + \sum_{s' \in \mathcal{S}} P_{s,s'}(\pi_0(s)) \max_{\pi_{1:T-1}} \left[V_{\pi_{1:T-1}}(s') \right] \right] \\
&= \max_{\pi_0(s)} \left[r(s, \pi_0(s)) + \sum_{s' \in \mathcal{S}} P_{s,s'}(\pi_0(s)) V_1^*(s') \right].
\end{aligned}$$

On a ainsi naturellement défini la fonction de valeur commune à toutes les politiques optimales, et cette définition fournit un algorithme récursif pour la calculer. Ce calcul, identique quel que soit l'état initial choisi et quel que soit le pas de temps, repose sur le *principe d'optimalité de Bellman* :

> *An optimal policy has the property that whatever the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decisions. (Bellman, 1954)*

Une fois ce calcul fait, il ne reste, pour obtenir une politique optimale π^*, qu'à suivre des actions gourmandes par rapport à la fonction de valeur calculée, c'est-à-dire permettant d'atteindre chaque maximum.

Implémenté de manière récursive, ce calcul aura l'inconvénient de passer de nombreuses fois par les mêmes états si plusieurs chemins sont possibles entre deux états à deux instants différents. La *programmation dynamique* est une procédure plus efficace consistant à appliquer ces formules du futur vers le passé, calculant la fonction V_t^* en fonction de la fonction V_{t+1}^*.

L'algorithme 1 détaille ce mécanisme en introduisant la *fonction d'action-valeur*

$$Q_{\pi_{t:T-1}}(s, a) = r(s, a) + \sum_{s' \in \mathcal{S}} P_{s,s'}(a) V_{\pi_{t+1:T-1}}(s')$$

et surtout sa version optimale Q_t^*.

Algorithme 1 : Programmation dynamique à horizon temporel fini

pour chaque $s \in \mathcal{S}$ **faire**
  $V_T^*(s) \leftarrow 0$
pour chaque $t \in T-1, \ldots, 0$ **faire**
  pour chaque $s \in \mathcal{S}$ **faire**
    pour chaque $a \in \mathcal{A}$ **faire**
      $Q_t^*(s, a) \leftarrow r(s, a) + \sum_{s' \in \mathcal{S}} P_{s,s'}(a) V_{t+1}^*(s')$
    $\pi_t^*(s) \in \operatorname{argmax}_{a \in \mathcal{A}} Q_t^*(s, a)$
    $V_t^*(s) \leftarrow \max_{a \in \mathcal{A}} Q_t^*(s, a)$
retourner π^*

2.1.3 Passage à l'horizon temporel infini

Des difficultés apparaissent dans le cas d'un horizon temporel infini.

La première difficulté est que le critère total (somme des récompenses cumulées) employé précédemment risque de conduire à des sommes divergentes. Un autre critère possible est le critère moyen, lequel

vise à maximiser la récompense moyenne par pas de temps. On va toutefois ici, comme souvent, lui préférer le critère pondéré, lequel modifie le critère total en pondérant les récompenses futures par un terme décroissant exponentiellement. Plus formellement, pour un *facteur d'atténuation* $\gamma \in [0, 1[$ donné, on cherche à optimiser, pour tout s, $\mathbb{E}_\pi \left[\sum_{\tau=0}^{\infty} \gamma^\tau R_\tau | S_t = s\right]$.

La seconde difficulté est qu'on ne peut pas exécuter l'algorithme 1 (même en y ajoutant le facteur γ) pour un horizon temporel infini, parce que cela demanderait un temps de calcul et une mémoire infinis.

Utilisons désormais l'indice k (au lieu de t), numéroté de 0 à l'infini (au lieu de $T = \infty$ à 0), et notons que la programmation dynamique consiste à appliquer de manière répétée l'*opérateur d'optimalité de Bellman* L défini, en notation vectorielle, par :

$$L : \begin{array}{ccc} \mathbb{R}^{|\mathcal{S}|} & \to & \mathbb{R}^{|\mathcal{S}|} \\ V & \mapsto & \max_\pi \left[r_\pi + \gamma P_\pi V\right], \end{array}$$

où r_π est le vecteur des récompenses pour chaque état quand la politique instantanée (pour un seul pas de temps) π est appliquée, et P_π est la matrice des probabilités de transition pour cette même politique. Or, parce que $\gamma \in [0, 1[$, l'opérateur L est contractant, donc la séquence des V_k obtenue par application récursive de L converge vers l'unique point fixe V^* solution de $V = LV$. On en déduit qu'une politique optimale stationnaire existe, laquelle agit de manière gourmande par rapport à V^*.

Ce résultat amène directement à l'algorithme d'*itération sur la valeur*, qui consiste à appliquer L de manière répétée jusqu'à ce que l'erreur résiduelle soit inférieure à un seuil $\epsilon > 0$:

$$\begin{aligned} \|V_k - V_{k-1}\|_\infty &= \max_{s \in \mathcal{S}} |V_k(s) - V_{k-1}(s)| \\ &< \epsilon. \end{aligned}$$

Alors, la politique π_k gourmande par rapport à V_k garantit (Williams et Baird, 1993) :

$$\|V_{\pi_k} - V^*\|_\infty < \frac{2\gamma}{1-\gamma}\epsilon.$$

En pratique, on emploie plutôt l'algorithme 2, une variante plus rapide qui utilise une nouvelle valeur dès qu'elle est calculée.

Algorithme 2 : Algorithme d'itération sur la valeur – Gauss-Seidel

pour chaque $s \in \mathcal{S}$ **faire**
  $V_0(s) \leftarrow 0$
$k \leftarrow 0$
répéter
  pour chaque $i \leftarrow 1$ **à** $|\mathcal{S}|$ **faire**
    $V_{k+1}(s_i) = \max_{a \in \mathcal{A}} \Big\{ r(s,a) + \gamma \sum_{1 \le j < i} p(s_j \mid s, a) V_{k+1}(s_j) \; + \; \gamma \sum_{i \le j \le |\mathcal{S}|} p(s_j \mid s, a) V_k(s_j) \Big\}$
  $k \leftarrow k + 1$
jusqu'à $\|V_{k+1} - V_k\| < \epsilon$
pour chaque $s \in \mathcal{S}$ **faire**
  $\pi(s) \in \operatorname{argmax}_{a \in \mathcal{A}} \{ r(s,a) + \gamma \sum_{s'} p(s'|s,a) V_k(s') \}$
retourner V_k, π

On citera brièvement deux autres types d'approches reposant sur le calcul de la fonction de valeur optimale :

la programmation linéaire, avec l'objectif de minimiser $\sum_{s \in \mathcal{S}} V(s)$ sous la contrainte que, pour tout $(s, a) \in \mathcal{S} \times \mathcal{A}$, $V(s) \ge r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P_{s,s'}(a) V(s')$; et

l'algorithme d'itération sur la politique, lequel consiste à alternativement (1) calculer la fonction de valeur V_k de la politique courante π_k, et (2) en déduire la prochaine politique π_{k+1} gourmande par rapport à V_k, comme illustré par l'algorithme 3.

La fonction de valeur V_π d'une politique stationnaire π s'obtient en résolvant $V_\pi = L_\pi V_\pi$, où L_π, l'opérateur de Bellman associé à π, est défini par :

$$L_\pi : \begin{array}{ccc} \mathbb{R}^{|\mathcal{S}|} & \to & \mathbb{R}^{|\mathcal{S}|} \\ V & \mapsto & r_\pi + \gamma P_\pi V. \end{array}$$

Cette résolution peut par exemple se faire par convergence vers un point fixe, ou en calculant $(I - \gamma P_\pi)^{-1} r_\pi$ (ce qui requiert une coûteuse inversion de matrice).

Algorithme 3 : Algorithme d'itération sur la politique

Initialiser π_0 aléatoirement
$k \leftarrow 0$
répéter
  Résoudre

$$V_k(s) = r(s, \pi_k(s)) + \gamma \sum_{s' \in S} p(s' \mid s, \pi_k(s)) V_k(s'), \quad \forall s \in \mathcal{S}$$

  pour chaque $s \in \mathcal{S}$ **faire**
    $\pi_{k+1}(s) \in \operatorname{argmax}_{a \in \mathcal{A}} \left\{ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s, a) V_k(s') \right\}$
  $k \leftarrow k + 1$
jusqu'à $\pi_k = \pi_{k+1}$
retourner V_k, π_{k+1}

Nous ne nous attarderons pas plus ici sur ces trois approches traitant le cas de l'horizon temporel infini et sur leurs propriétés respectives. On précisera seulement (1) que leurs complexités respectives sont polynomiales en le nombre d'états et d'actions (entre autres paramètres usuels), et (2) que des variantes de chacune existent, entre autres conçues pour traiter plus efficacement certaines classes de MDP.

2.2 Les MDP « trop gros »

Les algorithmes vus dans la section précédente permettent de traiter des problèmes de décision markoviens simples. Toutefois la taille des problèmes rencontrés peut croître très rapidement. Si on décrit les états possibles du système par plusieurs variables d'état multi-valuées (par exemple les angles des articulations d'un robot), la taille de $\mathcal{S}$ est exponentielle en le nombre de ces variables. Et la même croissance exponentielle s'observe sur $\mathcal{A}$ si les actions possibles s'écrivent comme la combinaison d'actions élémentaires (par exemple les forces appliquées sur les actionneurs d'un robot). On peut aussi faire face à des problèmes dans lesquels l'ensemble des états ou des actions est continu ou nécessiterait une discrétisation très fine.

Les problèmes posés par ces MDP de grande taille sont de plusieurs ordres. Le temps de convergence devient prohibitif. Il peut ne plus être possible de mémoriser la fonction de valeur calculée ou la politique, voire ne même plus être envisageable de seulement énumérer tous les couples états-actions.

Une grande partie des travaux de recherche sur les MDP s'attache ainsi à résoudre efficacement de tels MDP de grande taille. En réponse au « no free lunch theorem », [9] nombre d'entre eux s'attachent à exploiter des hypothèses particulières. La présente section décrit trois grandes classes d'approches : la programmation dynamique approchée (sec. 2.2.1), l'optimisation directe d'un contrôleur (sec. 2.2.2), et les recherches heuristiques ou arborescentes (sec. 2.2.3). En complément sont brièvement évoqués

9. Comme déjà dit en note en bas de la page 3, selon celui-ci, il n'existe pas, pour une classe de problème, un algorithme qui soit meilleur que les autres pour toutes sous-classe.

des travaux visant à exploiter la structure inhérente aux problèmes considérés pour les résoudre plus efficacement (sec. 2.2.4).

2.2.1 La programmation dynamique approchée

Une première idée assez naturelle est de tenter d'adapter les algorithmes de référence pour la résolution de MDP de « petite taille », et en particulier le principe de la programmation dynamique. On cherche donc encore à s'approcher au mieux d'une fonction de valeur optimale, pour ensuite pouvoir agir de manière gourmande par rapport à celle-ci (Munos, 2008). Se posent alors deux questions interdépendantes : Comment calculer cette fonction de valeur ? Et comment la représenter de manière compacte ?

2.2.1.1 Approximation de la fonction de valeur

En l'absence de propriétés particulières des fonctions de valeur rencontrées, on va devoir se restreindre à un sous-ensemble $\mathscr{F}$ des fonctions possibles en recourant à un approximateur de fonction $\mathscr{A}$ qui, à toute fonction de valeur (non-approchée), associe un élément de $\mathscr{F}$.

Un cas courant est celui de l'approximation linéaire dans laquelle, étant donné un ensemble de fonctions caractéristiques (*features*) associé à l'état :

$$\begin{aligned} \phi_1 &: \mathcal{S} \to \mathbb{R} \\ &\vdots \\ \phi_m &: \mathcal{S} \to \mathbb{R}, \end{aligned}$$

une fonction de valeur est spécifiée par un vecteur $\vec{w} \in \mathbb{R}^m$ et calculée comme une combinaison linéaire de ces caractéristiques avec ce vecteur :

$$V_w(s) = \vec{\phi}(s) \cdot \vec{w}, \ \forall s \in \mathcal{S}.$$

L'image d'une fonction f par $\mathscr{A}$ se trouve en effectuant une projection sur $\mathscr{F}$, c'est-à-dire en cherchant une fonction g de $\mathscr{F}$ qui minimise une distance $\|g - f\|$.

2.2.1.2 Itération sur la valeur approché

Dans ce cadre, l'algorithme d'*itération sur la valeur approché* (AVI) consiste à calculer une séquence de fonctions de valeur en appliquant itérativement l'opérateur d'optimalité de Bellman L et l'opérateur d'approximation $\mathscr{A}$:

$$V_{k+1} = \mathscr{A} L V_k. \tag{2.1}$$

On a ainsi une réponse possible à la question de la représentation de la fonction de valeur, même si les résultats obtenus dépendent de choix importants tels que celui de l'approximateur. Mais la question du calcul de la fonction de valeur effectué dans l'équation 2.1 est encore ouverte. Comment effectuer celui-ci en pratique quand il y a trop d'états ou d'actions pour pouvoir se permettre des calculs exacts ?

Une solution est, à l'étape k :

1. d'échantillonner un ensemble de N états $(s_n)_{1 \leq n \leq N}$ selon une distribution μ ;
2. de calculer la valeur $v_n \stackrel{\text{def}}{=} LV_k(s_n)$ de chacun d'entre eux ; et
3. d'employer un algorithme d'apprentissage supervisé avec les données d'apprentissage $\{(s_n, v_n)_{1 \leq n \leq N}\}$ qui retournera une fonction V_{k+1} de $\mathscr{F}$ minimisant l'erreur empirique.

Un problème est que, en utilisant AVI, on perd la garantie de convergence vers un point fixe. La suite générée peut osciller, voire diverger. Même si la fonction de valeur optimale était dans $\mathscr{F}$, on ne la trouverait pas nécessairement. Les résultats les plus généraux pour l'analyse d'AVI dépendent de l'*erreur d'approximation* $\|LV - \mathscr{A}LV\|_\infty$, c'est à dire de la pire erreur sur $\mathcal{S}$, laquelle est généralement difficile à contrôler.

2.2.1.3 Itération sur la politique approché

En très bref, l'algorithme d'*itération sur la politique approché* (API) remplace, dans PI, l'évaluation exacte d'une politique par une évaluation approchée. Différentes méthodes existent pour cela, telles que :
— les méthodes itératives (analogues à l'algorithme AVI),
— les méthodes par résolution d'un système linéaire,
— les méthodes de Monte-Carlo,
— les méthodes par différences temporelles (TD), et
— les méthodes par moindres carrés.

Bilan

Comme le montre cette rapide introduction, la programmation dynamique approchée fournit une variété de solutions pour traiter des MDP de grande taille, et a des liens forts avec la théorie de l'approximation de fonctions. En pratique, un certain nombre de succès ont été obtenus avec ces outils, même sans avoir toujours les garanties théoriques fortes que l'on pourrait souhaiter.

2.2.2 L'optimisation directe d'un contrôleur

Une limitation importante de la programmation dynamique approchée est le besoin de trouver, pour un MDP donné, un ensemble de fonctions $\mathscr{F}$ proche de la fonction de valeur optimale visée, et même des différentes fonctions de valeur calculées au cours de l'exécution d'AVI ou API. D'ailleurs, une faible erreur d'approximation peut amener à un changement important de politique.

Il est *a priori* plus simple (1) de trouver quelle est la meilleure action dans chaque état que (2) de connaître la Q-valeur optimale pour chaque couple état-action. En effet traiter ce deuxième problème fournit une solution au premier (alors que l'inverse est faux). Cette observation pourrait expliquer, par exemple, que les meilleurs algorithmes joueurs de Tetris ne reposent pas sur la programmation dynamique approchée (Thiéry, 2010). Aussi, il est parfois plus facile de trouver un espace de contrôleurs (politiques) dans lequel la recherche d'une bonne politique se fera efficacement.

Travailler ainsi directement dans un sous-ensemble de politiques revient à se ramener à un problème d'optimisation où l'on doit typiquement trouver un vecteur de paramètres θ maximisant un critère de performance f lié au contrôleur. On peut donc potentiellement appliquer de nombreuses méthodes issues de la théorie de l'optimisation.

Dans le cas mentionné précédemment du jeu de Tetris, un algorithme d'optimisation évolutionnaire (une stratégie évolutionnaire), la méthode de l'*entropie croisée* (CEM) (de Boer *et al.*, 2005; Mannor *et al.*, 2003; Szita et Lörincz, 2006), a été employée avec succès. D'autres algorithmes de ce type ont été employés, tels que CMA-ES (Hansen, 2011; Heidrich-Meisner et Igel, 2008), PI^2 (Theodorou *et al.*, 2010) ou PoWER (Kober et Peters, 2011), qui vont des algorithmes d'optimisation boîte noire à des algorithmes exploitant le caractère séquentiel du problème considéré. Stulp et Sigaud (2013b,a) mettent en avant un certain nombre de liens, similarités et différences, qui existent entre plusieurs de ces algorithmes.

2.2.2.1 Montées de gradient

Cette section va présenter de manière plus détaillée une famille importante d'approches pour l'optimisation directe d'un contrôleur, celle des montées de gradient (*policy gradient*), dont il sera plus souvent question dans la suite de ce mémoire.

Appliquer une montée de gradient suppose d'abord que le gradient de la fonction $f(\theta)$ considérée soit bien défini. Ce n'est pas évident si (1) l'espace d'états est discret ou (2) l'espace d'états est continu, mais il y a des discontinuités dans la dynamique (par exemple à cause de la présence d'un obstacle dans un environnement). La solution courante adoptée ici est, dans le cas d'un espace d'états discret, de considérer une politique paramétrée *stochastique*, ce qui permettra la bonne définition du gradient.

Optimiser une politique paramétrée avec une montée de gradient Historiquement, l'algorithme de référence pour la recherche directe de politique par montée de gradient est l'algorithme REINFORCE

(Williams, 1987, 1992). Ce travail a été ensuite généralisé et amélioré, entre autres, par Baird (1999); Baird et Moore (1999), puis par Baxter et Bartlett (2001); Baxter *et al.* (2001).

Dans le cas d'un MDP, l'objectif de la montée de gradient est de maximiser un critère de performance dépendant de la politique (stochastique) π appliquée. Notons par exemple $g(\pi)$ un critère de performance. Pour pouvoir employer une montée de gradient, la politique est écrite comme une fonction d'un vecteur paramètre $\vec{\theta}$: $\pi = h(\vec{\theta})$, de sorte que la mesure de performance est une fonction $f(\vec{\theta}) = g \circ h(\vec{\theta})$. Si cette fonction composée est dérivable, les conditions requises pour effectuer une montée de gradient sont satisfaites.

Le choix d'une forme paramétrée $\pi = h(\vec{\theta})$ définit un sous-espace de l'espace des politiques stochastiques. Idéalement, ce sous-espace devrait être aussi petit que possible, mais tout en contenant les meilleures politiques. Dans ce sous-espace, la politique optimale peut ne plus être déterministe, et une recherche peut mener à un optimum local. Supposons par exemple que, comme dans le cas de la programmation dynamique approchée, l'état courant est connu à travers un vecteur caractéristique $\vec{\phi}_s$ (par exemple des variables observables). Alors le vecteur paramètre du contrôleur $\vec{\theta}$ peut être décomposé en un vecteur $\vec{\theta}_a$ par action, et on peut choisir d'écrire la probabilité d'effectuer a avec le *softmax* suivant :

$$q(a|s;\vec{\theta}) = \frac{e^{\vec{\theta}_a^T \vec{\phi}_s}}{\sum_{b \in A} e^{\vec{\theta}_b^T \vec{\phi}_s}}. \tag{2.2}$$

Le nombre de paramètres à optimiser est ici $|\mathcal{A}| \cdot n_\phi$ où n_ϕ est le nombre de caractéristiques.

Le gradient de f au point $\vec{\theta}$ représente la sensibilité de la mesure de performance aux changements des paramètres de contrôle. Avec un modèle connu du MDP, on peut envisager de calculer le gradient de manière exacte, comme dans l'algorithme GAMP d'Aberdeen (2003). Mais ce gradient est le plus souvent estimé à travers des simulations, soit parce qu'aucun modèle n'est disponible, soit parce qu'une méthode exacte serait trop coûteuse (nécessitant de développer l'espace d'état-action complet). On appelle une telle approche une méthode de gradient stochastique.

Méthode des différences finies Une première approche possible pour estimer le gradient est la méthode classique des différences finies. Dans le cas mono-dimensionnel — pour une fonction dérivable $f : \mathbb{R} \to \mathbb{R}$ — cela revient à écrire la dérivée de f en x comme :

$$f'(x) = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x-\epsilon)}{2\epsilon}.$$

Dans notre cas où l'on n'accède à f qu'à travers des simulations, il faut choisir une petite valeur $\epsilon > 0$ et effectuer deux séries de simulations pour estimer $f(x+\epsilon)$ et $f(x-\epsilon)$.

L'emploi des différences finies requiert d'effectuer $2n$ simulations pour obtenir une estimation du gradient (n étant le nombre de paramètres dans $\vec{\theta}$). Le reste de cette section montre comment des estimations du gradient de f peuvent être calculées plus efficacement dans un cadre MDP.

Estimation du gradient ∇f – Cas de l'horizon temporal fini

Horizon temporel 1 Considérons d'abord un MDP dont l'horizon temporel est de longueur 1 (une seule décision doit être prise) et dont l'état initial est échantillonné aléatoirement selon une distribution de probabilité d. L'objectif est d'optimiser

$$J(\pi) = E_\pi[r] = \sum_{(s,a,s') \in S \times A \times S} \underbrace{d(s) q_\pi(a|s) p(s'|s,a)}_{Pr(s,a,s')} r(s,a,s').$$

Avec la paramétrisation $\vec{\theta}$, cette mesure de performance peut être ré-écrite :

$$f(\vec{\theta}) = \sum_{s,a,s'} d(s) q(a|s;\vec{\theta}) p(s'|s,a) r(s,a,s').$$

Effectuer une montée de gradient suppose, pour tout couple (s,a), que la fonction associant $\vec{\theta}$ à $q(a|s;\vec{\theta})$ soit différentiable. Supposant que les rapports de vraisemblance $\frac{\nabla q(a|s;\vec{\theta})}{q(a|s;\vec{\theta})}$ existent (et sont bornés), le gradient de f s'écrit alors :[10]

$$\begin{aligned}
\nabla E\left[r(s,a,s')\right] &= \nabla \left[\sum_{s,a,s'} d(s)q(a|s;\vec{\theta})p(s'|s,a)r(s,a,s')\right] \\
&= \sum_{s,a,s'} d(s)\nabla\left[q(a|s;\vec{\theta})\right] p(s'|s,a)r(s,a,s') \\
&= \sum_{s,a,s'} d(s)\left[\frac{\nabla q(a|s;\vec{\theta})}{q(a|s;\vec{\theta})} q(a|s;\vec{\theta})\right] p(s'|s,a)r(s,a,s') \\
&= \sum_{s,a,s'} \left[\frac{\nabla q(a|s;\vec{\theta})}{q(a|s;\vec{\theta})} r(s,a,s')\right] d(s)q(a|s;\vec{\theta})p(s'|s,a) \\
&= E\left[\frac{\nabla q(a|s;\vec{\theta})}{q(a|s;\vec{\theta})} r(s,a,s')\right].
\end{aligned}$$

Avec ce résultat, et avec N échantillons (s_i, a_i, s'_i) indépendants et identiquement distribués (iid) selon les distributions d, q et p, une estimation du gradient de f au point $\vec{\theta}$ est :

$$\tilde{\nabla} f(\vec{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \frac{\nabla q(a_i|s_i;\vec{\theta})}{q(a_i|s_i;\vec{\theta})} r(s_i,a_i,s'_i). \tag{2.3}$$

Utiliser ce résultat requiert d'abord de calculer $\frac{\nabla q(a|s;\vec{\theta})}{q(a|s;\vec{\theta})}$ (parfois appelé le log-gradient de la politique parce qu'il peut s'écrire $\nabla \ln q(a|s;\vec{\theta})$). Dans le cas de la formulation (2.2) par exemple, nous avons, selon qu'on dérive $\ln(q(a|s;\vec{\theta}))$ par rapport à un paramètre $\theta_{a,i}$ ou $\theta_{b,i}$:

$$\begin{aligned}
\frac{1}{q(a|s;\vec{\theta})}\frac{\partial q(a|s;\vec{\theta})}{\partial \theta_{a,i}} &= \frac{1}{q(a|s;\vec{\theta})}\left(\phi_{s,i}\frac{e^{\vec{\theta}_a^T\vec{\phi}_s}}{\sum_{a'\in\mathcal{A}} e^{\vec{\theta}_{a'}^T\vec{\phi}_s}} - \frac{e^{\vec{\theta}_a^T\vec{\phi}_s}}{\left[\sum_{a'\in\mathcal{A}} e^{\vec{\theta}_{a'}^T\vec{\phi}_s}\right]^2}\phi_{s,i}\ e^{\vec{\theta}_a^T\vec{\phi}_s}\right) \\
&= \phi_{s,i}(1-q(a|s;\theta)),\ \text{et} \\
\frac{1}{q(a|s;\vec{\theta})}\frac{\partial q(a|s;\vec{\theta})}{\partial \theta_{b,i}} &= \frac{1}{q(a|s;\vec{\theta})}\left(0 - \frac{e^{\vec{\theta}_a^T\vec{\phi}_s}}{\left[\sum_{a'\in\mathcal{A}} e^{\vec{\theta}_{a'}^T\vec{\phi}_s}\right]^2}\phi_{s,i}\ e^{\vec{\theta}_b^T\vec{\phi}_s}\right) \\
&= \phi_{s,i}(0-q(b|s;\theta)),
\end{aligned}$$

de sorte qu'on peut écrire :

$$\frac{\nabla q(a|s;\vec{\theta})}{q(a|s;\vec{\theta})} = \vec{\phi}_s(\vec{U}(a) - q(\cdot|s;\theta))^T, \tag{2.4}$$

où $\vec{U}(a)$ est un vecteur unitaire avec un 1 à la position a.

Mettre en œuvre l'équation (2.3) dans une montée de gradient requiert aussi d'ajouter à l'algorithme une boucle générant ces N échantillons pour estimer le gradient du vecteur paramètre courant.

10. Les gradients sont toujours pris par rapport au vecteur $\vec{\theta}$.

Horizon temporel T Passons maintenant au cas d'un horizon temporel fini $T \geq 1$. La mesure de performance considérée est la somme des récompenses pondérées avec $\gamma \in [0,1]$. Soit $V_t(s)$ la somme pondérée des récompenses de t à T si l'on part de l'état s. Alors,

$$\begin{aligned}
f(\vec{\theta}) &= E\left[V_0(s_0)\right] = \sum_s d(s) V_0(s), \\
V_t(s) &= E\left[r(s_t, a_t, s_{t+1}) + \gamma V_{t+1}(s_{t+1}) | s_t = s\right] && (\forall s \in \mathcal{S} \text{ et } t \in \{0, \ldots, T-1\}) \\
&= \sum_{a,s'} q(a|s;\vec{\theta}) p(s'|s,a) \left(r(s,a,s') + \gamma V_{t+1}(s')\right) \text{ et} \\
V_T(s) &= 0 && (\forall s \in \mathcal{S}).
\end{aligned}$$

Le gradient de f peut donc être calculé avec :

$$\begin{aligned}
\nabla f(\vec{\theta}) &= \sum_s d(s) \nabla V_0(s) = E\left[\nabla V_0(s_0)\right], \\
\nabla V_t(s) &= \sum_{a,s'} \nabla \left[q(a|s;\vec{\theta}) p(s'|s,a) \left(r(s,a,s') + \gamma V_{t+1}(s')\right) \right] && (\forall s \in \mathcal{S} \text{ et } t \in \{0, \ldots, T-1\}) \\
&= \sum_{a,s'} q(a|s;\vec{\theta}) p(s'|s,a) \left(\frac{\nabla q(a|s;\vec{\theta})}{q(a|s;\vec{\theta})} \left(r(s,a,s') + \gamma V_{t+1}(s')\right) + \gamma \nabla V_{t+1}(s') \right) \\
&= E\left[\frac{\nabla q(a_t|s_t;\vec{\theta})}{q(a_t|s_t;\vec{\theta})} \left(r_t + \gamma V_{t+1}(s_{t+1})\right) + \gamma \nabla V_{t+1}(s_{t+1}) \middle| s_t = s \right] \text{ et} && (\text{où } r_t = r(s_t, a_t, s_{t+1})) \\
\nabla V_T(s) &= \vec{0} && (\forall s \in \mathcal{S}).
\end{aligned}$$

En développant $\nabla f(\vec{\theta})$ et en factorisant les termes associés avec chaque récompense, on obtient :

$$\nabla f(\vec{\theta}) = E\left[\sum_{t=0}^{T-1} \gamma^t r_t \sum_{t'=0}^{t-1} \frac{\nabla q(a_{t'}|s_{t'};\vec{\theta})}{q(a_{t'}|s_{t'};\vec{\theta})} \right].$$

Comme avec l'horizon de longueur 1, le gradient peut être estimé par échantillonnage. Après acquisition d'un échantillon $s_0, a_0, \ldots, s_T$, on calcule une estimation non-biaisée $\vec{g}$ de $\nabla f(\vec{\theta})$ en itérant :

$$\begin{aligned}
\vec{z}_{t+1} &= \vec{z}_t + \frac{\nabla q(a_t|s_t;\vec{\theta})}{q(a_t|s_t;\vec{\theta})} \text{ et} \\
\vec{g}_{t+1} &= \vec{g}_t + \gamma^t r_t \vec{z}_t,
\end{aligned}$$

où $\vec{z}$ est une trace d'éligibilité (une estimation locale du log-gradient de la politique) et les deux suites sont initialisées avec le vecteur nul. Comme précédemment pour $T = 1$, une meilleure estimation (avec une plus faible variance) est obtenue comme la moyenne de N estimations $\vec{g}$ calculées indépendamment.

L'algorithme 4 résume les opérations à effectuer pour réaliser une montée de gradient dans un MDP à horizon temporel fini. La récompense instantanée r et le prochain état s' sont obtenus par interaction avec l'environnement réel ou un simulateur.

Horizon temporel infini Sur la base de cette approche, on peut passer à un horizon temporel infini (sans états absorbants) en effectuant une montée de gradient « en ligne ». Dans l'algorithme « On-Line POMDP » [11] de Baxter et Bartlett (2001); Baxter *et al.* (2001) (OLpomdp) par exemple, le gradient est estimé continuellement. Comme on le voit dans l'algorithme 5, l'estimation du gradient, $r\vec{z}$, repose sur la trace d'éligibilité $\vec{z}$. Or, du fait de l'horizon temporel infini, le calcul de cette trace requiert l'introduction d'un biais favorable aux transitions récentes (biais sans lequel cette trace peut diverger) à l'aide d'un facteur $\beta \in [0,1)$. Le choix du facteur β permet donc de faire un compromis biais-variance.

11. On notera qu'on peut traiter le cas d'un POMDP comme un MDP avec accès à de simples features.

Algorithme 4 : Montée de gradient pour MDP à horizon temporel fini $(\vec{\theta}, \alpha, N)$

Initialisation :
$\vec{\theta_0} \leftarrow \vec{\theta}$
$i \leftarrow 0$
`/* Boucle de la montée de gradient. */`
pour *toujours* **faire**
  $i \leftarrow i + 1$
  $\tilde{\nabla} f \leftarrow \vec{0}$
  `/* Boucle d'acquisition de` N `échantillons pour estimation du gradient. */`
  pour $n = 1, \ldots, N$ **faire**
    $\vec{z} \leftarrow \vec{0}$
    $\vec{g} \leftarrow \vec{0}$
    ObtientÉtatInitial(s) `/* Échantillonne état initial selon` $d(\cdot)$. `*/`
    `/* Boucle simulant le MDP pour obtention d'un échantillon. */`
    pour $t = 0, \ldots, T-1$ **faire**
      Échantillonne$(a, q(\cdot|s;\vec{\theta}))$
      Exécute(a)
      Obtient(s', r)
      $\vec{z} \leftarrow \vec{z} + \frac{\nabla q(a|s;\vec{\theta})}{q(a|s;\vec{\theta})}$
      $\vec{g} \leftarrow \vec{g} + \gamma^t r \vec{z}$
      $s \leftarrow s'$
    $\tilde{\nabla} f \leftarrow \tilde{\nabla} f + \frac{1}{N} \vec{g}$
  $\vec{\theta_i} \leftarrow \vec{\theta}_{i-1} + \alpha \tilde{\nabla} f$
retourner x_n

Algorithme 5 : `OLpomdp`$(\vec{\theta}, \alpha, \beta)$
Montée de gradient en-ligne pour (PO)MDPs

Initialisation :
$\vec{\theta_0} \leftarrow \vec{\theta}$; $\vec{z} \leftarrow \vec{0}$; $\vec{g} \leftarrow \vec{0}$
ObtientEtatInitial(s)
pour *toujours* **faire**
  Échantillonne$(a, q(\cdot|s;\vec{\theta}))$
  Exécute(a)
  Obtient(s', r)
  $\vec{z} \leftarrow \beta \vec{z} + \frac{\nabla q(a|s;\vec{\theta})}{q(a|s;\vec{\theta})}$
  $\vec{\theta} \leftarrow \vec{\theta} + \alpha r \vec{z}$
  $s \leftarrow s'$
retourner $\vec{\theta}$

Utilisation d'un critique On peut aussi voir dans l'algorithme 5 que l'on n'obtiendra pas le même comportement si la fonction de récompense r est remplacée par $r + 1000$, alors que le problème posé est identique. Intuitivement, on voudrait que le vecteur $\vec{\theta}$ suive la direction de $\vec{z}$ non pas quand r est positif, mais quand le gain futur espéré avec la transition observée $(r + V^{\vec{\theta}}(s'))$ est meilleur que le gain futur espéré *a priori* $(V^{\vec{\theta}}(s))$. Plus généralement, on souhaiterait remplacer r par $(r + V^{\vec{\theta}}(s')) - V^{\vec{\theta}}(s)$ dans la mise à jour de $\vec{\theta}$, à supposer que la fonction de valeur $V^{\vec{\theta}}$ de la politique courante soit connue.

Une telle approche est appelée descente de gradient acteur-critique. Le calcul de la fonction de valeur exacte n'étant typiquement pas possible (sinon on emploierait la programmation dynamique), le critique repose sur une approximation. Chaque itération d'une telle montée de gradient sera plus coûteuse calculatoirement, mais la vitesse de convergence (en prenant le nombre d'itérations, donc de transitions échantillonnées, comme unité de temps) sera plus grande parce que l'estimation du gradient est de plus faible variance.

2.2.3 Recherche heuristique / Recherche arborescente

Une dernière classe d'approches pour traiter les MDP de grande taille consiste à exploiter, le cas échéant, l'existence d'un état initial pour ne considérer que les états atteignables (sur lesquels on pourra calculer la fonction de valeur à l'aide du principe d'optimalité de Bellman), lesquels forment un graphe orienté (parfois acyclique, parfois même restreint à un arbre), comme illustré figure 2.2. Ce graphe est constitué de deux types de nœuds, les nœuds états et les nœuds actions, et souvent appelé graphe ET-OU (action→ET / état→OU) ou graphe max-espérance (état→max / action→espérance) du fait des évolutions ou calculs associés à ces deux types de nœuds.

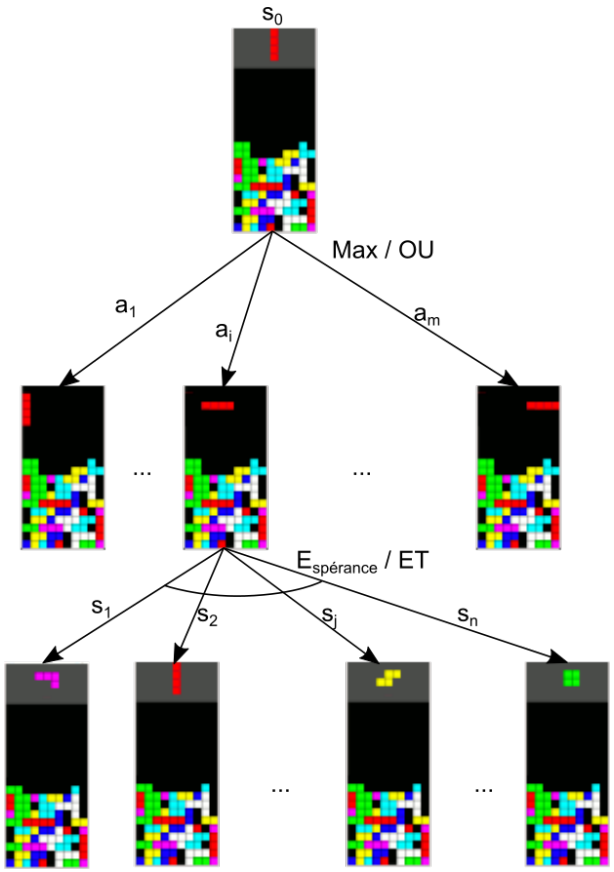


FIGURE 2.2 – Graphe expectimax (max-espérance / ET-OU) partiel du jeu de Tetris vu comme un MDP pour un état initial donné (figure largement inspirée par (Dutech *et al.*, 2008))

Mais, dans bien des cas, développer tous les futurs possibles reste trop onéreux. Cet arbre/espace de recherche est souvent élagué/restreint en exploitant le principe de l'*optimisme face à l'incertain*. Il s'agit, à supposer que l'on sait initialiser et mettre à jour une estimation optimiste de la fonction de valeur en chaque état, de ne développer que les branches les plus prometteuses, ce qui permet d'éviter, donc d'élaguer, un certain nombre de branches inutiles. Un autre moyen de restreindre l'espace exploré est de *faire des approximations*, soit en limitant l'horizon temporel, soit en échantillonnant seulement un sous-ensemble de branches.

On distinguera principalement les approches rencontrées selon qu'elles reposent sur des calculs exacts ou sur des estimations. On verra aussi qu'elles peuvent être employées de manière hors-ligne, en séparant une phase de planification de la phase d'exécution, ou en-ligne, profitant du temps de calcul disponible avant chaque action pour approfondir la planification.

2.2.3.1 Approches avec calculs exacts

Dans les approches avec calculs exacts apparaît clairement une affiliation avec les algorithmes de recherche heuristique classiques (c'est-à-dire dans un cadre déterministe). Pour être applicables, elles requièrent :

— la connaissance explicite du modèle, en particulier l'accès aux probabilités de transition, et
— souvent l'accès à des heuristiques admissibles (donc optimistes).

Ainsi, AO* (Nilsson, 1980) et LAO* (Loopy AO*) (Hansen et Zilberstein, 1998, 2001) généralisent A* (Hart *et al.*, 1968). Pour observer cela, notons d'abord que A* peut être vu [12] comme

1. maintenant une liste de chemins partiels depuis l'état initial (ce qui prend la forme d'un graphe incomplet), et
2. remplaçant à chaque itération le chemin partiel le plus prometteur de la liste — celui dont le coût total estimé de manière optimiste est le meilleur — par ses successeurs immédiats (c'est-à-dire que l'on développe une des feuilles du graphe incomplet), jusqu'à ce que le « plus prometteur » soit un chemin reliant un état but.

Or LAO* (pour aborder directement le cas le plus général) fait de même en maintenant une liste de politiques partielles et en développant, à chaque itération, l'un des états feuilles d'une des politiques les plus prometteuses, jusqu'à ce que la « plus prometteuse » soit une politique sans feuilles à développer.

De la même manière, (L)RTDP (Labeled Real-Time Dynamic Programming) (Barto *et al.*, 1995; Bonet et Geffner, 2003a) généralise LRTA* (Korf, 1990). En effet, il s'agit dans les deux cas de générer des trajectoires en agissant de manière gourmande, et en mettant à jour l'estimation (optimiste) actuelle de la fonction de valeur en chaque état rencontré. Dans LRTA*, le processus peut s'arrêter quand la même trajectoire est répétée sans modifications, ce que ne permet pas RTDP tel quel du fait du hasard des transitions. LRTDP ajoute à RTDP un mécanisme d'arrêt par analyse du graphe des états accessibles. L'algorithme 6 présente le détail des instructions suivies par RTDP, en notant ici la fonction de valeur U (pour *upper bound*) parce qu'elle surestime toujours la fonction optimale, ce qui requiert une initialisation avec une heuristique admissible (c'est-à-dire « surestimante » elle aussi). On notera que RTDP n'est applicable qu'aux problèmes de type « chemin le plus court », dont toute boucle a une récompense négative (/un coût positif), et dans lesquels un état terminal est toujours atteignable.

Fonctions heuristiques L'efficacité de ces algorithmes dépend de l'existence d'une fonction heuristique admissible (sur-estimante, autrement dit une fonction majorant la fonction de valeur optimale) (1) suffisamment informative et (2) calculable efficacement (c'est-à-dire dont le calcul est d'une complexité algorithmique moindre que la recherche heuristique elle-même). Certains algorithmes font aussi usage d'une fonction heuristique qui, à l'inverse, minore la fonction de valeur optimale.

Il existe de nombreux travaux en particulier sur la dérivation automatique d'heuristiques admissibles, majoritairement dans le cadre de la planification classique (déterministe). On rappellera simplement ici le principe général que

12. Cette vision n'est pas la plus classique, mais facilite l'analogie.

Algorithme 6 : Real-Time Dynamic Programming (RTDP)

Entrées : A,p,c, état initial s_0, heuristique admissible h

début
  tant que *condition d'arrêt non satisfaite* **faire**
    $s \leftarrow s_0$
    répéter
      $S' \leftarrow$ étatsAccessibles(s)
      pour $a \in A(s)$ **faire**
        $Q_U(s,a) \leftarrow r(s,a) + \gamma \cdot \sum_{s' \in S} p(s'|s,a)$ÉVALUE($s'$)
      $a \leftarrow \text{argmax}_{a \in A(s)} Q_U(s,a)$
      $U(s) \leftarrow Q_U(s,a)$
      $s \leftarrow$échantillon($p(\cdot|s,a)$)
    jusqu'à s *état terminal*
  retourner *fonction de valeur* U

Fct ÉVALUE(s')
  si s' *rencontré pour la première fois* **alors**
    retourner $h(s')$
  sinon retourner $U(s')$

— enlever des contraintes à un problème d'optimisation induit un problème (relâché) dont la meilleure solution sera de valeur supérieure ou égale (parce qu'on ne fait qu'augmenter l'espace des solutions admises), et
— ajouter des contraintes à un problème d'optimisation induit, à l'inverse, un problème dont la meilleure solution sera de valeur inférieure ou égale.

Toutefois, dans le cas général, ajouter ou enlever des contraintes ne garantit pas que le nouveau problème obtenu sera d'une complexité calculatoire plus faible.

On notera que, pour un MDP dont le modèle est connu (hypothèse requise ici), on peut généralement majorer la fonction de valeur par :
— $R_{\max}/(1-\gamma)$ dans le cas d'un problème avec facteur d'atténuation,
— $R_{\max}H$ dans le cas d'un problème à horizon temporel fini, ou
— $R_{\max}|S|/(1-p_{\max})$ dans le cas d'un problème à horizon temporel indéfini propre,[13] où $p_{\max}$ est la plus grande probabilité de transition inférieure à 1.

Toutefois, de tels majorants sont très lâches, et permettront rarement de guider une recherche heuristique efficacement.

2.2.3.2 Approches avec simulations et estimations

Les approches précédentes ne sont pas envisageables si le modèle du système ne fournit pas les probabilités de transition ou si l'on ne dispose pas d'une heuristique admissible suffisamment informative. Par contre, si le modèle permet au moins d'échantillonner des transitions, on peut remplacer les calculs exacts par des estimations. Ne connaissant (sauf cas particuliers) les états accessibles depuis un état donné qu'à travers des échantillonnages, il n'est pas garanti que tous les états pertinents seront couverts. C'est pourquoi il sera nécessaire de pouvoir faire ces calculs en ligne, avant chaque nouvelle prise de décision.

L'algorithme *Sparse Sampling* de Kearns *et al.* (2002) propose une première approche consistant

1. à construire un arbre des futurs possibles depuis l'état courant[14] (i) en négligeant le fait que le même état puisse être atteint de différentes manières, (ii) en échantillonnant les états accessibles depuis un nœud action donné, et (iii) en limitant l'horizon de recherche ; puis

13. Il s'agit des « chemins le plus court stochastiques » traités par RTDP.
14. Il s'agit bien d'un arbre et non d'un graphe parce que deux états identiques atteints par deux chemins différents donneront deux nœuds distincts.

2. à estimer la fonction de valeur optimale sur cet arbre en partant des feuilles.

Comme l'illustre l'algorithme 7, une formulation récursive permet d'éviter la construction explicite de l'arbre échantillonné en question. Comme démontré par Kearns *et al.* (2002, théorème 1), pour un $\epsilon > 0$ donné, l'horizon H et la largeur W peuvent être choisis de manière à garantir les propriétés (1) d'*efficacité*, c'est-à-dire que le temps de calcul est en $O((kW)^H)$, et (2) de *quasi-optimalité*, c'est-à-dire que, pour tout état s, $|V^{SS}(s) - V^*(s)| \leq \epsilon$.[15] On notera en particulier que ces paramètres, ainsi que la complexité computationnelle, sont indépendants de la taille de l'espace d'états.

Algorithme 7 : SparseSampling(s_0, H, W) (SS)

Entrées : simulateur du système, état courant s_t, horizon de raisonnement H, largeur W

début
  $\left(\hat{Q}^*_H(s_0, a_1), \ldots, \hat{Q}^*_H(s_0, a_{|\mathcal{A}|})\right) \leftarrow$ EstimeQ(s_0, a, H, W)
  retourner $\text{argmax}_{a \in \{a_1, \ldots, a_{|\mathcal{A}|}\}} \hat{Q}^*_H(s_0, a)$
Fct EstimeQ(s_t, h, W)
  pour chaque $a \in A$ **faire**
    $$\hat{Q}^*_h(s_t, a) \leftarrow \begin{cases} 0 & \text{si } h = 0 \\ r(s_t, a) + \gamma \frac{1}{W} \sum_{i=1}^{W} \text{EstimeV}(\text{échantillon}(p(.|s_t, a)), h-1, W) & \text{sinon} \end{cases}$$
  retourner $\left(\hat{Q}^*_h(s_t, a_1), \ldots, \hat{Q}^*_h(s_t, a_{|\mathcal{A}|})\right)$
Fct EstimeV(s_t, h, W)
  $\left(\hat{Q}^*_h(s_t, a_1), \ldots, \hat{Q}^*_h(s_t, a_{|\mathcal{A}|})\right) \leftarrow$ EstimeQ(s_t, h, W)
  retourner $\max_{a \in \{a_1, \ldots, a_{|\mathcal{A}|}\}} \hat{Q}^*_h(s_t, a)$

L'algorithme que nous venons de voir a l'inconvénient d'explorer les futurs possibles de manière uniforme, d'où un gâchis de ressources de calcul. Une première idée pour contourner ce problème est de construire de manière itérative un arbre des futurs possibles de plus en plus étoffé là où vont les meilleurs politiques. C'est ce que font par exemple les algorithmes de type *Monte Carlo Tree Search* (MCTS) ou *Monte Carlo Planning* — initialement développés pour des jeux à deux joueurs tels que le Go (Coulom, 2006) — en répétant, itération après itération, les étapes suivantes (en omettant les nœuds action) (Chaslot *et al.*, 2008) :

1. **sélection :** à l'aide d'une stratégie faisant un compromis entre exploration (branches les moins bien connues) et exploitation (branches les plus prometteuses), une trajectoire est générée jusqu'à atteindre un nœud état absent de l'arbre courant ;
2. **expansion :** ce nouveau nœud état est ajouté à l'arbre ;
3. **simulation :** une stratégie aléatoire permet de continuer la génération de la trajectoire jusqu'à terminaison ;
4. **rétro-propagation :** en remontant la trajectoire obtenue, les estimations de la fonction de valeur sont mises à jour, ainsi que les comptes des visites de chaque nœud.

Les algorithmes MCTS se distinguent les uns des autres par différents détails. Par exemple, l'action retournée au final peut être celle dont l'estimation actuelle (à la racine) est la meilleure, ou bien celle qui a été le plus souvent sélectionnée. Pour choisir une action dans la phase de sélection, on peut s'inspirer des stratégies pour bandits manchots, comme cela a été fait en utilisant la stratégie *Upper Confidence Bounds* (UCB) dans l'algorithme UCT (*UCB applied to Trees*) de Kocsis et Szepesvari (2006). L'algorithme 8 (inspiré de (Couetoux, 2013)) donne le détail d'UCT en confondant d'une part les nœuds états avec les états associés, et d'autre part les nœuds actions avec les couples états-actions associés.

15. V^{SS} est la valeur de la politique stochastique induite par l'algorithme.

On notera (1) que les algorithmes MCTS sont le plus souvent bien meilleurs que Sparse Sampling et (2) que leur estimation des actions disponibles à la racine tend vers les valeurs optimales. Toutefois, les comportements au pire cas peuvent être désastreux, comme démontré par Coquelin et Munos (2007). On notera aussi, entre autres choses, que :

— la stratégie d'exploration, qui est importante, peut être apprise ; et
— une fonction de valeur approchée apprise peut soit remplacer la phase de simulation pour évaluer les feuilles, soit servir à initialiser les valeurs des nœuds.

De telles méthodes ont fait le récent succès d'un algorithme MCTS dans le cadre du jeu de go face à un grand maître (sans handicap) (Silver *et al.*, 2016; Teytaud et Liu, 2016).

Algorithme 8 : MCTS, version UCT (s_0)

Entrées : état initial s_0
début
  répéter
    FairePousserArbre(s_0)
  jusqu'à *délai dépassé*
  retourner MeilleureAction(s_0))
Fct FairePousserArbre(s)
  /* Descendre dans l'arbre. */
  répéter
    $a \leftarrow$ Sélectionner(s, K)
    $enfants(s) \leftarrow enfants(s) \cup (s, a)$
    $s' \leftarrow p(\cdot|s, a)$
    $s \leftarrow s'$
  jusqu'à $n(s) = 0$ *ou* $terminal(s)$
  /* Évaluer la feuille atteinte. */
  $R \leftarrow$ Évaluer(s, ϕ)
  /* Rétro-propager les récompenses. */
  tant que $\neg racine(s)$ **faire**
    $n(s) \leftarrow n(s) + 1$
    $(s, a) \leftarrow pere(s)$
    $R \leftarrow R + r(s, a)$
    $T(s, a) \leftarrow T(s, a) + R$
    $n(s, a) \leftarrow n(s, a) + 1$
Fct Sélectionner(s)
  /* Les actions non-visitées sont prioritaires, d'où la valeur $+\infty$. */
  $a_s =$

$$\underset{a \in enfants(s)}{\operatorname{argmax}} \begin{cases} \frac{T(s,a)}{N(s,a)} + K\sqrt{\frac{\ln n(s)}{n(s,a)}} & \text{si } n(s,a) > 0 \\ +\infty & \text{sinon} \end{cases}$$

  retourner a_s
Fct Évaluer(s, ϕ)
  /* On cumule dans R les récompenses reçues en simulant la politique ϕ. */
  $R \leftarrow 0$
  tant que $\neg terminal(s)$ **faire**
    $a \leftarrow \phi(s)$
    $R \leftarrow R + r(s, a)$
    $s \leftarrow p(\cdot|s, a)$
  retourner R
Fct MeilleureAction(s)
  retourner $\operatorname{argmax}_{a \in enfants(s)} n(s, a)$

Hors-ligne vs en-ligne Les algorithmes vus ici partent d'un état initial connu. Certains, comme SS ou MCTS, sont faits pour être exécutés en ligne, en exploitant le temps de calcul disponible à chaque pas de décision. Pour SS, il faudra prévoir les paramètres (profondeur et largeur) compatibles avec le temps disponible. Pour MCTS, il suffit de l'interrompre quand c'est nécessaire puisque celui-ci est *anytime* ((interruptible)« à tout moment »). Par ailleurs, s'ils n'ont pas été particulièrement conçus dans ce but, des algorithmes comme LAO* ou (L)RTDP peuvent aussi être employés en ligne du fait aussi de leurs bons comportements *anytime*. Dans ce cas, le compromis entre qualité et coût de calcul des heuristiques reste critique, comme dans le cas hors-ligne.

La prise de décision en ligne est aussi souvent vue comme se faisant entre l'instant où le nouvel état courant est connu et celui où la prochaine action doit être exécutée. Or on peut anticiper la connaissance du prochain état courant pour exploiter plus de temps de calcul, comme expliqué par Chanel *et al.* (2014). À titre illustratif, dans un jeu comme le go ou les échecs, un joueur peut se mettre à la place de son adversaire et réfléchir en même temps que lui pendant qu'il choisit son prochain mouvement.

2.2.4 Complément : Exploitation de la structure

De nombreux travaux proposent par ailleurs d'exploiter la structure particulière de certains MDP.

Ainsi, une façon de résoudre de manière approchée certains problèmes est de résoudre une version approchée (simplifiée) de ces problèmes, obtenue en négligeant des détails. Agréger des états considérés comme proches permet par exemple de définir un MDP abstrait de différentes manières, typiquement selon qu'on est optimiste ou pessimiste dans la dérivation des probabilités de transition et des récompenses (Givan *et al.*, 2000). On notera sur ce sujet (i) qu'une question clef est de savoir comment obtenir ces agrégations à moindre coût ; et (ii) que raffiner progressivement une abstraction n'implique pas un raffinement progressif de la fonction de valeur obtenue (Tagorti *et al.*, 2013a).

Des travaux apparentés sont ceux exploitant la factorisation du MDP, les fonctions de transition et de récompense s'écrivant de manière structurée (comme un réseau bayésien dynamique pour ce qui est de la fonction de transition) à l'aide de plusieurs variables d'état, voire d'action. Ces variables peuvent évidemment fournir des fonctions caractéristiques (*features*) telles qu'employées en programmation dynamique approchée. Mais cette structure factorisée permet aussi de ré-écrire la fonction de valeur, l'opérateur de Bellman et la politique sous des formes plus efficaces, par exemple à l'aide d'arbres de décision (Boutilier *et al.*, 2000) ou de diagrammes de décision (Hoey *et al.*, 1999). Cette même idée a été exploitée non seulement avec des représentations propositionnelles, mais aussi du premier ordre, par exemple par Sanner et Boutilier (2006, 2007). Aussi, si les références précédentes concernent essentiellement des variantes (exactes ou approchées) des algorithmes d'itération sur la valeur ou d'itération sur la politique, des algorithmes reposant sur la programmation linéaire ont aussi été développés pour résoudre les MDP factorisés (Guestrin *et al.*, 2003b).

2.3 Quand on sort des MDP

Il y existe de nombreuses et diverses façons de « sortir des MDP », souvent en remettant en cause une composante du modèle original (le critère, les informations disponibles, le nombre d'entités agissantes...). La première question qui se pose quand on sort du cadre des MDP est de savoir comment traiter ce nouveau problème. A-t-on besoin d'outils radicalement nouveaux ? Peut-on adapter les outils connus ? Quelle est la complexité de ces nouveaux problèmes ? On va se contenter ici de présenter divers cas sur lesquels nous allons revenir dans la suite de ce mémoire.

2.3.1 Quand l'observabilité est partielle (POMDP (et PSR))

Un premier cas important est celui des MDP partiellement observables (Sondik, 1971; Smallwood et Sondik, 1973), cas dans lequel, à l'exécution, l'état courant n'est plus connu de façon certaine, mais des *observations* reçues à chaque pas de temps — souvent bruitées — fournissent des informations quant à cet état caché. On rencontre cette situation en robotique mobile, où les capteurs (caméras, odomètres, ...) ne donnent qu'une vue partielle de la situation, mais aussi en diagnostic médical, où l'état (à améliorer) du patient n'est connu qu'à travers les résultats d'analyses ou les effets de médicaments, et dans de nombreuses autres applications (Cassandra, 1998).

Formellement, à la définition d'un MDP est ajouté (1) un ensemble (souvent fini) d'observations $\mathcal{Z}$ et (2) une fonction d'observation $O_{a,s'}(z) = Pr(z|a,s')$ qui donne, pour chaque action a, chaque état s', et chaque observation z, la probabilité d'observer z quand l'action a conduit à l'état s'. Souvent, on précise aussi une distribution de probabilité sur l'état de départ (appelée *état de croyance* initial) b_0, et on considère, comme on le fera ici sauf précision contraire, un problème à horizon temporel infini. On obtient ainsi un tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{Z}, P, O, \mathcal{R}, b_0, T \rangle$, avec $T \in \mathbb{N}^* \cup \{\infty\}$. La figure 2.3 montre un modèle graphique représentant un POMDP.

2.3.1.1 Se ramener à la programmation dynamique à horizon fini

Supposant d'abord donnés un état de croyance initial et un horizon temporel fini, on peut raisonner sur l'arbre des futurs possibles, arbre expectimax (*cf.* section 2.2.3) dans lequel alternent des nœuds actions et des nœuds observations, en plus de la racine. Chaque nœud observation correspond à une situation

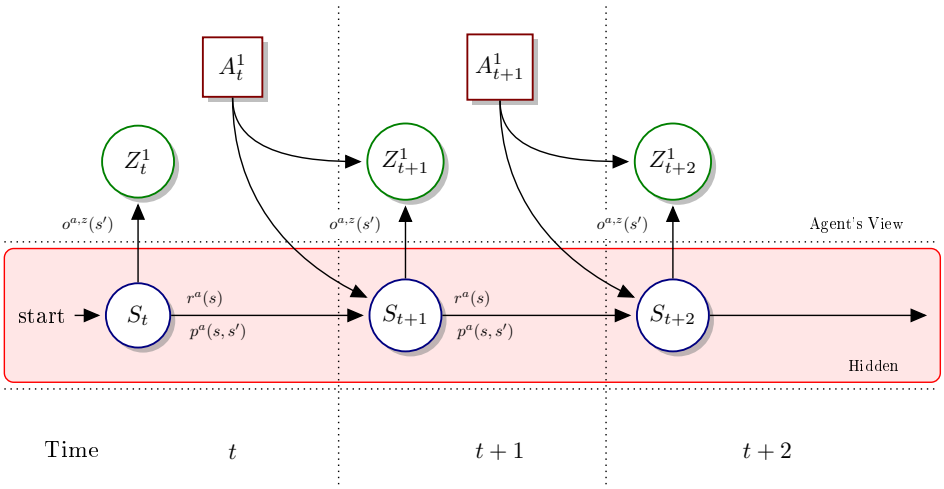


FIGURE 2.3 – Modèle graphique d'un POMDP

unique associée à l'historique des actions et observations passées. On peut, par filtrage bayésien, calculer l'état de croyance courant [16] b et, pour chaque nœud action fils, les probabilités de transition vers les nœuds observations suivants. De même, on peut calculer l'espérance de récompense associée à chaque transition : $r(b,a) = \sum_{s\in\mathcal{S}} b(s)r(s,a)$. Comme dans le cas complètement observable, on peut mettre en œuvre la programmation dynamique sur cet arbre expectimax.

Comme dans le cas complètement observable toujours, on peut lutter en partie contre l'explosion combinatoire de l'arbre construit avec des algorithmes de la famille MCTS qui permettront de concentrer les efforts sur les parties les plus prometteuses de l'arbre, y compris avec un horizon temporel infini, comme le permet l'algorithme POMCP de Silver et Veness (2010).

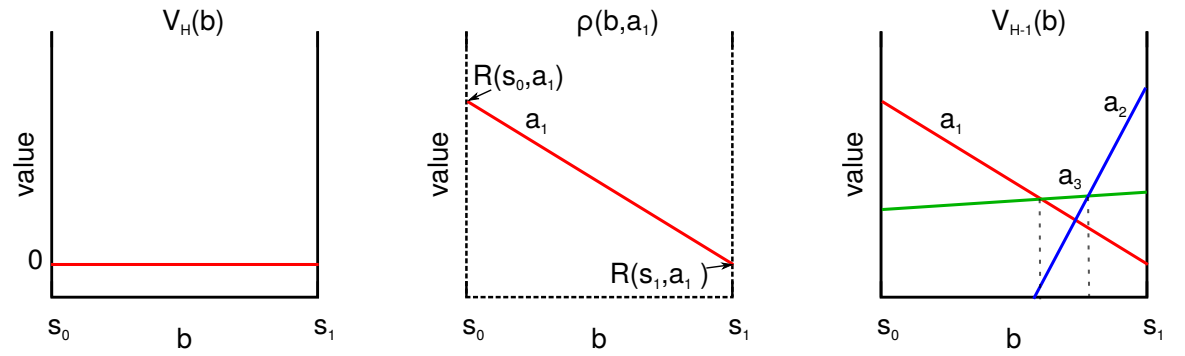
2.3.1.2 Calculer la fonction de valeur sur les états de croyance

On peut aussi remarquer qu'une information clef est celle contenue dans les états de croyance. Deux historiques d'actions-observations associées au même état de croyance vont être suivies des mêmes futurs possibles (avec les mêmes probabilités de transition), ce qui permet de fusionner les nœuds correspondant dans l'arbre, lequel devient ainsi un graphe. On obtient en fait ainsi un « méta-MDP », appelé *belief MDP* (bMDP), dont les « méta-états » sont les états de croyance du POMDP. Malheureusement, le nombre d'états de croyance atteignables peut croître indéfiniment avec l'horizon temporel. Calculer de manière exacte la fonction de valeur optimale semble donc exclu.

Toutefois, cette fonction de valeur a la particularité, dans le cas d'un horizon fini, d'être linéaire par morceaux et convexe (PWLC) sur l'espace des états de croyance. En effet elle s'obtient, en partant d'une fonction nulle au dernier pas de temps, par application répétée de l'opérateur d'optimalité de Bellman, lequel est constitué uniquement d'opérateurs préservant la linéarité par morceaux et la convexité. Graphiquement (voir figure 2.4), elle prend donc la forme, à chaque pas de temps, de l'enveloppe supérieure d'un ensemble d'hyperplans, à chacun étant associé au moins une action optimale.

La fonction de valeur peut ainsi être calculée non seulement de manière exacte à horizon fini, mais aussi de manière approchée à horizon infini (à la manière de l'algorithme d'itération sur la valeur). Du fait du nombre exponentiellement croissant d'hyperplans, il est indispensable d'élaguer au fur et à mesure ceux qui s'avèrent inutiles parce que dominés. On obtient ainsi des algorithmes tels que *Batch Enumeration* (Monahan, 1982) ou *Incremental Pruning* (Cassandra *et al.*, 1997).

16. L'état de croyance courant est la distribution de probabilité sur les états possibles étant donnés b_0 et l'historique des actions et observations passées.



(a) $V^*_H(b) = 0$ (b) $\rho(b,a) = \sum_{s \in \mathcal{S}} R(s,a)b(s)$ (c) $V^*_{H-1}(b) = \max_{a \in \mathcal{A}} (\rho(b,a) + 0)$

FIGURE 2.4 – Dans un POMDP à horizon fini, la fonction de valeur est PWLC pour tout t parce que (i) elle est nulle, donc PWLC, pour $t = H$ (*cf.* 2.4a), (ii) la fonction de récompense est linéaire en b (*cf.* 2.4b), et, (iii) les opérateurs arithmétiques impliqués préservent la linéarité et la convexité (*cf.* 2.4c).

2.3.1.3 Approches à base de points

Pour approcher de manière efficace l'enveloppe optimale d'hyperplans décrite ci-dessus, une idée proposée par Lovejoy (1991) est de choisir un ensemble fini d'états de croyance (distribués selon une grille), et de ne calculer des hyperplans qu'en ces points. Cette approche *« à base de points »* a été ensuite améliorée en considérant que, étant donné un état de croyance initial, il valait mieux sélectionner des points accessibles depuis celui-ci, en particulier par de bonnes politiques. C'est ce qui a conduit à des algorithmes tels que PBVI (Pineau *et al.*, 2003, 2006), PERSEUS (Spaan et Vlassis, 2005), HSVI(1&2) (Smith et Simmons, 2005; Smith, 2007), SARSOP (Kurniawati *et al.*, 2008) ou GapMin (Poupart *et al.*, 2011), lesquels diffèrent entre autres par leur stratégie de sélection des points.

Nous nous attardons ici sur HSVI (*Heuristic Search Value Iteration*). Cet algorithme de recherche heuristique ressemble à LRTA* et RTDP en ce qu'il génère répétitivement des trajectoires de manière gourmande par rapport à une majoration (continuellement mise à jour) de la fonction de valeur optimale. Sur ses prédécesseurs, HSVI a toutefois l'avantage que la mise à jour effectuée en un quelconque état de croyance b (l'hyperplan calculé localement) apporte de l'information pour tout un voisinage de b. Par ailleurs, en plus d'une fonction U majorant la valeur optimale, il maintient aussi une fonction L minorante (sous la forme de l'enveloppe inférieure d'un ensemble de points). Cela permet, en générant une trajectoire,

— de sélectionner la prochaine observation dont l'exploration semblerait la plus utile, et
— de s'arrêter quand un écart suffisamment faible entre minorant et majorant est observé.

L'algorithme 9 décrit le processus résultant où :

— la fonction principale `HSVI` lance la génération de trajectoires (fonction `EssayerRécursivement`) tant que l'excès au point b_0, lequel dépend du seuil ϵ choisi, est positif ;
— la fonction `EssayerRécursivement` est responsable de la génération d'une trajectoire par une procédure récursive ; elle choisit la prochaine action et le prochain état de croyance comme décrit précédemment, et demande la mise-à-jour de U et L en b avant et après récursion en appelant la fonction `MettreAJour`.

2.3.1.4 Autres approches

Comme pour les MDP, le calcul d'une fonction de valeur n'est pas la seule approche de résolution possible. On citera ici simplement l'utilisation d'une montée de gradient (Aberdeen, 2003) ou d'un algorithme génétique (Wells *et al.*, 1999). Dans de telles approches explorant directement l'espace des politiques (/contrôleurs), ces dernières peuvent prendre des formes diverses telles qu'un perceptron multi-couche (prenant un état de croyance en entrée, et retournant une distribution de probabilités sur les actions pos-

Algorithme 9 : Heuristic Search Value Iteration (HSVI) (Smith, 2007)

Fct `HSVI` (ϵ)
  Initialiser L et U
  tant que `excès` $(b_0, 0) > 0$ **faire**
    `EssayerRécursivement` $(b_0, d = 0)$
  retourner L
Fct `EssayerRécursivement` (b, d)
  si `excès` $(b, d) > 0$ **alors**
    `MettreAJour` (b)
    $a^* \in \text{argmax}_{a \in \mathcal{A}} Q^U(b, a)$
    $b^* \in \text{argmax}_{b' \in Suivants(b,a^*)} Pr(b, a^*, b')$`excès`$(b', d + 1)$
    `EssayerRécursivement` $(b^*, d + 1)$
    `MettreAJour` (b)
  retourner
Fct `MettreAJour` (b)
  $L \leftarrow \text{miseAJour}(L, b)$
  $U \leftarrow \text{miseAJour}(U, b)$
Fct `excès` (b, d)
  retourner $(U(b) - L(b)) - \epsilon\gamma^{-d}$

sibles en sortie), ou un automate à états fini (dont les transitions dépendent des actions et observations).

2.3.2 Quand plusieurs agents collaborent (Dec-POMDP)

Une autre extension naturelle des MDP vise à chercher non pas une politique pour un seul agent, mais un ensemble de politiques à répartir entre plusieurs agents partageant un même objectif.

Un premier modèle simple de cette situation est celui des MMDP (*Multiagent MDP*) (Boutilier, 1996), dans lequel tout agent a une observabilité complète de l'état courant. Cette connaissance commune fait que les agents peuvent raisonner de la même manière et donc agir « comme un seul homme ». Une solution simple est de résoudre un MDP dans lequel l'espace d'action joint les actions des différents agents. La politique jointe obtenue peut ensuite être séparée en une politique par agent, chacun avec son rôle propre. On notera que le formalisme MMDP a été introduit pour étudier le problème de la coordination en général, y compris dans le cas où, parce qu'il n'y a pas eu de planification *a priori*, les agents apprennent à agir ensemble (Boutilier, 1996; Claus et Boutilier, 1998).

Une situation plus complexe est celle où chaque agent n'a accès qu'à ses propres observations, ne pouvant prendre une décision qu'en fonction de l'historique de ses propres actions et observations passées. On entre ainsi dans le cadre des *POMDP décentralisés* (Dec-POMDP) (Bernstein *et al.*, 2000; Amato *et al.*, 2013). Un Dec-POMDP est défini formellement par un tuple $\langle I, \mathcal{S}, \mathcal{A}, \mathcal{Z}, P, O, \mathcal{R}, b_0, T \rangle$, qui se distingue de la définition d'un POMDP en ceci que :

- $I = \{1, \ldots, N\}$ est un ensemble d'agents ;
- $\mathcal{A} = \times_{i \in I} \mathcal{A}_i$, l'ensemble des actions *jointes* des agents, est la composition des ensembles d'actions $\mathcal{A}^i$ de chaque agent $i \in I$;
- $\mathcal{Z} = \times_{i \in I} \mathcal{Z}_i$, l'ensemble des observations *jointes* des agents, est la composition des ensembles d'observations $\mathcal{Z}^i$ de chaque agent $i \in I$.

On notera par ailleurs que (i) la transition de l'état du système dépend de l'action jointe ; (ii) les observations des différents agents ne sont pas nécessairement indépendantes ; (iii) les agents partagent la même fonction de récompense et la même croyance quant à l'état initial. La figure 2.5 montre un modèle graphique représentant un Dec-POMDP. Sauf mention contraire, on considérera ici un horizon temporel fini.

S'il y a une grande ressemblance entre les formalismes POMDP et Dec-POMDP, on ne peut pas

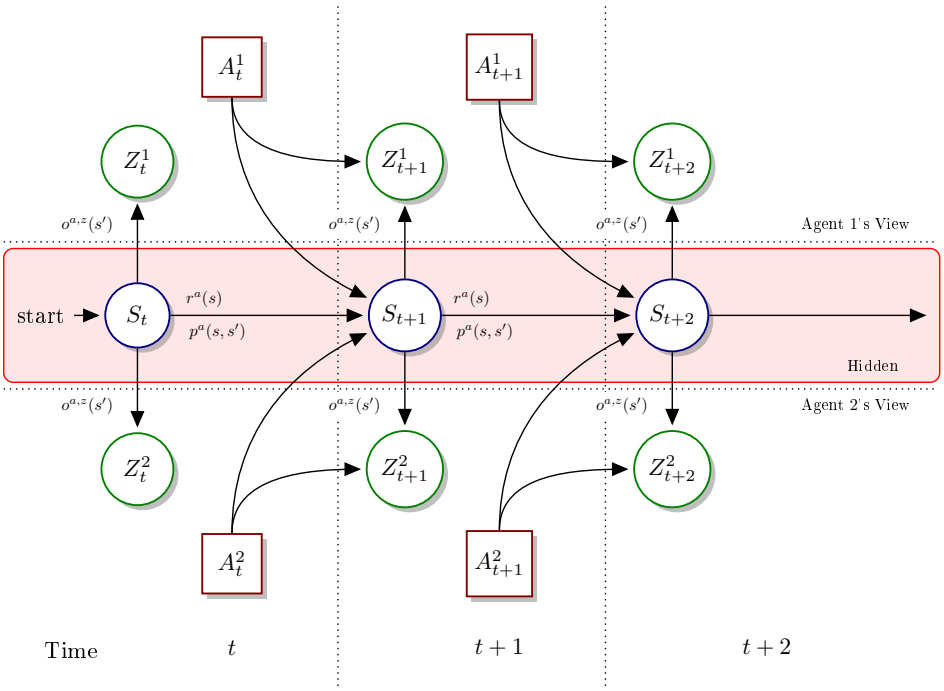


FIGURE 2.5 – Modèle graphique d'un Dec-POMDP

reprendre directement les mêmes solutions. En effet, on ne peut se ramener à un bMDP puisqu'il faut maintenant considérer la croyance de N agents et non plus 1 seul. Une politique (individuelle) sera donc une application de l'ensemble des historiques action-observation (individuelles) possibles vers celui des actions (individuelles), ce que l'on représentera souvent par un arbre (appelé arbre-politique). Par ailleurs, si l'on considère la situation courante pour deux agents après t pas de temps, ce que doit faire l'agent 1 face à une historique h^1 dépend de ce que doit faire l'agent 2 face à *toute* historique h^2 compatible (cohérente) avec h^1, [17] et réciproquement. Cette interdépendance fait qu'on ne peut se contenter de raisonner sur un agent à la fois.

Dans le cadre d'un horizon temporel fini, l'ensemble des arbres possibles pour chaque agent est fini, donc l'ensemble des solutions pour le groupe d'agents est fini et, par conséquent, théoriquement explorable de manière systèmatique. En pratique, du fait d'une complexité doublement exponentielle (Bernstein *et al.*, 2000, 2002), [18] une telle approche est impraticable. Nous présentons ci-après quelques solutions exactes plus adaptées (pour problèmes à horizons temporels finis), même si leurs capacités de passage à l'échelle restent très limitées.

2.3.2.1 Programmation dynamique

Notons d'abord qu'un début d'exécution d'une politique jointe π conduit à un situation à l'instant t décrite par l'état courant s et les historiques $h^1, \ldots, h^N$ des différents agents. Dans une telle situation, les comportements à venir des agents correspondent aux (sous) arbres-politiques $\pi^1_{h_1}$, …, $\pi^N_{h_N}$ attachés aux branches correspondant à leurs historiques actuelles $h_1, \ldots, h_N$. En notant π_h le vecteur des sous-arbres, on peut alors calculer la *valeur* de cette situation comme :

$$V_t^{\pi_h}(s) = \sum_{s' \in \mathcal{S}} \sum_{z \in \mathcal{Z}} P_{s,s'}(\hat{\pi}_h) O_{\hat{\pi}_h,s'}(z) \left[\mathcal{R}(s, \hat{\pi}_h) + \gamma V^{\pi_{h,z}}(s')\right],$$

17. Les agents 1 et 2 connaissent leurs propres historiques, mais pas leurs historiques réciproques.
18. On pourra observer cette explosion dans les algorithmes décrits ci-dessous.

où $\hat{\pi}_h$ est l'action jointe tirée des racines des arbres de π_h, et $\pi_{h,z}$ le vecteur de sous-arbres obtenu après réception de l'observation jointe z.

Comme proposé par Bellman pour les MDP, cette équation permet d'évaluer, à tout instant t, toute sous-politique jointe (de t à l'horizon) en tout état s. Ces valeurs permettent même d'évaluer une sous-politique jointe en n'importe quel état de croyance (en faisant une moyenne pondérée) — la fonction de valeur d'une sous-politique jointe étant un hyperplan — donc en particulier en b_0 pour une politique complète. De là se pose la question de dériver un principe d'optimalité comparable et donc une approche par programmation dynamique. C'est ce qu'ont proposé Hansen *et al.* (2004),[19] lesquels décrivent ainsi deux étapes : (1) la génération de sous-arbres pour l'instant t à partir de ceux obtenus pour $t+1$, et (2) l'élimination des sous-arbres dominés. Plus précisément, si on suppose donné (i) un ensemble d'arbres Θ_i^{t+1} pour chaque agent i, et (ii) une table des valeurs de chaque politique jointe (obtenue par combinaison d'arbres de chaque agent) pour chaque état,[20] alors les deux étapes fonctionnent comme suit :

1. Les sous-arbres de l'agent i à l'étape t peuvent être obtenus en plaçant une action à la racine, et en associant un sous-arbre de Θ_i^{t+1} à chacune des $|\mathcal{Z}|$ branches (une par observation possible). Une fois tous les sous-arbres possibles à l'étape t construits pour tous les agents, on calcule la table des valeurs associées, ce qui clôt l'étape de génération des sous-arbres.
2. Ensuite, pour chaque agent i, on peut éliminer tout sous-arbre θ_i^t dominé, c'est-à-dire tout sous-arbre tel que, quels que soient les sous-arbres associés des autres agents, il existe un sous-arbre $\tilde{\theta}_i^t$ de i de plus haute valeur en au moins un état de croyance. Cette élimination est effectuée par programmation linéaire.

2.3.2.2 Recherche heuristique

L'approche précédente permet d'obtenir une politique jointe optimale quel que soit l'état de croyance initial, au prix d'une grande complexité computationnelle. Étant donné un état de croyance initial et une fonction heuristique admissible, on peut envisager de développer des algorithmes plus efficaces, comme proposé par Szer *et al.* (2005). Leur approche, *Multi-Agent A** (MAA*), repose sur l'algorithme A* (Hart *et al.*, 1968). Chaque nœud de profondeur t correspond à une politique jointe partielle θ^t (de l'instant initial 0 à un instant t) et, depuis un nœud donné (non terminal : $t < T$), les nœuds accessibles correspondent à toutes les politiques jointes qui sont des extensions de θ^t (c'est-à-dire dont les sous-arbres pour chaque agent sont des extensions des sous-arbres de départ). L'évaluation $f(\theta^t) = g(\theta^t) + h(\theta^t)$ d'un nœud se fait en utilisant :

pour le passé (g) : l'espérance des récompenses reçues en appliquant θ^t depuis l'état de croyance initial, ce qui peut être calculé de manière itérative ;

pour le futur (h) : une fonction heuristique admissible, obtenue par exemple en résolvant le POMDP correspondant, c'est-à-dire en supposant qu'une seule entité reçoit les perceptions de tous les agents et contrôle toutes leurs actions, voire en supposant le système complètement observable.

2.3.2.3 Autres approches

Les deux approches exactes décrites ci-dessus ont été à l'origine de nombreux travaux. Des améliorations ont par exemple été faites en observant que certaines historiques étaient équivalentes, donc pouvaient être confondues, ce qui permet de réduire l'espace de recherche (Oliehoek *et al.*, 2013). D'autres travaux introduisent des heuristiques et approximations pour obtenir de bonnes politiques en des temps plus raisonnables, par exemple avec *Memory Bounded Dynamic Programming* (MBDP) de Seuken et Zilberstein (2007). D'autres méthodes exactes sont évidemment envisageables, telles que la programmation mathématique proposée par (Aras et Dutech, 2010). On notera aussi qu'un problème particulier est de s'attaquer aux horizons temporels infinis ou indéfinis, pour lesquels des politiques individuelles sous formes d'arbres ne sont plus appropriées.

19. Cette approche a été initialement développée dans le cadre plus général des jeux stochastiques partiellement observables (POSG), cadre dans lequel chaque agent a son propre objectif.

20. La table est donc de taille $|\mathcal{S}| \times |\Theta_1^{t+1}| \times \cdots \times |\Theta_N^{t+1}|$.

2.3.3 Quand le modèle est mal connu

Outre la capacité à percevoir l'état courant du système à contrôler, on peut aussi remettre en cause la connaissance que l'on a de ce système, à commencer par la validité des probabilités de transition, mais aussi éventuellement des récompenses. Dans les deux cas, il peut s'agir de données issues d'un modèle discutable ou d'estimations statistiques.

Face à un modèle mal connu, on peut principalement adopter deux stratégies :

— lors d'interactions avec le système réel, agir de manière à optimiser les récompenses reçues tout en en améliorant la connaissance du système, ce qui nous amène à l'*apprentissage par renforcement* (A/R ou RL) ; et

— chercher une meilleure politique alors que l'on fait des hypothèses pessimistes sur le modèle, ce qui nous amène par exemple à la *planification robuste* ou à la *planification possibiliste*.

Combiner ces deux stratégies est aussi envisageable.

2.3.3.1 Apprentissage par renforcement

L'apprentissage par renforcement est un problème des plus importants et étudiés dans le cadre des processus de décision markoviens. Ce sujet n'étant toutefois pas prépondérant dans mes travaux, nous ne ferons ici qu'en esquisser les grandes lignes.

Pour commencer, on peut reformuler la description précédente en disant simplement qu'il s'agit, pour un agent interagissant avec un système, d'optimiser les récompenses reçues (typiquement avec un des critères classiques des MDP). [21] En effet, cet objectif implique naturellement d'améliorer sa connaissance du système pour recueillir plus de récompenses. Il pose le problème de faire un compromis entre *exploiter* ses connaissances actuelles et *explorer* la dynamique du système pour améliorer ses connaissances.

Nous ferons ici en premier lieu la distinction entre les approches *sans modèle* (ou *directes*) d'une part, dans lesquelles un algorithme cherche à directement optimiser (/apprendre) une politique par *essai-erreur*, sans estimer le modèle du système, et les approches *avec modèle* (ou *indirectes*) d'autre part, dans lesquelles un algorithme cherche simultanément à estimer le modèle du système et à optimiser une politique.

Premières approches, sans contrôle de la vitesse de convergence Les premières approches d'apprentissage par renforcement sans modèle, telles que le Q-learning (Watkins, 1989; Watkins et Dayan, 1992) ou SARSA (Rummery et Niranjan, 1994; Sutton et Barto, 1998), reposent sur l'idée de transformer l'opérateur d'optimalité de Bellman en un estimateur de la fonction de valeur optimale. Pour une transition $s, a \mapsto s', r$ et avec un pas d'apprentissage $\alpha \in]0, 1[$, la mise à jour utilisée dans le Q-learning est ainsi :

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha[r + \gamma V(s')].$$

Sous certaines hypothèses, dont l'utilisation d'une stratégie d'exploration, la convergence de cette approche vers la fonction de valeur optimale est garantie. Par contre, on ne sait pas à quelle vitesse se fait cette convergence, et on contrôle mal le compromis exploration-exploitation. On peut aussi faire entrer dans cette catégorie certains algorithmes de planification reposant sur un modèle génératif (un simulateur). C'est le cas par exemple d'algorithmes de montée de gradient, en particulier s'ils essayent d'exploiter au mieux chaque nouvelle expérience.

Les premières approches d'apprentissage par renforcement avec modèle, telles que Dyna-Q (Sutton, 1990), à chaque pas de temps, (i) mettent à jour leur modèle, et (ii) calculent une décision optimale étant donné ce modèle (à moins qu'une action d'exploration soit choisie). Comme précédemment, il est souvent aisé de prouver la convergence vers une fonction de valeur (et donc une politique) optimale, mais pas de caractériser la vitesse de cette convergence.

21. On notera par ailleurs que la programmation dynamique approchée est aussi appelée « apprentissage par renforcement batch » parce qu'elle repose souvent sur l'estimation d'une fonction de valeur sur la base d'un ensemble d'échantillons de transitions (obtenues à l'aide d'un simulateur ou d'un système réel).

Un problème d'optimisation bien posé ? Une question que soulève en fait l'apprentissage par renforcement est de savoir s'il s'agit bien d'un problème d'optimisation « bien posé » (au sens large, pas au sens précis d'Hadamard). [22]

L'optimalité ne peut consister en la prise de décisions toutes MDP-optimales — c'est-à-dire optimales par rapport au MDP sous-jacent — ni dès le départ, ni au bout d'un certain temps. On n'est en effet jamais sûr que les expériences faites jusqu'à un instant donné n'ont pas été trompeuses.

Mais considérons plutôt un algorithme qui ne serait pas contraint à une exploration perpétuelle aussi contraignante que dans les algorithmes précédemment mentionnés, et s'autoriserait donc l'abandon au moins de certaines actions dans certains états. Parce que le hasard peut mal faire les choses, on ne peut garantir que cet algorithme convergera vers une politique MDP-optimale, ni-même vers une politique ϵ-MDP-optimale ($\epsilon > 0$). En outre, on voudrait des garanties sur la vitesse de convergence. Comme proposé par Kakade (2003), certains algorithmes peuvent être *PAC-MDP* (probablement approximativement correct par rapport au MDP sous-jacent), ce qui se traduit par le fait que, avec une probabilité $1 - \delta$, ils prennent toujours des décisions ϵ-MDP-optimales, sauf pour un nombre de pas de temps polynomial en les paramètres du problèmes. [23] Toutefois, si la propriété énoncée par Kakade fournit une information quantitative intéressante pour les algorithmes d'apprentissage par renforcement qui la vérifient, elle ne donne pas encore la définition d'un problème d'optimisation.

Une difficulté qui nous poursuit jusqu'ici est de toujours se référer au MDP sous-jacent, lequel n'est pas connu. Si l'on prend simplement comme critère l'espérance de la somme des récompenses futures décomptées, existe-t-il un algorithme (même idéal) qui peut garantir qu'il prendra les décisions maximisant ce critère malgré la méconnaissance initiale du modèle ? Un problème dans cette question est que le critère fait appel à une espérance mathématique. En effet, pour calculer une telle espérance, il faut que l'incertitude sur le modèle soit quantifiée (par une distribution de probabilités). Si, par exemple, seuls deux modèles sont possibles, cette espérance ne sera pas la même selon la probabilité associée à chacun de ces deux modèles.

A/R avec modèle bayésien En réponse aux dernières remarques, nous nous plaçons dans le cadre de l'apprentissage par renforcement avec modèle *bayésien*, c'est-à-dire dans lequel une telle distribution initiale sur les modèles possibles est donnée (Bellman, 1961; Martin, 1967). Chaque nouvelle transition effectuée permet — à l'aide de la formule de Bayes — de mettre à jour cette distribution comme, dans un POMDP, chaque transition permet de mettre à jour la croyance sur l'état caché. On peut en fait transformer ce problème d'apprentissage par renforcement en un POMDP, les paramètres inconnus du modèle devenant une partie fixe et cachée de l'état (Duff, 2002). Souvent, cette reformulation est dénommée *Bayes-Adaptive* (ou *Belief-Augmented*) MDP (BA-MDP). L'équivalence entre les deux problèmes montre que l'apprentissage par renforcement (avec modèle) bayésien est un problème bien posé (comme la résolution des POMDP). Par contre, l'espace d'états du POMDP en question est continu, donc l'espace des états de croyance est de dimension infinie. Ce problème est, en conséquence, très difficile à aborder. Parmi les quelques travaux qui se sont attaqués à l'apprentissage par renforcement bayésien sous cet angle, on citera ici BEETLE, un algorithme à base de points, (Poupart *et al.*, 2006), et deux approches par échantillonnage : *Bayesian Sparse Sampling* (Wang *et al.*, 2005) (inspiré de Sparse Sampling) et *Bayes-Adaptive Monte-Carlo Planning* (BAMCP) (Guez *et al.*, 2013) (inspiré de POMCP).

Propriétés liées à l'optimalité D'abord, pour bien les distinguer, nous continuerons à noter par la lettre V les fonctions de valeur calculées par rapport au MDP sous-jacent, et noterons par la lettre $\mathbb{V}$ les fonctions de valeurs calculées dans le BA-MDP. Avec ces notations, nous introduisons maintenant (de manière plus ou moins formelle) différents concepts liés à l'évaluation de la qualité d'un algorithme $\mathfrak{A}$ d'apprentissage par renforcement avec modèle bayésien :

— Un algorithme est *Bayes optimal* s'il résout de manière optimale le problème d'apprentissage par renforcement avec modèle bayésien, c'est-à-dire que $\mathbb{V}^{\mathfrak{A}}(b, s) = \mathbb{V}^{*}(b, s)$ pour tout b (tout croyance sur le modèle) et tout s (Duff, 2002).

22. Pour Hadamard, un problème est bien posé si (1) il existe une solution ; (2) la solution est unique ; et (3) le comportement de la solution change continûment avec les conditions initiales.

23. Cette dernière partie de la phrase est une forme de garantie sur la vitesse de convergence, même si l'on ne sait pas quand ces « erreurs » sont faites.

— Un algorithme est *presque (*near*) Bayes optimal* (ou *PAC-BAMDP*) (Kolter et Ng, 2009) si, pour $\epsilon > 0$, à chaque instant de décision t, et avec une probabilité au moins $1 - \delta$, la valeur associée à l'algorithme est ϵ-optimale :

$$\mathbb{V}^{\mathfrak{A}}(b_t, s_t) \geq \mathbb{V}^{*}(b_t, s_t) - \epsilon,$$

sauf pour un nombre de pas de temps polynomial en les paramètres du problème.

On notera que, s'il y a un parallèle fort entre les propriétés PAC-MDP et PAC-BAMDP, un algorithme peut être PAC-BAMDP sans être PAC-MDP, comme démontré par Kolter et Ng (2009).

Nous ne nous attarderons pas ici sur le choix d'un *a priori* pour le modèle, et reviendrons en section 4.4 sur le sujet de l'apprentissage par renforcement avec modèle bayésien. Toutefois, pour conclure temporairement, on notera qu'il existe aussi des algorithmes sans modèles qui sont dit bayésiens (Ghavamzadeh *et al.*, 2015, section 5), par exemple parce qu'une distribution sur les fonctions de valeur possibles est maintenue à l'aide d'un processus gaussien.

2.3.3.2 Planification robuste

L'apprentissage par renforcement, même avec une définition claire de ses objectifs, n'est pas nécessairement une solution adaptée quand le modèle du MDP est mal connu. C'est par exemple le cas si l'on veut concevoir hors ligne un contrôleur en partant d'un modèle, même incertain, pour ne plus le remettre en cause après. Ci-dessous, sauf précision contraire, on abordera le cas de la dynamique incertaine.

Dans ce cas, au lieu de chercher une politique optimale pour un modèle moyen, on peut vouloir être prudent et chercher une politique la meilleure possible face au pire modèle envisageable (application du principe de Wald (1945)). Cette forme de *planification robuste* requiert la donnée d'un ensemble de modèles possibles, [24] et la résolution d'un jeu à deux joueurs et à somme nulle. La plupart des approches font l'hypothèse que la dynamique du système est indépendante d'un couple état-action à l'autre, le jeu devenant un jeu alterné dans lequel, dans chaque état rencontré s :

1. le « contrôleur » choisit une action a ;
2. son adversaire choisit le pire modèle de transition depuis s pour l'action a ; et
3. le prochain état s' est échantillonné selon le modèle choisi.

En outre, on considère aussi généralement que l'ensemble des modèles est convexe. Les modèles locaux étant indépendants les uns des autres, cela implique que chaque ensemble de modèles locaux possibles pour un couple état-action est convexe. Dans ce cas, Bagnell *et al.* (2001) (Bagnell, 2003) montrent qu'il existe pour chaque joueur une politique optimale qui est à la fois markovienne, stationnaire et déterministe, et proposent l'algorithme d'itération sur la valeur robuste (*robust value iteration*).

Si la plupart des travaux ont abordé le sujet des dynamiques incertaines, il est aussi pertinent en pratique de traiter les fonctions de récompense incertaines. En effet, il peut par exemple être fastidieux de spécifier une fonction de récompense complète, auquel cas une spécification incomplète –on parle d'élicitation de préférences– amènera à une fonction de récompense incertaine. À ce sujet, voir par exemple les travaux de Regan et Boutilier (2011).

On notera, parmi les problématiques de la planification robuste,

— celle de la représentation efficace — en temps et en espace — du modèle incertain, comme abordée par exemple par Delgado *et al.* (2011) à l'aide de représentations factorisées, et
— les solutions très prudentes (pessimistes) obtenues, ce qui semble inhérent à cette approche.

2.3.4 Quand on optimise un critère non classique

Les sections précédentes ont déjà remis en cause, dans le formalisme original des MDP, l'observabilité complète de l'état du système, l'unicité de l'agent contrôleur, et la connaissance de modèle. Si plusieurs critères « classiques » existent, ils reposent tous sur une moyenne de récompenses cumulées, ce que nous allons remettre en cause ici à travers quelques exemples.

24. Ici aussi on restreindra la discussion à la dynamique du système, le pire modèle de récompense étant trivial à dériver.

2.3.4.1 Aversion au risque

Les approches robustes que nous avons rencontrées en section 2.3.3.2 raisonnaient déjà sur le pire modèle (parmi ceux possibles). On pourrait aussi adopter un point de vue pessimiste (ou, inversement, optimiste), même s'il n'y a pas d'incertitude sur le modèle, en fuyant le risque. Une première façon de faire est de maximiser la pire récompense cumulée envisageable lors d'une exécution. On notera qu'une telle approche (i) est on-ne-peut-plus pessimiste (respectivement optimiste), (ii) ne repose plus sur des incertitudes quantifiées (voir à ce sujet la planification non-déterministe (Cimatti *et al.*, 2003)), et (iii) ne requiert en fait que d'associer chaque politique à l'une de ses exécutions possibles.

Mais on peut aussi intégrer des mesures de risque. Supposons par exemple que l'on veuille pondérer l'espérance du retour par un terme proportionnel à l'écart-type sur ce retour, c'est-à-dire trouver une politique maximisant, pour un état initial s_0 donné :

$$V^\pi(s_0) = E_\pi \left[\sum_{t=0}^{\infty} \mathcal{R}(s_t, \pi(s_t)) \middle| s_0 \right] + \beta \sigma_\pi \left[\sum_{t=0}^{\infty} \mathcal{R}(s_t, \pi(s_t)) \middle| s_0 \right],$$

avec $\beta < 0$ pour un critère averse au risque ($\beta > 0$ pour une recherche de risque). Malheureusement, on ne sait pas calculer ce nouveau critère de manière récursive, ce qui autorisait la programmation dynamique. Il faut donc (i) soit trouver une autre méthode d'optimisation de la politique qui soit compatible avec ce critère (au risque de ne plus pouvoir exploiter le caractère séquentiel des MDP), (ii) soit utiliser un critère *temporellement cohérent* (*time-consistent* / *dynamically consistent*), c'est-à-dire compatible avec la propriété de Bellman, et donc une forme de programmation dynamique (Machina, 1989). On citera simplement ici deux travaux types : Howard et Matheson (1972) proposent d'optimiser un critère exponentiel

$$\frac{1}{\beta} log E_\pi \left[exp \left(\beta \sum_{t=0}^{T} \mathcal{R}(s_t, \pi(s_t)) \right) \right] \underset{\beta \mapsto 0}{\simeq} E_\pi \left[\sum_{t=0}^{\infty} \mathcal{R}(s_t, \pi(s_t)) \middle| s_0 \right] + \beta \; var_\pi \left[\sum_{t=0}^{\infty} \mathcal{R}(s_t, \pi(s_t)) \middle| s_0 \right],$$

et Wu et Lin (1999) proposent de maximiser la probabilité que le retour dépasse une certaine valeur seuil. Une référence plus récente (parmi d'autres) sur cette thématique de recherche est l'article de Baeuerle et Rieder (2014).

2.3.4.2 Modèles non probabilistes

Le cas de la planification robuste sus-cité peut aussi être vu comme une situation type où l'on change de critère parce que l'on sort de la dynamique stochastique usuellement connue. Dubois et Prade (1995) considèrent un problème à un seul pas de temps dans lequel on n'a accès qu'à un modèle *possibiliste* de la dynamique et des préférences. Ces premiers travaux ont ensuite été étendus à plusieurs pas de temps, à commencer pour l'horizon temporel fini par Fargier *et al.* (1998) avec un algorithme de programmation dynamique, puis pour l'horizon indéfini par Sabbadin (2001) avec un algorithme d'itération sur la valeur. Parce que le modèle est ici qualitatif et non quantitatif, il est naturel de ne plus considérer un calcul d'espérance (une utilité espérée), et la fonction de récompense n'a besoin que de spécifier une relation de préférence qualitative entre différentes situations. Divers critères peuvent être définis, par exemple (comme précédemment) selon qu'on adopte un point de vue pessimiste ou optimiste. En l'occurrence, ces critères sont temporellement cohérents, ce qui permet de dériver des algorithmes de type programmation dynamique ou itération sur la valeur.

Ce cadre possibiliste crée par contre certaines difficultés. Une première difficulté est de traiter des scénarios à horizon temporel infini : il n'y a pas d'équivalent du facteur d'atténuation γ. Par ailleurs, ne considérer que la pire ou la meilleure exécution d'une politique donnée est très réducteur. Plusieurs politiques peuvent partager une même meilleure exécution (si l'on adopte un point de vue optimiste), mais différer grandement sur les autres exécutions. Ce problème a été abordé dans le cas d'un seul pas de temps par Fargier et Sabbadin (2005), lesquels ont proposé des méthodes optimisant des critères plus décisifs (distinguant mieux les conséquences de deux politiques de décision).

Parmi les travaux plus récents sur ce sujet, on pourra citer ceux de Drougard *et al.* (2013); Drougard (2015), lesquels proposent en particulier une extension aux POMDP possibilistes.

2.3.4.3 Critères multiples

Les situations décrites précédemment sont souvent dues à une incertitude, soit liée à la stochasticité de la dynamique, soit liée à une méconnaissance du modèle. Une situation différente, mais aussi courante, est celle dans laquelle plusieurs critères existent, de natures diverses (donc incomparables) et potentiellement conflictuels (optimiser l'un n'optimise pas l'autre).

On entre ainsi dans le cadre de la prise de décision séquentielle *multi-critère* (/*multi-objectif*) (Roijers *et al.*, 2013). Dans ce cadre, étant donnée une fonction $\vec{f}$ à valeur dans $\mathbb{R}^n$, on va distinguer ici deux types de problèmes selon qu'on recherche :

— un *front de Pareto*, c'est-à-dire un ensemble maximal de solutions non dominées les unes par les autres,[25] ou

— un *meilleur compromis*, c'est-à-dire une solution maximisant une fonction g — dite d'agrégation ou de scalarisation — croissante sur l'espace des critères.

Pour les MDP multi-objectifs (moMDP), il ne faudra pas se limiter à considérer les politiques stationnaires déterministes. Sauf cas particulier (tel que celui d'une fonction d'agrégation linéaire), on relève généralement l'hypothèse de stationnarité. Cela peut se comprendre avec un moMDP à un seul état, dans lequel choisir des actions différentes selon le pas de temps (ou selon les récompenses déjà accumulées) permet de régler l'équilibre entre les différents objectifs. Ensuite, s'il est acceptable par l'utilisateur d'incorporer le hasard dans la prise de décision, on peut obtenir un meilleur espace de solutions en autorisant des politiques stochastiques ou, avec le même bénéfice, les mélanges de politiques déterministes. Dans ce cas, les valeurs atteignables couvrent l'enveloppe convexe des valeurs du front de Pareto.

Dans le cas non-séquentiel, de nombreuses approches reposent sur des algorithmes évolutionnaires (Ehrgott, 2005; Coello Coello *et al.*, 2007). Dans le cas des MDP multi-objectifs (Roijers *et al.*, 2013), on trouve des approches telles que, pour la recherche du front de Pareto :

— appliquer la programmation dynamique, en remplaçant l'opérateur de maximisation par un opérateur de dominance (lequel retiendra un ensemble de valeurs et non une seule) ; et

— à supposer qu'un état initial est connu, appliquer une recherche heuristique telle que MO-LAO* (Bryce *et al.*, 2007), elle-même apparentée à des recherches heuristiques multicritères pour le cas déterministe (Stewart et White, 1991).

La taille du front de Pareto pouvant croître très rapidement, on peut dans certains algorithmes se contenter de l'approcher à $\epsilon > 0$ près (c'est-à-dire en acceptant d'éliminer une solution si, à ϵ près, elle est dominée par une autre).

Supposons maintenant qu'une fonction d'agrégation est fixée, mais pas ses poids. Si elle est linéaire, on peut remplacer la recherche d'un front de Pareto par la recherche de son enveloppe convexe, laquelle correspond aux solutions optimales pour les différents poids possibles. En l'occurrence, le problème s'apparente à la résolution d'un POMDP dont l'état de croyance correspondrait au vecteur des poids (vu comme une distribution de probabilité sur les critères possibles), d'où un certain nombre d'approches particulières.

Des solutions existent pour trouver des solutions optimales pour (i) des fonctions d'agrégation particulières avec des approches telles que la programmation linéaire (Perny et Weng, 2010), ou (ii) des fonctions d'agrégation quelconques par recherche heuristique dans le cas déterministe (Galand et Perny, 2006).

La généralisation des MDP algébriques Perny *et al.* (2005); Weng (2006) ont introduit la classe abstraite des MDP algébriques, et montré que les MDP à critères non-classiques (et à horizon temporel fini) entrant dans cette classe pouvaient être résolus par le même schéma de programmation dynamique. Ceci facilite l'identification des classes de problèmes pouvant être résolus de cette manière, mais sans apporter de réponse dans un certain nombre de cas n'entrant pas dans ce cadre.

2.3.4.4 Jeux de Markov

Une extension naturelle à plusieurs joueurs des MDP est le jeu de Markov (ou jeu stochastique), dans lequel chaque agent a une observation complète et non bruitée, et son propre critère de performance. De

25. x est sur le front ssi, pour tout $x' \neq x$, (1) soit il existe $i \in \{1, \ldots, n\}$ tel que $f_i(x) > f_i(x')$, (2) soit $\vec{f}(x) = \vec{f}(x')$.

tels cas multi-joueurs peuvent être vus comme combinant

— les problématiques multi-agents, en particulier si l'observabilité est partielle, cas où la factorisation des politiques prend tout son sens ;

— et les critères multiples, chaque agent-joueur ayant son propre critère à maximiser.

Les jeux de Markov à deux joueurs et à somme nulle sont des cas assez simples à traiter, un algorithme de type itération sur la valeur ou Q-learning pouvant être appliqué (Littman, 1994; Littman et Szepesvári, 1996). Littman et Szepesvári (1996); Littman (1996) introduisent même le modèle abstrait des MDP généralisés, lequel permet d'étendre leurs résultats à d'autres problèmes satisfaisant certaines propriétés. En comparaison avec les MDP algébriques vus en section 2.3.4.3.0, ici les modèles de transition restent probabilistes (ce qui permet l'apprentissage par renforcement), et une récompense scalaire est employée (ce qui interdit un certain nombre de critères).

Les choses deviennent plus compliquées dans les jeux à somme générale. Comme démontré par Zinkevich *et al.* sur une classe de problèmes simples (à 2 joueurs, 2 états, 2 actions, transitions déterministes et à critère infini pondéré), calculer la fonction de valeur (vectorielle) n'est pas nécessairement suffisant pour trouver un équilibre, même si celui-ci est unique. Une difficulté particulière est déjà présente dans les jeux répétés, tels que le dilemme du prisonnier itéré, puisqu'on peut faire mieux que répéter un équilibre de Nash en s'accordant sur une solution dont l'application est garantie par l'existence d'une *menace de punition* (voir le théorème de folk (Osborne et Rubinstein, 1994)). Une telle punition est une réponse (coûteuse pour le joueur menacé) qui sera appliquée si un joueur ne respecte pas sa partie du contrat, c'est-à-dire s'il dévie de la politique commune choisie. Littman et Stone (2003) montrent comment résoudre ainsi des jeux répétés, et Murray et Gordon (2007); MacDermed et Isbell (2013) étendent cette approche aux jeux stochastiques.

On notera que, pour pouvoir appliquer une menace en cas de violation d'un contrat, il faut pouvoir détecter cette violation, ce qui est bien plus difficile en cas d'observabilité partielle, ou si seulement les actions des autres agents ne sont pas directement observables. Il n'existe pas à notre connaissance de solution pour les jeux stochastiques partiellement observables, à moins que la récompense soit commune (donc en se ramenant aux Dec-POMDP). Évidemment, le cas à horizon fini serait probablement plus simple à traiter puisqu'il n'y aurait pas de menace à mettre en œuvre.

2.3.5 Une vue d'ensemble

La figure 2.6 dévoile le tableau qui vient d'être dépeint touche par touche de différents modèles markoviens pour la prise de décision séquentielle (en incluant à cette photo de famille les deux proches parents que sont les chaînes de Markov et les HMM). Ces modèles sont organisés en un graphe que l'on peut voir comme un diagramme de classes ou comme un réseau phylogénétique. On notera que différents types de modèles comme de relations sont distingués (voir légende). La relation de reformulation d'un Dec-POMDP en oMDP n'est bien entendue valable que pour les problèmes à horizon fini (ou approchés de la sorte). Seuls les principaux modèles sont présentés, bien d'autres pouvant être obtenus en combinant des extensions, autrement dit par hybridation, comme par exemple les mo-pos-f-Dec-POMDP. On a aussi omis par exemple les problèmes liés à l'apprentissage par renforcement et les MDP généralisés de Littman et Szepesvári (1996); Szepesvári et Littman (1996) mentionnés en section 2.3.4.4.

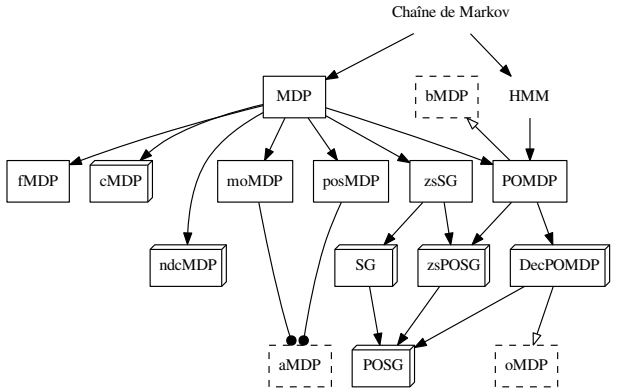
Une première remarque est que les différents types de relations identifiés ici peuvent prêter à discussion. Nous ne chercherons pas à définir de façon précise les différents termes utilisés. Il semblait toutefois intéressant de montrer par ces distinctions que les relations rencontrées paraissent être de diverses natures, et donc mériter des traitements différents. Étendre un modèle (par exemple en ajoutant l'observabilité partielle) n'est pas la même chose qu'abstraire un modèle ou le reformuler. En outre :

— étendre un modèle en un autre risque de rendre les méthodes de résolution habituelles invalides ;

— abstraire un modèle ne change pas les schémas algorithmiques applicables ; mais constater que deux modèles partagent une même abstraction permet d'envisager l'adaptation des solutions de l'un à l'autre (par analogie) ;

— reformuler un modèle sert typiquement à trouver des solutions (en se ramenant à un modèle connu).

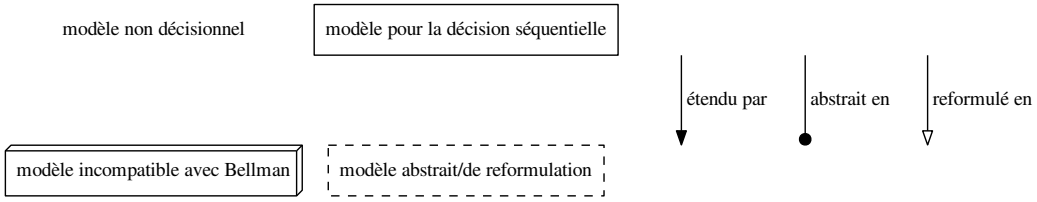
On voit par ailleurs que, parmi les modèles représentant des problèmes de décision séquentielle, plusieurs ne se prêtent pas (du moins directement) à l'emploi du principe d'optimalité de Bellman, et donc à certains schémas algorithmiques usuels (essentiellement la programmation dynamique et la recherche



(a) Des modèles markoviens et leurs relations

HMM	modèle de Markov caché
MDP	processus de décision markovien
POMDP	MDP partiellement observable
fMDP	MDP factorisé
posMDP	MDP possibiliste
moMDP	MDP multi-objectif (but : recherche du front de Pareto)
zs-SG	SG à 2 joueurs et somme nulle
ndcMDP	MDP non dynamiquement cohérent
cMDP	MDP contraint
Dec-POMDP	POMDP décentralisé
SG	jeu stochastique
POSG	jeu stochastique part. obs.
zs-POSG	POSG à 2 joueurs et somme nulle
bMDP	belief MDP
oMDP	occupancy MDP
aMDP	MDP algébrique

(b) Les acronymes utilisés ci-contre



(c) Légende du diagramme ci-dessus (les 4 types de modèles et 3 types de relations rencontrés)

FIGURE 2.6 – Divers modèles markoviens présentés avec leurs relations. 4 types principaux de modèles et 4 types de relations sont distingués par des encadrements et des extrémités de flèches différents.

heuristique). Comme on l'a déjà évoqué, dans le cas de certains critères non classiques, on parle de problèmes non *dynamiquement (/temporellement) cohérents* (Machina, 1989).

Une question sur laquelle nous reviendrons est de savoir dans quels cas on peut, en reformulant le problème considéré, se ramener à un modèle dans lequel le principe d'optimalité de Bellman est à nouveau applicable. On peut même chercher quelles généralisations des MDP permettent d'employer les mêmes types de solution, comme cela a été fait avec les MDP algébriques (Weng, 2006).

2.4 Discussion

En guise de discussion, cette section cherche à prendre du recul par rapport aux travaux évoqués dans cet état de l'art selon trois dimensions qui, à mon sens, illustrent la richesse du domaine de la prise de décision séquentielle dans l'incertain.

2.4.1 Comment résoudre un *MDP en pratique ?

L'état de l'art effectué jusqu'ici a permis d'évoquer une assez grande variété de situations, dont une assez grande variété d'extensions du cadre des MDP, laquelle nous désignerons ici par l'acronyme *MDP. [26] Cela montre non seulement la richesse de ce domaine de recherche, mais aussi la difficulté qu'on pourra avoir à trouver la bonne façon de résoudre un problème donné. En effet, automatiser par un algorithme la résolution d'un problème ou d'une famille de problèmes (pas nécessairement un *MDP) requiert à la fois de

— *modéliser le système* considéré (pour un *MDP, souvent à travers une dynamique et une fonction d'observation) en fonction des données disponibles,
— spécifier un *critère à optimiser* (pour un *MDP, à travers une fonction de récompense et une fonction objectif en dépendant),

26. On emploiera l'acronyme générique *MDP même si certains acronymes ne contiennent pas ces trois lettres, par exemple pour les jeux stochastiques (SG).

— identifier les *contraintes computationnelles* (puissance de calcul et espace mémoire disponibles hors ligne (avant exécution) et en ligne (pendant l'exécution) le cas échéant), et
— choisir/concevoir un *algorithme de résolution*.

Cette tâche est d'autant plus délicate que ces différents aspects sont interdépendants, comme l'illustrent les quelques exemples suivants pris dans le cadre qui nous intéresse ici :

modèle → algorithme Si l'on ne peut énumérer les actions ou les transitions, on devra *a priori* se contenter d'un modèle génératif (un simulateur du système) et employer des algorithmes adéquats.

CPU → algorithme S'il n'est pas possible de calculer une politique optimale complète hors ligne, mais que du temps est disponible en ligne entre deux décisions, alors celui-ci peut être exploité par des algorithmes spécifiques pour prendre une décision pour la situation courante.

modèle → critère → algorithme Si le modèle de la dynamique est mal connu, on peut (1) modéliser cette incertitude sur les probabilités de transition et (2) choisir (par exemple) un critère optimisant la politique face au pire modèle possible, ce qui requiert des algorithmes dédiés.

Toutefois, on listerait essentiellement les mêmes questions dans le cadre plus général de l'optimisation.

2.4.2 Questions scientifiques

Les différents travaux évoqués jusqu'ici illustrent les divers types de questions auxquels l'étude des *MDP peut conduire. On peut citer, à titre d'exemples, les questions suivantes :

— Quel algorithme utiliser pour résoudre un problème donné ?
— Le problème A se ramène-t-il à un problème B mieux connu ?
— L'algorithme X converge-t-il ?
— L'algorithme X s'arrête-t-il ?
— Quelles sont les complexités (en temps et en espace) de l'algorithme X ?
— Quel compromis entre temps, espace et qualité ?
— Quand passer de l'algorithme X à l'algorithme Y ? (selon les paramètres du problème)
— L'algorithme X se généralise-t-il ?
— Comment dériver de bonnes heuristiques ?
— Comment exploiter de la connaissance experte (donnée *a priori*) ?
— Comment exploiter la structure du problème ? (dans un algorithme de résolution ou dans une heuristique)
— Étant donnés des contrôleurs (/politiques), quelles sont les propriétés du système dynamique obtenu ?

Toutefois, on listerait essentiellement les mêmes questions dans le cadre plus général de l'optimisation.

2.4.3 De nombreux sujets abordés

Par ailleurs, les sujets abordés sont nombreux.

L'optimisation est présente sous différentes formes : programmation dynamique, optimisation linéaire, recherche heuristique, optimisation sous contraintes, descentes/montées de gradient, ... L'approximation de fonction est elle-même un problème d'optimisation puisqu'il s'agit de trouver les paramètres minimisant la distance entre une fonction paramétrée et une fonction de référence.

L'incertain est de même très présent. C'est d'abord le cas parce que la nature du problème amène à considérer des modèles probabilistes (ou autres formes d'incertitudes) et, par exemple, le raisonnement bayésien. C'est aussi le cas quand les algorithmes utilisés sont randomisés, reposant par exemple sur des méthodes de Monte-Carlo, d'où des analyses employant des inégalités de concentration.

Enfin (mais sans clore cette liste), on citera les sujets de la théorie de la décision et de la théorie des jeux, voire la théorie du choix social (pour aider à prendre une décision équitable quand plusieurs critères, éventuellement liés chacun à un agent, sont en conflit).

Chapitre 3

Mes travaux sur les MDP

Sommaire

Note : Pour plus de clareté, les intitulés de sections ne touchant qu'à de la planification classique (déterministe) sont précédés par [dét].

3.1 Introduction

Ce chapitre présente un premier ensemble de mes contributions, lesquelles ont en commun de rester pour l'essentiel dans le cadre des MDP (i) à modèle connu et factorisé, et (ii) à observation complète.

3.1.1 La planification probabiliste

On s'intéresse ici souvent à la *planification probabiliste*, c'est-à-dire la résolution de MDP de type « chemin le plus court » [27] formulé à l'aide d'un modèle factorisé de type STRIPS probabiliste (même si encodé en PPDDL). [28]

Un problème STRIPS probabiliste est décrit par un tuple $\langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$, où :

— $\mathcal{P} = \{p_1, \ldots, p_{|\mathcal{P}|}\}$ est un ensemble (fini) de *prédicats* — assimilés à des variables binaires — qui permettent de décrire l'état du monde ;
— $\mathcal{O}$ est un ensemble d'*opérateurs* (voir plus bas) qui permettent de faire évoluer l'état du monde ;
— $\mathcal{I}$ est l'*état initial* ; et
— $\mathcal{G}$ est une description des *états terminaux (/buts)* à travers une conjonction de prédicats.

Un opérateur $o \in \mathcal{O}$ est décrit par un tule $\langle pre(o), c(o), out(o) \rangle$, où :

— $pre(o)$, la *pré-condition* de o, est une conjonction de prédicats devant être vérifiée pour que l'action soit possible ;
— $c(o) \in \mathbb{R}^+$ est le *coût* de o ; et
— $out(o)$ est une *distribution* sur un ensemble d'effets possibles $\{out_1, \ldots, out_{|out(o)|}\}$, où un *effet* (*outcome*) est donné par
  — *add*, une liste de prédicats à *ajouter* (mettre à vrai), et
  — *del*, une liste de prédicats à *effacer* (mettre à faux).

Cette description conduit naturellement à un MDP (i) en transformant les coûts en récompenses et (ii) en décidant de la manière de traiter les trajectoires ne pouvant pas atteindre un état but. Elle permet d'abord de décrire une variété de grands problèmes d'une manière compacte, mais aussi, en particulier dans le cadre déterministe, de calculer des heuristiques admissibles de manière efficace.

Les problèmes de planification décrits en STRIPS, qu'ils soient déterministes ou probabilistes, sont souvent abordés à l'aide de recherches heuristiques, retournant des plans d'action (conditionnels si besoin). Le calcul efficace d'heuristiques admissibles se fait typiquement en relâchant des contraintes, par exemple :

— dans le cas déterministe, en faisant fi des effacements de prédicats, et
— dans le cas probabiliste, en supposant qu'on peut contrôler le hasard.

Cette dernière relaxation, qui consiste à se ramener au cas déterministe, illustre la nature différente des heuristiques selon que la dynamique est certaine ou pas.

Quand la dynamique est stochastique, non seulement plusieurs exécutions sont possibles pour une politique donnée, mais en plus ces exécutions peuvent rester dans des graphes récurrents pendant des durées indéterminées. Ces phénomènes rendent difficiles non seulement les recherches heuristiques elles-mêmes, mais aussi le calcul d'heuristiques admissibles non triviales.

3.1.2 Plan du chapitre

Les travaux présentés ici cherchent le plus souvent, dans le cas de MDP factorisés tels que rencontrés en planification probabiliste, donc « contenant » souvent une certaine structure, le moyen de trouver ou exploiter celle-ci. Ce chapitre décrit ainsi :

— en section 3.2, des travaux abordant des sujets relativement classiques en planification (classique et probabiliste) puisqu'ils visent à *exploiter des abstractions et les heuristiques associées* ;
— en section 3.3, un planificateur probabiliste innovant en ce que, au lieu d'utiliser une recherche heuristique, il repose sur l'*optimisation des paramètres d'un contrôleur par montée de gradient*, puis son adaptation au cas où la dynamique est mal connue ;

27. On cherche à atteindre un but en minimisant des coûts.
28. STRIPS est l'acronyme désignant un algorithme de planification classique (Fikes et Nilsson, 1971) et, par extension, son langage de représentation. PDDL est l'acronyme de *Planning Domain Definition Language* (langage de définition de domaines de planification) (McDermott *et al.*, 1998), et PPDDL (*Probabilistic PDDL*) son extension avec transitions stochastiques (Younes et Littman, 2004).

— en section 3.4, des travaux qui essayent non de résoudre un problème donné, mais d'*apprendre des règles génériques utiles pour une famille de problèmes* ; et
— en section 3.5, des projets auxquels ces travaux ont contribué.

3.2 Exploiter abstractions et heuristiques

Dans cette section, nous considérons des problèmes de planification formalisés de manière structurée, mais en nous intéressant à des approches qui abstraient les états pour relâcher les problèmes et, typiquement, obtenir des estimations heuristiques optimistes.

3.2.1 Agrégation d'états

Note : Des tableaux comme le suivant listent, pour chaque travail de recherche décrit dans ce mémoire, les publications associées.

HSDIP'13 (Tagorti *et al.*, 2013a)

Un point de départ du travail de thèse de Manel Tagorti — co-encadrée par Bruno Scherrer et Joerg Hoffmann — était d'explorer les liens entre des techniques de la planification classique (déterministe) et la planification dans le cadre des MDP. Or la planification classique s'appuie souvent sur l'utilisation d'abstractions par agrégation d'états, par exemple pour calculer des fonctions heuristiques admissibles. Nous nous sommes donc demandé dans quelle mesure cette approche pouvait s'étendre aux MDP.

Une abstraction par agrégation consiste à définir une partition des états, chaque élément de la partition définissant un état abstrait. Dans le cas d'une dynamique déterministe, (i) une transition (/action) abstraite existe entre deux états abstraits σ et σ' s'il existe une action (concrète) a menant d'un état $s \in \sigma$ à un état $s' \in \sigma'$, (ii) le coût d'une transition est le coût de la moins chère action associée, et (iii) un état abstrait est un état but s'il contient un état but du problème original. Le problème ainsi obtenu est une relaxation du problème de départ, et sa résolution permet donc d'estimer de manière optimiste la valeur d'une solution optimale de ce problème de départ. Par ailleurs, plus une abstraction est fine (c'est-à-dire proche du problème original), plus les heuristiques obtenus ont des valeurs proches des valeurs optimales des états.

Dans le cas d'une dynamique stochastique, il faut non seulement définir les actions et transitions possibles, mais aussi dériver les probabilités associées, ainsi que les récompenses. Plusieurs méthodes sont en fait possibles pour complétement définir un MDP abstrait. Nous avons distingué deux méthodes principales : le *MDP moyen* qui utilise la probabilité de transition moyenne et la récompense moyenne, et le *MDP encadrant* (*bounded*), qui considère les intervalles encadrant les probabilités de transition et les récompenses possibles. Dans ce dernier cas, on peut calculer des fonctions de valeurs optimistes (majorantes) et pessimistes (minorantes) en se ramenant à un jeu à deux joueurs dans lequel le second joueur choisi le meilleur (ou le pire) modèle possible parmi les modèles autorisés. [29]

Nos travaux ont montré que :
— pour l'abstraction par MDP moyen, la fonction de valeur calculée ne minore ni ne majore la fonction de valeur optimale du problème de départ ;
— pour l'abstraction par MDP encadrant, si on obtient bien des minorations et majorations de la fonction de valeur, il ne suffit pas de raffiner une abstraction pour resserrer l'encadrement ; une condition supplémentaire (non détaillée ici) est nécessaire pour garantir un resserrement, laquelle n'est pas trivialement vérifiée, sauf dans le cas de MDP déterministes ; par ailleurs, on sait construire un MDP et une abstraction associée tels que, quel que soit le raffinement de cette abstraction, l'erreur d'approximation augmente.

Concernant les résultats obtenus avec l'abstraction par MDP encadrant, on notera qu'on peut tomber dans des cas particuliers quand le choix des agrégations est tel que l'abstraction est une bisimulation du MDP original : la simulation est « parfaite », le MDP encadrant ayant des intervalles (probabilités et récompenses) réduits à des singletons. On comprend bien que raffiner une telle abstraction parfaite risque de nuire. Un problème ouvert est celui de la recherche, à coût raisonnable, d'une telle abstraction bisimilaire de taille minimale. À défaut, il serait déjà très utile de trouver un critère facilement vérifiable

29. Il y a donc un lien fort avec la planification robuste discutée en sections 2.3.3.2 et 3.3.2.

(voire approximativement vérifiable) qui permette de choisir comment raffiner une abstraction de manière à réduire l'erreur.

3.2.2 Maximiser les chances de succès

JAIR (16)	(Steinmetz *et al.*, 2016a)
ICAPS'16	(Steinmetz *et al.*, 2016b)

Problématique

La planification probabiliste introduite en section 3.1.1 a idéalement un double objectif : (1) d'abord maximiser la probabilité de succès (c'est-à-dire d'atteindre un but), et (2) secondairement maximiser une récompense cumulée (/minimiser un coût total). La plupart des travaux abordent seulement l'un des deux problèmes, souvent avec des algorithmes de recherche heuristique comme LAO* ou (L)RTDP.

La recherche d'un chemin le plus court (stochastique) aborde le second critère en supposant qu'on peut atteindre le but avec probabilité 1. On notera qu'elle peut aisément bénéficier d'heuristiques, par exemples issues de la planification classique, pour estimer de manière optimiste la distance au but.

La recherche d'une politique maximisant la probabilité de succès, bien qu'objectif premier, s'avère moins étudiée, peut-être parce que plus difficile. En effet, il est actuellement bien moins aisé d'estimer la probabilité d'atteindre le but depuis un état donné. Étant dépourvues de guide, on s'attend donc à ce que les recherches heuristiques soient inefficaces.

Travaux

Dans le cadre de la thèse de Marcel Steinmetz, encadrée par Joerg Hoffmann, nous avons décidé d'étudier plus avant les possibilités offertes par les recherches heuristiques. Nous avons ainsi (Steinmetz *et al.*, 2016b,a) :

- effectué une *analyse des algorithmes* considérés : *topological VI* (Dai *et al.*, 2011) [30], AO*, LRTDP (section 2.2.3.1), ILAO* (Hansen et Zilberstein, 2001), HDP (Bonet et Geffner, 2003b), et le cadre FRET (Kolobov *et al.*, 2011) (lequel permet d'appliquer LRTDP même en présence de culs-de-sac) ;
- défini, à l'aide de cette analyse, des *familles d'algorithmes* partageant une structure commune et se distinguant par certaines caractéristiques (telles que le calcul d'un minorant et/ou d'un majorant, les mises à jour en descendant ou en remontant, le test de terminaison par labelisation ou par seuil) ; les trois principales familles sont celles de AO*, LRTDP, et DFHS (*Depth-First Heuristic Search*, laquelle regoupe ILAO* et HDP) ;
- proposé une *méthode d'abstraction* par bisimulation déterminée (création d'un problème équivalent de plus petite taille) ;
- adjoint des *heuristiques* — issues de la planification classique, et mises en œuvre en déterminisant nos problèmes — qui permettent de détecter certains culs-de-sac ; et
- considéré différentes *stratégies de sélection des nœuds* action (et des nœuds états quand le choix est possible).

Sur cette base, nous avons mené une étude expérimentale assez large sur des bancs d'essais issus de l'état de l'art (avec des modifications pour certains problèmes de type « chemin le plus court »), mais aussi inspirés du problème des tests de pénétration en sécurité informatique (voir section 4.5.2), et avec trois objectifs :

MaxProb : maximiser la probabilité d'atteindre le but ;

AtLeastProb : trouver une solution de probabilité supérieure à un seuil $\theta \in [0;1]$ (ou prouver qu'il n'en existe pas) ; et

ApproxProb : trouver une solution optimale à $\epsilon \in [0;1]$ près.

30. TVI n'effectue pas une recherche heuristique, et requiert une énumération de l'espace d'état pour proposer un ordre de mise à jour efficace.

Dans l'ensemble, cette étude montre que l'emploi de ces heuristiques, en particulier h^{max}, permet d'améliorer nettement le taux de couverture[31] par rapport à une recherche non guidée. En effet, tout cul-de-sac détecté (ou tout but atteint) est une information utile que les algorithmes vont remonter et qui va leur permettre de ré-orienter leur exploration. Comme il faut aller assez loin dans les trajectoires pour trouver une information nouvelle à remonter, les recherches en profondeur (familles LRTDP et DFHS) s'avèrent globalement plus efficaces que les autres.

Par ailleurs, l'observation des comportements anytime (l'évolution de la valeur de l'état initial pendant la planification) montre qu'il y a un compromis à faire en vitesse de résolution et consommation mémoire : dans les problèmes acycliques, l'algorithme le plus rapide pour trouver une politique optimale, $\text{AO}^*|_{\text{L}}$,[32] développe une grande partie de l'espace d'état, et est donc aussi le premier à consommer les ressources mémoire. On note aussi que les critères AtLeastProb et ApproxProb permettent une terminaison bien plus tôt, ce qui devrait encourager leur emploi quand il répond aux besoins.

Les travaux futurs incluent (i) l'utilisation d'abstractions pour dériver des heuristiques permettant d'encadrer plus finement la probabilité d'atteindre le but (voir section 3.2.1) ; (ii) la combinaison avec des recherches arborescentes Monte Carlo (MCTS) (voir section 2.2.3.2) ; et (iii), dans le cas par exemple des problèmes avec ressources limitées, la généralisation des encadrements en exploitant les relations de dominance (de manière analogue à la généralisation permise par la propriété PWLC dans les POMDP).

3.2.3 [dét] Planification factorisée (projet SuperCom)

IJCAI'07 (Kelareva *et al.*, 2007)

Le projet SuperCom visait à proposer des méthodes de supervision (diagnostic et planification) de *systèmes composites*, c'est-à-dire formés de multiples composants organisés en réseaux, tels que des réseaux de capteurs ou des services web. Du fait de ce contexte, et au cours d'un stage d'Elena Kelareva co-encadré par Jinbo Huang et Sylvie Thiébaux, nous avons abordé des problèmes de planification déterministes *factorisés* au sens que leurs domaines (les variables et actions sur lesquelles ils reposent) peuvent être décomposés en sous-domaines peu dépendant les uns des autres. Un sous-domaine est défini par

- les actions, dites *réelles*, propres à ce sous-domaine,
- les variables apparaissant dans la définition de ces actions, et
- des actions *abstraites*, lesquelles permettent de remplacer des actions propres à d'autres sous-domaines.

Typiquement, dans un système composite, chaque composant sera associé à un tel sous-domaine. Nous illustrerons notre approche à l'aide du *problème de la fenêtre* décrit figure 3.1, dans lequel une personne veut lancer une balle à travers une fenêtre sans la casser, et où chaque groupe d'actions est réduit à un singleton.

Une telle forme de factorisation peut être exploitée différemment de ce qu'un planificateur STRIPS commun ferait. À l'époque de ce travail, les deux planificateurs factorisés de référence étaient PartPlan (Amir et Engelhardt, 2003) et LID-GF (Brafman et Domshlak, 2006), et reposaient tous deux sur la programmation dynamique, avec des besoins en mémoire potentiellement énormes. Nous avons au contraire proposé une approche par recherche arborescente dont voici les principaux ingrédients :

- les sous-domaines sont d'abord organisés selon un *decomposition Tree* (dTree) (Darwiche, 2001; Darwiche et Hopkins, 2001) dont les feuilles sont les groupes d'actions (voir figure 3.2-b) ;
- ce dTree permet ensuite d'ordonner les sous-domaines de sorte que (i) deux sous-domaines consécutifs partagent le plus de variables possible, et (ii) le premier sous-domaine contienne le plus de variables apparaissant dans le but (voir figure 3.2-a) ;
- notre recherche arborescente a un niveau de profondeur par sous-domaine (dans l'ordre précédemment défini), et à chaque nœud correspond un problème de planification local au sous-domaine, spécifié par des conditions de départ et d'arrivée sur ses variables partagées avec les sous-domaines qui le précède ;

31. Le taux de couverture est la proportion de problèmes résolus sur l'ensemble des problèmes testés. Un temps et une mémoire limites étaient fixés.

32. Variante de AO* n'utilisant qu'une minoration de la fonction de valeur.

Variables :

```
open ? {true, false}
broken ? {true, false}
ball ? {inside, outside}
```

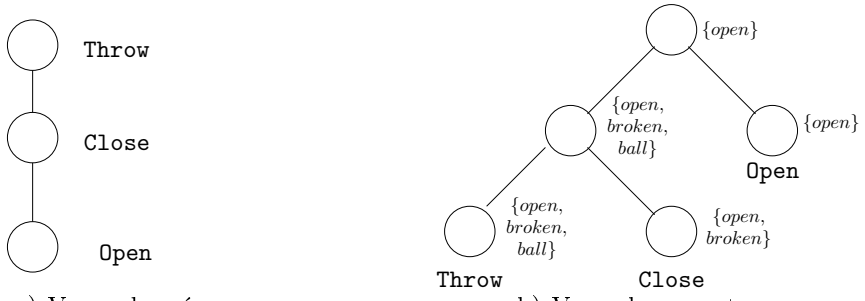
Actions :

```
  Open:
(open=false)                -> Set(open=true)
  Close:
(open=true & broken=false) -> Set(open=false)
  Throw:
(open=true & ball=outside) -> Set(ball=inside)
(open=true & ball=outside) -> Set(ball=inside
                & broken=true & open=true)
```

Init state : `ball=outside & open=false & broken=false`
Goal state : `ball=inside & open=false & broken=false`

FIGURE 3.1 – Le problème de la fenêtre

— dans un nœud, un planificateur énumère les solutions possibles et, pour chacune d'elles, transforme les séquences d'actions abstraites qu'elle contient (dénommées « trous ») en un problème de planification pour le nœud suivant, ce qui permet un processus récursif en partant, à la racine, du problème original.



a) Vue ordonnée b) Vue arborescente

FIGURE 3.2 – Deux vues des sous-domaines de notre exemple

L'algorithme obtenu, dTreePlan, bénéficie en plus de quelques détails d'implémentation tels que (i) l'expansion progressive de la taille de plan maximale autorisée, (ii) l'utilisation, dans chaque nœud, d'un planificateur SAT (Kautz et Selman, 1992) (utilisant le solveur zChaff de Moskewicz *et al.* (2001)), et (iii) l'emploi d'une mémoire cache se souvenant des sous-problèmes déjà rencontrés, qu'ils aient pu être résolus ou pas.

Des expériences ont été menées sur une famille de problèmes en faisant varier certains paramètres de l'algorithme. Les résultats ont montré la validité de l'approche, mais le problème de la planification factorisée reste difficile et mal compris, comme expliqué par Fabre *et al.* (2010). On pourra citer, parmi divers travaux ultérieurs apparentés, ceux sur la planification multi-agent distribuée de Nissim et Brafman (2014), lesquels sont allés plus loin dans l'utilisation d'un algorithme de recherche arborescente.

3.3 Recherche directe d'un contrôleur et robustesse

Comme expliqué en section 3.1.1, la planification probabiliste est souvent abordée à l'aide de recherches heuristiques. Dans le cadre du projet DPOLP (*cf.* section 3.5.1), avant mon arrivée, Aberdeen *et al.*

(2004) ont ainsi employé LRTDP pour résoudre des problèmes assez génériques Après mon arrivée, nous avons proposé avec Douglas Aberdeen de résoudre ces mêmes problèmes non pas en cherchant un plan conditionnel (une politique) mais en optimisant un contrôleur paramétré, comme détaillé en section 3.3.1. Nous avons aussi abordé la question de la planification robuste (introduite en section 2.3.3.2) à la fois dans le cadre d'une recherche heuristique (sections 3.3.2.1 et 3.3.2.2) et dans celui d'une montée de gradient (section 3.3.2.3). La table 3.1 donne une vue d'ensemble de ces différents travaux.

TABLE 3.1 – Grille de lecture des travaux de cette section

planification	normale	robuste
avec (L)RTDP	(Aberdeen *et al.*, 2004)	sec. 3.3.2.1 et 3.3.2.2
par montée de gradient	sec. 3.3.1	sec. 3.3.2.3

3.3.1 Planification probabiliste par montée de gradient (FPG)

PLAPUDD'05	(Aberdeen et Buffet, 2005)
IPC'06	(Buffet et Aberdeen, 2006b)
IPC'06 ?	(Buffet et Aberdeen)
ICAPS'07 a	(Buffet et Aberdeen, 2007a)
ICAPS'07 b	(Aberdeen et Buffet, 2007)
AIJ	(Buffet et Aberdeen, 2009)
PSUWS'10	(Buffet et Hoffmann, 2010)

3.3.1.1 Planification par montée de gradient

Optimiser un contrôleur — en l'occurrence par montée de gradient — est une approche très inhabituelle pour un problème de planification probabiliste, entre autres parce qu'elle implique une résolution approchée. En effet, à moins que les contrôleurs soient très expressifs, donc réglés par un grand nombre de paramètres, il n'est pas garanti qu'ils permettent d'encoder une politique optimale.

Forme des contrôleurs Le contrôleur doit prendre en entrée les prédicats décrivant l'état du monde et retourner une action. Ici, l'entrée est un vecteur binaire (plutôt qu'une liste de prédicats) $\vec{s}_t = (p_{1,t}, \ldots, p_{n,t})$, ce qui lui permet d'être de taille fixe. Par ailleurs, la sortie est une distribution de probabilité sur les actions possibles dont on pourra échantillonner une action à un instant donné, les contrôleurs stochastiques étant préférables dans le contexte d'une telle montée de gradient à espace d'action discret.

En pratique, nous utilisons un approximateur affine (un perceptron sans couche cachée) suivi d'une fonction de régression logistique restreinte aux actions dont les préconditions sont satisfaites dans l'état $\vec{s}_t$ (ensemble $\mathcal{A}(\vec{s}_t)$). En notant $\vec{\theta} = (\vec{\theta}_1, \ldots, \vec{\theta}_{|\mathcal{A}|})$ le vecteur paramètre (décomposé en $|\mathcal{A}|$ vecteurs, chacun de dimension $|\mathcal{P}|$) du contrôleur, la probabilité d'exécuter une action $i \in \mathcal{A}(\vec{s}_t)$ est donnée par :

$$\pi(a_t = i|\vec{s}_t; \vec{\theta}) = \frac{exp(\vec{s}_t^{\intercal}\vec{\theta}_i)}{\sum_{j\in\mathcal{A}(\vec{s}_t)} exp(\vec{s}_t^{\intercal}\vec{\theta}_j)}.$$

Algorithme En planification probabiliste, une exécution (une trajectoire dans l'espace des états partant de l'état initial) peut conduire (i) à un état de succès (où la condition but $\mathcal{G}$ est satisfaite), (ii) à un état d'échec (parce qu'aucune action n'est exécutable), ou (iii) dans un sous-graphe absorbant dénué d'états de succès ou d'échec. Nous ne sommes pas confrontés à un problème de plus court chemin stochastique « simple », tel que considéré par Barto *et al.* (1995), dans lequel toute trajectoire suivie par l'algorithme (donc en s'adaptant si besoin) conduit nécessairement au but. Comme expliqué par Bryce *et al.* (2007), une façon naturelle de bien poser le problème de la planification probabiliste est de prendre en compte deux critères : d'abord en maximisant la probabilité d'atteindre le but et, ensuite (si plusieurs politiques sont optimales du point de vue de ce premier critère), en minimisant l'espérance du coût d'une trajectoire.

Une descente de gradient ne pouvant traiter une telle combinaison de deux critères, nous nous ramenons au seul critère de maximiser la probabilité de succès (en ignorant les coûts des actions). En outre,

comme il est difficile de détecter les sous-graphes absorbants (voir la section 3.2.2), une trajectoire sera considérée comme un échec si et seulement si soit elle atteint un état sans action possible, soit elle dépasse une certaine durée $T_{\max}$.

Parce que toute trajectoire se termine, nous pourrions utiliser une descente de gradient pour processus régénératifs (Buffet, 2008). En pratique, nous employons toutefois l'algorithme OLpomdp (algorithme 5, page 18) pour son caractère en-ligne : à chaque pas de temps, le vecteur de paramètres est mis à jour. En comparaison avec des algorithmes tels que *Natural Actor-Critic*, on préfère ici un algorithme qui profite de la génération rapide de nombreuses trajectoires, quitte à moins bien exploiter chaque nouvelle expérience.

En théorie, parce que les trajectoires ont des durées variables, OLpomdp optimise non pas la probabilité de succès, mais la fréquence à laquelle un état de succès (associé à une récompense $r_s > 0$) est atteint. Cela pourrait être corrigé en ajoutant des pas de temps fictifs, mais au prix d'une moins grande efficacité en pratique. Par ailleurs, en guise d'heuristique, la fonction de récompense est, à chaque instant, proportionnelle au nombre de prédicats appartenant au but ajoutés ou retirés de l'état courant. De manière plus formelle, on peut l'écrire $\lambda * [n_{\oplus}(s, s') - n_{\ominus}(s, s')]$ où $\lambda > 0$, et $n_{\oplus}(s, s')$ (respectivement $n_{\ominus}(s, s')$) est le nombre de prédicats du but apparaissant (resp. disparaissant) lors de la transition $s \to s'$. Ce choix permet d'encourager les rapprochements vers un état but et de décourager les éloignements.

L'algorithme résultant (et le planificateur développé) est appelé FPG pour « *Factored Policy Gradient* ».

Implémentation Le planificateur FPG repose essentiellement sur l'interfaçage entre

— MDPSim, un simulateur de MDP originellement développé par Younes *et al.* (2005) pour permettre d'évaluer les compétiteurs de l'IPPC (*International Probabilistic Planning Competition*), et

— libPG, une bibliothèque d'apprentissage par renforcement principalement orientée vers les montées de gradient, initiée par Douglas Aberdeen.

Les problèmes traités sont formalisés en PPDDL (*Probabilistic Planning Domain Description Language*) (Younes, 2003; Younes et Littman, 2004), sans fluents,[33] donc en logique du premier ordre. Une première difficulté est alors d'identifier les prédicats et actions réellement envisageables, pour éviter de multiplier les entrées et sorties inutiles, ce qui se fait à l'aide d'une analyse d'accessibilité partant de l'état initial.

Résultats Nous présentons ici des résultats expérimentaux obtenus sur les problèmes de la 5^e compétition internationale de planification (IPC), catégorie probabiliste, à laquelle FPG a participé en 2006, et qui propose divers bancs d'essais reconnus par la communauté.

La compétition évaluait les planificateurs concurrents sur 9 domaines spécifiés en PPDDL, avec 15 problèmes par domaine, et 30 simulations d'exécutions pour chaque problème. Ce nombre de simulations est insuffisant pour une mesure de performances fiable, mais des contraintes de temps nécessitaient une telle limitation. La compétition imposait une limite de 40 minutes pour chaque problème, incluant l'optimisation (calculs hors- et en-ligne) et les 30 simulations. Plusieurs de ces domaines, tels que le monde des blocs (*Blocksworld* (BW)), sont des domaines déterministes classiques auxquels du bruit a été ajouté aux effets.

La table 3.3 montre le résumé des résultats, par domaine et par planificateur. Ces planificateurs incluent, en plus des concurrents, FF-replan (Yoon *et al.*, 2007) (précédent gagnant de la compétition, lequel repose sur la déterminisation du problème et l'utilisation du planificateur classique FF (Hoffmann et Nebel, 2001)) et FQL (*Factored $Q(\lambda)$-Learning*, un algorithme de $Q(\lambda)$-learning qui reprend le même type d'approximateur factorisé que dans FPG, et proposé en sus de celui-ci). Les paramètres de FPG et FQL (table 3.2) ont été réglés manuellement sur différents domaines d'IPC-4 et sur des bancs d'essais préliminaires d'IPC-5, mais pas sur les problèmes d'IPC-5.

FPG et FF-replan s'avèrent bien plus robuste à une variété de domaines que les autres planificateurs. Cela s'explique en partie par la maturité logicielle : FPG et FF-replan reposent en majeure partie sur des logiciels stables (FPG←(MDPSim+LibPG), FF-replan←(FF)), alors que deux des trois autres planificateurs n'étaient pas exempts de bugs au moment de la compétition. Dans certains domaines, Paragraph

33. Les seules fonctions des objets du monde sont des prédicats, donc à valeurs booléenne, et non à valeurs sur de plus grands ensembles.

TABLE 3.2 – Principaux réglages des paramètres de FPG lors d'IPC-5 et de FQL. D'autres facteurs d'atténuation de différences temporelles ont été expérimentés ($\lambda = 0.8$ et $\lambda = 1$) avec des résultats quasi-identiques.

	pas size	facteur d'atténuation	λ	récompense pour succès	récompense pour progrès
FPG	$\alpha = 0.00005$	$\beta = 0.85$	N/A	$r_s = 1000$	$r_{progrs} = 100$
FQL	$\alpha = 0.001$	$\gamma = 0.85$	$\lambda = 0$	$r_s = 1000$	$r_{progrs} = 100$

TABLE 3.3 – Résumé des résultats sur les bancs d'essai d'IPC-5. Les valeurs sont des pourcentages de simulations de plans ayant atteint l'objectif (sur 30 essais). Un tiret indique que le planificateur n'a pas été exécuté, ou a échoué à fournir des résultats.

Domaine	FOALP	sfDP	FPG	Paragraph	FF-replan	FQL
BW	100	29	63	–	86	0
Exploding BW	24	31	43	31	52	7
Tire	82	–	75	91	82	11
Zeno	–	7	27	7	100	7
Drive	–	–	63	9	71	11
Elevator	100	–	76	–	93	1
Pitchcatch	–	–	23	–	54	0
Schedule	–	–	54	1	51	54
Random	–	–	65	5	100	10

et FOALP — qui, à la différence de FPG et FF-replan, sont optimaux — ont les meilleurs résultats sur la plupart des problèmes, comme décrit plus en détails par Buffet et Aberdeen (2006b).

FQL s'avère compétitif seulement sur quelques problèmes des domaines Drive, Random, Schedule et Tireworld. Nous présumons que FPG réussit mieux que FQL parce que FQL cherche à approcher une Q-valeur optimale pour chaque couple état-action alors qu'il suffit à FPG de retourner une bonne action dans chaque état. C'est cohérent avec des observations faites dans le jeu Tetris (Thiéry et Scherrer, 2009). Dans le domaine Schedule, FQL et FPG ont des performances comparables pour chaque problème. On notera par ailleurs que FQL est plus difficile à régler que FPG. De meilleurs réglages pourraient avoir donné de meilleurs résultats pour FQL ou pour FPG.

Extension aux problèmes temporels à actions concurrentes FPG désigne aussi un algorithme analogue qui permet de traiter des problèmes plus complexes. Ces problèmes se distinguent de la planification probabiliste considérée jusqu'ici en ce que :

— des ressources (donc des variables numériques) doivent être gérées ;
— les actions consistent à lancer un ou plusieurs opérateurs en parallèle ;
— les opérateurs ont des durées propres et incertaines, décrites par des distributions de probabilités ; des nouveaux opérateurs peuvent être lancés alors que d'autres sont en cours d'exécution.

Ces trois dimensions amènent des difficultés parce qu'elles impliquent des entrées continues, une multiplication des instants de décision possibles (qui ne correspondent pas qu'aux instants où des événements ont lieu (Cushing *et al.*, 2007; Mausam et Weld, 2008)), et à une explosion du nombre des actions possibles.

En résumé, la variante de FPG conçue pour de tels problèmes repose sur les choix suivants :

— les instants de décisions correspondent à des occurrences d'événements ;
— les ressources ne servent pas d'entrées aux contrôleurs, mais leur insuffisance permet d'interdire des actions ;
— pour chaque opérateur, un approximateur affine et une fonction de régression logistique sont employés pour calculer une probabilité d'exécution ;
— tant que des opérateurs incompatibles (mutuellement exclusifs) sont parmi les opérateurs échantillonnés, l'un d'eux est éliminé.

FPG (y compris dans une variante remplaçant l'approximateur affine par un arbre de décision) a été comparé expérimentalement avec Prottle (Little *et al.*, 2005), et un planificateur basé sur LRTDP (Aberdeen *et al.*, 2004). Les résultats décrits par Buffet et Aberdeen (2009) montrent la capacité de l'algorithme à fournir des politiques satisfaisantes sur des problèmes de grande taille.

Conclusion Ces travaux nous ont permis de montrer que, en planification probabiliste, l'optimisation directe de contrôleur était une solution souvent viable, et parfois plus adaptée que des approches plus classiques telles que les recherches heuristiques, en particulier quand les espaces d'états et d'actions sont très grands.

3.3.2 Planification robuste (LRTDP, FPG)

IJAIT	(Buffet, 2007a)
IJCAI'05	(Buffet et Aberdeen, 2005a)
ICTAI'05	(Buffet, 2005a)
PLAPUDD'05	(Buffet, 2005b)
RLNSE'05	(Buffet et Aberdeen, 2005c)
PLMUDW'06	(Buffet et Aberdeen, 2006c)

3.3.2.1 (L)RTDP robuste

Nous avons déjà vu en section 2.3.3.2 que, dans le cas où le modèle (de la dynamique) est incertain, la planification robuste cherche une meilleure politique face au pire modèle envisageable. On fait alors couramment l'hypothèse que le modèle associé à un couple état-action est indépendant des autres couples état-action. Sous cette condition, chercher une politique maximisant le gain face au pire modèle revient à transformer le problème de prise de décision séquentielle de départ en un jeu alterné à deux joueurs, le second joueur choisissant le pire modèle pour le couple (s, a) après que le premier a choisi l'action a dans l'état s.

Les premiers travaux de la littérature sur le sujet ont montré comment des algorithmes tels que l'itération sur la valeur pouvaient être adaptés à ce cas. Avec Douglas Aberdeen, nous avons montré comment RTDP (et LRTDP, décrits en section 2.2.3.1) peut aussi être rendu robuste, par une adaptation similaire, tout en préservant ses propriétés de convergence (malgré un schéma algorithmique moins symétrique qu'itération sur la valeur) (Buffet et Aberdeen, 2005a). L'approche ressemble à celle qui permet d'adapter RTDP aux jeux stochastiques, mais avec la contrainte de devoir considérer tous les états atteignables sous un quelconque modèle possible, et non en se limitant aux états atteignables sous un pire modèle.

3.3.2.2 Analyse d'accessibilité pour (L)RTDP robuste

(L)RTDP (normal) ne fonctionne que sur des problèmes tels que (i) les récompenses sont toutes négatives (donc représentent des coûts) et (ii) toute trajectoire suivie par l'algorithme atteint un état terminal avec probabilité 1. La plupart du temps, il est aisé de vérifier ces deux hypothèses, ou du moins de détecter si l'état courant peut encore conduire à un état terminal (en utilisant une planification déterministe).

Dans le cas de la variante robuste, le graphe représentant l'accessibilité entre états (selon les actions) n'est pas fixe. Vérifier qu'un état terminal est toujours accessible, quel que soit le modèle, est une tâche plus complexe. Nous avons donc développé des algorithmes spécifiques d'analyse d'accessibilité tenant compte de la forme variable du graphe (Buffet, 2005a,b, 2007a).

Pour ce faire, on va caractériser les états selon qu'ils sont :

atteignants (*reaching*) : il existe une politique telle que, quel que soit le modèle, la probabilité d'atteindre un état but est non nulle ;

cul-de-sac (*dead-end*) : quels que soient le modèle et la politique, la probabilité d'atteindre un état but est nulle ; et

dangereux (*dangerous*) : quels que soient le modèle et la politique, la probabilité d'atteindre un cul de sac est non nulle.

On notera qu'un état ne peut être atteignant et cul-de-sac simultanément, mais que les autres combinaisons sont possibles. Pour caractériser ainsi les états atteignables on va raisonner, pour chaque état s et chaque couple (s, a), sur les listes (minimales) d'états qui *ne peuvent pas être interdits simultanément* par l'adversaire (en choisissant le modèle local). À l'aide de ces listes, on va d'abord déterminer, en partant des états buts, quels sont les états atteignants, puis quels sont les états dangereux. Les états dangereux et non atteignants sont les états cul-de-sac. La figure 3.3 permet d'illustrer ces concepts sur deux exemples.

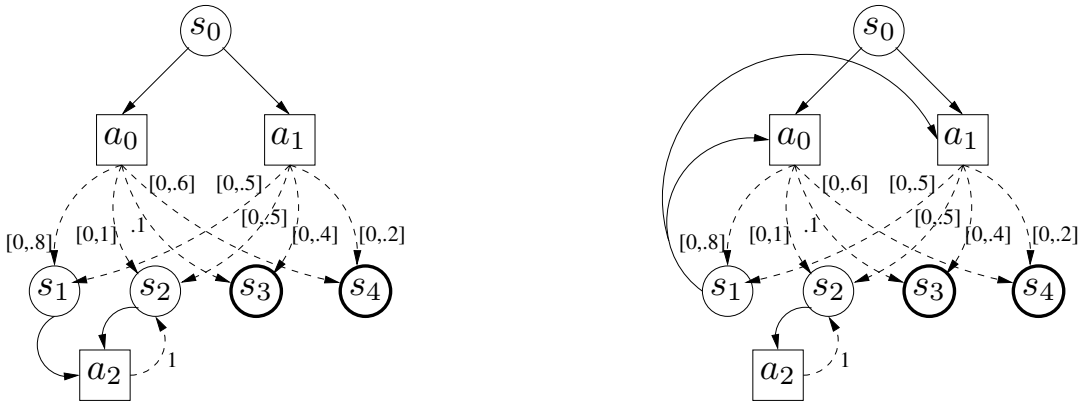


FIGURE 3.3 – Deux des problèmes tests employés. s_3 et s_4 sont des états buts. Les récompenses ne sont pas représentées. Dans les deux cas, la meilleure politique en s_0 consiste à choisir a_0, ce qui donne (au pire) une probabilité $0,1$ d'atteindre l'état but s_3 (et une probabilité $0,9$ d'atteindre l'état cul-de-sac s_2).

On peut comparer les processus mis en jeu à la recherche d'un *plan fort* pour un problème de planification non-déterministe particulier (Cimatti *et al.*, 2003). Notre cadre est toutefois plus complexe puisque l'adversaire ne peut choisir précisément le résultat d'une transition, mais restreint les résultats possibles.

Cette analyse d'accessibilité nécessite le développement du graphe des états accessibles, lequel peut être de très grande taille. Des approches exploitant la factorisation d'un modèle pourraient faciliter une telle analyse.

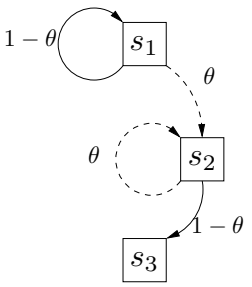
3.3.2.3 FPG robuste

Comme on l'a vu, il est courant que le modèle de la dynamique d'un système soit factorisé (*cf.* Probabilistic STRIPS ou PPDDL), donc que le même schéma d'action soit applicable en différents états décrits par des variables d'états.

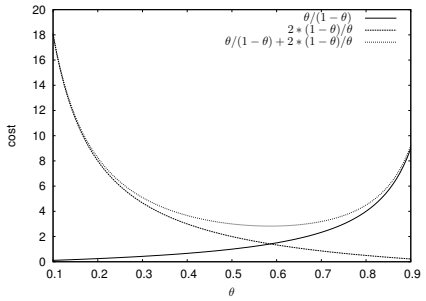
En revanche, les approches robustes font l'hypothèse que les modèles d'action sont indépendants d'un couple état-action à l'autre, hypothèse qui n'est donc plus valide dans un cadre factorisé. Les approches de planification robuste courantes ne fonctionnent donc plus : on ne peut pas se ramener à un jeu à deux joueurs (et à somme nulle) alterné. La figure 3.4 illustre le problème rencontré sur une chaîne de Markov (un MDP à une seule action) incertaine dans laquelle le modèle de transition depuis les états s_1 et s_2 dépend du même paramètre $\theta \in [0, 1]$. Si l'objectif est de minimiser le temps moyen pour atteindre s_3 (mais que le « temps » est deux fois plus coûteux en s_2 qu'en s_1), il faut régler $\theta \simeq 0,58$.

Par contre, globalement, le principe de chercher une politique la meilleure possible face au pire modèle reste valide. C'est donc ce que nous avons cherché à faire dans nos travaux (Buffet et Aberdeen, 2005c, 2006c). Notre objectif étant avant tout de trouver une telle politique (et non un pire modèle), il est naturel de prendre un point de vue proche des jeux de Stackelberg (Fudenberg et Tirole, 1991; von Stackelberg, 2011), le second joueur choisissant sa stratégie (son modèle) en connaissance de la stratégie (la politique stochastique) du premier. Au lieu de chercher un équilibre, donc une paire de stratégies pour chaque joueur, on peut se contenter de chercher une meilleure politique sachant que l'adversaire y donnera sa meilleure réponse possible. On a ainsi un double problème d'optimisation (hiérarchisé).

En l'occurrence, il peut ne pas exister de politique déterministe qui soit optimale. On doit donc chercher dans l'espace des politiques stochastiques. Ici, on décide d'employer deux descentes de gradient simultanées : l'une (lente) pour l'optimisation de la politique, l'autre (rapide) pour l'optimisation du



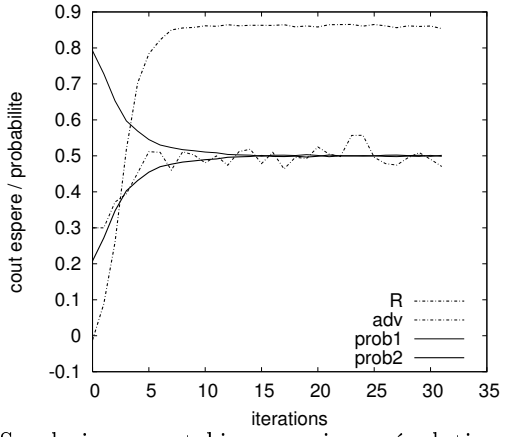
a- Une chaîne de Markov incertaine à 2 boucles successives (sur s_1, puis s_2) avec des probabilités de boucler complémentaires



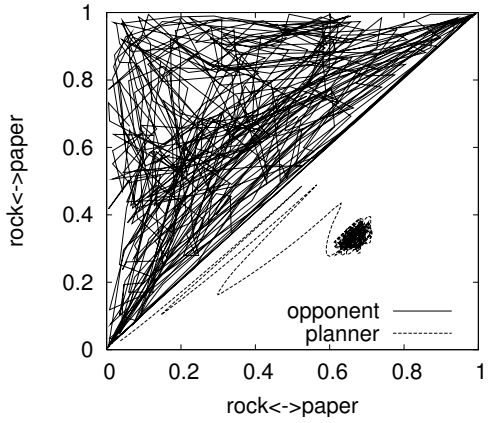
b- Temps moyen avant d'arriver dans l'état but s_3

FIGURE 3.4 – Une chaîne de Markov incertaine avec un compromis optimal

modèle. Une difficulté est alors de régler les pas d'apprentissage de manière à ce que la seconde converge (la première n'ayant pas nécessairement le même besoin de pouvoir se stabiliser). La figure 3.5 illustre le comportement de l'algorithme dit « à deux équipes » (une équipe avec un agent par état, l'autre avec un agent par action paramétrée) sur deux simples jeux de matrice.



a- Sur le jeu « matching pennies », évolution des probabilités des deux actions du premier joueur (Prob1 et Prob2), de la probabilité de la première action du second joueur (adv), et de la récompense moyenne, en fonction de l'itération.



b- Sur le jeu « pierre-papier-ciseaux » (/roshambo), évolution des politiques des deux joueurs vues comme des points dans des simplexes de dimension 3 (triangle supérieur pour l'adversaire, inférieur pour le joueur principal)

FIGURE 3.5 – Deux vues de la dynamique de notre algorithme « à deux équipes » sur deux problèmes différents

Dans ce cadre, la recherche d'un plan robuste est plus difficile. L'algorithme proposé est sans garantie de convergence, et nous n'avons pas proposé d'analyse d'accessibilité.

3.4 Apprendre des règles génériques

En planification classique ou probabiliste, si les problèmes sont généralement décrits en logique relationnelle (en mettant en relation des objets du monde), le moindre ajout ou retrait d'un objet implique de re-lancer une planification. Or, si l'on sait que l'on va être confronté à de nombreux problèmes de la

même famille,[34] on pourrait souhaiter qu'un algorithme sache automatiquement :

— soit concevoir un contrôleur efficace quel que soit le problème ;

— soit optimiser automatiquement le planificateur qui sera ensuite employé.

Nous allons voir dans cette section deux approches visant à apprendre des règles génériques (i) l'une pour choisir des actions, en effectuant un apprentissage incrémental, et (ii) l'autre pour détecter des actions à éviter.

3.4.1 Planification/RL développemental

EWRL'01	(Buffet et Dutech, 2001)
AAMAS'02	(Buffet *et al.*, 2002b)
ECAI'02	(Buffet *et al.*, 2002a)
AAMAS'03	(Buffet *et al.*, 2003a)
EWRL'03	(Buffet et Dutech, 2003)
SAB'04	(Buffet *et al.*, 2004)
RIA a	(Buffet *et al.*, 2005)
RIA b	(Buffet *et al.*, 2006)

Nous abordons ici un travail conduit pendant mon doctorat, mais présenté alors avec une terminologie un peu différente. La terminologie employée ici est plus adaptée au reste de ce document. Les problèmes considérés entrent dans le cadre de la planification probabiliste *partiellement observable*, c'est-à-dire que seuls certains prédicats et certains fluents[35] sont accessibles à l'agent. Par ailleurs, si le modèle de la dynamique est connu, c'est ici uniquement à travers un simulateur.

L'idée de départ est d'apprendre des comportements « de base », chacun défini pour (i) un uplet de types d'objets et (ii) une partie de la fonction de récompense (supposée décomposable additivement). Un tel comportement pourra s'appliquer à un quelconque uplet d'objets de types compatibles, en prenant en entrée toutes les observations (tous les prédicats et fluents) associés à ces objets.

La figure 3.6 montre une situation dans laquelle un agent interagit avec deux types d'objets — tuile et trou — et a deux comportements de base :

— mettre une tuile dans un trou (en l'y poussant), applicable deux fois parce qu'il y a deux tuiles et un trou dans l'environnement, et

— éviter un trou (applicable ici une seule fois).

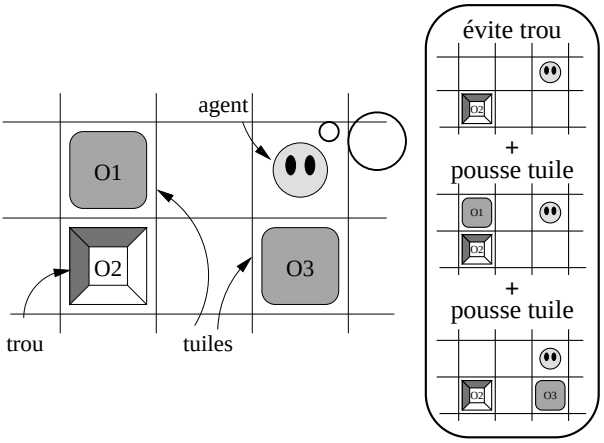


FIGURE 3.6 – Une scène avec quelques objets : la tâche globale est une combinaison de sous-tâches

3.4.1.1 Combiner des comportements de base

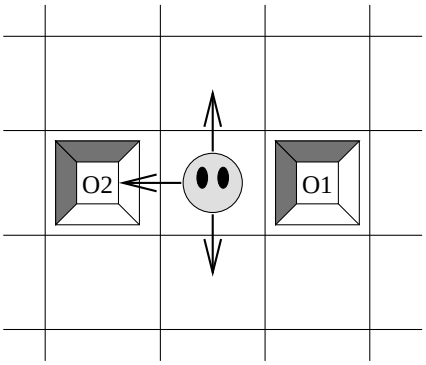
Dans un premier temps, on a appris un à un des comportements de base pré-sélectionnés, à l'aide d'une montée de gradient, ainsi que la fonction de valeurs associée. Le problème posé était alors de

34. La terminologie PDDL parle de « domaine ».

35. Là où les prédicats sont des « fonctions » des objets à valeurs booléennes, les fluents peuvent prendre des valeurs dans d'autres ensembles.

savoir comment combiner ces comportements, problème qui nous fait entrer dans le champ de la *sélection d'actions* (Tyrrell, 1993; Humphrys, 1997). Plus précisément, à chaque instant, chaque comportement a des préférences sur les décisions possibles, et on veut en déduire la décision à prendre.

Une approche classique est de mettre en œuvre un critère permettant de sélectionner un comportement à suivre à un instant donné. Il n'est toutefois pas évident d'établir un tel critère, et le fait de ne suivre qu'un seul comportement peut conduire à une très mauvaise décision qui aurait pu être évitée en tenant compte d'autres comportements, comme illustré par la figure 3.7.



Ici, avec un algorithme winner-take-all, l'agent a une chance sur deux de ne se soucier que du trou O_1, et ainsi d'obéir à une politique le faisant aller dans n'importe quelle direction parmi nord, sud et ouest. Dans ce dernier cas, l'agent va chuter dans l'autre trou.

FIGURE 3.7 – Un agent devant éviter deux trous

Une autre idée est de sélectionner l'action suggérée par le plus grand nombre de comportements. Toutefois, cela ignore (i) l'importance relative des actions (estimable à travers les Q-valeurs) et (ii) les informations fournies par certains comportements qui visent plutôt à *éviter* de mauvais choix, comme c'est le cas avec l'évitement de trou (figure 3.7). Pour cette raison, nous avons pondéré les recommandations des différents comportements à l'aide des *valeurs absolues* des Q-valeurs, ce qui requiert que les récompenses soient nulles par défaut.

Nous avons ainsi proposé et évalué plusieurs méthodes de sélection d'action par combinaison de politiques stochastiques, avec pondération par l'importance estimée à travers les Q-valeurs. Pour chaque comportement, un poids global était appris, les Q-valeurs reflétant (i) l'importance relative d'une action par rapport à une autre au sein d'un comportement, mais pas (ii) l'importance relative d'un comportement par rapport à un autre. À titre comparatif, nous avons aussi évalué une politique stochastique obtenue à l'aide des Q-valeurs. On notera que, si tous les comportements obtenus étaient potentiellement stochastiques, l'ajout d'un bruit permettait de sortir de certaines situations de blocage.

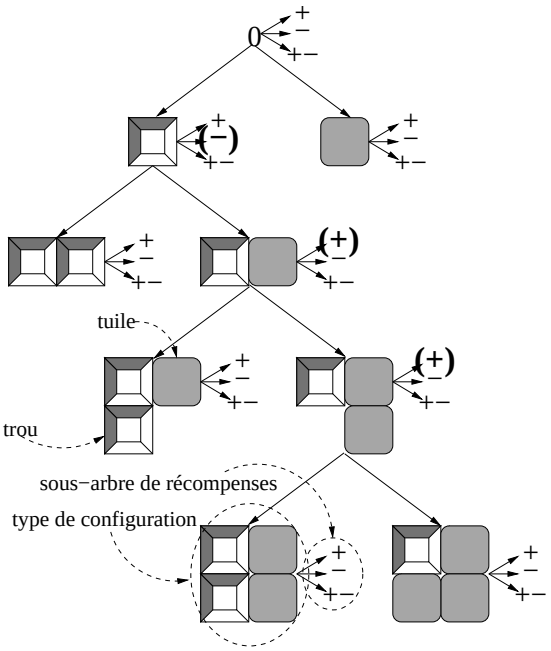
3.4.1.2 Apprendre incrémentalement des comportements de base

Les cas de blocages rencontrés suggèrent qu'une simple combinaison de comportements de base n'est pas une solution suffisante. Les choix effectués ne sont pas les bons (i) dans les situations de blocage elles-mêmes, mais aussi (ii) dans les situations qui conduisent aux blocages. Il est donc tentant de chercher à apprendre des comportements de plus haut niveau (tout en restant les plus simples possibles) visant essentiellement à éviter ces blocages.

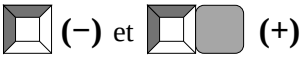
Nous avons ainsi proposé un algorithme qui construit incrémentalement un arbre de comportements de base. Comme le montre la figure 3.8, on considère des uplets de types d'objets de plus en plus complexes,

— en tenant compte à chaque fois des différentes combinaisons de fonctions de récompenses élémentaires possibles (ici notées '+' (pousser une tuile dans un trou) et '-' (tomber dans un trou)) ;
— en ne retenant un nouveau comportement de base appris que si ses résultats sont meilleurs (selon un critère pré-défini) qu'une combinaison des comportements déjà retenus ; et
— en ne testant que des uplets qui étendent d'un type d'objet (ici une tuile ou un trou) un uplet appartenant à un comportement déjà retenu.

Contrairement à ce que suggère la figure 3.8, si le graphe des uplets considérés est orienté et acyclique, ce n'est pas nécessairement un arbre.



En exécutant l'algorithme proposé, aux deux comportements que l'on peut intuitivement juger comme suffisants pour le monde des tuiles :



vient s'ajouter un comportement résolvant l'un des blocages apparus dans nos essais :

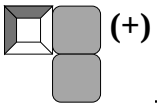


FIGURE 3.8 – L'arbre des comportements testés et (entre parenthèses) ceux qui ont été retenus.

Les expérimentations ont montré que cette approche permet de découvrir un comportement supplémentaire utile dans le scénario considéré. En outre, cette approche permet d'apprendre ce comportement à l'aide des comportements antérieurs, alors qu'un apprentissage direct s'avère difficile (du fait du grand espace de recherche)

Les travaux décrits ici s'apparentent à la planification et à l'apprentissage par renforcement pour MDP relationnels (Dzeroski *et al.*, 2001; Guestrin *et al.*, 2003a; Gretton et Thiébaux, 2004; Fern *et al.*, 2006; Gretton, 2007), cadres dans lesquels on cherche une politique relationnelle optimale pour une famille de problèmes. Les techniques employées relèvent alors souvent de la régression du premier ordre (pour calculer une fonction de valeur généralisée) ou de l'induction d'une politique généralisée à partir d'exemples. Mais, s'il s'agit de pré-calculer (hors-ligne) une fonction de valeur ou une politique généralisée, l'utilisation de ces fonctions ou politiques en ligne sera d'une complexité exponentielle en le nombre d'objets présents à un instant donné.

En outre, dans l'exemple « tuiles/trous » évoqué, il semble qu'on pourra toujours mettre cette approche — apprendre des comportements de base à combiner — en échec en ajoutant un nombre suffisant de trous et de tuiles. Mais cela reste à démontrer, et on peut se demander quelles garanties théoriques cette approche peut fournir, et sous quelles conditions.

3.4.2 [dét] *Learning bad actions* (apprendre les mauvaises actions)

ECAI'14	(Krajnanský *et al.*, 2014a)
JFPDA'14	(Krajnanský *et al.*, 2014b)

3.4.2.1 Apprendre pour planifier

L'apprentissage de règles génériques (comme déjà discuté dans la section précédente) a été développé en bonne partie dans le contexte de l'apprentissage pour la planification (*learning for planning*), promu entre autres par une épreuve dédiée de la compétition internationale de planification. Il s'agit ici plus particulièrement d'employer un certain nombre d'exemples de problèmes (classiques, c'est-à-dire déterministes) d'une même famille (d'un même domaine) pour en tirer des informations facilitant la résolution d'autres problèmes, typiquement plus gros, de la même famille. Cet apprentissage peut prendre des formes

diverses, comme la sélection de l'algorithme ou de l'heuristique qui semble le plus adapté. Une approche classique est de chercher à apprendre des règles génériques, ce qui nous ramène au cadre des MDP relationnels décrit à la fin de la section précédente. Toutefois, l'objectif peut être d'appliquer directement ces règles, considérées comme une politique, ou de s'en servir de guide (d'heuristique) pour un algorithme de planification, comme c'est plus souvent le cas.

3.4.2.2 Identifier les mauvaises actions

Pour de nombreux problèmes, il s'avère difficile d'apprendre des règles identifiant correctement les actions à appliquer dans chaque état. C'est par exemple le cas dans le jeu de Sokoban,[36] un puzzle de transport (NP-difficile) dans lequel, à chaque niveau, un gardien d'entrepôt doit trouver le moyen de pousser des caisses jusqu'à leurs destinations sans les coincer. Face à ce constat, nous — Joerg Hoffmann, Michal Krajňanský, Alan Fern, Valerie Poser, Alvaro Torralba et moi-même — essayons d'apprendre des règles décrivant, à l'inverse, certaines mauvaises actions, c'est-à-dire des actions à *éviter*. L'objectif est d'employer cette information pour restreindre l'espace de recherche d'un algorithme de planification en élaguant des branches, d'où le terme « règles d'élagage » (*pruning rules*). Sont ici plus particulièrement décrits les premiers travaux réalisés par Krajnanský *et al.* (2014a,b).

Cet apprentissage est effectué en résolvant par recherche heuristique un certain nombre de (petits) problèmes de la famille considérée. Plus précisément :

- on construit des arbres contenant *toutes* les solutions optimales, en augmentant chaque état d'un prédicat décrivant le but courant ; s'arrêter à une solution optimale ne permettrait pas de savoir si certaines actions sont réellement mauvaises ou si elles pourraient conduire au but dans le cadre d'une autre solution ;
- les couples ⟨état (augmenté), action⟩ tels que l'état fait partie d'une solution optimale sont alors mis dans les ensembles (i) d'exemples positifs si le couple fait partie d'une solution optimale, et (ii) d'exemples négatifs sinon ;
- ces ensembles d'exemples alimentent un algorithme de programmation logique inductive (ILP) qui apprend des règles d'élagage de la forme

$$r[Y] = \neg a[X] \Leftarrow l_1[X_1] \wedge \cdots \wedge l_n[X_n]$$

où $a[X]$ est un schéma d'action du domaine courant D, les $l_i[X_i]$ sont des littéraux du premier ordre, et $Y = X \cup \bigcup_i X_i$ est l'ensemble des variables apparaissant dans la règle.

La forme des règles et la procédure d'apprentissage ont été choisis pour leur simplicité.

Des variantes de cet apprentissage peuvent être obtenues selon que :

- l'on considère toutes les solutions optimales ou seulement une ;
- l'on ne garde dans les exemples que les *actions utiles* (*helpful actions*) identifiées par l'heuristique de FF ou toutes les actions ;
- l'on autorise dans les règles les littéraux négatifs (qui permettent une plus grande expressivité) ou pas (pour réduire l'espace de recherche) ;
- l'on autorise dans les règles les contraintes d'inégalité — qui permettent d'interdire que deux variables désignent le même objet — ou pas.

3.4.2.3 Utilisation des règles

Le principe d'utilisation des règles d'élagage apprises est simplement, lors d'une recherche heuristique, de ne développer, dans un état donné, que les actions non élaguées par les règles. Toutefois, parce que la recherche des règles applicables peut devenir très coûteuse, on doit limiter la taille des règles apprises à $L = 6$ littéraux (ce qui limite aussi l'espace de recherche de règles).

En pratique, d'un domaine à l'autre, les meilleurs résultats sont obtenus avec différents paramétrages de notre algorithme d'apprentissage. Pour cette raison, nous avons décidé d'employer :

- un portefeuille de solveurs, chacun ayant droit à un certain temps pour s'exécuter, et
- un algorithme de réglage automatique des paramètres, SMAC (Hutter *et al.*, 2009), éventuellement autorisé à sélectionner les règles apprises.

36. `https://fr.wikipedia.org/wiki/Sokoban`

Les résultats expérimentaux obtenus avec ces approches, en particulier SMAC, sont très encourageants.

3.5 Projets et applications

Nous décrivons ici des travaux principalement liés à des applications de modèles tels que les MDP. S'ils ont une visée plus directement pratique, ces travaux n'excluent toutefois pas des questions et contributions théoriques.

Dans chaque cas sont d'abord mentionnés les publications liées (ou consécutives) au projet, et les principaux collaborateurs impliquées.

3.5.1 DPOLP / planification d'opérations

journaux	IJAIT	(Buffet, 2007a)
	AIJ	(Buffet et Aberdeen, 2009)
confs. int.	IJCAI'05	(Buffet et Aberdeen, 2005a)
	ICTAI'05	(Buffet, 2005a)
	IPC'06	(Buffet et Aberdeen, 2006b)
	ICAPS'07	(Buffet et Aberdeen, 2007a)
	ICAPS'07	(Aberdeen et Buffet, 2007)
	AISTATS'07	(Aberdeen *et al.*, 2007)
workshops	PLMUDW'06	(Buffet et Aberdeen, 2006c)
	RLNSE'05	(Buffet et Aberdeen, 2005c)
	PLAPUDD'05	(Buffet, 2005b)
	PLAPUDD'05	(Aberdeen et Buffet, 2005)
	IFORS'05	(Naguleswaran *et al.*, 2005)
	JFPDA'07	(Buffet et Aberdeen, 2007b)
	CAp'06	(Buffet et Aberdeen, 2006a)
	CAp'05	(Buffet et Aberdeen, 2005b)
rapports	NICTA	(Buffet et Aberdeen, 2005d)
	NICTA	(Buffet, 2004)
	NICTA	(Buffet et Aberdeen, 2004)

Collaborateurs principaux :

Nom	Affiliation	DPOLP	Brazil	STaR
Douglas Aberdeen	NICTA	✓	✓	✓
Sylvie Thiébaux	ANU	✓	✓	✓
Owen Thomas	NICTA	✓	✓	✓
Silvia Richter	NICTA			✓
Jin Yu	NICTA			✓
Lin Zhang	DSTO	✓		
Lang White	U. of Adelaide	✓		
Sanjeev Naguleswaran	U. of Adelaide	✓		
Sarah Hickmott	U. of Adelaide	✓		

Pendant mon post-doctorat au sein du NICTA à Canberra (Australie), j'ai principalement travaillé dans le cadre de DPOLP (*Dynamic Planning, Optimisation and Learning Project*), un projet en collaboration avec le DSTO [37], l'université nationale australienne (ANU) et l'université d'Adelaide (UoA) dont l'objectif était de développer des outils de planification d'opérations militaires. Le problème soulevé était de pouvoir planifier automatiquement des tâches

— nombreuses et variées, allant d'envois de troupes à des tâches logistiques (par exemple pour la gestion de l'alimentation ou de l'énergie) en passant par des soutiens aériens ;
— durables (/temporelles) et concurrentes, puisque chaque tâche peut avoir sa durée propre et que des tâches peuvent être initiées alors que d'autres sont en cours ; et
— incertaines, une tâche pouvant réussir ou échouer (à supposer que seuls ces deux résultats soient attendus).

Il s'agit donc d'un problème demandant des descriptions très expressives. À mon arrivée, j'ai pu participé à la formalisation en cours de ce problème, laquelle a requis un travail de nettoyage, ainsi qu'à sa résolution à l'aide d'un planificateur (`MOP`, *Military Operation Planning*) reposant sur LRTDP (voir section 2.2.3.1).

Nos travaux, principalement avec Douglas Aberdeen, Owen Thomas et Sylvie thiébaux, se sont alors portés vers (i) la résolution de tels problèmes par descente de gradient factorisée et (ii) la planification robuste (aux incertitudes sur le modèle), comme décrit en section 3.3. Mais nous avons aussi naturellement étendu notre champ d'application en abordant des scénarios, tels que les chantiers de construction, dans lesquels l'incertitude est principalement liée à la durée des tâches. Cela nous a conduit à développer un

37. *Defence Science and Technology Organisation*

solveur, `Brazil`, dans lequel une durée peut être définie selon une distribution de probabilité. La même bibliothèque d'apprentissage par renforcement, `libpg`, aussi réalisée par nos soins, servait de moteur d'apprentissage/résolution aussi bien pour les problèmes d'opération militaire que pour la compétition de planification (IPC 2006, remportée par `fpg`) et pour `Brazil`.

Dans le même cadre de travail, j'ai aussi abordé avec Sylvie Thiébaux et des collègues de l'université d'Adélaide — Lang White, Sanjeev Naguleswaran et Sarah Hickmott — les sujets (i) du lien entre planification et dépliement de réseaux de Petri, et (ii) de la planification MDP par calcul quantique (Naguleswaran *et al.*, 2005).

3.5.2 SuperCom

IJCAI'07	(Kelareva *et al.*, 2007)

Collaborateurs principaux :

Nom	Affiliation
Sylvie Thiébaux	ANU
Elena Kelareva	Univ. of Melbourne
Jinbo Huang	NICTA

[Ce projet a déjà été décrit en section 3.2.3.]

Pour rappel, il visait à proposer des méthodes de supervision (diagnostic et planification) de *systèmes composites*, c'est-à-dire formés de multiples composants organisés en réseaux, tels que des réseaux de capteurs ou des services web. Je n'y ai participé qu'à ses débuts.

3.5.3 AGATA / planification, diagnostic et simulation avec un même modèle

Rapport de recherche	(Buffet, 2007c)
Rapport de recherche	(Buffet, 2007b)

Collaborateurs principaux :

Nom	Affiliation
Félix Ingrand	LAAS - CNRS
Gérard Verfaillie	ONERA
Marie-Claire Charmeau	CNES
Elodie Chanthery	LAAS - INSA
Louise Travé-Massuyès	LAAS - CNRS
Michel Lemaître	ONERA

Mon post-doctorat au LAAS (CNRS) à Toulouse était financé par le projet AGATA (Architecture Générique pour l'Autonomie, Tests et Applications), collaboration avec le CNES et l'ONERA. Celui-ci portait sur le contrôle de systèmes embarqués autonomes. Un scénario type est la gestion d'un satellite d'observation de la Terre en interaction avec son centre de contrôle, plus précisément le satellite Frankie dont l'objet est de détecter et surveiller des points chauds (feux, volcans, ...) (Lemai-Chenevier et Charmeau, 2006; Lemai-Chenevier *et al.*, 2006), et tel qu'illustré par la figure 3.9. Ce satellite peut identifier de nouveaux points chauds, et effectuer des prises de vue en fonction des demandes qui lui sont faites, de son énergie disponible et de sa mémoire disponible.

Mon travail, pour partie inspiré du projet SuperCom, a consisté à proposer une approche permettant de modéliser de manière modulaire un tel système, de sorte à ce que le même modèle permette ensuite d'effectuer des tâches de simulation, de diagnostic ou de planification. J'ai ainsi réalisé un démonstrateur dans lequel

— le satellite Frankie est décrit dans un ensemble de fichiers (au format dédié « AGA »), chaque fichier représentant un composant, éventuellement construit sur la base de sous-composants ;
— le programme « TA » permet de simuler l'exécution d'un scénario (une série d'événements pré-programmés) sur un système donné (tel que le satellite Frankie).

3.5.4 DOPEC / planification pour la collecte d'informations

IFAC'11	(Godichaud *et al.*, 2011a)

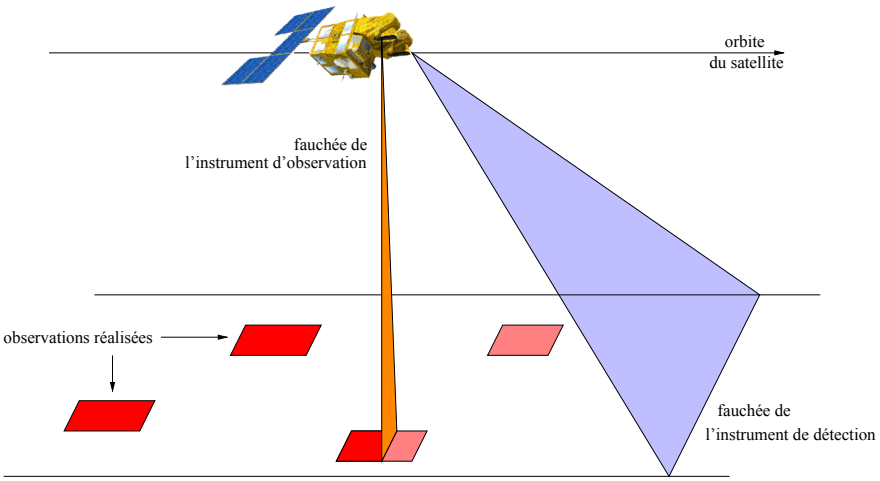


FIGURE 3.9 – Représentation de Frankie, satellite d'observation de la Terre capable de détecter des points chauds (avec un capteur à large bande) et de les photographier (avec un instrument d'observation plus précis) — (image tirée du mémoire de thèse de Beaumet (2008))

Collaborateurs principaux :

Nom	Affiliation
Marc Contat	EADS
Thierry Oms	EADS
Elodie Chanthery	LAAS - INSA
Matthieu Godichaud	LAAS - CNRS

J'ai enfin participé à un autre projet dont la thématique est proche à la fois de DPOLP et d'AGATA. Ce projet, dénommé DOPEC et effectué en collaboration avec EADS et le LAAS (CNRS et INSA Toulouse), visait à la réalisation d'un *Démonstrateur pour l'OPtimisation de l'Emploi des systèmes-Capteurs*. Pour être plus précis, nous nous intéressions à la planification de tâches d'acquisition d'information dans un cadre militaire. Des requêtes d'information étant définies, et chaque requête pouvant être satisfaite d'une ou plusieurs manières, par un ou plusieurs types de systèmes de capteurs (humains, appareils fixes (radars), volants (UAV) ou roulants, ...), le problème était d'affecter au mieux les systèmes de capteurs disponibles à ces requêtes, et d'ordonnancer les observations à effectuer. Nous avons proposé une formalisation de ce problème de collecte d'informations — qui est ici vu comme concurrent, temporel, et déterministe — ainsi qu'un algorithme pour le résoudre, dont la phase de planification repose sur A* (Godichaud *et al.*, 2011a).

3.5.5 BARQ

journaux	JAIR (16)	(Dibangoye *et al.*, 2016)
confs int.	IJCAI'13	(Dibangoye *et al.*, 2013a)
	AAMAS'14	(Dibangoye *et al.*, 2014b)
	ECML'14	(Dibangoye *et al.*, 2014a)
	IJCAI'15	(Dibangoye *et al.*, 2015a)
	IJCAI'15	(Dibangoye *et al.*, 2015b)
workshops	JFPDA'13	(Dibangoye *et al.*, 2013b)
	JFPDA'13	(Tagorti *et al.*, 2013b)
	HSDIP'13	(Tagorti *et al.*, 2013a)
	JFPDA'14	(Dibangoye *et al.*, 2014c)
rapports	INRIA	(Dibangoye *et al.*, 2014d)

Collaborateurs principaux :

Nom	Affiliation
Jörg Hoffmann	INRIA / U. Sarrebrück
Bruno Scherrer	INRIA
Boris Lesner	INRIA
Manel Tagorti	INRIA
Jilles Dibangoye	INRIA

J'ai aussi participé aux travaux menés dans le cadre d'une chaire d'excellence ANR portée initialement par Joerg Hoffmann et intitulée BARQ pour « Abstraction, Raffinement, et Bornes de Qualité ». Son objectif était d'étudier les méthodes d'abstraction pour calculer des minorants — en particulier dans des problèmes de planification (déterministe ou probabiliste) — et les raffinements d'abstraction pour améliorer incrémentalement ces minorants.

Les travaux conduits avec Manel Tagorti, Bruno Scherrer et Joerg Hoffmann sur l'agrégation d'états se situaient dans ce cadre (section 3.2.1), ainsi qu'une partie des travaux effectués avec Jilles Dibangoye sur la résolution de Dec-POMDP (qui sont présentés ultérieurement en section 4.3.3).

3.6 Discussion

3.6.1 Sur les aspects théoriques

Comme on vient de le voir, mes travaux sur les MDP (et la planification déterministe) ont porté sur des sujets assez divers. Je reviens ici sur quelques points qui me semblent plus particulièrement importants.

Concernant les fonctions heuristiques, lesquelles sont d'usage bien plus courant dans le cadre déterministe que dans le cadre stochastique, nous avons montré : (i) que de bonnes propriétés importantes dans le cas déterministe ne se transposaient pas dans le cas stochastique (raffiner une agrégation n'améliore pas nécessairement l'approximation), et (ii) que maximiser la probabilité d'atteindre un état but pouvait se faire de manière assez efficace malgré les heuristiques peu informatives à disposition. L'utilisation d'heuristiques et d'abstractions n'est donc pas efficace que dans le cas déterministe, mais les incertitudes de la dynamique créent des difficultés et amènent à chercher des solutions spécifiques.

Concernant la recherche directe d'un contrôleur, nous avons d'abord montré que c'était une approche viable pour la planification probabiliste en développant un outil relativement simple et robuste. Cette approche, plutôt liée à l'apprentissage et à l'optimisation, est assez différente des approches plus usuelles reposant par exemple sur des recherches heuristiques et sur le principe d'optimalité de Bellman. Si elle ne bénéficie pas des mêmes garanties théoriques de convergence, elle généralise naturellement ce qui a été appris d'une situation à une autre. En outre, elle permet assez facilement d'intégrer des variables continus, des durées d'actions incertaines, ou des contraintes (de la connaissance experte) sur la forme attendue du contrôleur. Cette approche permet aussi de traiter plus finement les problèmes de planification robuste en ne faisant pas l'hypothèse que le modèle d'une action dépend de l'état courant.

Nous avons aussi employé de l'apprentissage pour obtenir des règles de comportement qui fonctionne dans des environnements de taille variable (par exemple des nombres d'objets variables), que ce soient des règles de recommandation (il faut faire l'action a) ou d'interdiction (il ne faut pas faire l'action a). Mais de telles méthodes restent difficiles à mettre en œuvre. On trouve en effet facilement des scénarios dans lesquels les règles apprises peuvent conduire à de mauvaises décisions. Une question intéressante est de savoir si, dans certains cas, on peut vérifier (à coût raisonnable) la validité d'une règle apprise par induction.

3.6.2 Remarques sur DPOLP, SuperCom, AGATA et DOPEC

DPOLP, Supercom, AGATA et DOPEC étaient quatre projets particulièrement ambitieux mettant en œuvre des systèmes complexes avec de nombreux composants et donc de nombreuses actions possibles. Le tableau 3.4 présente une synthèse (un peu simplificatrice) des caractéristiques de ces projets. Il aurait idéalement fallut dans ces quatre cas tenir compte d'incertitudes (fait uniquement dans DPOLP) en plus de la durée et de la concurrence des actions. Par ailleurs, on imagine que, dans bien des scénarios de ces projets, il est plutôt abusif de supposer que l'état est complètement observable, une hypothèse pourtant faite dans chacun de ces projets.

Dans les quatre cas, un gros travail a dû être réalisé en même temps sur la modélisation (+formalisation) et sur la résolution du problème, illustrant le fait que ces deux opérations sont intimement liées. Il faut savoir trouver la bonne façon de décrire le problème pour ne pas faire trop d'hypothèses simplificatrices, mais que l'on puisse en même temps proposer un algorithme de résolution efficace.

TABLE 3.4 – Principales caractéristiques des quatre projets DPOLP, SuperCom, AGATA et DOPEC (tels qu'ils ont été abrodés)

projet	cadre	incertitude	temps/conc.	ressources
DPOLP	militaire+industriel	✓	✓	✓
SuperCom	industriel	✗	✓	✗
AGATA	aérospatial	✗	✓	✓
DOPEC	militaire	✗	✓	✓

En outre, chaque fois qu'experts d'un domaine et chercheurs ont collaboré à une telle modélisation, un effort a été nécessaire de part et d'autre pour se comprendre, préciser progressivement les choses, et parfois revenir en arrière en cas de méprises. Sans entrer dans le détail, le même constat a été effectué dans les projets présentés dans le chapitre suivant (par exemple autour du pen-testing en sécurité informatique et sur la protection d'espèces).

Chapitre 4

Mes travaux sur les *POMDP (et le BRL)

Sommaire

4.1 Introduction

Ce chapitre présente un deuxième ensemble de contributions, contributions qui portent sur les POMDP et leurs extensions (telles que les Dec-POMDP), lesquels modèles seront désignés dans leur ensemble par l'acronyme *POMDP. Comme dans le chapitre précédent, il est principalement question de planification

d'action, le modèle étant connu. Mais il sera aussi question d'apprentissage par renforcement bayésien avec modèle, cadre qui s'apparente à celui de la résolution de POMDP si l'on considère le modèle méconnu comme une variable d'état cachée.

Dans le chapitre précédent, l'optimisation d'un contrôleur par montée de gradient a été employée pour la planification probabiliste et pour l'apprentissage de règles comportementales (section 3.3). Ce dernier emploi était illustré par un problème jouet (dans lequel un agent déplace des tuiles) qui tient en fait plus du POMDP que du MDP. Nous allons voir en section 4.2 deux autres travaux concernant cette même famille d'approches : l'une abordant la conception d'agents collaboratifs, l'autre comparant le cadre des POMDP à celui des PSR. La section 4.3 revient ensuite à des approches plus usuelles pour montrer comment la propriété PWLC [38] est exploitée dans trois cadres étendant les POMDP de diverses manières : (i) en supposant certaines variables d'état visibles, (ii) pour les problèmes de collecte d'information, et (iii) pour le contrôle décentralisé. Puis, la section 4.4 décrit des travaux sur l'apprentissage par renforcement bayésien dans lesquels, malgré la parenté avec la résolution des POMDP, on proposera une approche d'une toute autre forme (optimiste), approche accompagnée de bonnes propriétés théoriques de convergence. Pour terminer, quelques projets et applications reposant sur la résolution de POMDP et l'apprentissage par renforcement sont présentés en section 4.5

4.2 Montée de gradient et POMDP

Aparté : Les travaux présentés en section 3.3 mettent aussi en jeu une montée de gradient et sont aussi applicables à des problèmes partiellement observables. Il semble toutefois plus naturel de les classer avec les travaux sur les MDP du chapitre précédent.

4.2.1 maRL incrémental

Agents'01	(Buffet *et al.*, 2001)
IJCAI'01	(Dutech *et al.*, 2001)
AAMAS'07	(Buffet *et al.*, 2007)

Pendant mon stage de DEA et le début de ma thèse, j'ai travaillé sur l'utilisation de l'apprentissage par renforcement pour la conception de systèmes multi-agents collaboratifs. Il n'existait alors pas d'algorithmes permettant de concevoir (/planifier) des politiques individuelles pour un groupe d'agents participant à une tâche commune, sauf dans le cas où chaque agent a une observation complète de l'état (Boutilier, 1996). Vue la nature distribuée du problème, nous avons naturellement considéré l'emploi de l'apprentissage par renforcement, tous les agents essayant simultanément d'optimiser leurs comportements. Ne sont alors plus respectées les propriétés qui permettent la convergence

— non seulement du Q-learning ou de SARSA parce que l'observabilité est partielle,
— mais aussi des montées de gradient parce que le système, vu d'un agent, n'est pas markovien puisqu'il contient d'autres agents dont le comportement évolue aussi.

Ne disposant pas d'algorithme adapté à des agents évoluant de concert, nous avons fait le choix d'employer, malgré tout, une montée de gradient en ligne. Des expérimentations ont été conduites sur deux problèmes situés dans des grilles carrées toriques :

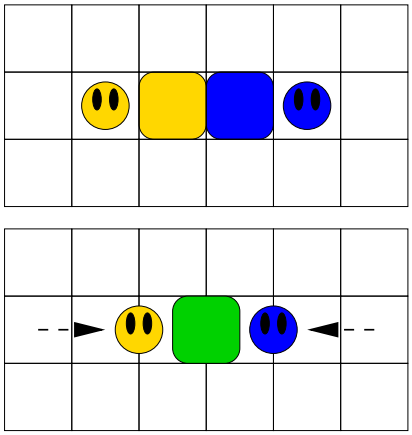
fusion de blocs (fig. 4.1), dans lequel des agents doivent se coordonner pour pousser des paires de blocs (une verte, une bleue) l'une contre l'autre pour les faire disparaître ; et

prédateurs-proie (fig. 4.2), dans lequel quatre prédateurs doivent apprendre à entourer une proie se déplaçant aléatoirement, et ce, dans une grille torique.

Dans les deux cas les observations sont partielles parce qu'un agent ne perçoit d'un objet ou autre agent que sa direction (nord/sud/est/ouest) et sa distance (proche/éloignée). Par ailleurs, dans le cas de la fusion de blocs, ne sont visibles que l'agent de couleur opposé le plus proche et la paire de blocs de couleurs opposées les plus proches (de sorte que le nombre d'observations possibles reste le même quand le nombre d'agents ou de blocs évolue).

Les premiers résultats ont montré que l'apprentissage était difficile et, en particulier, que plus le nombre d'agents et objets présents était grand, plus il était difficile d'apprendre à agir au mieux. Cela nous a

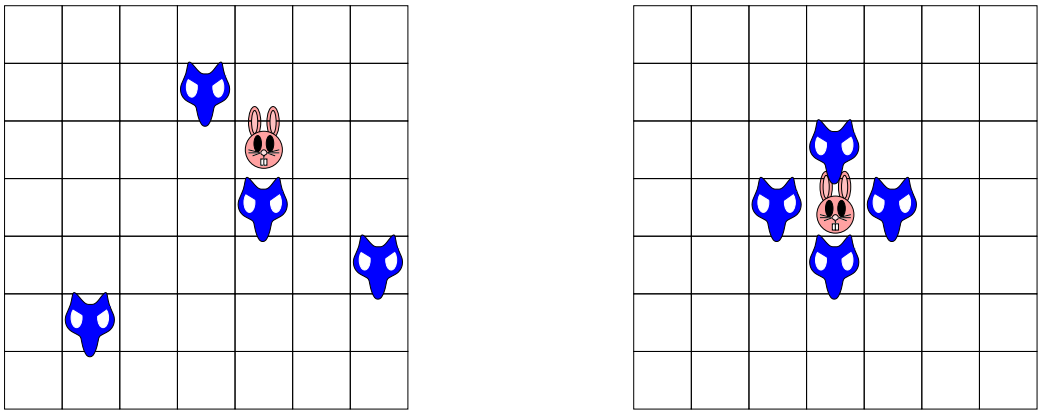
38. Linéarité par morceaux et convexité de la fonction de valeur dans les POMDP.



Ici, agent et bloc de gauche sont en jaune, et bloc et agent de droite sont en bleu (mais les couleurs des blocs pourraient être échangées).

On a représenté en vert les blocs fusionnant avant de disparaître. Les **deux** agents doivent pousser les blocs pour que l'opération soit effectuée. Il ne suffit pas que l'un pousse et l'autre bloque.

FIGURE 4.1 – Un exemple de fusion de blocs



a- situation quelconque b- situation recherchée

FIGURE 4.2 – Deux exemples de situations possibles du problème prédateurs-proie

conduit à proposer d'effectuer un apprentissage progressif, non seulement en augmentant progressivement le nombre d'agents et objets dans l'environnement (pour la fusion de blocs), mais aussi en proposant aux agents des situations de plus en plus éloignées des possibilités de récompenses.

Cette approche d'aide à l'apprentissage, comparable en certains points à celle d'Asada *et al.* (1996), a permis une nette amélioration des résultats sur le problème de la fusion de blocs, avec à la fois une accélération de l'apprentissage, mais aussi une capacité à atteindre de meilleurs niveaux de performance. Ses effets ont par contre été contre-productifs sur le problème prédateurs-proie, les prédateurs perdant du temps à désapprendre certaines règles comportementales acquises.

Cet apprentissage progressif pourrait être amélioré en automatisant la progression de la complexité des tâches (La progression en nombre d'agents/objets est simple.). Une façon de faire pourrait être de résoudre la version mono-contrôleur et totalement observable du problème, pour ensuite mesurer la distance des états à de possibles récompenses, voire identifier des états dans lesquelles le choix d'action est critique.

Une limitation est que les agents doivent ici avoir un ensemble d'observations indépendant du nombre d'agents et d'objets dans leur environnement. C'est pour pallier ce défaut que nous avons abordé l'apprentissage de comportements « combinables » décrit en section 3.4.1.

4.2.2 Montée de gradient : POMDP vs PSR

AISTATS'07 (Aberdeen *et al.*, 2007)

Nous avons déjà vu que les montées de gradient peuvent être employées pour optimiser des contrôleurs

pour POMDP ne prenant que des observations en entrée. Ne prendre une décision qu'en fonction de l'observation courante n'est évidemment pas optimal, et peut être catastrophique (Singh *et al.*, 1994). Aberdeen et Baxter (2002) ont montré comment on pouvait étendre cette approche en ajoutant au contrôleur une mémoire interne, laquelle doit être dimensionnée de manière appropriée. L'apprentissage a alors la tâche délicate de trouver comment employer cette mémoire pour représenter une information utile présente dans l'historique des actions et observations passées.

Dans ce travail réalisé avec Douglas Aberdeen et Owen Thomas, nous avons décidé d'utiliser en entrée des contrôleurs des statistiques suffisantes. La statistique suffisante, dans le cas d'un POMDP, est usuellement l'état de croyance, une distribution de probabilité sur les états possibles. Mais un autre formalisme, celui des *représentations prédictives d'états* (PSR) (Littman et R. Sutton, 2001), permet de décrire les mêmes problèmes, avec l'avantage de ne pas avoir à connaître le nombre d'états sous-jacents. Dans ce cas, la statistique suffisante est le *vecteur de prédiction*, c'est-à-dire un vecteur (pas nécessairement une distribution) de probabilités sur des *tests*, tests qui correspondent à des séquences d'actions-observations futures possibles.

Pour évaluer les différentes approches possibles, nous avons comparé :

— des contrôleurs réactifs simples : l'un aléatoire, les autres appris (avec une montée de gradient simple), prenant en entrée des observations plus ou moins limitées : d'aucune observation à une mémoire de quelques pas de temps ;
— des contrôleurs pour les statistiques suffisantes des PSR comme des POMDP optimisés avec la montée de gradient NAC (Peters *et al.*, 2005), laquelle incorpore un critique et fait meilleur usage des échantillons obtenus qu'une montée simple ;
— les mêmes contrôleurs optimisés avec le même algorithme, mais en estimant le modèle simultanément : POMDP à l'aide d'un algorithme d'espérance-maximisation (EM) de type Baum-Welch (Chrisman, 1992), et PSR à l'aide d'un algorithme de McCracken et Bowling (2006).

Les expérimentations conduites sur six bancs d'essais ont d'abord confirmé l'intérêt de prendre des décisions sur la base de statistiques suffisantes plutôt que d'informations partielles. Elles ont aussi montré que, si l'apprentissage en-ligne des modèles fonctionne, l'optimisation des contrôleurs est plus efficace quand le modèle est déjà connu, avec une plus grande stabilité des résultats. Enfin, elles suggèrent deux remarques nouvelles concernant la comparaison entre POMDP et PSR : (1) apprendre un POMDP, y compris le nombre d'états, n'est pas nécessairement plus difficile qu'apprendre un PSR, et (2) l'optimisation d'un contrôleur parait plus difficile dans le cadre PSR que dans le cadre POMDP.

Concernant cette dernière remarque, nous avons montré qu'un contrôleur reposant sur un perceptron suivi d'un soft-max pouvait, en prenant un état de croyance en entrée, au moins reproduire l'heuristique votante de Simmons et Koenig (1995). [39] Mais d'autres résultats suggèrent des similitudes quant à l'usage qui peut être fait d'un vecteur de prédiction ou d'un état de croyance. Il a en effet été montré que, dans un PSR, la fonction de valeur était PWLC (linéaire par morceaux et convexe) dans l'espace des vecteurs de prédiction (James *et al.*, 2004), ce qui permet d'appliquer les mêmes algorithmes que pour les POMDP (mais avec plus de difficultés à faire de l'élagage parce qu'on ne sait pas délimiter cet espace comme le simplexe des états de croyance).

4.3 Exploitation de la propriété PWLC

De nombreux solveurs de POMDP exploitent le fait que la fonction de valeur optimale est linéaire par morceaux est convexe (PWLC) (*cf* section 2.3.1.2). Nous allons voir ici des travaux reprenant cette propriété dans des variantes de POMDP.

4.3.1 MOMDP

ICTAI'10 (Araya-López *et al.*, 2010b)

39. Cette heuristique donne à chaque état un poids correspondant à sa probabilité, et le fait voter pour une action optimale dans le MDP correspondant. L'action ayant le plus de votes est choisie.

Parce qu'elle se situait dans le cadre du projet COMAC (*cf* section 4.5.3) et du diagnostique actif (le choix d'actions épistémiques [40] à effectuer pour déterminer l'état d'un objet), la thèse de Mauricio Araya-López s'est naturellement orientée vers la planification dans les POMDP. En creusant cette problématique, nous (avec ses encadrants Vincent Thomas et François Charpillet) nous avons découvert une propriété courante qui n'était pas exploitée par les planificateurs. La plupart des algorithmes considèrent en effet que l'état du système est complètement caché, et donc que l'état de croyance est un point à l'intérieur du simplexe des distributions sur les états.

Il est toutefois courant que, parmi plusieurs variables décrivant l'état du système à contrôler, certaines soient complètement observables. On peut décrire ces *MDP à observabilité mixte* (MOMDP) en se ramenant à deux variables (discrètes) s_v et s_h correspondant respectivement aux parties visibles et cachées (*hidden*) de l'état, comme illustré figure 4.3. Les états de croyances possibles se situent alors dans $|S_v|$ simplexes chacun de dimension $|S_h|$, et il est possible de représenter la fonction de valeur comme $|S_v|$ enveloppes d'hyperplans de dimension réduite par rapport au problème initial (voir figure 4.4). Les algorithmes de l'état de l'art exploitant la propriété PWLC peuvent en conséquence être adaptés à cette structure en réduisant considérablement leur coût de calcul.

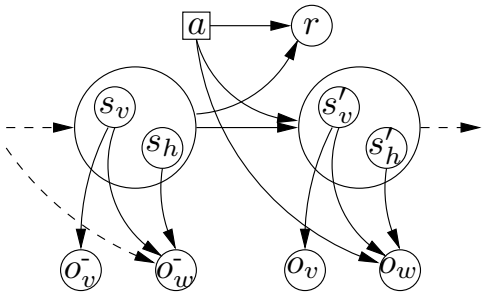


FIGURE 4.3 – Un MOMDP vu comme un diagramme d'influence dynamique

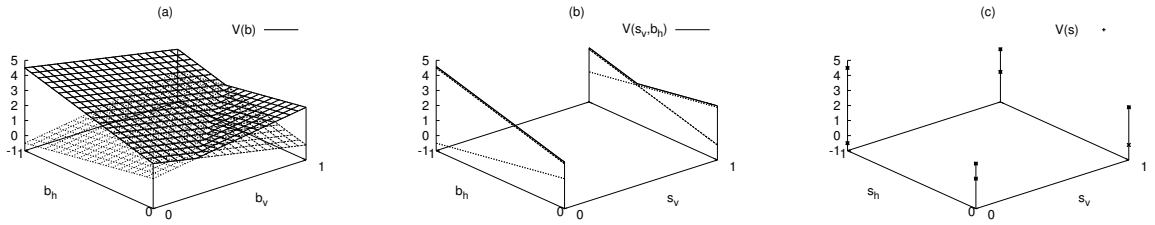


FIGURE 4.4 – La même fonction de valeur vue dans un POMDP, un MOMDP, et un MDP. — Note : Cette représentation est pratique mais abusive : une fonction de valeur sur 4 états requiert une représentation en 4 dimensions.

Après avoir discuté du gain en complexité obtenu par cette approche, nous avons comparé l'algorithme *Incremental Pruning* (IP) expérimentalement avec et sans la prise en compte de cette structure, avec des gains importants autant en temps de calcul qu'en mémoire consommée. En parallèle, Kurniawati *et al.* (2008) ont développé SARSOP, un algorithme à base de points qui fait maintenant partie de l'état de l'art et qui exploite avec le même succès ce même moyen de réduction de la dimensionalité.

4.3.2 ρ-POMDP

NIPS'10 (Araya-López *et al.*, 2010a)

40. Une action épistémique vise essentiellement à recueillir des informations, par opposition aux actions ontiques.

Toujours dans le contexte de la thèse de Mauricio Araya-López, les problèmes de diagnostique actif, dont ceux soulevés par le projet COMAC, nous ont amenés à nous interroger sur les problèmes que le formalisme des POMDP permet de traiter. En effet, un certain nombre de scénarios envisagés amenaient à vouloir récompenser non pas l'arrivée dans certains états ou la réalisation de certaines actions, comme c'est typiquement le cas dans les MDP et POMDP, mais plutôt des gains d'information dans l'état de croyance. À titre d'exemple, la performance du contrôleur peut mesurer :

— la quantité d'information moyenne par pas de temps si on cherche à surveiller continuellement un système, ou

— la quantité d'information obtenue après avoir dépensé un certain « budget » d'action.

Nous avons ainsi été amenés à proposer un nouveau formalisme, celui des ρ-POMDP, dans lequel la fonction de récompense, notée ρ, dépend de l'état de croyance.[41] Le changement de fonction de récompense pose la question du maintien de la propriété PWLC et des moyens de résoudre ces nouveaux problèmes. Quand l'objectif est d'acquérir des informations, ce qui est courant, cette fonction est généralement convexe, croissant quand on se rapproche des sommets du simplexe. Mais, pour pouvoir nous ramener aux outils reposant sur la propriété PWLC, nous avons dû démontrer que :

1. si ρ est convexe dans l'espace des états de croyance, alors la fonction de valeur optimale est aussi convexe (parce que les opérateurs apparaissant dans l'opérateur d'optimalité de Bellman préservent la convexité) ; et
2. si ρ et convexe est linéaire par morceaux, il en est de même pour la fonction de valeur optimale (par le même raisonnement).

Or il est souvent naturel de vouloir définir ρ à partir d'une mesure d'information telle que l'entropie ou la divergence de Kullback-Leibler entre l'état de croyance courant et une distribution représentant l'ignorance. Une fonction $f : \mathcal{D} \mapsto \mathbb{R}$ avec $\mathcal{D} \subset \mathbb{R}^n$ comme celles-ci a malheureusement la particularité de ne pas être Lipschitzienne (grossièrement, « de dérivée bornée »), mais α-hölderienne pour $\alpha \in (0, 1]$, c'est-à-dire qu'elle satisfait :

$$\exists K_\alpha > 0, \text{ tel que } |f(\vec{x}) - f(\vec{y})| \leq K_\alpha \|\vec{x} - \vec{y}\|_1^\alpha.$$

En évitant de trop s'approcher des bords du domaine $\mathcal{D}$ (du simplexe dans notre cas), nous avons pu montrer :

— comment approcher une fonction α-hölderienne convexe par une enveloppe d'hyperplans ; et, de là,

— comment approcher (en contrôlant l'erreur faite) la fonction de valeur optimale par une enveloppe d'hyperplans en partant d'une approximation assez fine (c'est-à-dire reposant sur un ensemble de points couvrant le simplexe de manière assez dense) de la fonction de récompense α-hölderienne ρ.

Nous avons pu mettre en œuvre l'approche obtenue dans différents algorithmes à base de points de l'état de l'art pour l'étudier expérimentalement sur différents bancs d'essais, dont par exemple une adaptation du problème *Rock Sampling* (Smith et Simmons, 2004), renommée *Rock Diagnosis*, et dans laquelle un robot doit identifier au mieux des roches sur un terrain martien (voir figure 4.5). Les résultats obtenus montrent l'intérêt de cette approche en comparaison de l'utilisation (courante dans certains domaines) d'une heuristique myope consistant, à chaque prise de décision, à maximiser le gain en information à un pas de temps. Par ailleurs, nous avons pu observer la difficulté croissante à bien approcher la fonction de récompense (et donc la fonction de valeur optimale) quand la dimensionalité augmente. Une fonction de récompense PWLC simple peut alors s'avérer plus pratique d'emploi.

4.3.3 Dec-POMDP & co.

IJCAI'13	(Dibangoye *et al.*, 2013a)
AAMAS'14	(Dibangoye *et al.*, 2014b)
IJCAI'15	(Dibangoye *et al.*, 2015a)
JAIR	(Dibangoye *et al.*, 2016)

41. Cette fonction peut aussi éventuellement dépendre de l'état et de l'action.

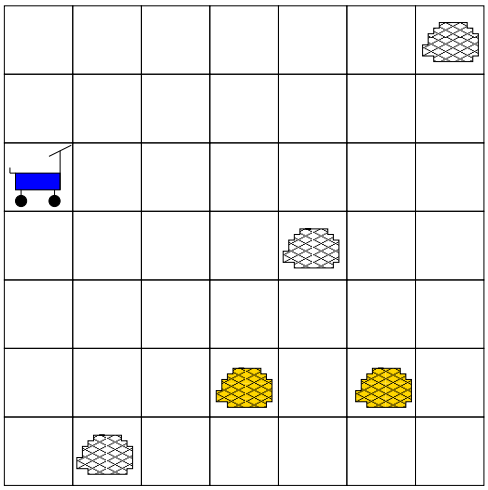


FIGURE 4.5 – Représentation graphique du problème *Rock Diagnosis* avec 5 roches et une grille de taille 7×7. Les roches jaunes sont les « bonnes » roches (avec de la valeur scientifique) et les autres sont les « mauvaises » roches (sans valeur scientifique), toutefois l'agent ne connaît pas le type des différentes roches. Notez que, pour le diagnostic, une « bonne » roche n'est pas plus importante qu'une « mauvaise », parce que l'objectif est de les désambiguiser toutes, et non de n'échantillonner que les « bonnes » comme dans le problème *Rock Sampling* original.

4.3.3.1 Dec-POMDP → ιMDP → oMDP

J'ai eu l'opportunité de m'intéresser de plus près à la planification pour POMDP décentralisés à l'occasion du postdoctorat nancéen de Jilles Dibangoye, en collaboration à François Charpillet et Christopher Amato. Comme déjà évoqué en section 2.3.2, la résolution de Dec-POMDP a été abordée de différentes manières, dont une recherche heuristique (MAA*) inspirée de A*, mais aussi la programmation dynamique ou la programmation mathématique. Sans que ses auteurs ne le disent explicitement, l'approche adoptée par MAA* transforme le Dec-POMDP de départ en un MDP déterministe. Il y a là une analogie avec les POMDP couramment transformés en belief MDP. Comme expliqué ci-après, nous avons poussé cette analogie plus loin pour en tirer bénéfice.

Dans ce MDP déterministe, on prend le point de vue non pas d'un ou de plusieurs agents du problème, mais d'un algorithme planificateur extérieur qui raisonne sur tous les futurs possibles. Ce planificateur raisonne sur les séquences (partielles, mais commençant à $t = 0$) de règles de décision jointes. À une telle séquence correspond un ensemble de situations (combinant état et historiques des différents agents) qui peuvent être atteintes. On note ι un tel *état d'information*, et on définit un *information MDP* (ιMDP) dont la dynamique est déterministe — parce qu'ajouter une règle de décision jointe ne peut amener qu'à une seule nouvelle séquence de règles — et dont la résolution fournit une politique jointe optimale pour le Dec-POMDP de départ.

Le calcul de la récompense pour une transition donnée d'un ιMDP requiert de passer par le calcul (coûteux) de la distribution de probabilité sur les couples états/historiques-jointes pour l'état d'information courant. Une telle distribution ξ_t, appelée *état d'occupation*, s'avère pouvoir être calculée en fonction de ξ_{t-1} et de la dernière règle de décision jointe choisie, ce qui permet de définir une chaîne de Markov contrôlée (déterministe) dans l'espace de ces distributions. On peut prouver en outre que sa connaissance (plus la connaissance de la fonction de valeur à $t + 1$) est suffisante pour prendre une décision optimale. De là, on définit un *occupancy MDP* (oMDP) dont la résolution fournit une politique optimale pour le Dec-POMDP de départ.

Résolution et représentations compactes L'intérêt des occupancy MDP tient non seulement (1) au calcul récursif possible d'une statistique suffisante nécessaire et (2) à ce qu'un état d'occupation peut représenter plusieurs états d'information équivalents, mais aussi (3) à ce que la fonction de valeur est linéaire par morceaux et convexe (PWLC) dans l'espace des états d'occupation — comme, pour les POMDP, cette même fonction est PWLC dans l'espace des états de croyance associé. Cette propriété fournit un moyen d'approcher la fonction de valeur, et permet de réutiliser divers algorithmes efficaces développés pour les POMDP. En l'occurrence, nous avons employé *Heuristic Search Value Iteration* (HSVI), lequel exploite à la fois une minoration et une majoration de la fonction de valeur de manière à contrôler la convergence de l'algorithme.

Toutefois, le nombre d'historiques jointes croît toujours doublement exponentiellement, ce qui induit

une explosion combinatoire de la dimension de l'espace des états d'occupation comme du nombre de règles de décisions possibles. Nous avons donc cherché à réduire le nombre d'historiques nécessaires à la prise de décision. Nous partons pour cela de l'idée de regrouper les historiques individuelles *probabilistiquement équivalentes*, c'est-à-dire impliquant les mêmes probabilités de transition (et observation). Identifier (confondre) deux historiques individuelles probabilistiquement équivalentes permet de réduire non seulement le nombre d'historiques individuelles, mais aussi d'historiques jointes, comme cela a été mis en œuvre dans GMAA*-ICE (Oliehoek *et al.*, 2013). Deux problèmes se posent toutefois, auxquels nous apportons des réponses :

— il reste coûteux de tester si des historiques individuelles sont probabilistiquement équivalentes, parce qu'il faut tenir compte de tous les états possibles et toutes les historiques jointes des autres agents ; pour pallier cette difficulté, nous nous restreignons à ne chercher des politiques probabilistiquement équivalentes que si elles sont identifiables par un suffixe commun ; la réduction de dimensionalité reste souvent importante, les historiques récentes contenant souvent beaucoup d'information, et s'avère bien plus rapide à trouver ;

— des historiques peuvent être probabilistiquement équivalentes dans un état d'information (ou d'occupation) donné, mais pas dans un autre ; les états d'occupations compacts (EOC) s'écrivent donc chacun à l'aide de ses propres caractéristiques (*features* — les classes d'équivalence identifiées) ; nous avons toutefois démontré qu'il reste possible d'appliquer la propriété PWLC pour calculer un minorant comme un majorant de la fonction de valeur en généralisant les connaissances déjà acquises ; cela repose sur l'idée qu'on peut évaluer en un EOC ξ les connaissances obtenues en un autre EOC ξ' en mettant en correspondance leurs classes d'équivalences (leurs caractéristiques) ; mais, plutôt que de chercher la meilleure mise en correspondance, une seule est choisie de manière heuristique.

Même si des représentations compactes sont employées, énumérer les règles de décision optimales jointes pour trouver la meilleure possible en un état d'occupation compact donné reste très coûteux. Nous avons donc remplacé cette énumération par la résolution d'un problème de satisfaction de contraintes pondéré (Weighted CSP), résolution effectuée en pratique à l'aide de ToulBar2 (de Givry *et al.*, 2005).

Malgré les différentes transformations décrites ci-dessus, imposées par le passage des POMDP aux Dec-POMDP, on peut observer que l'algorithme 10 obtenu, *Feature-Based HSVI*, est très similaire dans sa structure à son parent HSVI.

Algorithme 10 : The FB-HSVI Algorithm

```
function FB-HSVI((L_{M̃,t})_{t∈{0,1,···,T}}, (U_{M̃,t})_{t∈{0,1,···,T}})
    initialize L_{M̃,t} and U_{M̃,t} for all t ∈ {0, ··· , T − 1}.
    while ¬ Stop (ξ̃_0, 0) do Explore(ξ̃_0, 0)
function Explore(ξ̃_t, g_t)
    if ¬ Stop (ξ̃_t, g_t) then
        d̃_t ∈ arg max_{d̃'_t ∈ Ã} R̃(ξ̃_t, d̃'_t) + U_{M̃,t+1}(P̃(ξ̃_t, d̃'_t))
        Update U_{M̃,t}
        Explore(P̃(ξ̃_t, d̃_t), R̃(ξ̃_t, d̃_t) + g_t)
        Update L_{M̃,t}
    return g_t
function Stop(ξ̃_t, g_t)
    return U_{M̃,t}(ξ̃_t) ≤ L_{M̃,t}(ξ̃_t) ∨ L_{M̃,0}(ξ̃_0) ≥ g_t + U_{M̃,t}(ξ̃_t)
```

Les tables 4.1 et 4.2 présentent une comparaison de FB-HSVI avec les principaux autres solveurs exacts de Dec-POMDP. L'algorithme que nous avons proposé s'avère bien plus efficace que ses concurrents sur cet ensemble de bancs d'essai. Les gains en temps de calcul étant de plusieurs ordre de grandeur, il n'est pas particulièrement gênant que les différentes évaluations n'aient pu toutes être menées sur le même

ordinateur. On notera que l'utilisation de représentations compactes est nécessaire à cette efficacité, l'exploitation de la propriété PWLC seule s'avérant insuffisante.

T	MILP	LPC	IPG	ICE	OHSVI	**FB-HSVI**	$L_{0,\tilde{M}}(\xi_0)$
MULTI-AGENT TIGER							
2	–	0.17	0.32	0.01	0.161	**0.03**	−4.00
3	4.9	1.79	55.4	0.01	28.567	**0.40**	5.1908
4	72	534	2286	108		**1.36**	4.8027
5				347		**9.65**	7.0264
6						**24.42**	10.381
7						**33.11**	**9.9935**
8						**41.21**	**12.217**
9						**58.51**	**15.572**
10						**65.57**	**15.184**
RECYCLING ROBOT							
2	–	–	0.30	36	0.04	**0.01**	7.000
3	–	–	1.07	36	0.555	**0.10**	10.660
4	–	–	42.0	72	696.8	**0.30**	13.380
5	–	–	1812	72		**0.34**	16.486
10						**0.52**	31.863
30						**1.13**	93.402
70						**2.13**	216.47
100						**2.93**	**308.78**
MEETING IN A 3X3 GRID							
2	–	–	–	–	93.029	**0.03**	0.0
3	–	–	–	–		**0.04**	0.133
4	–	–	–	–		**0.79**	0.432
5	–	–	–	–		**1.30**	0.894
6						**1.88**	**1.491**
7						**2.55**	**2.19**
8						**18.06**	**2.96**
9						**24.39**	**3.80**
10						**34.42**	**4.68**
20						**291.1**	**14.35**
30						**456.6**	**24.33**

TABLE 4.1 – Expérimentations comparant les temps de calcul (en secondes) de tous les solveurs exacts (1ère partie). Les violations de la limite de temps sont indiquées par une espace « », alors que « – » indique des valeurs inconnues. Les entrées en gras correspondent aux meilleurs résultats connus pour ces bancs d'essai, à la fois en termes de temps de calcul et de valeur espérée ($L_{0,\tilde{M}}(\xi_0)$ est le minorant obtenu en l'état d'occupation initial).

4.3.3.2 Approximations contrôlées

Une suite naturelle à ces premiers travaux sur les Dec-POMDP a été de chercher comment faire de FB-HSVI un algorithme approché tout en contrôlant les erreurs faites. En nous plaçant dans le cas d'un horizon temporel infini avec facteur d'atténuation $\gamma > 0$, nous avons identifié trois opérateurs clefs qui permettent de telles approximations :

— **G**, l'opérateur gourmand de sélection d'une règle de décision ; un WCSP peut en effet être résolu à $\alpha > 0$ près ;
— **P**, l'opérateur d'estimation de l'état d'occupation, en autorisant des compressions avec perte — c'est-à-dire en employant une quasi-équivalence probabiliste — du moment que la distance variationnelle totale à l'état d'occupation exact ne dépasse pas $\delta > 0$; et
— **T**, l'opérateur de mise-à-jour de Bellman, en interrompant les explorations quand l'écart (*gap*) entre minorant et majorant dépasse un seuil $\frac{\epsilon}{\gamma^t}$.

T	MILP	LPC	IPG	ICE	OHSVI	**FB-HSVI**	$L_{0,\tilde{M}}(\xi_0)$
BROADCAST CHANNEL							
2	–	–	–	–	0.036	**0.02**	2
3	–	–	–	–	3.446	**0.22**	2.99
4	–	–	–	–		**0.32**	3.89
5	–	–	–	–		**0.33**	4.79
10						**0.78**	9.29
30						**14.0**	27.42
50						**41.7**	45.50
100						**473.3**	90.76
GRID SMALL							
2	0	–	0	36	0.911	**0**	0.37
3	0.65	–	0.18	36		**0.1**	0.91
4	1624	–	4.09	1512		**0.73**	1.55
5		–	77.4	242605		**1.39**	2.24
6						**3.51**	2.97
7						**8.30**	3.71
8						**42.2**	**4.47**
9						**69.03**	**5.23**
10						**581.2**	**6.03**
COOPERATIVE BOX-PUSHING							
2	–	–	1.07	36	0.294	**0.1**	17.600
3	–	–	6.43	540		**0.457**	66.081
4	–	–	1138	2596		**0.622**	98.593
5						**5.854**	**107.72**
6						**10.7**	**120.67**
7						**24.96**	**156.42**
8						**28.97**	**191.22**
9						**184.3**	**210.27**
10						**293.7**	**223.74**
MARS ROVERS							
2	–	–	83	1.0	0.027	**0.10**	5.80
3	–	–	389	1.0	1.881	**0.23**	9.38
4				103		**0.47**	10.18
5						**0.82**	**13.26**
6						**3.97**	**18.62**
7						**5.81**	**20.90**
8						**22.8**	**22.47**
9						**26.5**	**24.31**
10						**62.7**	**26.31**

TABLE 4.2 – Expérimentations comparant les temps de calcul (en secondes) de tous les solveurs exacts (2ème partie).

Par ailleurs, en remplaçant l'interruption des explorations en fonction de l'écart par un horizon fini $T = \lceil \log_\gamma ((1-\gamma)\varepsilon/\|r\|_\infty) \rceil$, et en considérant le cas où les paramètres α et δ dépendent du pas de temps, nous avons démontré que l'erreur est bornée par :

$$2\|r\|_\infty \sum_{t=0}^{T-1} \gamma^t \left[1 - \prod_{k=1}^{t}(1-\boldsymbol{\delta}(k))\right] + \left(\sum_{t=0}^{T-1} \gamma^t \alpha(t)\right) + \varepsilon. \tag{4.1}$$

D'après nos résultats expérimentaux, l'algorithme obtenu, *Error-Bounded FB-HSVI* (EB-FB-HSVI), fournit de manière générale des solutions de qualité meilleure ou au moins comparable aux sept algorithmes de l'état de l'art considérés, en étant nettement plus rapide. Il s'avère offrir le meilleur compromis entre qualité de la solution obtenue et ressources computationnelles nécessaires (temps et mémoire). En outre, il fournit des garanties quant à l'erreur effectuée. Nous avons aussi pu comparer (i) l'erreur *a priori* et (ii) *a posteriori*[42] avec (iii) l'écart entre minorant et majorant pour l'état initial. Comme attendu, ces trois chiffres vont décroissant, sauf quand le temps de calcul est dépassé. Par ailleurs, l'erreur *a posteriori* s'avère souvent bien plus proche de l'écart que de l'erreur *a priori*.

4.3.3.3 Network-Distributed POMDP

Une autre façon de résoudre des Dec-POMDP de manière plus efficace est de s'intéresser à une sous-classe pour exploiter sa structure et ses propriétés. Une classe particulièrement intéressante (et étudiée) de Dec-POMDP est celle des *POMDP distribués en réseaux* (*Network-Distributed POMDP* / ND-POMDP), c'est-à-dire le cas dans lequel chaque agent n'a d'interactions directes — à travers sa dynamique, ses observations et la fonction de récompense — qu'avec un sous-ensemble fixe des autres agents. Plus précisément, un ND-POMDP est défini par un tuple $\langle I, U, \mathcal{S}, \mathcal{A}, \mathcal{Z}, P, O, \mathcal{R}, b_0, T\rangle$, où :

- I est un ensemble fini de n *agents* et U est un ensemble fini de *sous-groupes d'agents* (appelés ici *voisinages*) ;
- $\mathcal{S}^i$ est l'ensemble fini des *états privés* s^i de l'agent i et $\mathcal{S}^0$ est l'ensemble fini des états non affectés[43] s^0 ; on dénote alors $\mathcal{S}^u = \times_{i\in u} \mathcal{S}^i$ l'ensemble *des états de voisinage* s^u pour tout voisinage $u \in U$, et $\mathcal{S} = \mathcal{S}^0 \times_{i\in I} \mathcal{S}^i$ l'ensemble des *états* s ;
- $\mathcal{A}^i$ est un ensemble fini *d'actions privées* de l'agent i ; on dénote alors $\mathcal{A}^u = \times_{i\in u} \mathcal{A}^i$ l'ensemble des *actions de voisinage* a_u pour chaque voisinage $u \in U$, et $\mathcal{A} = \times_{i\in I} \mathcal{A}^i$ l'ensemble des *actions jointes* a ;
- $\mathcal{Z}^i$ est l'ensemble fini des *observations privées* de l'agent i ; on dénote alors $\mathcal{Z}^u = \times_{i\in u} \mathcal{Z}^i$ l'ensemble des *observations de voisinage* z^u pour tout voisinage $u \in U$, et $\mathcal{Z} = \times_{i\in I} \mathcal{Z}^i$ l'ensemble des *observations jointes* z ;
- P^i est le *modèle de transition privé* de l'agent i : $P^i(s^0, s^i, a^i, \dot{s}^i)$ est la probabilité de transiter vers l'état privé $\dot{s}^i$ après exécution de l'action a^i dans les états (s^0, s^i) ; et P^0 dénote la fonction de transition non-affectable ; on dénote alors P — la fonction de transition (jointe) du processus — la *fonction de transition (multiplicativement faiblement séparable)* :

$$P(s, a, \dot{s}) = P^0(s^0, \dot{s}^0) \prod_{i\in I} P^i(s^0, s^i, a^i, \dot{s}^i); \tag{4.2}$$

- O^i est le *modèle d'observation privé* de l'agent i : $O^i(a^i, \dot{s}^0, \dot{s}^i, \dot{z}^i)$ est la probabilité d'observer l'observation privée $\dot{z}^i$ dans l'état $(\dot{s}^0, \dot{s}^i)$ après avoir effectué l'action privée a^i ; on dénote alors O — la fonction d'observation (jointe) — une *fonction d'observation* :

$$O(a, \dot{s}, \dot{z}) = \prod_{i\in I} O^i(a^i, \dot{s}^0, \dot{s}^i, \dot{z}^i); \tag{4.3}$$

42. Ces erreurs sont calculées en utilisant la formule 4.1 avec (i) les seuils de tolérance fixés d'une part vs (ii) les erreurs d'approximation observées par l'algorithme d'autre part.

43. Les états non affectés font référence aux composantes de l'état du monde qui ne peuvent être impactés par les décisions prises par les agents (par exemple, les facteurs tels que la météo, ou les positions de cibles dans des systèmes de suivi multi-capteurs).

— $\mathcal{R}^u$ est une *fonction de récompense de voisinage* pour tout voisinage $u \in U$, où $\mathcal{R}^u(s^0, s^u, a^u)$ est la récompense obtenue à l'exécution de l'action de voisinage a^u dans les états (s^0, s^u) ; on dénote alors $\mathcal{R}$ le *modèle de récompense* :

$$\mathcal{R}(s, a) = \sum_{u \in U} \mathcal{R}^u(s^0, s^u, a^u); \tag{4.4}$$

— b_0^i est une *distribution sur les états initiaux* pour tout agent $i \in I$; on dénote alors b_0^0 la distribution initiale sur les états non affectés, et $\boldsymbol{b}_0$ la *distribution initiale sur les états* :

$$b_0(s) = b_0^0(s^0) \prod_{i \in I} b_0^i(s^i); \tag{4.5}$$

— T est l'horizon temporel fini.

La figure 4.6 illustre ce formalisme sur un problème de suivi de cible à l'aide de plusieurs capteurs fonctionnant en réseau, deux capteurs voisins devant examiner leur zone commune simultanément pour bien localiser la cible si elle s'y trouve.

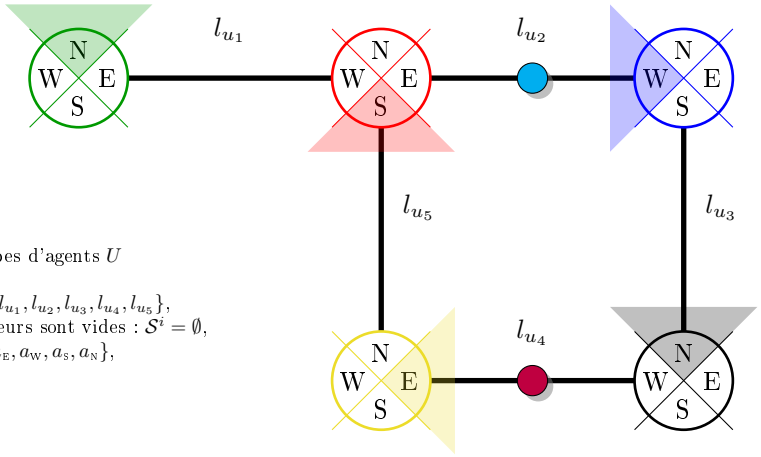


FIGURE 4.6 – Une illustration du problème de suivi de cible à cinq capteurs

Des travaux antérieurs avaient déjà expliqué comment exploiter la structure particulière d'un ND-POMDP pour le résoudre plus efficacement qu'avec une approche générique pour Dec-POMDP (Nair *et al.*, 2005; Kumar *et al.*, 2011) comme cela a été fait aussi pour des MDP factorisés. Nous avons pour notre part fait de même dans le cadre de notre approche par occupancy MDP, ce qui repose en premier lieu sur les résultats suivants :

— la fonction de valeur d'une politique jointe partielle est *additivement faiblement séparable et linéaire* (AWSL), c'est-à-dire qu'elle s'écrit comme la somme d'une fonction linéaire pour chaque facteur ;
— les états d'occupation marginaux, c'est-à-dire des restrictions de l'état d'occupation à chaque facteur, constituent une statistique suffisante pour la planification optimale.

On peut ainsi, de manière plus efficace, (i) mettre à jour une statistique suffisante, et (ii) évaluer les minorants et majorants de la fonction de valeur. En outre, si l'on ne peut pas s'affranchir de la recherche centralisée d'une meilleure règle de décision en un état d'occupation donné, le WCSP employé tire parti des représentations particulières utilisées.

Les résultats expérimentaux obtenus montrent d'abord que *Separable FB-HSVI* (SFB-HSVI) est nettement plus rapide que FB-HSVI, seul autre planificateur exact utilisable sans dépasser trop vite les limites computationnelles. Dans des comparaisons avec deux planificateurs approchés de l'état de l'art, si SFB-HSVI se montre parfois plus lent, il fournit souvent des solutions de bien meilleure qualité (et dont l'optimalité est garantie).

Dans la lignée de ces travaux, nous avons proposé une méthodologie pour adapter plus systématiquement les algorithmes à différentes classes de Dec-POMDP (Dibangoye *et al.*, 2015a). Comme on le verra dans les travaux futurs, cette piste reste toutefois à développer.

4.4 BRL avec BOLT

ICML'12	(Araya-López *et al.*, 2012a)
EWRL'11	(Araya-López *et al.*, 2011a)

Dans le cadre de la thèse de Mauricio Araya-Lòpez, nous avons aussi travaillé sur l'apprentissage par renforcement bayésien (avec modèle). Comme expliqué en section 2.3.3.1, un problème d'apprentissage par renforcement bayésien peut être reformulé comme un POMDP particulier (appelé BA-MDP) dont l'état caché correspond aux paramètres inconnus du modèle. Une telle reformulation ne permet toutefois pas d'utiliser directement les algorithmes développés pour résoudre les POMDP, principalement parce que l'espace d'états est alors continu — et donc l'espace des croyances infini-dimensionnel.

Parmi les approches de résolution de l'AR bayésien, on distingue typiquement :

— celles qui travaillent directement sur cette reformulation ;
— celles qui planifient les actions selon le modèle courant et ajoutent une exploration aléatoire ; et
— celles qui adoptent le principe de l'optimisme face à l'incertain.

Nous nous sommes placés dans ce dernier cas. Comme dans le cadre de la planification par recherche heuristique, le principe de *l'optimisme face à l'incertain* consiste à surestimer l'évaluation des actions dont les conséquences sont les plus incertaines. Certaines des approches reposant sur ce principe résolvent le MDP induit par le modèle espéré courant (à une étape donnée) en ajoutant une récompense d'exploration qui favorise les transitions avec des modèles moins bien connus, comme dans R-MAX (Brafman et Tennenholtz, 2003), BEB (Kolter et Ng, 2009), ou avec des récompenses basées sur la variance (Sorg *et al.*, 2010). Une autre approche, utilisée dans BOSS (Asmuth *et al.*, 2009), est de résoudre une estimée optimiste du vrai MDP (obtenue en fusionnant plusieurs modèles échantillonnés).

De notre côté, comme l'incertitude porte souvent sur la dynamique, nous avons proposé un algorithme, BOLT (Bayesian Optimistic Local Transitions), dans lequel l'optimisme apparaît non dans la fonction de récompense, mais dans la fonction de transition. Pour ce faire, pour chaque couple état-action (s, a), BOLT modifie la distribution sur les prochains états pour favoriser l'état s' associé au meilleur retour $(R(s, a, s') + \gamma V(s'))$. [44] Plus un couple état-action est visité, plus ce bonus devient négligeable. Cette évolution ayant lieu dans tout le MDP, l'algorithme converge.

Nous avons prouvé théoriquement de cet algorithme non seulement qu'il est optimiste et converge, mais aussi qu'il est PAC-BAMDP (probablement approximativement correct par rapport au BAMDP sous-jacent), ce qui donne des garanties sur la qualité des décisions prises. Comme c'est le cas aussi pour d'autres algorithmes tels que BEB, la preuve de ce dernier résultat repose sur l'utilisation d'inégalités de concentration (la borne de Hoeffding).

Par ailleurs, dans nos expérimentations sur quelques problèmes jouets, BOLT s'est avéré compétitif face à l'état de l'art. En outre, comme l'illustrent les figures 4.7 et 4.8, si on le compare avec BEB, concurrent direct entre autres parce que chacun d'eux n'a qu'un paramètre à régler, BOLT a l'avantage notable qu'il est bien moins sensible au réglage de son paramètre. [45]

4.5 Projets et applications [(B)RL / POMDP]

Nous décrivons ici des travaux applicatifs de modèles tels que les POMDP et l'apprentissage par renforcement. S'ils ont une visée plus directement pratique, ces travaux n'excluent toutefois pas des questions et contributions théoriques.

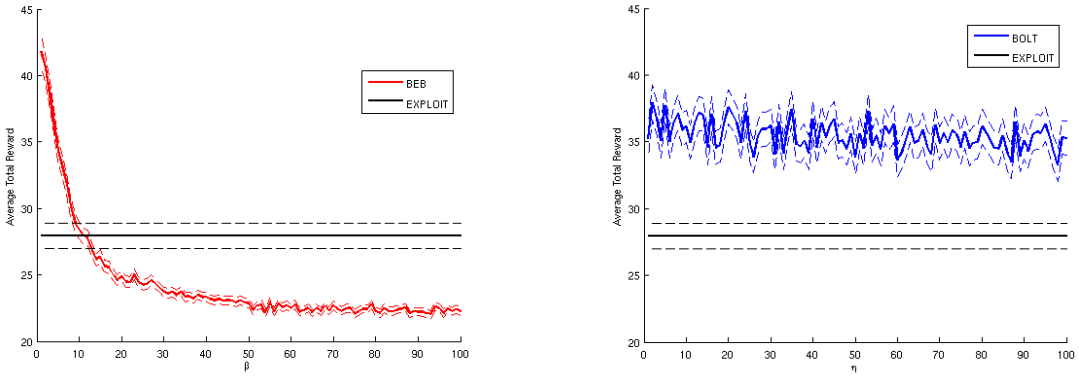
Dans chaque cas sont d'abord mentionnés les publications liées (ou consécutives) au projet, et les principaux collaborateurs impliquées.

4.5.1 Biologie de la conservation (Adaptive Managment) [BRL/POMDP]

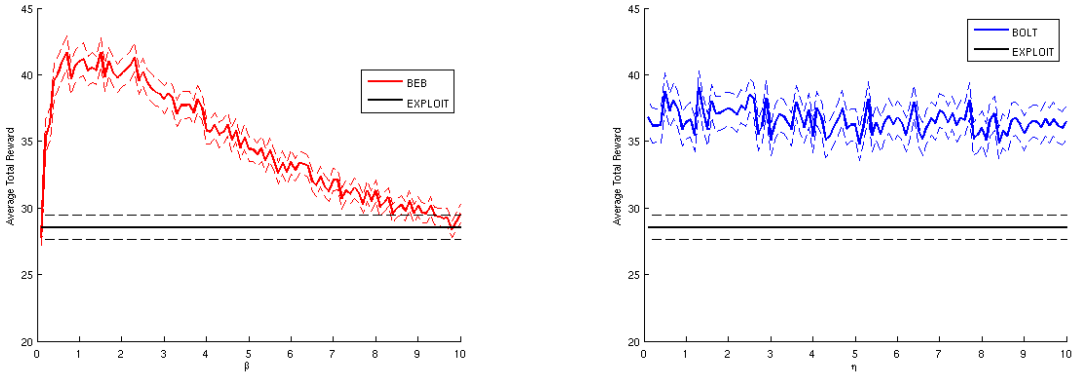
AAAI'12	(Chadès *et al.*, 2012)
IJCAI'13	(Nicol *et al.*, 2013)

44. On peut aussi voir ce mécanisme comme la résolution d'un nouveau MDP dans lequel on choisit une action *et* un état futur à favoriser.

45. EXPLOIT est un algorithme sans exploration qui sert essentiellement de référence.



(a) Analyse basse résolution : $\beta, \eta \in [1, 100]$



(b) Analyse haute résolution : $\beta, \eta \in [0.1, 10]$

FIGURE 4.7 – Résultats du réglage de paramètre sur le problème de la chaîne. Récompense totale moyenne sur 300 essais pour un horizon de 150 pas de temps, et pour des paramètres β et η entre 1 et 100 (en haut), et entre 0.1 et 10 (en bas). Pour référence, la valeur obtenue par EXPLOIT est aussi dessinée. Tous les résultats sont montrés avec un interval de confiance de 95%.

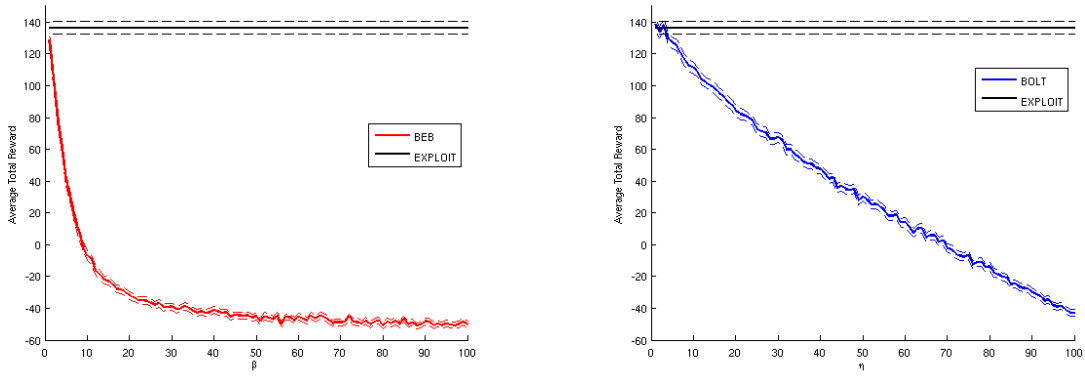
Collaborateurs principaux :

Nom	Affiliation
Iadine Chadès	CSIRO
Samuel Nicol	Univ. of Alaska Fairbanks → CSIRO
Josie Carwardine	CSIRO
Tara Martin	CSIRO
Régis Sabbadin	INRA
Takuya Iwamura	Stanford University

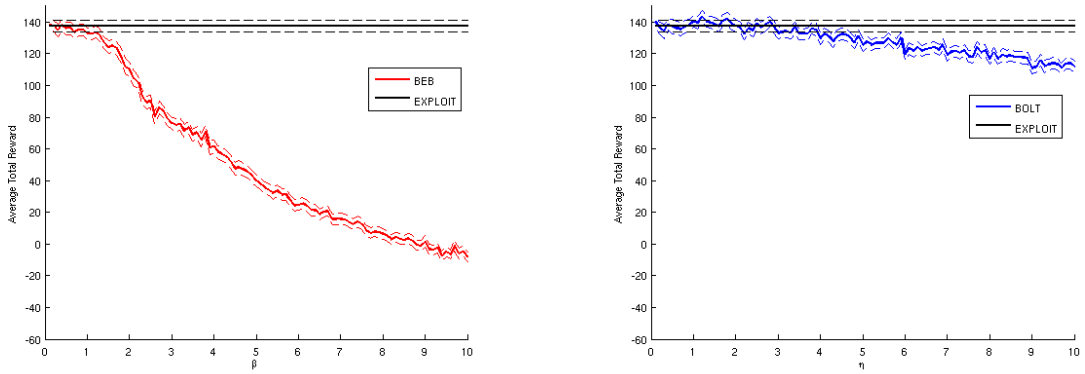
J'ai l'opportunité depuis quelques années de travailler occasionnellement avec Iadine Chadès (du CSIRO à Brisbane) et ses collaborateurs, dont Samuel Nicol (CSIRO) et Régis Sabbadin (INRA), sur des problèmes de biologie de la conservation. Cette discipline traite des questions de perte, maintien ou restauration de biodiversité.[46] Nous nous intéressons plus particulièrement à la recherche de stratégies permettant de sauvegarder des espèces menacées. Comme nous allons le voir à travers les exemples de deux travaux, ces problèmes de décision impliquent d'agir en cherchant à acquérir des informations.

Protection du diamant de Gould Le diamant de Gould (figure 4.9(a)) est une espèce de passereau menacée dans la région du Kimberley en Australie. Dans le scénario considéré, quatre actions sont possibles : ne rien faire (Do-Nothing), améliorer la gestion des feux et des pâturages (Fire-and-Grazing), contrôler les chats sauvages (Cats) et fournir des boîtes de nidification (Nesting-boxes) ; et deux états

46. https://fr.wikipedia.org/wiki/Biologie_de_la_conservation



(a) Analyse basse résolution : $\beta, \eta \in [1, 100]$



(b) Analyse haute résolution : $\beta, \eta \in [0.1, 10]$

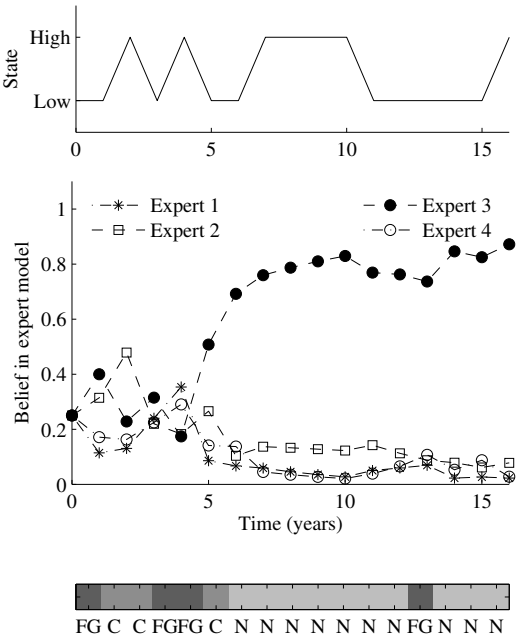
FIGURE 4.8 – Résultats du réglage de paramètre sur le problème Paint/Polish. Récompense totale moyenne sur 300 essais pour un horizon de 150 pas de temps, et pour des paramètres β et η entre 1 et 100 (en haut), et entre 0.1 et 10 (en bas). Pour référence, la valeur obtenue par EXPLOIT est aussi dessinée. Tous les résultats sont montrés avec un interval de confiance de 95%.

sont distingués selon que la probabilité de persistence de la population est *faible* ou *élevée*. On a toutefois une mauvaise connaissance des conséquences de ces actions. Quatre fonctions de transition différentes sont obtenues de quatre experts.

Supposant que l'un de ceux-ci a raison, nous avons modélisé ce problème comme un MDP à observabilité mixte (MOMDP), la partie cachée de l'état indiquant quelle est la bonne dynamique parmi les quatre hypothèses. On voit dans les expérimentations présentées dans la figure 4.9(b) que la politique obtenue identifie rapidement le modèle le plus vraisemblable et agit en conséquence. Ce travail a permis de montrer que les outils du domaine des MDP étaient adaptés pour aborder de tels problèmes à travers des approches de gestion adaptatives, c'est-à-dire intégrant la recherche active d'information pour mieux prendre les décisions ultérieures (à l'opposé d'approches passives plus classiques). Nous avons au passage prouvé que décider s'il existe une solution d'une certaine qualité à un MOMDP de ce type — dans lequel l'état caché est fixe — reste un problème PSPACE-complet.

Protection d'oiseaux migrateurs face à la montée du niveau de la mer La voie migratoire de l'Asie orientale et de l'Australasie permet à de nombreuses espèces d'oiseaux de rivage, après une période de reproduction en Sibérie, de traverser l'Asie vers le sud — avant de retourner en Sibérie — en faisant étape sur des vasières, habitats temporaires qui sont les nœuds d'un réseau des déplacements possibles des oiseaux (voir figure 4.10). Ces vasières sont malheureusement menacées par la montée du niveau de la

(a) Diamant de Gould



(b) Suivi d'une exécution de l'algorithme

FIGURE 4.9 – Illustration du problème de protection du diamant de Gould

mer consécutive au réchauffement climatique, et le développement urbain côtier limite les possibilités de déplacement de ces milieux vers l'intérieur des terres. Pour protéger ces oiseaux migrateurs, des actions de gestion de ces habitats temporaires peuvent être accomplies permettant de les adapter à la montée des eaux.

Nous avons modélisé ce problème à nouveau comme un MOMDP (voir le diagramme d'influence des principales variables figure 4.11), mais dont l'état caché, l'élévation du niveau de la mer M, évolue. On ne considère ici que trois élévations possibles : Om, 1m et 2m. L'état visible du système contient (i) la taille X de la population, discrétisée en 4 valeurs, et (ii) un état de protection V_n pour chaque nœud n du graphe, avec autant de valeurs possibles que de niveaux d'élévation. La dynamique stochastique décrit :

— comment l'état de protection d'une vasière peut se dégrader d'un niveau de manière aléatoire ou bien monter d'un niveau si une action de gestion est entreprise ;
— l'évolution stochastique du niveau de la mer avec un modèle simple tel que, en moyenne, 20 ans séparent deux élévations d'un niveau ; et
— enfin, l'évolution de la population en fonction des états de protection et du niveau de la mer, en déterminant le nombre d'oiseaux d'une génération pouvant effectuer la migration selon les capacités des nœuds du graphe.

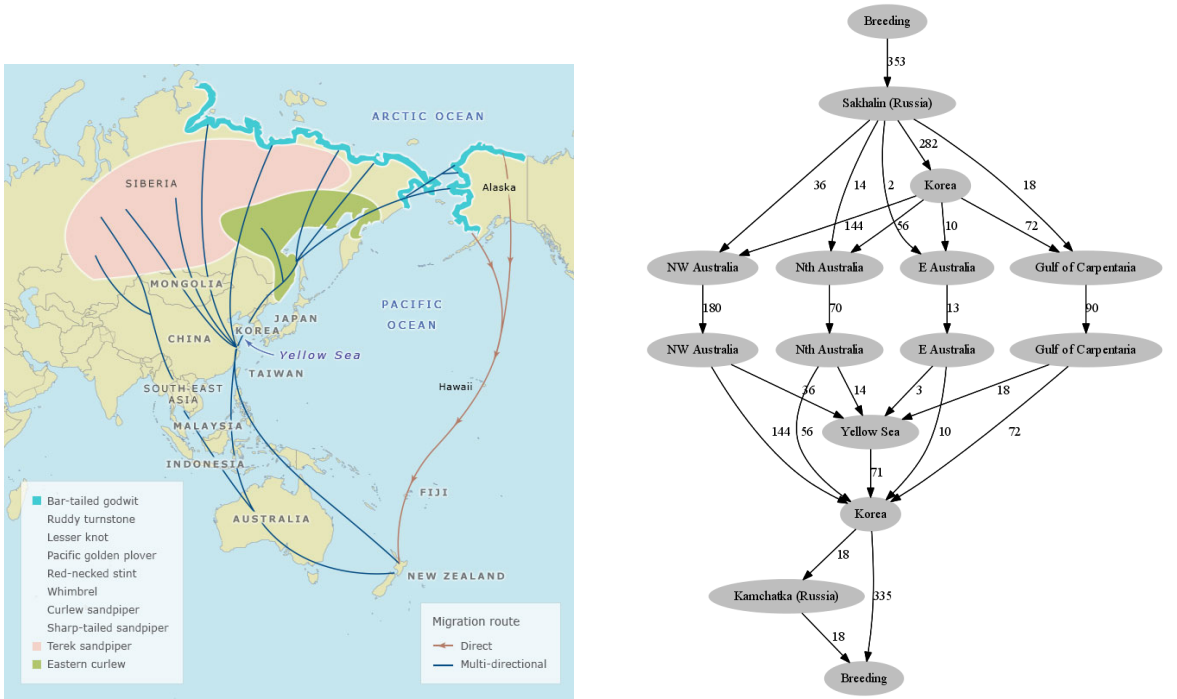
La fonction de récompense, elle, fait intervenir les coûts des actions effectuées et des coûts dépendant du nombre d'oiseaux qui auraient pu utiliser les habitats perdus.

Sur la base de ce modèle, nous avons effectué quelques expérimentations pour différentes espèces, [47] mais surtout mis à disposition un jeu de données pouvant servir de banc d'essai aux algorithmes de résolution de POMDP.

4.5.2 pen-testing [POMDP]

SecArt'11	(Sarraute *et al.*, 2011)
AAAI'12	(Sarraute *et al.*, 2012a)

47. Ici on ne s'intéresse qu'à la protection d'une espèce à la fois, chaque espèce ayant son propre modèle.



(a) Vue générale

(b) Réseau migratoire pour le bécasseau de l'Anadyr dans l'hypothèse d'une montée des eaux de 1m

FIGURE 4.10 – Voies migratoires de l'Asie orientale et de l'Australasie

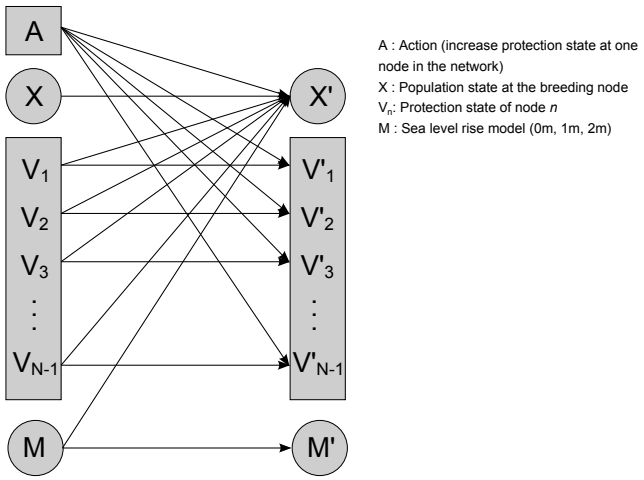
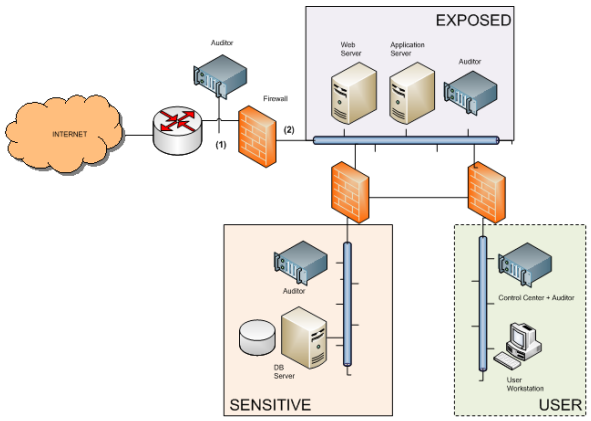


FIGURE 4.11 – Diagramme d'influence entre variables d'états (et variable d'action) pour la gestion des voies migratoires de l'Asie orientale et de l'Australasie

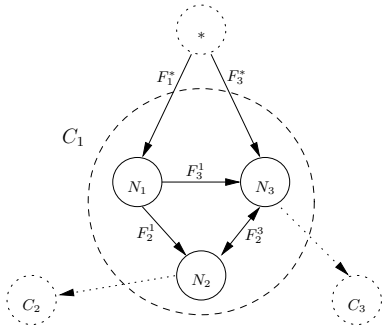
Collaborateurs principaux :

Nom	Affiliation
Jörg Hoffmann	INRIA → Saarland Univ.
Carlos Sarraute	Core Security

Bien loin à première vue de la biologie de la conservation, nous avons aussi travaillé dans le domaine de la sécurité informatique. Le problème posé ici est celui de l'évaluation de la sécurité d'un réseau par la réalisation d'un *test de pénétration* (*pentest*). Historiquement, il a été abordé entre autres comme un problème de planification d'attaque, connu comme le domaine « Cyber Security » (Boddy *et al.*, 2005). Cette approche a aussi été mise en avant indépendamment par l'entreprise Core Security (Lucángeli Obes *et al.*, 2010), une entreprise spécialisée dans ce domaine. Pour notre part, nous abordons le pentesting automatique régulier plutôt tel qu'il est effectué dans l'outil « Core Insight Enterprise » de Core Security.



(a) Un exemple de réseau à tester, la machine contrôlée de départ étant typiquement à l'extérieur



(b) Représentation d'un composant biconnexe ; Deux sous-réseaux i et j sont séparés par un pare-feu F_j^i

FIGURE 4.12 – Illustration du pentesting

Dans cet outil, une première phase d'analyse — coûteuse en temps et en trafic réseau — permet d'acquérir des informations sur les configurations, et donc les vulnérabilités, des machines. Sur la base de la configuration la plus probable, un problème de planification d'attaque est encodé en PDDL (Lucángeli Obes *et al.*, 2010) avant d'utiliser un planificateur standard — en l'occurence Metric-FF (Hoffmann, 2003) — pour trouver une séquence d'*exploits* à mettre en œuvre. Sarraute *et al.* (2011) ont cherché à mieux tenir compte des incertitudes en une probabilité de succès à chaque exploit, probabilité transformée en coût.

Une première contribution importante réalisée avec Sarraute et Hoffmann a été de revoir la modélisation du pentesting pour mieux représenter sa nature. En faisant l'hypothèse courante que la topologie du réseau est connue (voir par exemple la figure 4.12(a)), nous en sommes ainsi arrivés à formuler le problème du pentesting comme la résolution d'un POMDP dans lequel :

— l'état caché de chaque machine décrit sa configuration logicielle (et donc ses potentielles vulnérabilités) ;
— l'état visible décrit les machines contrôlées par l'attaquant ;
— les actions possibles correspondent à des tests ou des attaques sur les machines accessibles depuis les machines contrôlées.

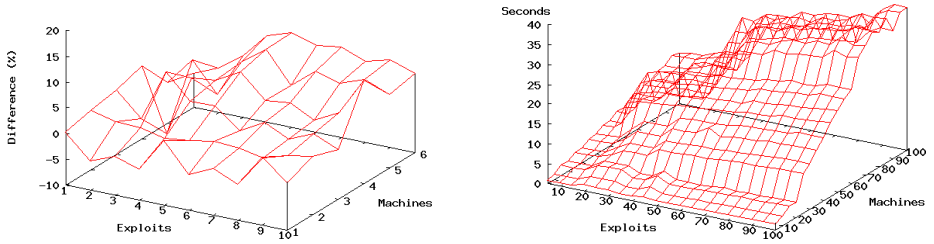
On notera entre autres que, (i) d'une part, les actions sont essentiellement déterministes et il ne sert à rien de les répéter ; (ii) d'autre part, l'incertitude n'est liée qu'à la méconnaissance initiale des configurations des machines.

Une seconde contribution a été d'analyser le problème obtenu et de proposer un algorithme exploitant ses caractéristiques, et particulièrement la structure du réseau. En effet, on peut voir ce réseau comme un graphe (enraciné en une machine contrôlée initiale notée *) dont les nœuds sont des sous-réseaux complètement connectés — donc dont il n'est pas utile de contrôler toutes les machines — et dont les

arcs représentent les connexions entre ces sous-réseaux. L'algorithme proposé, 4AL (pour *4 Abstract Layers*), suit les étapes suivantes :

1. Ce graphe de sous-réseaux est décomposé en un arbre de *composants* bi-connexes. On va donc raisonner sur chaque composant dans son ensemble comme décrit à l'étape 2, en remontant dans l'arbre. [48]
2. Dans un composant C (voir figure 4.12(b)), on considère tous les chemins P joignant des nœuds associés à des objectifs. Chaque chemin est alors remonté en appliquant l'étape 3 dans chaque sous-réseau, et en accumulant les valeurs retournées.
3. Étant donnés un sous-réseau N et un pare-feu F, pour chaque machine $m \in N$, calculer la valeur qu'il y aurait à l'attaquer avant les autres. Pour cela, on calcule avec l'étape 4 (i) la valeur de l'attaque de toute machine $m' \in N \setminus \{m\}$ sans pare-feu, puis (ii) la valeur de l'attaque de m sachant que son succès apporterait les valeurs des autres machines individuellement.
4. Pour une machine m, l'attaque se ramène à la résolution d'un POMDP. On mémorise les résultats pour éviter de répéter les mêmes calculs.

Cet algorithme, s'il ne fournit pas une stratégie optimale, permet de réduire drastiquement la complexité du problème. Par ailleurs, comme nous l'avons démontré, il garantit, à travers chaque étape, de ne jamais surestimer la valeur (maximisée) de la stratégie obtenue. Des résultats expérimentaux (voir figure 4.13) ont permis, sur des problèmes simples, de montrer (i) que 4AL fournit des solutions de qualité comparable à une résolution de POMDP globale, et (ii) que les temps de calculs restent raisonnables quand on augmente les nombres de machines et d'exploit dans un réseau type.



(a) Qualité relative des stratégies obtenues (b) Temps d'exécution de 4AL

FIGURE 4.13 – Résultats expérimentaux de 4AL comparé à un modèle POMDP global

4.5.3 Projet COMAC

confs int.	ICML'12	(Araya-López *et al.*, 2012a)
	EWRL'11	(Araya-López *et al.*, 2011a)
	NIPS'10	(Araya-López *et al.*, 2010a)
	ICTAI'10	(Araya-López *et al.*, 2010b)
confs nat.	JFPDA'13	(Araya-López *et al.*, 2013)
	JFPDA'12	(Araya-López *et al.*, 2012b)
	JFPDA'11	(Araya-López *et al.*, 2011b)
	CAp'11	(Araya-López *et al.*, 2011c)
	JFPDA'10	(Araya-López *et al.*, 2010c)
rapports	INRIA	(Araya-López *et al.*, 2012c)
	INRIA	(Araya-López *et al.*, 2010d)

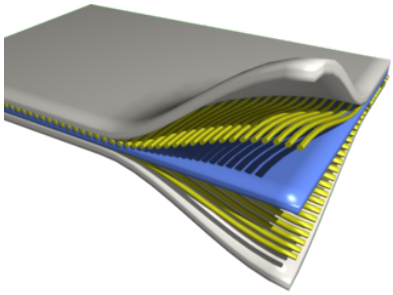
Note : ces publications présentent des travaux de recherche fondamentale liées au projet COMAC à travers la thèse de Mauricio Araya-López, mais pas de tentatives d'application au cadre du projet.

Collaborateurs principaux :

Nom	Affiliation
Vincent Thomas	Université de Lorraine
Marie Tonnelier	INRIA
Mauricio Araya-López	INRIA
François Charpillet	INRIA
Laurent Bougrain	Université de Lorraine

48. C'est la stratégie adoptée dans *Topological Value Iteration* (Dai *et al.*, 2011).

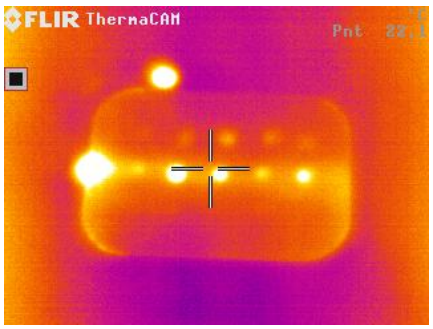
Le projet COMAC (Contrôle Optimisé Multi-techniques des Aérostructures Composites) portait sur le contrôle non-destructif (CND) de pièces en matériaux composites pour l'aéronautique. Un matériau composite est obtenu par assemblage d'au moins deux matériaux non miscibles, par exemple par superpositions de couches de plusieurs matériaux, comme illustré par la figure 4.14. Un tel matériau a l'avantage d'améliorer la qualité de la matière face à une certaine utilisation. En effet le nouveau matériau ainsi constitué possède des propriétés que les éléments seuls ne possèdent pas. À l'inverse, un tel matériau a l'inconvénient (i) que sa description est complexe du point de vue mécanique, et (ii) d'être non recyclable.



(a) principe d'un matériau composite multi-couche

(b) analyse ultrasonore par C-scan jet d'eau

(c) analyse optique de déformations en surface par shearographie



(d) observation d'inserts dans une plaque par thermographie

FIGURE 4.14 – Illustrations de techniques de contrôle non-destructif de pièces en matériaux composites

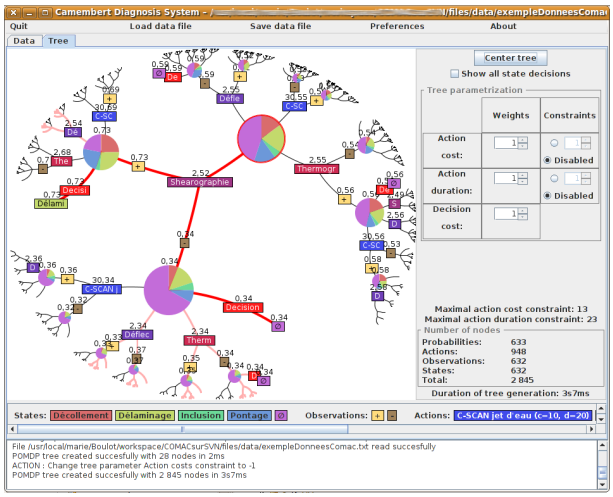
Ce projet impliquait plusieurs industriels ou laboratoires spécialisés dans différentes méthodes de contrôle, un but principal du projet étant de les évaluer selon plusieurs critères tels que leur efficacité — en tenant compte de la variété des défauts envisageables — , leur coût et leur portabilité. Les défauts peuvent être de différents types tels que des décollages de couches, des inserts (présence d'un élément étranger entre deux couches), ou des fissures. Les méthodes de contrôle peuvent reposer par exemple sur des moyens d'observation optiques (observant les déformations en surface), ultrasonores (observant la transmission des ondes sonores), ou thermographiques (observant les temps de transmission de la chaleur).

Une part de notre travail a consisté à proposer un outil logiciel permettant de stocker les résultats des tests effectués et de faire des recherches parmi ceux-ci selon différents critères tels que la méthode utilisée, le défaut ou la pièce considérée. Mais nous avons aussi montré comment ce problème de contrôle non-destructif (CND) avec plusieurs examens possibles était comparable au choix d'une séquence d'examens médicaux et pouvait donc être modélisé comme un problème de diagnostic actif tel qu'abordé dans le cadre de la thèse de Mauricio Araya-López (voir sections 4.3.1, 4.3.2, et 4.4). On notera que, comme pour le pentesting, il est supposé inutile de répéter un examen. Par contre, la modélisation concrète du problème étant trop peu avancée (manque important de données fiables) et ce problème étant potentiellement très complexe (avec des espaces continus), nous n'avons pas chercher à développer un outil pour réellement résoudre ce problème de diagnostic. En revanche, à des fins essentiellement pédagogiques, nous avons développé un logiciel — dénommé *Camembert Diagnosis System* (CDS) et employé comme preuve de

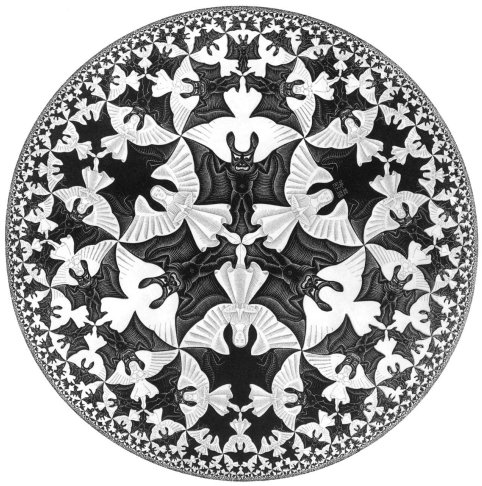
concept — permettant :

- de spécifier un problème de diagnostic actif avec différents états cachés possibles, différents examens, différentes observations, et les coûts associés aux examens (ainsi que des durées éventuelles) ;
- de résoudre ce problème avec une approche par programmation dynamique — coûteuse mais simple — qui développe l'arbre des futurs possibles avant de remonter les valeurs selon le principe d'optimalité de Bellman ; et
- de visualiser les valeurs résultantes et le sous-arbre solution obtenu.

La figure 4.15(a) montre une copie d'écran du logiciel CDS et sa visualisation sur un disque hyperbolique de Poincaré qui permet de se déplacer et zoomer facilement dans l'arbre obtenu. Une question qui mériterait d'être approfondie est d'adapter ce principe de visualisation à d'autres algorithmes qui ne construisent pas nécessairement l'arbre des futurs possibles.



(a) Une copie d'écran du logiciel *Camembert Diagnosis System* montrant une politique comme un arbre des futurs possibles, représenté dans un disque hyperbolique de Poincaré



(b) « Cirkellimiet iv », gravure sur bois de M.C. Escher (1960) représentant un pavage sur un disque hyperbolique de Poincaré

FIGURE 4.15 – Deux usages du disque hyperbolique de Poincaré

4.5.4 Suivi de cibles [HMM(→POMDP)]

ECAI'14	(Fansi Tchango *et al.*, 2014a)
STAIRS'14	(Fansi Tchango *et al.*, 2014b)
FUSION'14	(Fansi Tchango *et al.*, 2014c)
AAMAS'14	(Fansi Tchango *et al.*, 2014d)

Collaborateurs principaux :

Nom	Affiliation
Arsène Fansi Tchango	Thales / INRIA
Vincent Thomas	Université de Lorraine
Fabien Flacher	Thales ThereSIS
Alain Dutech	INRIA

Dans le cadre de la thèse CIFRE d'Arsène Fansi-Tchango, encadrée par Vincent Thomas et Alain Dutech d'une part, et Fabien Flacher d'autre part, nous avons travaillé sur un problème de suivi de cibles. L'objectif à long terme est de disposer de systèmes de surveillance automatiques capables de détecter des comportements « suspects » d'individus dans une foule. Il ne s'agit donc pas au départ d'un problème de contrôle de type MDP/POMDP. Un exemple type (figure 4.16) est l'identification, au sein d'une station de métro, de personnes dont le comportement ne correspondrait ni à un usager (qu'il aille prendre un métro, sorte de la station, ou s'achète à manger), ni à un employé (surveillant, technicien, ...).

À défaut d'aller jusqu'à cette détection d'anomalies, nous avons cherché à effectuer le suivi d'individus en supposant que (i) leur comportements incertains est modélisé par un simulateur, et (ii) un module de perception (typiquement reposant sur des caméras fixes) permet de localiser les individus quand ils ne sont

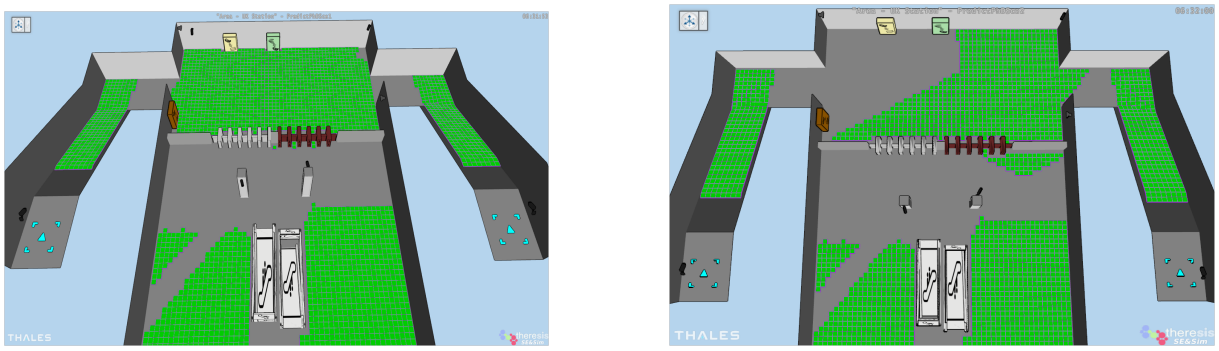


FIGURE 4.16 – Une même station de métro (avec 2 entrées, 2 distributeurs, des barrières à ticket et des escalators) avec deux couvertures par caméras différentes, représentées par les zones en vert.

pas cachés. Nous avons abordé ce problème comme un problème de filtrage bayésien, et plus précisément de filtrage particulaire pour pouvoir traiter des systèmes dont la dynamique est généralement complexe.

4.5.4.1 Filtrage avec des modèles riches

Dans un premier temps, nous avons adopté l'idée qu'il est souvent nécessaire de raisonner conjointement sur la localisation et l'activité d'un individu, comme cela a été fait auparavant par Wilson et Atkeson (2005); Manfredotti *et al.* (2011); Cattelani *et al.* (2014). Nous avons montré qu'il est possible d'effectuer ainsi le suivi d'un individu unique en employant un simulateur particulièrement riche dans la mesure où il permet de tenir compte de variables internes (faim, soif, endurance, destination, ...), d'interactions avec des objets de l'environnement (distributeurs, barrières, ...) et d'événements exogènes. Le système de suivi développé s'est avéré relativement robuste étant donné le degré de complexité des comportements mis en jeu, et était en particulier capable de gérer des périodes d'occlusion (sans observations) assez longues. En outre, il exploitait le parallélisme possible dans un tel filtre particulaire (avec 4 machines à 16 cœurs chacune dans nos expérimentations).

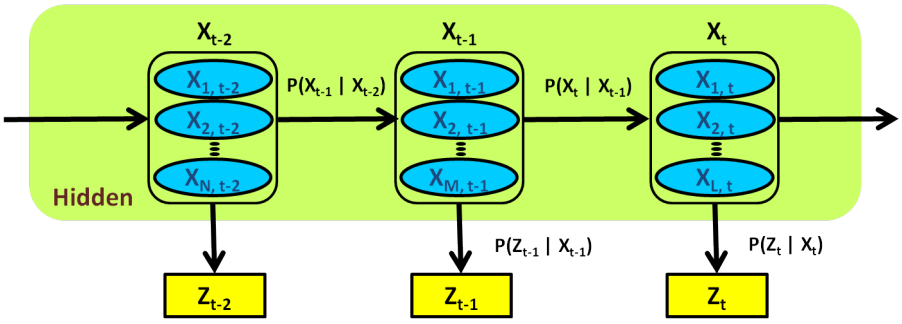
4.5.4.2 Suivi multi-cible par factorisation

Dans un deuxième temps, notre intérêt s'est porté sur le cas du suivi de plusieurs cibles (ou *Multi-Target Tracking* (Reid, 1979; Fortmann *et al.*, 1983; Schulz *et al.*, 2003; Liu *et al.*, 2007)). Sous l'hypothèse de cibles en interaction, le filtre bayésien doit alors idéalement maintenir une distribution de probabilité sur les états possibles du système complet, c'est-à-dire sur des états décrivant toutes les cibles, ce qui conduit très vite à une explosion combinatoire (voir figure 4.17a). On identifie alors deux natures d'interactions posant problème :

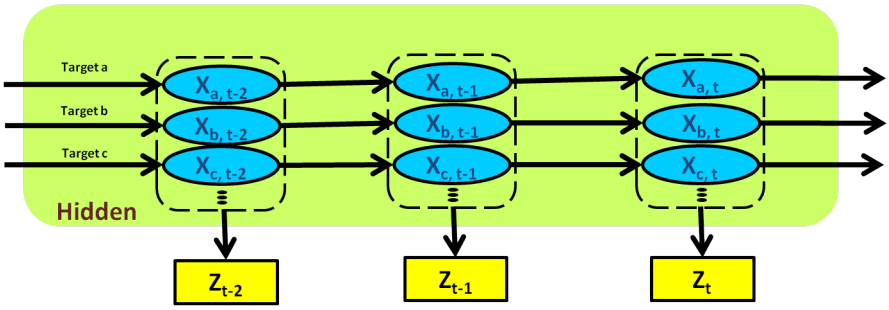
— l'*interaction observationnelle*, c'est-à-dire le fait que, dans les observations (/mesures) reçues, il y ait confusion possible entre les cibles ; en d'autres termes, une perception peut avec une certaine probabilité être associée à l'une ou l'autre cible ; cette difficulté a été abordée par exemple dans les approches MHT (Multiple Hypothesis Tracker) (Reid, 1979) et JPDAF (Joint Probabilistic Data Association Filter) (Fortmann *et al.*, 1983) ;
— l'*interaction au sein de la dynamique*, c'est-à-dire le fait que les dynamiques des cibles ne soient pas indépendantes, que les déplacements et autres actions de l'une dépendent des autres ; cette difficulté a été abordée par exemple par Khan *et al.* (2005).

Nous avons choisi d'adopter JPDAF pour gérer l'interaction observationnelle, et de nous tourner vers une approche par factorisation comparable à celle de Khan *et al.* (2005) pour gérer l'interaction dynamique.

Forme de la dynamique Toutefois, là où Khan *et al.* considèrent des dynamiques couplées *a posteriori* (les cibles évoluent *a priori* indépendamment les unes des autres, mais des contraintes influencent *a posteriori* les états atteignables), nous considérons des *dépendances conditionnelles markoviennes* (Daduna et Szekli, 1995; Pattison et Robins, 2002) sur les états des cibles. En d'autres termes, nous faisons



(a) Filtre bayésien classique dans lequel l'état $\vec{x}_t$ du système complet au temps t est une collection $\{\vec{x}_{k,t}\}$ d'états des différentes cibles présentes



(b) Filtre bayésien factorisé avec un « sous-filtre » par cible

FIGURE 4.17 – Modèle graphique d'un problème de suivi multi-cible

l'hypothèse que l'état d'une cible à l'instant courant dépend des états de toutes les cibles à l'instant précédent, ce qui s'écrit formellement :

$$p(\vec{x}_t|\vec{x}_{t-1}) = \prod_{k=1}^{K} p(\vec{x}_{k,t}|\vec{x}_{t-1}),$$

où $\vec{x}_{k,t}$ est le vecteur état de la cible k à l'instant t, et $\vec{x}_t$ le vecteur état joint de toutes les cibles à l'instant t. Cette façon de modéliser les dépendances entre cibles nous parait plus réaliste et relativement commune dans les systèmes physiques (Harel et Pnueli, 1985; Reynolds, 1999).

Factorisation et représentants En outre, pour qu'une factorisation de la distribution sur l'état joint en une distribution par cible soit pertinente, il ne faut pas estimer la dynamique en recombinant les distributions, [49] sans quoi on perdrait tout gain en complexité. Pour ne pas recombiner les distributions, Khan *et al.* (i) associent à chaque cible-distribution un unique représentant (typiquement la moyenne de la distribution), (ii) identifient les représentants en interactions les uns avec les autres (en l'occurrence parce qu'ils sont proches les uns des autres, puis, (iii) pour chaque cible, simulent l'évolution de chaque particule conjointement avec les représentants des cibles voisines identifiées.

Le choix d'un unique représentant pose toutefois problème puisqu'on risque d'omettre des interactions potentielles importantes. Nous avons donc proposé d'utiliser plusieurs représentants par cible, ceux-ci étant choisis à l'aide d'une procédure d'agrégation (*clustering*). Différentes méthodes d'agrégation classiques sont envisageables, par exemple fixant le nombre de représentants par cible (donc le coût computationnel par pas de temps), ou bien réglant plutôt la qualité de l'agrégation (donc la qualité du filtrage/suivi).

Dans le cas d'un filtre particulaire, l'utilisation de plusieurs représentants par cible fait que chaque particule p d'une cible k au temps t a autant de descendants à $t+1$ qu'il y a de combinaisons de

49. Dans le cas d'un filtre particulaire, cela consisterait à effectuer une simulation pour chaque combinaison d'une particule par cible.

représentants voisins, d'où une explosion combinatoire. Nous avons donc mis en place une procédure de réduction des ensembles de particules. Ce problème est assimilable au *problème du maintien de la diversité* des algorithmes évolutionnaires (AE), pour lesquels on doit sélectionner intelligemment, à chaque itération, les individus d'une population qui seront maintenus à la prochaine itération (*cf. l'algorithme de partage de fitness* de Goldberg et Richardson (1987) ou, plus récemment, la *méthode de nettoyage* de Petrowski (1996)). Dans notre cas, nous avons proposé d'effectuer, pour chaque cible, une procédure en deux étapes :

1. l'agrégation des N_p particules générées pour cette cible en une partition $SP = \{SP_s, \breve{w}_s\}$ de $|SP|$ sous-ensembles (agrégats ou sous-populations) où SP_s est le sous-ensemble de particules de taille $|SP_s|$ appartenant au s^{eme} agrégat et $\breve{w}_s$ est son poids (la somme des poids de ses particules) ; et
2. la sélection d'un sous-ensemble de $\sim \frac{|SP_s|}{N_p} N$ individus dans chaque agrégat SP_s par échantillonnage sans ou avec remplacement (respectivement un échantillonnage stratifié (Kitagawa, 1996) ou un échantillonnage résiduel (Liu et Chen, 1998)), avant re-pondération.

L'approche complète proposée a été implémentée et évaluée expérimentalement sur un scénario jouet implémentant des cibles à comportements réactifs de type *boids* (Reynolds, 1999).

4.5.4.3 Vers un problème de contrôle

Le lecteur aura pu remarquer que le travail effectué ici touche au filtrage et non au contrôle. Le contrôle est toutefois un objectif à long terme de ce travail, un problème important étant de concevoir un système capable d'orienter et faire (dé)zoomer des caméras au mieux pour compenser des champs de vision fixes limités. Le problème ainsi posé se formule comme un POMDP ou un ρ-POMDP.

4.5.5 Jeu du démineur [(PO)MDP/UCT]

ICS'12 (Lin *et al.*, 2012)

Collaborateurs principaux :

Nom	Affiliation
Alain Dutech	INRIA
Olivier Teytaud	INRIA
Chang-Shing Lee	National University of Tainan
Woan-Tyng Lin	National University of Tainan

Un dernier travail en lien avec la prise de décision séquentielle que je souhaite mentionner, même brièvement, est la résolution du jeu du démineur (*Minesweeper*, figure 4.18). Dans ce jeu, une grille représentant un champ de mines doit être nettoyée. Pour cela, on doit cliquer dans les cases supposées vides, lesquelles laissent alors apparaître le nombre de leurs immédiates voisines minées, jusqu'à ce qu'il ne reste que des cases minées. On notera que, dans les versions de référence du jeu, la première case cliquée est toujours vide, et qu'il y a régulièrement des situations où l'on doit tenter sa chance, n'étant pas sûr de sélectionner une case vide.

Les travaux scientifiques liés au jeu du démineur se sont souvent intéressés à la détermination des cases vides ou minées (ou indéterminées) étant donnés les indices courants (problème NP-difficile). Toutefois, le problème dans son ensemble est un POMDP dont le sous-problème précédent ne sert qu'à calculer la fonction de transition. La meilleure décision à un instant donnée correspond à un compromis entre faible probabilité de tomber sur une mine et gain potentiel d'information.

J'ai personnellement d'abord travaillé, dans le cadre d'un projet étudiant co-encadré avec Alain Dutech, sur le calcul efficace des probabilités de présence de mines et sur des heuristiques de prise de décision. Ensuite, j'ai eu l'opportunité de collaborer avec Olivier Teytaud, Woanting Lin et Chang-Shing Lee sur la résolution de ce jeu à l'aide de l'algorithme UCT, considéré comme pertinent pour affronter le très large espace de recherche du problème. L'algorithme obtenu donnait alors les meilleurs résultats sur une grande variété de types de grilles (différentes tailles, différents nombres de mines).

FIGURE 4.18 – Une situation de jeu du démineur ; on peut ici déterminer avec certitude, pour chaque case de la frontière, si elle contient une mine ou pas.

4.6 Discussion

Comme pour les MDP, mes travaux sur les POMDP (et extensions) contiennent principalement une partie reposant sur de la recherche directe de politique — en l'occurrence une montée de gradient —, et une autre reposant sur le calcul de la fonction de valeur optimale.

4.6.1 Montées de gradient

Les montées de gradient ont servi d'une part à apprendre des contrôleurs simples pour des situations difficiles, insolubles par des solutions exactes actuelle, dans lesquelles plusieurs agents doivent collaborer à une tâche sous observabilité partielle. Nous avons montré que la convergence de l'algorithme pouvait être aidée, mais que définir l'aide à apporter était en soit un problème difficile.

D'autre part nous avons évalué la résolution de problèmes de contrôle sous observabilité partielle par montée de gradient en donnant en entrée du contrôleur une statistique suffisante : soit l'état de croyance (point de vue POMDP), soit un vecteur de prédiction (point de vue PSR). Ces travaux essentiellement empiriques ont mené à plusieurs observations telles que (i) la confirmation de l'utilité d'employer des statistiques suffisantes, (ii) apprendre un POMDP, y compris le nombre d'états, n'est pas nécessairement plus difficile qu'apprendre un PSR (avec les algorithmes de référence du moment), et (iii) l'optimisation d'un contrôleur parait plus difficile dans le cadre PSR que dans le cadre POMDP.

Tout ceci confirme que, comme pour les MDP, les montées de gradient s'avèrent donc des outils pratiques pour aborder certains problèmes de contrôle quand des approches exactes ne seraient pas appropriées.

4.6.2 Extensions des POMDP

Mes plus importantes contributions reposant sur le calcul de la fonction de valeur optimale ont consisté à montrer comment résoudre certaines extensions des POMDP — en l'occurrence des ρ-POMDP et des Dec-POMDP — de la même manière que les POMDP eux-mêmes. À chaque fois, on se ramène à la résolution d'un MDP à espace d'états continu particulier, ce qui était déjà connu pour les Dec-POMDP, et en particulier au calcul d'une fonction de valeur linéaire par morceaux et convexe (PWLC).

En passant des MDP aux POMDP, puis aux Dec-POMDP, on généralise des travaux pour qu'ils rendent possible la résolution de problèmes de moins en moins contraints. Pour pouvoir gérer l'observabilité partielle, on ne raisonne plus sur l'état (désormais mal connu), mais sur l'information détenue. Pour pouvoir contrôler séparément plusieurs agents, on adopte un point de vue extérieur à ceux-ci. On

notera que ces façons de faire paraissent assez simples à comprendre, mais que les solutions efficaces obtenues intègrent des ingrédients dont la découverte est moins évidente, comme la propriété PWLC ou la compression d'états d'occupation.

Aussi, la plupart des preuves théoriques développées dans le cadre de mes travaux sont essentiellement des adaptations de preuves développées dans d'autres cadres. L'approximation bornée d'une fonction convexe α-hölderienne par une fonction PWLC est peut-être le résultat le plus innovant.

De tout cela je retiens qu'il est souvent question dans mes travaux (comme dans certains autres) de savoir ré-utiliser des idées connues dans de nouvelles situations. En un sens, cela rejoint l'idée que la créativité se nourrit de l'expérience passée.

4.6.3 BRL

Par ailleurs, nous avons proposé une nouvelle approche pour l'apprentissage par renforcement bayésien (avec modèle). Celle-ci repose sur l'idée d'être optimiste par rapport au modèle en cours d'apprentissage, optimisme qui se répercute sur la politique d'action à adopter. L'algorithme obtenu, BOLT, a la bonne propriété d'être PAC-BAMDP. Sur les expérimentations que nous avons menées, cet algorithme s'avère en outre assez peu sensible à son unique paramètre, contrairement à un algorithme optimiste de référence, BEB.

4.6.4 À propos de la modélisation (encore)

On peut noter sur les diverses applications présentées que, si les POMDP et apparentés s'avèrent très pertinents pour aborder une variété de problèmes, les modèles utilisés simplifient souvent notablement la réalité pour rendre un tant soit peu applicables les algorithmes existants. Par ailleurs, même s'il n'était pas utile de procéder à de telles simplifications, ces modèles resteraient assez « incertains », d'où l'intérêt d'étudier la sensibilité des solutions aux paramètres des modèles. D'ailleurs, si les échelles de temps entre deux décisions réelles peuvent être longues, il est utile aux praticiens de pouvoir expérimenter avec modèles et algorithmes, ce qui implique d'obtenir des solutions assez rapidement. Un autre point important pour nombre de praticiens (par exemple en écologie) est, dans la mesure du possible, d'obtenir des règles de décision simples à suivre.

Chapitre 5

Mes travaux autres

Sommaire

Le présent chapitre revient sur les quelques travaux sortant de la prise de décision séquentielle auxquels j'ai eu l'occasion de participer. Un premier ensemble, décrit en section 5.1, ne nous éloigne pas trop des préoccupations du reste de ce mémoire puisqu'il est question dans chacun des trois travaux (i) de systèmes dynamiques complexes (en ce qu'ils mettent en œuvre de multiples entités simples en interaction) et (ii) d'algorithmes de recherche. Le quatrième travail, décrit en section 5.2, reste pour sa part dans le domaine de la théorie de l'utilité, puisqu'il traite de systèmes de recommandation par filtrage collaboratif.

5.1 Sur des systèmes dynamiques complexes et des algorithmes de recherche

En guise d'introduction, la table 5.1 montre, pour les travaux dont la description suit, leurs liens avec des systèmes dynamiques complexes et des algorithmes de recherche.

	système dynamique complexe	algorithme(s) de recherche
détection de cycles	fourmis	EVAW (LRTA sans but)
ordonnancement temps réel	jobs/tâches à exécuter	solveur CSP et recherches arborescentes
gestion du trafic routier	véhicules et intersection	recherche locale

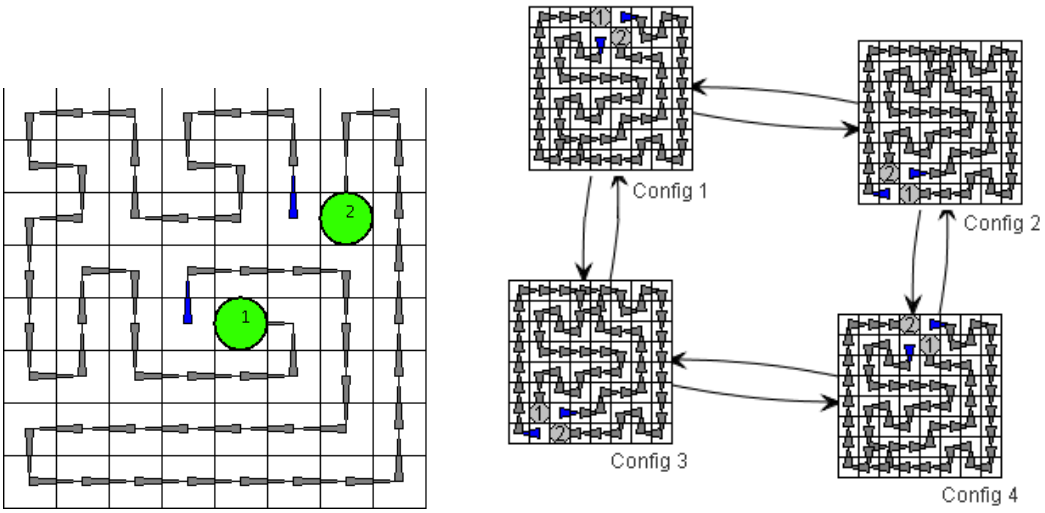
TABLE 5.1 – Pour chacun des travaux présentés dans cette section, le système dynamique complexe et le(s) algorithme(s) de recherche au(x)quel(s) il est associé.

5.1.1 Détection de cycles

ECAI'08	(Glad *et al.*, 2008)
SASO'09	(Glad *et al.*, 2009a)
AAMAS'10	(Glad *et al.*, 2010)

La thèse d'Arnaud Glad, encadrée par Olivier Simonin et François Charpillet, portait principalement sur l'étude d'algorithmes fourmis pour la patrouille d'un environnement discret. Il s'agit de comportements

réactifs adoptés par des agents mobiles, capables de marquer leur environnement, et tels qu'un groupe d'agents est assuré de parcourir de manière répétée, indéfiniment, cet environnement.[50] La figure 5.1a montre un tel comportement de patrouille d'une grille où deux agents suivent un même cycle.



(a) 2 agents sur un même cycle déterministe

(b) Sous-graphe absorbant simplifié avec 2 agents et 4 nœuds non-déterministes

FIGURE 5.1 – Exemples de patrouilles de grilles avec 2 agents

Un algorithme type est VAW (Chu *et al.*, 2007), dans lequel chaque agent-fourmi, à chaque instant :

1. cherche une cellule y (voisine de sa cellule courante x) de plus faible marquage : $y \in \min_{x' \in N(x)} m(x')$, où $N(x)$ désigne les cellules voisines d'une cellule x, et $m(x')$ désigne la valeur marquée sur une cellule x' ;
2. marque la cellule courante du pas de temps courant : $m(x) \leftarrow t$; et
3. se déplace vers la cellule choisie : $x \leftarrow y$.

Ce comportement est très proche de celui de l'algorithme RTA* (Real-Time A*) de Korf (1990) — lequel fournit un comportement d'agent cherchant un (plus) court chemin vers un but tout en explorant son environnement — dans le cas où il n'y a en fait pas de but à trouver, et donc où la recherche se poursuit indéfiniment.

Ensemble nous avons d'abord démontré l'équivalence entre EVAW, une variante de VAW obtenue en intervertissant deux opérations, et un autre algorithme de l'état de l'art, EVAP (Wagner *et al.*, 1999). En effet, l'évaporation progressive de phéromones marquant l'environnement dans EVAP (décroissance géométrique de la trace au cours du temps) est strictement équivalent au marquage par la date de dernière visite de EVAW. Nous avons aussi montré que, comme VAW dans le cas d'un seul agent, les algorithmes de ce type permettaient effectivement de patrouiller un environnement, le temps entre deux re-visites étant borné (majorant théorique souvent très pessimiste comparé aux simulations).

Continuant à étudier la dynamique de ces systèmes, nous avons remarqué que, dans le cas d'un seul agent-fourmi, après une première couverture de l'environnement, la dynamique du système devient déterministe. L'espace des états étant fini, on démontre alors que le comportement du système boucle nécessairement. En revanche, dans le cas multi-agent, il peut y avoir des décisions aléatoires (i) parce que deux agents sont en conflit pour aller sur une même case, ou (ii) parce qu'un agent a le choix entre deux cellules de même valeur minimale. Le système obtenu peut alors être décrit par une chaîne de Markov sur un espace fini, comme illustré par la figure 5.1b.

50. On ne confondra pas ces algorithmes fourmis et l'optimisation par colonie de fourmis (ACO) (Dorigo et Gambardella, 1997a,b).

Pour pouvoir étudier le comportement à long terme des algorithmes de patrouille considérés, au vu des résultats précédents nous nous sommes inspirés de l'algorithme du lièvre et de la tortue de Floyd (1967), lequel sert à détecter l'apparition d'un cycle dans un système déterministe. Nous avons proposé une variante qui permet, au cours d'une simulation d'une chaîne de Markov, de détecter quand un sous-graphe récurrent est atteint tout en le construisant.

Équipés de cet outil, nous avons mené une étude expérimentale sur des variantes de VAW, en fonction des nombres d'agents-fourmis et des environnements, pour observer par exemple leurs temps de convergence vers un comportement stable et la fréquence d'obtention de cycles hamiltoniens (chaque nouvelle couverture ne passant qu'une fois par cellule).

5.1.2 Ordonnancement temps-réel

XRTS'09	(Cucu-Grosjean et Buffet, 2009)
RTSOPS'10	(Buffet et Cucu-Grosjean, 2010a)
Evolve'11	(Buffet et Cucu-Grosjean, 2011a)
RTNS'11	(Maxim *et al.*, 2011)
ROADEF'11	(Buffet et Cucu-Grosjean, 2011b)

Liliana Cucu-Grosjean m'a pour sa part amené à m'intéresser à *l'ordonnancement temps-réel*, c'est-à-dire à l'ordonnancement de tâches répétitives sur des systèmes temps-réel, donc sous contraintes computationnelles fortes. Une tâche τ_i est typiquement décrite au moins par sa périodicité et sa durée au pire cas. Alors que la prise de décision séquentielle voit habituellement l'ordonnancement de tâches comme un problème d'optimisation combinatoire à résoudre, ce champ de recherche s'intéresse plus souvent aux bonnes propriétés — parfois contre-intuitives — d'algorithmes simples tels que donner la priorité aux tâches dont l'échéance est la plus proche. On fera attention dans les quelques travaux ci-dessous aux différentes hypothèses de travail.

Dans un premier temps (Cucu-Grosjean et Buffet, 2009; Buffet et Cucu-Grosjean, 2011b), nous avons abordé l'ordonnancement temps-réel, dans le *cas multi-processeur pré-emptif*,[51] comme un problème d'optimisation combinatoire, c'est-à-dire en supposant que l'ordonnancement complet pouvait être précalculé et mémorisé. Un algorithme de recherche heuristique dédié a été proposé pour résoudre ce problème de satisfaction de contraintes, avec différentes heuristiques, lequel montrait de très bonnes performances en comparaison avec un solveur générique.

En réponse à un appel à communication particulier nous avons ensuite proposé un problème ouvert lié à l'ordonnancement temps-réel *uniprocesseur*, nous demandant quel serait l'impact d'abandonner certains jobs (des instances de tâches). Nous avons considéré deux cas :

- Dans le premier cas (Buffet et Cucu-Grosjean, 2010a), chaque tâche τ_i a une *durée d'exécution décrite par une distribution de probabilité* (à support fini) $\mathcal{C}_i$ et doit être exécutée avec *probabilité au moins* p_i. Une analyse d'ordonnancement permet de calculer les temps de réponse pour chaque job sur une hyper-période (la période commune des différentes tâches) en supposant une règle d'ordonnancement donnée. Notre proposition (préliminaire) est alors d'abandonner les jobs ayant de grandes probabilités d'échec, ce qui laisserait du temps pour d'autres jobs.
- Dans un second cas (Buffet et Cucu-Grosjean, 2011a), nous considérons que les *durées d'exécutions des tâches sont déterministes*, mais que chaque tâche τ_i doit toujours être *exécutée avec probabilité au moins* p_i. Nous proposons alors une procédure pour déterminer si un job est condamné à échouer, auquel cas il peut être abandonné à l'avance. On montre alors que cette procédure, combinée à un algorithme exécutant les jobs suivant des priorités de tâches pré-définies, garantit d'améliorer la faisabilité de l'ordonnancement.

Une piste pour des travaux futurs est d'autoriser l'abandon d'un job chaque fois que la probabilité p_i associé à sa tâche τ_i est satisfaite.

Enfin, avec Dorin Maxim, Lucas Santinelli et Robert I. Davis, nous avons proposé dans (Maxim *et al.*, 2011) des algorithmes pour résoudre des problèmes d'ordonnancement temps-réel *uniprocesseur, préemptif, et à durées d'exécutions stochastiques* sous différentes hypothèses. En particulier, nous démontrons que des algorithmes gourmands permettent de satisfaire ou de minimiser des taux d'échec, mais qu'un tel algorithme ne peut minimiser le pire taux d'échec. Pour trouver des priorités fixes minimisant le pire taux d'échec, nous proposons un algorithme de recherche en profondeur.

51. Une tâche est préemptive si elle peut être interrompue et relancée.

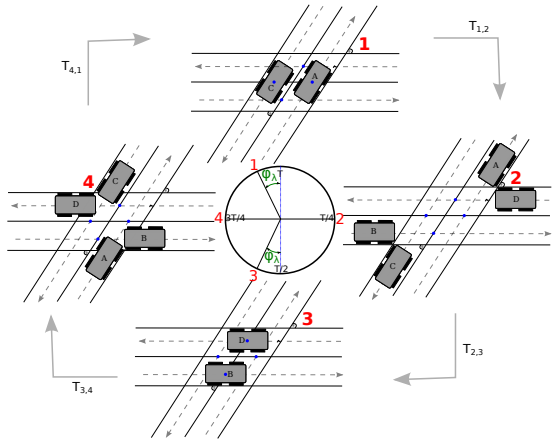
5.1.3 Gestion du trafic routier (projet InTraDE)

ICTAI'12	(Tlig *et al.*, 2012)
AATMO'13	(Tlig *et al.*, 2013a)
JFSMA'13 (démo)	(Tlig *et al.*, 2013b)
RJCIA'13	(Tlig *et al.*, 2013c)
ICALT/ITS'14	(Tlig *et al.*, 2014b)
ECAI/PAIS'14	(Tlig *et al.*, 2014a)
RIA	(Tlig *et al.*, 2016)
INRIA	(Tlig *et al.*, 2014c)

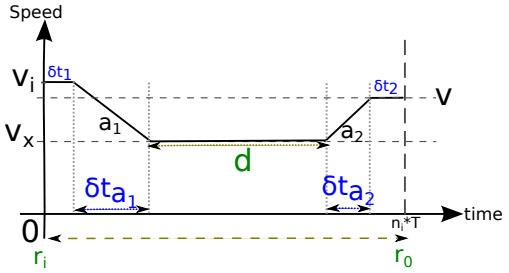
Le projet européen InTraDE[52] (InterReg IV-B) portait sur le développement d'un véhicule guidé automatiquement (AGV) pour le transport de conteneurs à l'intérieur des ports maritimes. Dans ce contexte, la thèse de Mohamed Tlig, co-encadrée avec Olivier Simonin, s'intéressait à la gestion du trafic de tels véhicules automatisés. Éviter les congestions et optimiser un tel trafic sont en effet des difficultés majeures à aborder non seulement dans un tel environnement portuaire, mais aussi au sein de sites industriels ou en milieu urbain.

Après un premier travail sur des stratégies de partage d'une voie par deux files de véhicules autonomes (par exemple dans l'hypothèse où un véhicule en panne bloque une autre voie) (Tlig *et al.*, 2012, 2013a), nous avons approfondi l'idée d'éviter aux véhicules de s'arrêter aux intersections d'un réseau routier. Cette idée est issue des travaux de Albouze (2010), lequel proposait un mécanisme pour alterner le passage de deux flux de véhicules à une intersection.

Nous avons d'abord déterminé les périodes de passage optimales pour deux flux de véhicules identiques traversant des intersections à vitesse constante, quel que soit l'angle formé par les deux voies considérées. Ensuite ces calculs ont été étendus au cas d'une intersection de deux routes à deux voies (opposées) chacune (*cf.* figure 5.2a). La résolution d'une équation du second degré fournit alors, si une solution existe, un profil de vitesse simple pour que les véhicules arrivent à l'entrée des intersections au bon moment et à la bonne vitesse (*cf.* figure 5.2b).



(a) Quatre situations clefs pour déterminer la période et les instants de passage

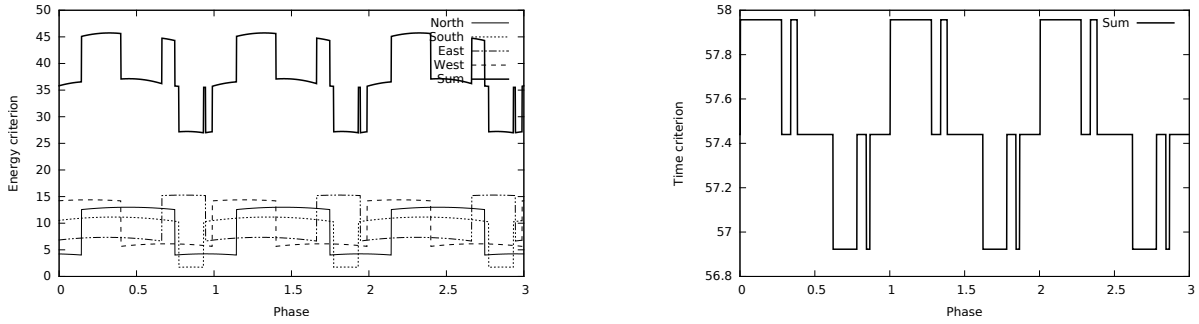


(b) Profil de vitesse type pour qu'un véhicule arrive à l'entrée d'une intersection à l'instant et à la vitesse voulus

FIGURE 5.2 – Synchronisation de véhicules pour effectuer des croisements alternés

Équipés de ces premiers outils permettant de synchroniser au mieux des véhicules à une intersection, nous avons ensuite proposé une approche par optimisation distribuée pour synchroniser des intersections entre elles de manière à minimiser globalement : soit la consommation énergétique, soit le temps de traversée. Nos travaux ont permis de démontrer entre autres (i) que l'optimisation des critères choisis pouvait être distribuée sans perte d'optimalité, et (ii) que, au vu des profils locaux des critères aux intersections (en créneaux, *cf.* figure 5.3), il valait mieux préférer une recherche assez large dans l'intervalle des phases considéré. En outre, une étude expérimentale en simulation a permis de mettre en avant que l'approche « alterner les véhicules » était plus adaptée que l'approche « premier arrivé, premier servi » si les flux sont relativement denses.

52. Intelligent Transportation for Dynamic Environment `http://www.intrade-nwe.eu/`



a b

FIGURE 5.3 – Comparaison entre les courbes d'optimisation obtenues pour le critère énergie (a) et pour le critère temps (b) dans les mêmes configurations

5.2 Sur les systèmes de recommandation

5.2.1 Filtrage collaboratif à base de préférences

ICTAI'10	(Brun *et al.*, 2010a)
PL'10	(Brun *et al.*, 2010c)
RFIA'10	(Brun *et al.*, 2010b)

Les systèmes de recommandation ont pour but de recommander des items (des films, des articles, ...) à des utilisateurs. Les méthodes de filtrage collaboratif permettent de réaliser de tels systèmes. Dans ces méthodes, des données liées aux goûts de multiples utilisateurs permettent d'estimer les goûts d'une personne.

Un exemple type est l'emploi de notes (par exemple entre 1 et 5) données à des items, notes qui peuvent servir à déterminer :

— si deux utilisateurs ont des goûts proches parce qu'ils donnent des notes similaires ;
— si deux items sont proches parce que les mêmes utilisateurs les apprécient ; ou
— quelle note un utilisateur x donnerait à un item i au vu des notes données par des utilisateurs proches sur le même item (ou des items proches).

Or l'emploi de notes (c'est-à-dire de préférences quantitatives) pose problème parce qu'il suppose que les différents utilisateurs partagent et maintiennent une échelle de valeur absolue commune. En pratique, certains utilisateurs risquent de noter les films plus durement que d'autres, et cette dureté de notation peut évoluer avec le temps.

Partant de ce constat, nous avons décidé avec Anne Boyer, Armelle Brun et Ahmad Hamad d'étudier la possibilité d'utiliser non pas des notes, mais des comparaisons entre items. L'utilisateur a ainsi non pas à évaluer chaque item indépendamment dans l'absolu, mais à le comparer à d'autres items. Cela nous a amené à travailler avec des relations de préférence, à proposer un calcul de similarité entre utilisateurs dans ce cadre, ainsi qu'une méthode de recommandation. Des évaluations empiriques ont été conduites comparant une recommandation reposant sur des notes à une recommandation reposant sur les préférences dérivées de ces notes. Les résultats obtenus étaient prometteurs en ce qu'ils ne montraient pas de dégradation significative de la qualité des prédictions effectuées.

5.3 Discussion

Comme dit en introduction, ces travaux qui sortent du cadre des *MDP et de la planification automatique touchent quand même au contrôle de systèmes dynamiques ou à la théorie de l'utilité.

Dans les trois premiers, il s'est agit de proposer ou d'étudier des algorithmes simples, typiquement des algorithmes de recherche arborescente ou locale, pour contrôler au mieux un système dynamique assez spécifique, souvent complexe (mettant en œuvre des entités en interaction). On reste donc dans la même famille d'outils algorithmiques, avec d'ailleurs à la fois des algorithmes hors-ligne — pré-calculant une

solution qui n'a plus qu'à être exécutée — et des algorithmes en-ligne. Ce sont les propriétés particulières des systèmes considérés qui vont conduire à des solutions et des analyses spécifiques, par exemple les interactions entre cycles de plusieurs fourmis, ou les interactions entre jobs à différents niveaux de priorité.

Dans le quatrième travail, nous cherchons une alternative qualitative à un système d'utilités (quantitatives). Il serait intéressant de regarder s'il y a des liens entre ceci et la théorie de la décision qualitative, par exemple de groupe ou sous incertitude. Pour rappel (voir section 2.3.4.2), des pendants qualitatifs des MDP ont déjà été proposés et étudiés.

Chapitre 6

Mes travaux futurs (projet de recherches)

Sommaire

"I am awaiting the day when people remember the fact that discovery does not work by deciding what you want and then discovering it." – David Mermin

Ce chapitre vise à présenter un certain nombre de travaux futurs possibles — des pistes à explorer et des questions à creuser — dans une certaine continuité avec mes travaux passés et présents. Comme « la prévision est difficile, surtout lorsqu'elle concerne l'avenir » (Pierre Dac), il ne s'agit que de projets de recherches, pas d'un programme (qui devrait plutôt être un plan conditionnel). Par ailleurs, ces projets évolueront vraisemblablement au gré des opportunités, des applications, des rencontres, des échanges...

6.1 Autour des états d'occupation

Les travaux conduits avec Jilles Dibangoye sur l'utilisation d'états d'occupation pour résoudre des Dec-POMDP ont permis une avancée majeure. Ils offrent aussi diverses voies de recherche très prometteuses dont celles que je vais évoquer ici. La plupart ont été envisagées avec Jilles Dibangoye dans le cadre de notre collaboration.

6.1.1 Popularisation de l'approche

Les quelques principes sur lesquels repose notre approche ne paraissent pas particulièrement complexes. Au fait de se ramener à un problème de décision séquentielle déterministe (ce qui n'est pas tout à fait nouveau), nous ajoutons en premier lieu l'utilisation d'une statistique particulière pour calculer la fonction de valeur (ou son approximation). Toutefois, il nous a semblé assez tôt qu'elle s'avérait difficile d'accès. La rédaction de l'article de revue (Dibangoye *et al.*, 2016) qui étend l'article de conférence (Dibangoye *et al.*, 2013a) nous a conduit à un important travail de réflexion pour mieux comprendre et expliquer l'approche.

Il reste toutefois encore à faire pour que plus de chercheurs et praticiens s'y intéressent et se l'approprient. Cela inclut toujours une réflexion sur la façon de voir et présenter les choses (sans chercher nécessairement à se limiter à un seul point de vue). Cela passe aussi par la mise à disposition d'un outil logiciel permettant d'utiliser le planificateur Dec-POMDP développé, mais aussi de l'étendre pour en proposer des variantes, par exemple pour traiter des sous-classes particulières.

Comme on va le voir par la suite, la transformation d'un problème en *occupancy MDP* (oMDP) est une solution pour résoudre non seulement des Dec-POMDP, mais aussi d'autres classes de problèmes de décision séquentielle. Le travail de popularisation devra donc aussi être conduit auprès d'une plus vaste population de chercheurs et praticiens.

6.1.2 Que peut-on résoudre avec l'approche oMDP ? Comment ?

Nous avons proposé une méthode M (« l'approche oMDP ») pour résoudre un problème P donné (les Dec-POMDP). De là, nous avons montré comment spécialiser cette méthode sur des sous-classes P', P", ... (telles que les ND-POMDP) en particulier en établissant des statistiques suffisantes plus petites, donc permettant un meilleur passage à l'échelle. Nous avons même proposé une méthode pour travailler sur de telles sous-classes, et il reste à faire dans ce domaine pour fournir des outils réutilisables, sans chercher pour autant à traiter tous les cas possibles.

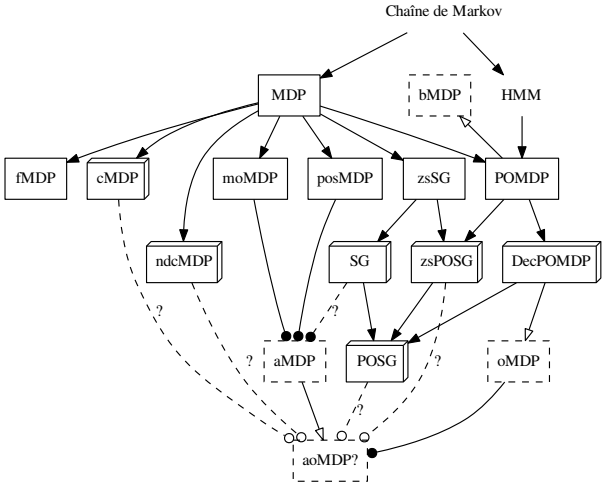
De ce point de vue, il est naturel de chercher aussi quelles « sur-classes » des Dec-POMDP peuvent être abordées par l'approche oMDP et, plus généralement, quelles autres classes de problèmes peuvent être abordées utilement par cette approche.[53] Or, comme on l'a vu à l'aide de la figure 2.6 (page 36), on peut trouver des solutions à des modèles à l'aide :

— de reformulations comme c'est le cas pour les Dec-POMDP, ou
— d'abstractions, si elles permettent de montrer que deux modèles sont analogues parce qu'abstraits par le même troisième.

On peut donc peut-être abstraire les oMDP en des « *oMDP algébriques* » (ou une autre classe abstraite, cette terminologie étant choisie ici seulement par analogie avec les aMDP, et donc abusivement).

La figure 6.1 illustre ce questionnement en étendant la figure 2.6. L'ajout principal est la classe des aoMDP suscitée, et des liens hypothétiques vers les classes des aMDP et des aoMDP. Concernant les jeux stochastiques simples, sachant qu'on peut leur appliquer des algorithmes de type programmation dynamique, une première question est de savoir s'ils entrent aussi dans le cadre aMDP, ou peut-être dans un cadre plus abstrait. On pourrait ainsi les voir comme une autre forme de problème multi-critère compatible avec la propriété de Bellman. On notera par ailleurs que les classes répertoriées ici comme ne permettant pas d'appliquer directement la propriété de Bellman[54] subissent des contraintes (du fait par exemple de critères non classiques ou de la présence d'agents indépendants) telles qu'on ne peut pas considérer plusieurs situations possibles à un instant t, issues d'exécutions différentes, indépendamment. Il semble donc nécessaire de raisonner sur plusieurs situations en même temps en faisant usage d'états d'occupations (ou analogue).

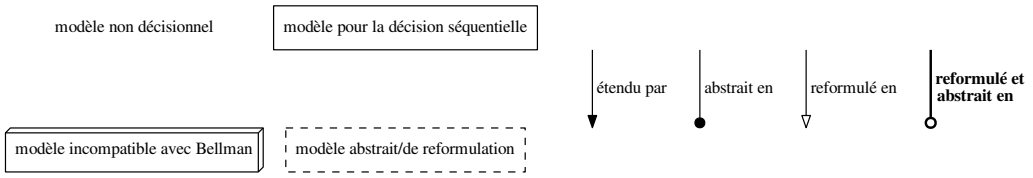
53. On peut résoudre des MDP ou POMDP aussi, mais vraisemblablement sans gain computationnel.
54. Nous n'avons pas identifié d'autres classes satisfaisant cette propriété.



(a) Des modèles markoviens et leurs relations

HMM	modèle de Markov caché
MDP	processus de décision markovien
POMDP	MDP partiellement observable
fMDP	MDP factorisé
posMDP	MDP possibiliste
moMDP	MDP multi-objectif (but : recherche du front de Pareto)
SG	jeu stochastique
ndcMDP	MDP non dynamiquement cohérent
cMDP	MDP contraint
Dec-POMDP	POMDP décentralisé
POSG	jeu stochastique part. obs.
zs-POSG	POSG à 2 joueurs et somme nulle
bMDP	belief MDP
oMDP	occupancy MDP
aMDP	MDP algébrique
aoMDP	**oMDP algébrique**

(b) Les acronymes utilisés ci-contre



(c) Légende du diagramme ci-dessus (les 4 types de modèles et 3+**1** types de relations rencontrés)

FIGURE 6.1 – Divers modèles markoviens présentés avec leurs relations. On ajoute ici un modèle (les aoMDP) hybridant les oMDP et les aMDP, et d'hypothétiques liens vers celui-ci.

6.1.2.1 Jeux

On obtient une surclasse des Dec-POMDP, celle des jeux stochastiques partiellement observables (POSG), en relâchant la contrainte que tous les agents partagent une fonction de récompense commune. [55] La définition d'un état d'occupation donnée dans le cadre Dec-POMDP reste valable. On peut donc se demander si « l'approche » oMDP est adaptable. Nous avons déjà conduit des travaux préliminaires sur ce sujet, dont le fruit est présenté ci-dessous.

Concepts solution Une première difficulté est de spécifier plus précisément le (ou les) problème(s) posé(s). En effet, plusieurs concepts de solution sont envisageables, et il faut trouver lesquels donnent accès à des solutions algorithmiques. Nos premières réflexions se sont tournées vers les équilibres de Nash parfaits en sous-jeux, c'est-à-dire des solutions à des problèmes séquentiels dans lesquelles, à chaque étape du jeu, et chaque joueur ayant connaissance des stratégies des autres joueurs, aucun d'entre eux n'a pas intérêt à modifier unilatéralement sa propre stratégie (Fudenberg et Tirole, 1991).

Jeux stochastiques complètement observables À notre connaissance, il n'existe pas d'algorithmes de recherche heuristique « meilleur d'abord » pour jeux stochastiques (complètement observables). [56] Cela soulève donc une première question : Comment guider une telle recherche ? Quel est l'équivalent du comportement optimiste d'un agent seul ?

De premiers résultats préliminaires (théoriques) semblent indiquer que, dans le cadre 2 joueurs, somme nulle, une approche valide est d'utiliser un majorant et un minorant de la fonction de valeur, et que chaque joueur agisse de manière optimiste. Plus précisément, un joueur (resp. l'autre joueur) doit choisir une

55. Dans certains travaux antérieurs, les Dec-POMDP sont vus comme des POSG particuliers pour lesquels des solutions particulières sont proposées (Hansen *et al.*, 2004; Emery-Montemerlo *et al.*, 2004).

56. Mais des recherches en profondeur d'abord ont été étudiées, par exemple par Saffidine *et al.* (2012).

stratégie individuelle à l'équilibre par rapport au majorant (resp. du minorant). Cela amènerait à réviser les minorants et majorants jusqu'à ce que les deux joueurs tombent sur un équilibre commun avec des valeurs exactes.

Jeux à deux joueurs et à somme nulle Avant d'aborder le cas le plus général, j'évoquerai le cas des jeux à deux joueurs et à somme nulle, le gain de l'un étant alors la perte de l'autre. Cette situation d'opposition parfaite est particulière parce que la performance d'un agent-joueur est l'exact opposé de la performance de l'autre. Cela permet de ne considérer qu'un seul critère de performance (qu'on notera encore V), mais en sachant que l'un va chercher à la maximiser pendant que l'autre tentera de la minimiser.

Mais cette fonction de valeur vérifie-t-elle toujours la propriété PWLC (linéarité par morceaux et convexité) dans l'espace des états d'occupation ? Intuitivement, la convexité était liée — pour les Dec-POMDP — au fait que les gains futurs espérés les plus importants correspondent à des situations parmi les plus certaines (état et historiques jointes) du point de vue du planificateur. Ceci n'est plus nécessairement vrai dans un jeu à deux joueurs, à moins d'adopter le point de vue d'un joueur. Par contre, nous avons remarqué que la fonction de valeur est lipschitzienne, ce qui permet de la minorer et la majorer aisément avec des approximateurs en dents de scie, et au besoin d'utiliser des algorithmes d'optimisation optimiste (voir par exemple les travaux de (Munos, 2011, 2014)).

Sans aller plus loin dans cette direction, se pose donc la question de la forme de la fonction de valeur : Outre le fait d'être lipschitzienne, est-elle particulière? Permet-elle des calculs plus efficaces que des évaluations locales (en chaque état d'occupation considéré) ? Dans la même veine, si l'on veut adopter des schémas algorithmiques comparables à fb-HSVI, comment peut-on encadrer cette fonction de valeur ? Quelles relaxations résoudre pour obtenir des minorants ou majorants initiaux ? Comment mettre à jour ces minorants et majorants ? Comment guider une recherche heuristique ? (voir cas complètement observable) Comment prouver la convergence ? Se fait-elle en temps fini ? A ϵ près ?

De premières réponses apparaissent à certaines questions. Une première idée pour obtenir minorants ou majorants initiaux par relaxation est de considérer un joueur (ou une équipe collaborant au même objectif) à la fois. Pour calculer un majorant, on peut alors (i) relâcher des contraintes pour ce joueur, par exemple en lui donnant une observabilité complète, ou (ii) ajouter des contraintes aux autres, par exemple en les rendant aveugles. Toutefois, il faut vérifier que modifier les valeurs d'un jeu à l'avantage d'un joueur ne peut que lui apporter de meilleurs résultats. De la même manière, il faut trouver une méthode de mise à jour de ces minorants et majorants (1) qui préserve leurs qualités de minorants et majorants, et (2) qui assure d'une convergence à terme.

Jeux à somme générale En abordant les jeux à somme générale, on passe d'une fonction objectif unique à plusieurs fonctions objectif. Peut-on alors, en particulier avec le concept de solution choisi précédemment, définir une fonction de valeur par joueur ? Les questions posées précédemment restent pertinentes concernant les formes de ces fonctions, la possibilité de les encadrer, de les mettre à jour, de mettre en œuvre une recherche heuristique, ...

On notera que les réponses seront différentes. Entre autres, si on considère un joueur i, rendre les autres joueurs clairvoyants peut être aussi bien bénéfique que mauvais pour i. Des travaux récents relatifs aux jeux stochastiques (à information complète) pourraient nous guider pour répondre à ces questions (MacDermed et Isbell, 2013; Murray et Gordon, 2006, 2007).

6.1.2.2 MDP contraints

Hormi les classes de problèmes directement apparentées aux Dec-POMDP, une façon de chercher où l'approche oMDP peut servir est d'identifier des problèmes de décision séquentiels dans lesquels le principe d'optimalité de Bellman n'a pas pu être mis en œuvre jusque là. C'est le cas des *MDP contraints* (Altman, 1999), c'est-à-dire des MDP dans lesquels, à la fonction de coût c dont on dérive un critère de performance à minimiser,[57] s'ajoutent K autres fonctions de coûts $c^{(1)}, \ldots, c^{(K)}$ dont on va dériver K contraintes à satisfaire. À horizon H, avec facteur d'atténuation, il faudrait ainsi trouver une politique π

57. Ici je parle de coût plutôt que de récompense pour respecter une notation plus commune à ce cadre particulier.

minimisant

$$E\left[\sum_{t=0}^{H}\gamma^t c_t\right]$$

en respectant, pour tout $k \in \{1,\ldots,K\}$, la contrainte

$$E\left[\sum_{t=0}^{H}\gamma^t c_t^{(k)}\right] \leq M^{(k)}.$$

On notera que les politiques optimales ne seront pas les mêmes si l'on change d'état initial. En effet, si la meilleure action (au sens du critère à minimiser) dans l'état s est une action coûteuse pour un critère k, alors elle n'est peut-être pas acceptable si s est l'état initial, alors qu'elle serait acceptable si s ne risque d'être rencontré que tardivement. Par ailleurs, les politiques optimales peuvent être stochastiques (voir figure 6.2) ou non stationnaires.

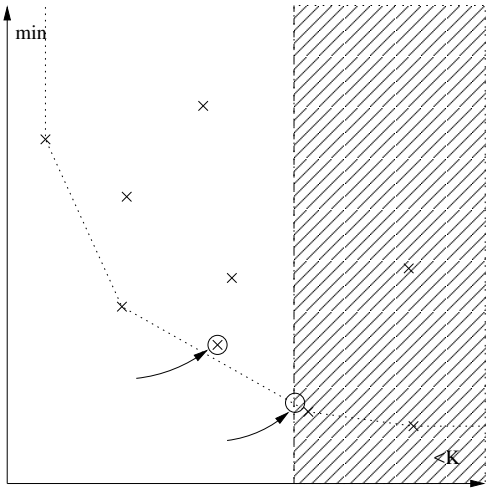


FIGURE 6.2 – Exemple d'espace de solutions d'un problème d'optimisation sous une contrainte ; chaque solution « pure » (déterministe) est représentée par un point dont les coordonnées sont le critère contrainte (en abscisse) et le critère à optimiser (en ordonnée) ; la meilleure solution « mixte » (stochastique) est à l'intersection entre le front de Pareto des différents critères et la frontière des solutions faisables.

Ce qui empêche ici l'application de la programmation dynamique classique est que les politiques futures possibles depuis l'état courant dépendent non seulement des coûts futurs et passés, mais aussi des coûts des autres chemins qui auraient pu être suivis. L'approche oMDP semblerait une solution possible raisonnant sur tous les états atteignables à un instant t étant donnée la politique suivie jusque là (l'espace des états d'occupation serait alors analogue à un espace des états de croyance). Mais, comme précédemment, se pose la question de savoir si la fonction de valeur optimale a une forme particulière exploitable. Pour toute politique $\pi_{t:H}$, tout critère (optimisation ou contrainte) a une valeur linéaire dans l'espace des occupations. S'il n'y avait pas de contraintes, la fonction de valeur optimale respecterait la propriété PWLC. Mais les politiques acceptables dépendent des coûts passés et de l'état d'occupation. Plus précisément, à coûts passés fixes, les contraintes réduisent la zone d'applicabilité d'une politique donnée à un polyèdre convexe dans l'espace des états d'occupation. Toujours à coûts passés fixes, l'espace des états d'occupation se découpe en une partition de polyèdres, à chacun correspondant un sous-ensemble de politiques futures applicables. La propriété PWLC n'est donc applicable que sur chaque élément de cette partition. En revanche, les différentes fonctions sont peut-être lipschitziennes, et la même piste évoquée pour les jeux stochastiques — une approximation en dents de scie et de l'optimisation optimiste — est à considérer.

Aussi, dans la perspective d'un algorithme similaire à fb-HSVI, comment une telle recherche heuristique peut-elle intégrer des contraintes sur les solutions faisables ? La question se poserait d'ailleurs déjà pour A* ou LRTA*. Dans une telle situation plus simple, en un état donné, les actions possibles sont telles qu'aucune contrainte n'est violée par les minorants des coûts totaux pour ces actions — minorants obtenus comme la somme (i) des coûts cumulés passés, (ii) des coûts immédiats et (iii) des minorants connus sur les coûts cumulés futurs. Si aucune action n'est éligible, il faut invalider l'état courant, mettre à jour ses minorants, et re-considérer ses propres parents. On voit ainsi que, du fait des contraintes, les recherches heuristiques vont devoir effectuer des retours arrières pour revoir les meilleurs chemins possibles.

6.1.3 Approximations

Comme évoqué en section 4.3.3.2, de premiers travaux ont été menés permettant d'introduire des approximations tout en contrôlant les erreurs faites. Ces approximations pouvaient être introduites à différents points de l'algorithme fb-HSVI (et *a priori* dans des points équivalents d'algorithmes à base de points comparables). Le même raisonnement devrait être applicable à d'autres cas que les Dec-POMDP.

Du travail reste à faire pour mieux comprendre l'effet des différentes approximations possibles, ainsi que leurs relations, de manière à mieux savoir contrôler les approximations faites en fonction de la structure du problème.

Par ailleurs, la plupart des étapes impliquant des approximations permettent non seulement de limiter les erreurs faites, mais aussi d'estimer les erreurs effectivement faites, et ces erreurs observées *a posteriori* peuvent être nettement plus faibles que les erreurs requises *a posteriori*.[58] De tels retours en cours d'exécution permettraient donc de réviser les erreurs demandées à chaque étape au lieu de n'utiliser que des limites pré-calculées au départ.

Enfin, on pourrait raisonner sur des modèles (dynamique et observabilité) approchés. Prenons par exemple le cas d'un ND-POMDP tel que certaines dépendances entre voisins pourraient être plus faibles que d'autres. Dans ce cas, il serait intéressant de considérer une approximation de ce ND-POMDP dans laquelle les dépendances les plus faibles sont éliminées, de sorte que les dimensionalités des sous-espaces à manipuler seront plus faibles. En sachant mesurer la distance entre modèle original et modèle approché, on peut envisager de majorer l'erreur faite en raisonnant sur ce modèle approché. Un des problèmes ouverts est évidemment le moyen d'approcher un modèle par un autre. Ici, cela reviendrait à un problème de réduction de dimensionalité (avec perte d'information) en tenant compte à la fois de la dynamique et de l'observabilité.

6.1.4 D'autres méthodes de compression de l'état d'occupation ?

Un des ingrédients clef de la réussite de fb-HSVI pour les Dec-POMDP est l'utilisation d'une représentation d'ensembles d'historiques *probabilistiquement équivalents* (PE) qui s'avère compacte dans la plupart des problèmes rencontrés.[59] Il s'est en effet avéré que, très souvent, raisonner sur une historique d'actions et d'observations courte suffisait pour prendre, à coup sûr, des décisions optimales. Cela se comprend par exemple dans le problème du tigre multi-agent, dans lequel on peut considérer que la mémoire est réinitialisée de manière régulière parce que l'un des deux agents au moins à dû ouvrir la porte. Il est toutefois assez aisé de construire des problèmes dans lesquels une mémoire tronquée n'est pas utilisable. Il suffit pour cela que la meilleure décision dépende d'un événement qui a pu se passer arbitrairement loin dans le passé.

Mais la troncature n'est qu'une façon simple de trouver un dénominateur commun à un ensemble d'historiques PE. Il faut donc envisager d'autres méthodes pour discriminer ces ensembles d'historiques PE les uns des autres. On peut voir là un problème de classification — multi-classe — de séquences (Xing *et al.*, 2010), ou plutôt d'inférence grammaticale (de la Higuera, 2005). Mais vient s'ajouter la difficulté

58. D'après nos expérimentations, les erreurs totales estimées *a posteriori* sont souvent très faibles comparées aux erreurs attendues *a priori*.

59. Deux historiques individuelles sont *probabilistiquement équivalentes* si, suivant une politique donnée après l'une ou l'autre historique, le système évoluera de la même façon.

qu'on va chercher à opérer de manière incrémentale, en reprenant les résultats à l'étape t pour obtenir ceux de l'étape $t+1$.

6.1.5 D'autres questions

D'autres questions se posent dans le cadre de oMDP, telles que :
— Quels algorithmes employer à des étapes telles que l'optimisation, en un état d'occupation, de la règle de décision à appliquer ? Pour les Dec-POMDP, y a-t-il mieux à faire qu'employer un solveur de wCSP ?
— Quels schémas algorithmiques (outre celui de fb-HSVI) seraient applicables ? À quelles conditions ?
— Dans certains jeux, comment s'assurer que chacun partage honnêtement sa fonction d'utilité avec les autres ?
— Comment aborder les scénarios multi-agents dans lesquels les agents n'ont pas d'horloge commune, donc ne peuvent pas se synchroniser de manière sûre ?

6.2 ρ-POMDP

La recherche active d'information modélisée par des ρ-POMDP s'avère être un problème assez proche des POMDP classiques. Il y a toutefois quelques difficultés spécifiques qui mériteraient des recherches propres.

6.2.1 Heuristiques

Une première question particulière est de savoir comment initialiser les heuristiques d'algorithmes de résolution de ρ-POMDP.

Pour initialiser un minorant dans un cadre POMDP, il est courant d'employer une politique aveugle (Hauskrecht, 1997; Smith, 2007), laquelle suppose que la même action (la meilleure possible) est répétée. Cela ne requiert que l'évaluation, peu coûteuse, d'une telle politique, laquelle donne une fonction linéaire (un hyperplan) par « action-politique ». Une fois cette évaluation fait pour chaque action-politique, l'état de croyance courant permet de déterminer laquelle est la meilleure, et donc quelle valeur heuristique adopter.

La même approche pourrait être envisagée pour les ρ-POMDP à récompense PWLC (ou en prenant toute approximation PWLC, puisqu'elle sera pessimiste). Toutefois la fonction de valeur est alors elle-même PWLC, avec un nombre d'hyperplans qui va croître rapidement, et peut-être pas de possibilités d'élagage (question ouverte). Peut-être est-il néanmoins possible, en calculant les fonctions de valeurs de différentes politiques simultanément, de les comparer au fur et à mesure pour élaguer des hyperplans manifestement inutiles.

Pour initialiser un majorant dans un cadre POMDP, il est courant d'employer une heuristique clairvoyante (en résolvant le MDP associé) ou le « majorant vite informé » (*Fast Informed Bound*) (Hauskrecht, 1997; Smith, 2007). Or être clairvoyant, c'est-à-dire connaître l'état, dévoilerait toute l'information immédiatement, ce qui est sans intérêt. Peut-on quand même s'inspirer du majorant FIB ? On notera que la fonction de récompense convexe devra être approchée par une enveloppe inférieure de points (ou une fonction en dents de scie).

6.2.2 Approximations de V^* et ρ

Les principaux résultats théoriques que nous avons établis montrent que, avec une approximation PWLC — à erreur bornée — de la fonction de récompense ρ, l'erreur faite par un algorithme à base de points tel que PBVI est aussi bornée, et dépend de la densité de l'ensemble de points utilisé. On n'est toutefois pas contraint d'utiliser une approximation PWLC pré-calculée de ρ de cette seule manière. Si l'on considère par exemple un algorithme tel que HSVI, qui génère des trajectoires et minore autant qu'il majore V^*, les deux options suivantes sont envisageables :

— au lieu de pré-calculer une approximation PWLC de ρ, calculer des valeurs ou hyperplans locaux à la demande si ça n'est pas trop coûteux ;
— pré-calculer de premières approximations PWLC (minorant) et en-dents-de-scie (majorant) — si possible en estimant leurs erreurs — de ρ pour lancer une première fois HSVI jusqu'à atteindre un certain écart (*gap*) fonction de l'erreur sur les approximations de ρ, puis recommencer avec des approximations plus fines de ρ.

Pour une fonction de récompense α-Holderienne dont on connaîtrait la constante α, on peut aussi envisager d'affiner l'approximation de ρ quand l'état de croyance courant est trop éloigné des autres points supports.

6.2.3 Cas non-convexe

S'il est assez naturel de définir, dans des problèmes de collecte d'information, des fonctions de récompense convexes, on pourrait aussi vouloir utiliser des fonctions non-convexes, par exemple constantes par morceaux. On pourrait même rencontrer des problèmes dans lesquels on veut « perdre » de l'information plutôt qu'en trouver, éventuellement avec une fonction de récompense concave. À titre d'exemple, un joueur de cartes pourrait chercher combien de fois battre son jeu pour bien le mélanger sans perdre trop de temps pour autant.

Que peut-on dire de la fonction de valeur dans ces cas ? Doit-on abandonner les algorithmes exploitant la propriété PWLC ? Des évaluations heuristiques (minorants ou majorants) sont-elles encore possibles, ne serait-ce que ponctuellement ?

Si la fonction de récompense est lipschitzienne, la même piste qu'évoqué précédemment pour les POSG est à creuser, avec l'approximation en dents de scie du minorant comme du majorant. Si la fonction de récompense est non-lipschitzienne, une approximation lipschitzienne est envisageable avec éventuellement des bornes d'erreur.

6.3 Algorithmes MCTS

Jusqu'ici mes travaux de recherche ont peu abordé les algorithmes MCTS (*Monte-Carlo Tree Search*, section 2.2.3.2), à part sur le jeu du démineur (section 4.5.5) et dans le cadre de projets étudiants. Ces algorithmes offrent certaines possibilités pour répondre à des difficultés rencontrées avec des algorithmes « plus classiques », par exemple pour traiter des POMDP à grands espaces d'état, d'action ou d'observation. Ceci m'amène donc à envisager certaines pistes de recherche sur ces algorithmes.

6.3.1 ... pour ρ-POMDP

Nos travaux sur les ρ-POMDP nous ont amené à voir (i) comment adapter certains algorithmes à base de points exploitant la propriété PWLC, et (ii) que d'autres algorithmes, comme AEMS2, pouvaient fonctionner sans modification (sans approximation PWLC de ρ) du moment qu'on sait évaluer rapidement la fonction de récompense. Le principal algorithme MCTS pour POMDP, POMCP (Silver et Veness, 2010), pose pour sa part de nouvelles difficultés, alors qu'un algorithme de cette famille semblerait pertinent pour gérer des problèmes de grande taille. En effet l'état de croyance n'y est jamais calculé exactement dans les nœuds de l'arbre construit, et n'est estimé que progressivement par un ensemble d'états (qu'on appellera « particules » par analogie avec les filtres particulaires).

Se pose donc la question de comment adapter le principe des algorithmes MCTS aux ρ-POMDP. Si l'on veut éviter les calculs exacts d'états de croyance (ce qui est souvent pertinent, en particulier si les états sont nombreux) l'estimation de la récompense en un point d'une trajectoire générée va nécessiter de produire dès le départ des ensembles de particules. Dans le cas d'une récompense liée à un calcul d'entropie, on pourra employer des méthodes d'estimation telles que décrites par Beirlant *et al.* (1997). [60]

60. Voir aussi `https://en.wikipedia.org/wiki/Entropy_estimation`

6.3.2 ... pour POMDP continus

Il y a déjà eu des travaux sur la résolution de POMDP continus ou de grande taille, que l'on considère les états, les actions ou les observations (Hoey et Poupart, 2005; Porta *et al.*, 2006; Erez et Smart, 2010; Zamani *et al.*, 2012; Brechtel *et al.*, 2013). Concernant plus particulièrement les approches MCTS, à ma connaissance, seuls ont été abordés les grands espaces d'états et d'action, et ce dans un contexte complètement observable. Les techniques d'élargissement progressif (*progressive widening*) de l'arbre construit offrent par exemple des solutions dans le cas complètement observable (Couetoux *et al.*, 2011; Couetoux, 2013).[61] Celles-ci reposent sur l'idée de n'ajouter des nouveaux fils à un nœud qu'à un certain rythme (de plus en plus lent).

Peut-on adapter ce même principe de l'élargissement progressif au cas partiellement observable, par exemple au sein de POMCP ? Or, alors que POMCP est guidé par des simulations de la dynamique du système (et qu'il stocke les états utilisés comme des particules pour estimer l'état de croyance en chaque nœud), les schémas reposant sur l'élargissement progressif vont souvent choisir des branches en fonction de l'arbre courant, lequel repose sur des observations. Il faudrait donc « réconcilier » ces deux points de vue, peut-être à nouveau en générant à chaque trajectoire un ensemble d'états-particules.

6.3.3 ... et discrétisation du temps

Par ailleurs, dans un certain nombre de scénarios, un choix de modélisation typique est celui de la discrétisation du temps. Il faut trouver un compromis entre un pas de temps court pour une plus grande finesse, ou long pour réduire l'explosion combinatoire des futurs possibles à horizon temporel (absolu) donné. Mais ne pourrait-on pas travailler avec plusieurs discrétisations ? Évidemment, une hypothèse préalable est de disposer d'un modèle (surtout la dynamique du système) qui s'y prête.

À supposer cette hypothèse satisfaite, une première idée, envisageable pour les algorithmes en ligne en général, est de relancer le même algorithme plusieurs fois avec des pas de temps de plus en plus petits. Toutefois, dans le cas d'une approche MCTS, si on repart à chaque fois d'un arbre vide, quelle garantie a-t-on que la dernière itération effectuée fournira une meilleure décision que les précédentes ? Combien de temps faut-il laisser à l'itération $n+1$ pour qu'elle fasse mieux (au moins avec une forte probabilité) que l'itération n ? De manière très liée, de quelle façon le pas de temps doit-il évoluer ?

Ceci-dit, plutôt que de repartir d'un arbre vide, ne pourrait-on pas continuer sur le même arbre, lequel contiendrait simplement des branches associées à différents pas de temps ? L'algorithme qui le fait croître pourrait alors, dans un nœud décision, choisir entre suivre une branche existante (quelle que soit la durée associée) et ajouter une branche avec le pas de temps courant.

6.3.4 D'autres questions

De nombreuses autres questions (nouvelles ou déjà connues) sont envisageables concernant les algorithmes MCTS, telles que :

— Dans un cadre POMDP, peut-on exploiter la propriété PWLC ? Ou n'est-ce qu'une perte de temps ?
— Dans le cas d'actions continues, comment conduire l'exploration ?
— Comment résoudre aussi des Dec-POMDP ou des jeux ? (Cazenave, 2015)
— Comment apprendre un contrôleur pour guider MCTS ?
— Peut-on généraliser encore plus le schéma algorithmique de MCTS ?

6.4 Bien d'autres sujets

J'ai présenté dans ce chapitre quelques voies assez précises que j'envisage de poursuivre à court et moyen terme.

Deux directions secondaires qui pourraient être pertinentes au vu de mes travaux et intérêts récents, mais que je préfère ne pas développer ici sont :

61. Voir aussi (Bubeck *et al.*, 2009; Van den Broeck et Driessens, 2011; Weinstein et Littman, 2012)

— la planification par recherche heuristique, pour laquelle il est toujours utile de mieux comprendre et améliorer les recherches heuristiques, ainsi que de chercher des moyens de dériver des heuristiques efficaces ; et
— l'utilisation de méthodes d'apprentissage (par renforcement ou supervisé, éventuellement profond) pour mieux guider des recherches heuristiques comme des algorithmes MCTS.

Au delà de ces pistes, comme l'état de l'art de ce mémoire (chapitre 2) essaye de le mettre en avant, les problèmes à aborder dans le vaste domaine de la planification automatique se distinguent selon diverses dimensions dont :

— le déterminisme ou l'incertitude (par exemple dans la dynamique du système),
— l'observabilité partielle ou complète,
— la connaissance ou non du modèle,
— la présence d'un ou plusieurs agents,
— le modèle énumératif ou factorisé,
— l'objectif (atteindre un but, minimiser un coût total, ...),
— la possibilité de faire des calculs hors ligne ou en ligne, et
— la possibilité d'apprendre pour planifier, c'est-à-dire d'analyser (automatiquement) un certains nombre de problèmes d'une famille pour améliorer (toujours automatiquement) un planificateur dédié à cette famille de problèmes.

Par ailleurs, les outils employés pour aborder ces différentes situations sont multiples, qu'il s'agisse

— d'algorithmes d'optimisation exploitant la propriété de Bellman (programmation dynamique ou recherche heuristique) ou pas (programmation linéaire, descente de gradient, optimisation sous contraintes),
— de raisonnement bayésien, de méthodes de Monte-Carlo, d'apprentissage automatique, ou
— d'algèbre ou d'analyse, voire de géométrie.

Il y a donc bien de quoi faire, et j'aurais bien du mal à prédire quelles directions mes travaux vont suivre à long terme. Mais il ne s'agit pas que de prendre des ingrédients existants pour tenter de nouveaux mélanges et pour compliquer des recettes. Il s'agit aussi de prendre du recul pour se demander s'il n'y a pas des ingrédients, ou des façons de les traiter, qui nous auraient échappé. Je pense ici par exemple à des travaux récents sur les défauts envisageables d'un agent rationnel ou, en d'autres termes, sur les erreurs de conception qui pourraient amener un agent à un comportement inattendu non souhaité, et sur la façon de les éviter (Orseau et Armstrong, 2016; Everitt *et al.*, 2016). Une autre tâche importante est de prendre du recul par rapport à tous ces travaux, et éventuellement ceux d'autres domaines connexes, pour essayer d'en tirer des leçons, pour comprendre par exemple si on ne peut pas « unifier » certains travaux pour les voir sous un nouveau jour mieux éclairé. Je pense par exemple aux travaux de Stulp et Sigaud (2013a) unifiant différents algorithmes d'optimisation de politiques, de Scherrer (2014) comparant différents schémas d'itération sur les politiques approché, et à bien des tutoriels. Mais tout ceci n'est pas très nouveau, et je demande au lecteur de m'excuser pour avoir enfoncé quelques portes ouvertes.

Références

D. Aberdeen et J. Baxter. Scaling internal-state policy-gradient methods for POMDPs. Dans *Proceedings of the Nineteenth International Conference on Machine Learning (ICML'02)*, July 2002.

D. Aberdeen. *Policy-Gradient Algorithms for Partially Observable Markov Decision Processes.* Thèse de doctorat, The Australian National University, Canberra, Australia, March 2003.

D. Aberdeen, S. Thiébaux, et L. Zhang. Decision-theoretic military operations planning. Dans *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS'04)*, June 2004.

S. Albouze. Stratégie réactive pour le croisement sans arrêt de deux platoons de véhicules décentralisés. Stage master de recherche, Université Henri Poincaré, Nancy, France, février 2010.

E. Altman. *Constrained Markov Decision Processes.* Chapman and Hall/CRC, 1999.

C. Amato, G. Chowdhary, A. Geramifard, N. K. Ure, et M. J. Kochenderfer. Decentralized control of partially observable Markov decision processes. Dans *Proceedings of the Fifty-Second IEEE Conference on Decision and Control (CDC-13)*, 2013.

E. Amir et B. Engelhardt. Factored planning. Dans *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'03)*, 2003.

R. Aras et A. Dutech. An investigation into mathematical programming for finite horizon decentralized POMDPs. *Journal of Artificial Intelligence Research*, 37:329–396, 2010.

M. Asada, S. Noda, S. Tawaratsumida, et K. Hosodaal. Purposive behavior acquisition for a real robot by vision-based reinforcement learning. *Machine Learning*, 23(2–3):279–303, 1996.

J. Asmuth, L. Li, M. Littman, A. Nouri, et D. Wingate. A Bayesian sampling approach to exploration in reinforcement learning. Dans *Proceedings of the Twenty-Fifth Annual Conference on Uncertainty in Artificial Intelligence (UAI'09)*, 2009.

N. Baeuerle et U. Rieder. More risk-sensitive Markov decision processes. *Mathematics of Operations Research*, 39(1):105–120, 2014.

J. Bagnell, A. Y. Ng, et J. Schneider. Solving uncertain Markov decision problems. Technical Report CMU-RI-TR-01-25, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 2001.

J. Bagnell. *Bagnell's PhD Thesis.* Thèse de doctorat, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 2003.

L. Baird. *Reinforcement Learning Through Gradient Descent.* Thèse de doctorat, Carnegie Mellon University, Pittsburgh, PA 15213, 1999.

L. Baird et A. Moore. Gradient descent for general reinforcement learning. Dans *Advances in Neural Information Processing Systems 11 (NIPS'99)*. The MIT Press, 1999.

A. Barto, S. Bradtke, et S. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72, 1995.

J. Baxter et P. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

J. Baxter, P. Bartlett, et L. Weaver. Experiments with infinite-horizon, policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:351–381, 2001.

G. Beaumet. *Planification continue pour la conduite d'un satellite d'observation agile autonome*. Thèse de doctorat, Université de Toulouse, 2008.

J. Beirlant, E. J. Dudewicz, L. Gyorfi, et E. C. van der Meulen. Nonparametric entropy estimation : An overview. *International Journal of Mathematical and Statistical Sciences*, 6:17–39, 1997.

R. Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60 (6):503–515, 1954.

R. Bellman. A Markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957.

R. Bellman. *Adaptive Control Processes*. Princeton University Press, Princeton, New-Jersey, 1961.

D. Bernstein, S. Zilberstein, et N. Immerman. The complexity of decentralized control of Markov decision processes. Dans *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI'00)*, 2000.

D. Bernstein, R. Givan, N. Immerman, et S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

D. Bertsekas et J. Tsitsiklis. *Neurodynamic Programming*. Athena Scientific, 1996.

M. S. Boddy, J. Gohde, T. Haigh, et S. A. Harp. Course of action generation for cyber security using classical planning. Dans *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS'05)*, 2005.

P. de Boer, D. Kroese, S. Mannor, et R. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005.

B. Bonet et H. Geffner. Labeled RTDP : Improving the convergence of real-time dynamic programming. Dans *Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling (ICAPS'03)*, 2003a.

B. Bonet et H. Geffner. Faster heuristic search algorithms for planning with uncertainty and full feedback. Dans *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 1233–1238, 2003b.

C. Boutilier, R. Dearden, et M. Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 1–2(121):49–107, 2000.

C. Boutilier. Planning, learning and coordination in multiagent decision processes. Dans Y. Shoham, éditeur, *Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge (TARK '96)*, pages 195–210, March 1996.

R. Brafman et C. Domshlak. Factored planning : How, when and when not. Dans *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI'06)*, 2006.

R. Brafman et M. Tennenholtz. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research (JMLR)*, 3:213–231, 2003.

S. Brechtel, T. Gindele, et R. Dillmann. Solving continuous POMDPs : Value iteration with incremental learning of an efficient space representation. Dans *Proceedings of the Thirteenth International Conference on Machine Learning (ICML'13)*, 2013.

G. Van den Broeck et K. Driessens. Automatic discretization of actions and states in Monte-Carlo tree search. Dans *Proceedings of the ECML/PKDD 2011 Workshop on Machine Learning and Data Mining in and around Games*, pages 1–12, 2011.

R. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15, 1990.

D. Bryce, W. Cushing, et S. Kambhampati. Probabilistic planning is multi-objective ! Technical Report ASU CSE TR 07-006, Arizona State University, June 2007.

S. Bubeck, G. Stoltz, C. Szepesvári, et R. Munos. Online optimization in X-armed bandits. Dans *Advances in Neural Information Processing Systems (NIPS'09)*, pages 201–208, 2009.

A. Cassandra, M. Littman, et N. Zhang. Incremental pruning : A simple, fast, exact method for partially observable Markov decision processes. Dans *Proceedings of the thirteenth Conference on Uncertainty in Artificial Intelligence (UAI'97)*, 1997.

A. Cassandra. A survey of POMDP applications. Dans *AAAI Fall Symposium*, 1998.

L. Cattelani, C. E. Manfredotti, et E. Messina. A particle filtering approach for tracking an unknown number of objects with dynamic relations. *Journal of Math. Model. Algorithms in OR*, 13 (1):3–21, 2014.

T. Cazenave. Sequential halving applied to trees. *IEEE Transactions on Computational Intelligence and AI in Games*, 7(1):102–105, 2015.

C. Chanel, C. Lesire, et F. Teichteil-Königsbuch. A robotic execution framework for online probabilistic (re)planning. Dans *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS'14)*, 2014. URL `http://www.aaai.org/ocs/index.php/ICAPS/ICAPS14/paper/view/7928`.

G. Chaslot, S. Bakkes, I. Szita, et P. Spronck. Monte-Carlo tree search : A new framework for game AI. Dans *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*, 2008.

L. Chrisman. Reinforcement learning with perceptual aliasing : The perceptual distinctions approach. Dans *Proceedings of the National Conference on Artificial Intelligence (AAAI'92)*, 1992.

H. Chu, A. Glad, O. Simonin, F. Sempé, A. Drogoul, et F. Charpillet. Swarm approaches for the patrolling problem, information propagation vs. pheromone evaporation. Dans *Proceedings of the International Conference on Tools with Artificial Intelligence (ICTAI'07)*, pages 442–449, 2007.

A. Cimatti, M. Pistore, M. Roveri, et P. Traverso. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence*, 147(1–2):35–84, 2003. doi : http://dx.doi.org/10.1016/S0004-3702(02)00374-0.

C. Claus et C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. Dans *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI'98)*, pages 746–752, 1998.

C. A. Coello Coello, G. B. Lamont, et D. A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems, (2nd ed.)*. New York : Springer, 2007.

P.-A. Coquelin et R. Munos. Bandit algorithms for tree search. Dans *Proceedings of the International Conference on Uncertainty in Artificial Intelligence (UAI)*, Vancouver, Canada, 2007. URL `https://hal.inria.fr/inria-00150207`.

A. Couetoux. *Monte Carlo Tree Search for Continuous and Stochastic Sequential Decision Making Problems*. Thèse de doctorat, Université Paris Sud, 2013.

A. Couetoux, J.-B. Hoock, N. Sokolovska, O. Teytaud, et N. Bonnard. Continuous upper confidence trees. Dans *Proceedings of the 5th International Conference on Learning and Intelligent OptimizatioN (LION'11)*, 2011.

R. Coulom. Efficient selectivity and backup operators in Monte-Carlo tree search. Dans *Proceedings of the Fifth International Conference on Computer and Games (CG-2006)*, 2006.

W. Cushing, S. Kambhampati, Mausam, et D. S. Weld. When is temporal planning really temporal ? Dans *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI'07)*, 2007.

H. Daduna et R. Szekli. Dependencies in Markovian networks. *Advances in applied probability*, pages 226–254, 1995.

P. Dai, Mausam, D. S. Weld, et J. Goldsmith. Topological value iteration algorithms. *Journal of Artificial Intelligence Research*, 42:181–209, 2011.

A. Darwiche et M. Hopkins. Using recursive decomposition to construct elimination orders, join trees, and dtrees. Dans *Proceedings of the Sixth European Conference for Symbolic and Quantitative Approaches to Reasoning under Uncertainty (ECSQARU'01)*, 2001.

A. Darwiche. Recursive conditioning. *Artificial Intelligence Journal*, 125(1–2):5–41, 2001.

K. V. Delgado, S. Sanner, et L. N. de Barros. Efficient solutions to factored MDPs with imprecise transition probabilities. *Artificial Intelligence*, 175(9–10):1498–1527, 2011. doi : http://dx.doi.org/10.1016/j.artint.2011.01.001. URL `http://www.sciencedirect.com/science/article/pii/S0004370211000026`.

R. Descartes. *Discours de la méthode pour bien conduire la raison, et chercher la vérité dans les sciences*. 1637.

M. Dorigo et L. M. Gambardella. Ant colonies for the traveling salesman problem. *BioSystems*, 43:73–81, 1997a.

M. Dorigo et L. M. Gambardella. Ant colony system : A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997b.

N. Drougard, F. Teichteil-Konigsbuch, J.-L. Farges, et D. Dubois. Qualitative possibilistic mixed-observable MDPs. Dans *Proceedings of the Twenty-Ninth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI'13)*, 2013.

N. Drougard. *Exploiting Imprecise Information Sources in Sequential Decision Making Problems under Uncertainty*. Thèse de doctorat, Institut Supérieur de l'Aéronautique et de l'Espace (ISAE), décembre 2015. URL `https://hal.archives-ouvertes.fr/tel-01296749`.

D. Dubois et H. Prade. Possibility theory as a basis for qualitative decision theory. Dans C. Mellish, éditeur, *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 1924–1930, San Francisco, 1995. Morgan Kaufmann.

M. Duff. *Optimal learning : Computational procedures for Bayes-adaptive Markov decision processes*. Thèse de doctorat, University of Massassachusetts Amherst, 2002.

A. Dutech, B. Scherrer, et C. Thiery. La carotte et le bâton... et Tetris. *Interstices*, 2008. URL `https://interstices.info/jcms/c_32764/la-carotte-et-le-baton-et-tetris`.

S. Dzeroski, L. D. Raedt, et K. Driessens. Relational reinforcement learning. *Machine Learning*, 43:7–52, 2001.

M. Ehrgott. *Multicriteria Optimization*, volume 491 de *Lecture Notes in Economics and Mathematical Systems*. Springer, 2005.

R. Emery-Montemerlo, G. Gordon, J. Schneider, et S. Thrun. Approximate solutions for partially observable stochastic games with common payoffs. Dans *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'04)*, July 2004.

T. Erez et W. Smart. A scalable method for solving high-dimensional continuous POMDPs using local approximation. Dans *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence (UAI'10)*, 2010.

T. Everitt, D. Filan, M. Daswani, et M. Hutter. Self-modification of policy and utility function in rational agents. Dans *Proceedings of the International Conference on Artificial General Intelligence (AGI'16)*, 2016.

E. Fabre, L. Jezequel, P. Haslum, et S. Thiébaux. Cost-optimal factored planning : Promises and pitfalls. Dans *Proceedings of the International Conference on Automated Planning and Scheduling*, 2010. URL `http://www.aaai.org/ocs/index.php/ICAPS/ICAPS10/paper/view/1442`.

H. Fargier, J. Lang, et R. Sabbadin. Towards qualitative approaches to multi-stage decision making. *International Journal of Approximate Reasoning*, 19:441–471, 1998.

H. Fargier et R. Sabbadin. Qualitative decision under uncertainty : back to expected utility. *Artificial Intelligence*, 164(1–2):245–280, 2005. doi : http://dx.doi.org/10.1016/j.artint.2004.12.002.

A. Fern, S. Yoon, et R. Givan. Approximate policy iteration with a policy language bias : Solving relational Markov decision processes. *Journal of Artificial Intelligence Research*, 25:85–118, 2006.

R. E. Fikes et N. J. Nilsson. STRIPS : A new approach to the application of theorem proving to problem solving. Dans *Proceedings of the Second International Joint Conference on Artificial Intelligence (IJCAI-71)*, 1971. URL `http://dl.acm.org/citation.cfm?id=1622876.1622939`.

R. Floyd. Non-deterministic algorithms. *Journal of the ACM*, 14(4):636–644, October 1967. URL `http://doi.acm.org/10.1145/321420.321422`.

T. Fortmann, Y. Bar-Shalom, et M. Scheffé. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE J. Oceanic Eng.*, 8, July 1983.

D. Fudenberg et J. Tirole. *Game Theory*. The MIT Press, 1991.

L. Galand et P. Perny. Search for compromise solutions in multiobjective state space graphs. Dans *Proceedings of the seventeenth European Conference on Artificial Intelligence*, 2006. URL `http://www-desir.lip6.fr/publications/pub_447_1_lgppecai06.pdf`.

M. Ghavamzadeh, S. Mannor, J. Pineau, et A. Tamar. Bayesian reinforcement learning : A survey. *Foundations and Trends in Machine Learning*, 8(5–6):359–483, 2015.

R. Givan, S. Leach, et T. Dean. Bounded parameter Markov decision processes. *Artificial Intelligence*, 122(1-2):71–109, 2000.

S. de Givry, F. Heras, M. Zytnicki, et J. Larrosa. Existential arc consistency : Getting closer to full arc consistency in weighted CSPs. Dans *Proceedings of the International Joint Conference Artificial Intelligence*, pages 84–89, 2005.

D. E. Goldberg et J. Richardson. Genetic algorithms with sharing for multimodal function optimization. Dans *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*, pages 41–49, 1987.

C. Gretton et S. Thiébaux. Exploiting first-order regression in inductive policy selection. Dans *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence (UAI'04)*, 2004.

C. Gretton. Gradient-based relational reinforcement-learning of temporally extended policies. Dans *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling (ICAPS'07)*, 2007.

A. Guez, D. Silver, et P. Dayan. Scalable and efficient Bayes-adaptive reinforcement learning based on Monte-Carlo tree search. *Journal of Artificial Intelligence Research*, 48:841–883, 2013. doi : 10.1613/jair.4117.

C. Guestrin, D. Koller, C. Gearhart, et N. Kanodia. Generalizing plans to new environments in relational MDPs. Dans *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'03)*, August 2003a.

C. Guestrin, D. Koller, R. Parr, et S. Venkataraman. Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research (JAIR)*, 19:399–468, 2003b.

E. Hansen et S. Zilberstein. Heuristic search in cyclic and/or graphs. Dans *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI'98)*, 1998.

E. Hansen et S. Zilberstein. LAO* : A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129:35–62, 2001.

E. Hansen, D. Bernstein, et S. Zilberstein. Dynamic programming for partially observable stochastic games. Dans *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI'04)*, 2004.

N. Hansen. The CMA evolution strategy : A tutorial. Technical report, LRI, June 2011. URL `http://www.lri.fr/~{}hansen/cmatutorial.pdf`.

P. Hart, N. Nilsson, et B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, July 1968. doi : 10.1109/TSSC.1968.300136.

D. Harel et A. Pnueli. *On the development of reactive systems*. Springer, 1985.

M. Hauskrecht. Incremental methods for computing bounds in partially observable Markov decision processes. Dans *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI)*, 1997.

V. Heidrich-Meisner et C. Igel. Evolution strategies for direct policy search. Dans *Proceedings of the Tenth international conference on Parallel Problem Solving from Nature (PPSN X)*, 2008.

C. de la Higuera. A bibliographical study of grammatical inference. *Pattern Recognition*, 38(9):1332–1348, 2005. doi : http://dx.doi.org/10.1016/j.patcog.2005.01.003.

J. Hoey, R. St-Aubin, A. Hu, et C. Boutilier. Spudd : Stochastic planning using decision diagrams. Dans *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI'99)*, 1999.

J. Hoey et P. Poupart. Solving POMDPs with continuous or large discrete observation spaces. Dans *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'05)*, 2005.

J. Hoffmann et B. Nebel. The FF planning system : Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

J. Hoffmann. The Metric-FF planning system : Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research (JAIR)*, 20:291–341, 2003.

R. Howard et J. Matheson. Risk-sensitive Markov decision processes. *Management Science*, (18):356–369, 1972.

M. Humphrys. *Action Selection methods using Reinforcement Learning.* Thèse de doctorat, University of Cambridge, 1997.

F. Hutter, H. H. Hoos, K. Leyton-Brown, et T. Stützle. ParamILS : An automatic algorithm configuration framework. *Journal of Artificial Intelligence Research (JAIR)*, 36:267–306, 2009.

M. R. James, S. Singh, et M. Littman. Planning with predictive state representations. Dans *Proceedings of the International Conference on Machine Learning and Applications (ICMLA)*, pages 304–311, 2004.

S. Kakade. *On the Sample Complexity of Reinforcement Learning.* Thèse de doctorat, University College London, 2003.

H. A. Kautz et B. Selman. Planning as satisfiability. Dans *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI'92)*, 1992.

M. Kearns, Y. Mansour, et A. Ng. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Machine Learning*, 49:193–208, 2002.

Z. Khan, T. Balch, et F. Dellaert. MCMC-based particle filtering for tracking a variable number of interacting targets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27, 2005.

G. Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

J. Kober et J. Peters. Policy search for motor primitives in robotics. *Machine Learning*, 84:171–203, 2011.

L. Kocsis et C. Szepesvari. Bandit based Monte-Carlo planning. Dans *Proceedings of the Seventeenth European Conference on Machine Learning (ECML'06)*, 2006.

A. Kolobov, Mausam, D. S. Weld, et H. Geffner. Heuristic search for generalized stochastic shortest path MDPs. Dans *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS'11)*, 2011.

J. Kolter et A. Ng. Near-Bayesian exploration in polynomial time. Dans *Proceedings of the Twenty-Sixth International Conference on Machine Learning (ICML'09)*, 2009.

R. Korf. Real-time heuristic search. *Artificial Intelligence*, 42(2-3):189–211, 1990.

H. Kurniawati, D. Hsu, et W. Lee. SARSOP : Efficient point-based POMDP planning by approximating optimally reachable belief spaces. Dans *Robotics : Science and Systems IV*, 2008.

A. Kumar, S. Zilberstein, et M. Toussaint. Scalable multiagent planning using probabilistic inference. Dans *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI'11)*, pages 2140–2146, 2011.

S. Lemai-Chenevier et M.-C. Charmeau. Exemple de satellite d'observation de la terre simplifié - architecture matérielle. Technical Report AGATA-NT-AA_AR-19-CN, CNES, 2006.

S. Lemai-Chenevier, A. Orlandini, F. Perrot, F. Py, et G. Verfaillie. Exemple de satellite d'observation de la terre simplifié - entrées pour l'élaboration de modèles de planification, contrôle d'exécution et diagnostic. Technical Report AGATA, CNES, LAAS/CNRS, ONERA, 2006.

M. L. Littman et P. Stone. A polynomial-time nash equilibrium algorithm for repeated games. Dans *Proceedings 4th ACM Conference on Electronic Commerce (EC-2003)*, 2003. doi : 10.1145/779928.779935. URL `http://doi.acm.org/10.1145/779928.779935`.

M. Littman et C. Szepesvári. A generalized reinforcement learning model : Convergence and applications. Dans *Proceedings of the International Conference on Machine Learning (ICML'96)*, 1996.

I. Little, D. Aberdeen, et S. Thiébaux. Prottle : A probabilistic temporal planner. Dans *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI'05)*, 2005.

M. Littman. Markov games as a framework for multi-agent reinforcement learning. Dans *Proceedings of the Eleventh International Conference on Machine Learning (ICML'94)*, 1994.

M. Littman. *Algorithms for Sequential Decision Making*. Thèse de doctorat, Brown University, Department of Computer Science, Providence, 1996.

M. Littman et S. S. R. Sutton. Predictive representations of state. Dans *Advances in Neural Information Processing Systems 14 (NIPS'01)*, 2001.

J. S. Liu et R. Chen. Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93:1032–1044, 1998.

J. Liu, M. Chu, et J. Reich. Multi-target Tracking in distributed sensor networks. *Signal Processing Magazine, IEEE*, 24(3):36–46, 2007.

W. Lovejoy. Computationally feasible bounds for partially observed Markov decision processes. *Operations Research*, 39(1):162–175, 1991.

J. Lucángeli Obes, C. Sarraute, et G. Richarte. Attack planning in the real world. Dans *Proceedings of the AAAI 2010 workshop on Intelligent Security (SecArt'10)*, 2010.

M. Machina. Dynamic consistency and non-expected utility models of choice under uncertainty. *Journal of Economic Literature*, 27:1622–1668, December 1989.

L. C. MacDermed et C. Isbell. Solving stochastic games. Dans *Advances in Neural Information Processing Systems 22*, 2013.

C. Manfredotti, D. J. Fleet, H. J. Hamilton, et S. Zilles. Simultaneous tracking and activity recognition. Dans *Proceedings of the International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 189–196, Washington, DC, USA, 2011.

S. Mannor, R. Rubinstein, et Y. Gat. The cross entropy method for fast policy search. Dans *Proceedings of the Twentieth International Conference on Machine Learning (ICML'03)*, pages 512–519, 2003.

J. Martin. *Bayesian decision problems and Markov chains*. Wiley, 1967.

Mausam et D. Weld. Planning with durative ations in stochastic domains. *Journal of Artificial Intelligence (JAIR)*, 31(1):33–82, 2008.

P. McCracken et M. Bowling. Online discovery and learning of predictive state representations. Dans *Advances in Neural Information Processing Systems 18 (NIPS'05)*, 2006.

D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, et D. Wilkins. PDDL—the planning domain definition language. Technical Report CVC TR98003/DCS TR1165, 1998.

G. A. Miller. The cognitive revolution : a historical perspective, trends in cognitive sciences. *Trends in Cognitive Sciences*, 7(3), March 2003.

G. Monahan. A survey of partially observable Markov decision processes. *Management Science*, 28:1–16, 1982.

M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, et S. Malik. Chaff : engineering an efficient SAT solver. Dans *Proceedings of the Thirty-Eighth Conference on Design Automation*, pages 530–535, 2001.

R. Munos. *Processus décisionnels de Markov en intelligence artificielle*, volume 2, chapter 1- Programmation dynamique avec approximation de la fonction de valeur, pages 19–50. Lavoisier - Hermes Science Publications, 2008.

R. Munos. Optimistic optimization of a deterministic function without the knowledge of its smoothness. Dans *Advances in Neural Information Processing Systems (NIPS'11)*, 2011.

R. Munos. From bandits to Monte-Carlo Tree Search : The optimistic principle applied to optimization and planning. *Foundations and Trends in Machine Learning*, 7(1):1–130, 2014. URL `http://hal.archives-ouvertes.fr/hal-00747575`.

C. Murray et G. J. Gordon. Multi-robot negotiation : Approximating the set of subgame perfect equilibria in general-sum stochastic games. Dans *Advances in Neural Information Processing Systems 19 (NIPS'06)*, 2006.

C. Murray et G. J. Gordon. Finding correlated equilibria in general sum stochastic games. Technical Report CMU-ML-07-113, Carnegie Mellon University, 2007.

R. Nair, P. Varakantham, M. Tambe, et M. Yokoo. Networked distributed POMDPs : A synthesis of distributed constraint optimization and POMDPs. Dans *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, pages 133–139, 2005.

N. Nilsson. *Principles of artificial intelligence*. Morgan Kaufmann Publishers, 1980.

R. Nissim et R. Brafman. Distributed heuristic forward search for multi-agent planning. *Journal of Artificial Intelligence Research (JAIR)*, 51(1):293–332, septembre 2014. URL `http://dl.acm.org/citation.cfm?id=2750423.2750431`.

F. Oliehoek, M. T. J. Spaan, C. Amato, et S. Whiteson. Incremental clustering and expansion for faster optimal planning in Dec-POMDPs. *Journal of Artificial Intelligence Research*, 46:449–509, 2013.

L. Orseau et S. Armstrong. Safely interruptible agents. Dans *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence (UAI'16)*, 2016.

M. J. Osborne et A. Rubinstein. *A Course in Game Theory*. The MIT Press, 1994.

P. Pattison et G. Robins. Neighborhood-based models for social networks. *Sociological Methodology*, 32(1):301–337, 2002.

P. Perny, O. Spanjaard, et P. Weng. Algebraic Markov decision processes. Dans *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, 2005.

P. Perny et P. Weng. On finding compromise solutions in multiobjective Markov decision processes. Dans *In proceedings of the ECAI Multidisciplinary Workshop on Advances in Preference Handling*, pages 55–60, 2010.

A. Petrowski. A clearing procedure as a niching method for genetic algorithms. Dans *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 798–803, May 1996.

J. Peters, S. Vijayakumar, et S. Schaal. Natural actor-critic. Dans J. Gama, R. Camacho, P. Brazdil, A. Jorge, et L. Torgo, éditeurs, *Proceedings of the Sixteenth European Conference on Machine Learning (ECML'05)*, volume 3720 de *Lecture Notes in Computer Science*. Springer-Verlag, October 2005.

J. Pineau, G. Gordon, et S. Thrun. Point-based value iteration : An anytime algorithm for POMDPs. Dans *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1025–1032, 2003.

J. Pineau, G. Gordon, et S. Thrun. Anytime point-based approximations for large POMDPs. *Journal of Artificial Intelligence Research (JAIR)*, 27:335–380, 2006.

J. Porta, N. Vlassis, M. T. Spaan, et P. Poupart. Point-based value iteration for continuous POMDPs. *Journal of Machine Learning Research*, 7:2329–2367, 2006.

P. Poupart, K.-E. Kim, et D. Kim. Closing the gap : Improved bounds on optimal POMDP solutions. Dans *Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling (ICAPS)*, 2011.

P. Poupart, N. Vlassis, J. Hoey, et K. Regan. An analytic solution to discrete Bayesian reinforcement learning. Dans *Proceedings of the Twenty-Third International Conference on Machine Learning (ICML'06)*, 2006.

J. Prévert. *Paroles*, chapter Inventaire. 1946.

M. L. Puterman. *Markov Decision Processes – Discrete Stochastic Dynamic Programming*. John Wiley and Sons, Inc., New York, USA, 1994.

K. Regan et C. Boutilier. Robust online optimization of reward-uncertain MDPs. Dans *Proceedings of the Twenty-Second Joint Conference on Artificial Intelligence (IJCAI'11)*, 2011.

D. B. Reid. An algorithm for tracking multiple targets. *IEEE Trans. Automat. Contr.*, 24:843–854, 1979.

C. Reynolds. Steering behaviors for autonomous characters. Dans *Game Developers Conference 1999*, pages 763–782, 1999.

D. M. Roijers, P. Vamplew, S. Whiteson, et R. Dazeley. A survey of multi-objective sequential decision-making. 48:67–113, 2013. URL `http://dx.doi.org/10.1613/jair.3987`.

G. A. Rummery et M. Niranjan. On-line Q-learning using connectionist systems. CUED/F-INFENG/TR 166, Cambridge University Engineering Department, September 1994. URL `ftp://svr-ftp.eng.cam.ac.uk/reports/rummery_tr166.ps.Z`.

S. Russell et P. Norvig. *Artificial Intelligence : A Modern Approach*. Englewood Cliffs, NJ : prentice Hall, 2010.

S. Russell. Rationality and intelligence. *Artificial Intelligence*, 94(1–2):57–77, 1997. URL `http://dx.doi.org/10.1016/S0004-3702(97)00026-X`.

R. Sabbadin. Possibilistic Markov decision processes. *Engineering Applications of Artificial Intelligence*, 14(3):287–300, 2001. doi : http://dx.doi.org/10.1016/S0952-1976(01)00007-0.

A. Saffidine, H. Finnsson, et M. Buro. Alpha-Beta pruning for games with simultaneous moves. Dans *Proceedings of the 26th AAAI Conference (**AAAI**)*, pages 556–562, Toronto, Canada, juillet 2012. AAAI Press.

S. Sanner et C. Boutilier. Probabilistic planning via linear value-approximation of first-order MDPs. Dans *Proceedings of the Fifth International Planning Competition (IPC-5)*, 2006.

S. Sanner et C. Boutilier. Approximate solution techniques for factored first-order MDPs. Dans *Proceedings of the Seventeenth Conference on Automated Planning and Scheduling (ICAPS'07)*, 2007.

C. Sarraute, G. Richarte, et J. Lucangeli. An algorithm to find optimal attack paths in nondeterministic scenarios. Dans *ACM Workshop on Artificial Intelligence and Security (AISec'11)*, 2011.

D. Schulz, W. Burgard, D. Fox, et A. B. Cremers. People tracking with a mobile robot using sample-based joint probabilistic data association filters. *Int. Journal of Robotics Research*, 22(2):99–116, 2003.

B. Scherrer. Approximate policy iteration schemes : A comparison. Dans *Proceedings of the International Conference on Machine Learning (ICML'14)*, 2014.

S. Seuken et S. Zilberstein. Improved memory-bounded dynamic programming for decentralized POMDPs. Dans *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence (UAI'07)*, 2007.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, et D. Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529 (7587):484–489, janvier 2016. URL `http://dx.doi.org/10.1038/nature16961`.

D. Silver et J. Veness. Monte-Carlo planning in large POMDPs. Dans *Advances in Neural Information Processing Systems 24 (NIPS'10)*, 2010.

R. Simmons et S. Koenig. Probabilistic robot navigation in partially observable environments. Dans *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'95)*, 1995.

S. Singh, T. Jaakkola, et M. Jordan. Learning without state estimation in partially observable Markovian decision processes. Dans *Proceedings of the Eleventh International Conference on Machine Learning (ICML'94)*, July 1994.

R. Smallwood et E. Sondik. The optimal control of partially observable Markov decision processes over a finite horizon. *Operation Research*, 21:1071–1088, 1973.

B. de Smit et H. W. Lenstra Jr.. Artful mathematics : The heritage of M. C. Escher. *Notices of the AMS*, 50(4):446–457, April 2003.

T. Smith et R. Simmons. Heuristic search value iteration for POMDPs. Dans *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, 2004.

T. Smith et R. Simmons. Point-based POMDP algorithms : Improved analysis and implementation. Dans *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence (UAI)*, 2005.

T. Smith. *Probabilistic Planning for Robotic Exploration.* Thèse de doctorat, The Robotics Institute, Carnegie Mellon University, 2007.

E. Sondik. *The Optimal Control of Partially Observable Markov Decision Processes.* Thèse de doctorat, Standford University, 1971.

J. Sorg, S. Singh, et R. Lewis. Variance-based rewards for approximate Bayesian reinforcement learning. Dans *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, 2010.

M. Spaan et N. Vlassis. Perseus : Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220, 2005. URL `http://www.aaai.org/Papers/JAIR/Vol24/JAIR-2406.pdf`.

H. von Stackelberg. *Market Structure and Equilibrium.* Springer, 2011. 1st Edition Translation into English, Bazin, Urch and Hill.

B. S. Stewart et C. C. White, III. Multiobjective A*. *Journal of the ACM*, 38(4):775–814, octobre 1991. ISSN 0004-5411. doi : 10.1145/115234.115368.

F. Stulp et O. Sigaud. Policy improvement : Between black-box optimization and episodic reinforcement learning. Dans *Journées Francophones Planification, Décision, et Apprentissage pour la conduite de systèmes*, 2013a.

F. Stulp et O. Sigaud. Robot skill learning : From reinforcement learning to evolution strategies. *Paladyn. Journal of Behavioral Robotics*, 4(1):49–61, September 2013b.

R. Sutton. Integrated architecture for learning, planning, and reacting based on approximating dynamic programming. Dans *Proceedings of the 7th International Conference on Machine Learning (ICML'90)*, pages 216–224, 1990.

R. Sutton et G. Barto. *Reinforcement Learning : an introduction*. Bradford Book, MIT Press, Cambridge, MA, 1998.

D. Szer, F. Charpillet, et S. Zilberstein. MAA* : A heuristic search algorithm for solving decentralized POMDPs. Dans *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI'05)*, 2005. URL `https://hal.inria.fr/inria-00000204`.

C. Szepesvári et M. Littman. Generalized Markov decision processes : Dynamic-programming and reinforcement-learning algorithms. Technical Report (CS-96-11), Brown University, Department of Computer Science, Providence, RI, 1996.

I. Szita et A. Lörincz. Learning tetris using the noisy cross-entropy method. *Neural Computation*, 18:2936–2941, 2006.

O. Teytaud et J. Liu. Jeu de go par ordinateur : Google-Deepmind vs Fan-Hui, la machine gagne 5-0., 2016. URL `https://docs.google.com/document/d/1lBh-2l9LvkVU6ArZo5e5iRo6ODClYZ8Ak4Ue3tXsPKs`.

E. Theodorou, J. Buchli, et S. Schaal. A generalized path integral control approach to reinforcement learning. *Journal of Machine Learning Research*, 11:3137–3181, 2010.

C. Thiéry et B. Scherrer. Building controllers for Tetris. *International Computer Games Association Journal*, 2009.

C. Thiéry. *Itération sur les politiques optimiste et apprentissage du jeu de Tetris*. Thèse de doctorat, Université Henri Poincaré – Nancy 1, 2010.

A. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42), 1936.

A. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, oct 1950.

T. Tyrrell. *Computational Mechanisms for Action Selection*. Thèse de doctorat, University of Edinburgh, 1993.

I. Wagner, M. Lindenbaum, et A. Bruckstein. Distributed covering by ant-robots using evaporating traces. *IEEE Transactions on Robotics and Automation*, 15(5), october 1999.

A. Wald. Statistical decision functions which minimize the maximum risk. *The Annals of Mathematics*, 46(2):265–280, 1945.

T. Wang, D. Lizotte, M. Bowling, et D. Schuurmans. Bayesian sparse sampling for on-line reward optimization. Dans *Proceedings of the Twenty-Second international conference on Machine learning (ICML'05)*, 2005.

C. Watkins. *Learning from delayed rewards*. Thèse de doctorat, King's College of Cambridge, UK., 1989.

C. Watkins et P. Dayan. q-learning. *Machine Learning*, 8:279–292, 1992.

A. Weinstein et M. L. Littman. Bandit-based planning and learning in continuous-action Markov decision processes. Dans *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling (ICAPS'12)*, 2012.

C. Wells, C. Lusena, et J. Goldsmith. Genetic algorithms for approximating solutions to POMDPs. Technical Report 290-99, Univ. of Kentuckty CS Dept, 1999.

P. Weng. *Modèles qualitatifs et approches algébriques pour la décision dans l'incertain : fondements axiomatiques et application à la décision séquentielle*. Thèse de doctorat, Université Pierre et Marie Curie, décembre 2006.

D. H. Wilson et C. Atkeson. Simultaneous tracking and activity recognition (STAR) using many anonymous, binary sensors. Dans *Proceedings of the Third International Conference on Pervasive Computing*, pages 62–79, Berlin, Heidelberg, 2005. Springer-Verlag.

R. J. Williams et L. C. I. Baird. Tight performance bounds on greedy policies based on imperfect value functions. Technical Report NU-CCS-93-14, Northeastern University, 1993.

R. Williams. A class of gradient-estimating algorithms for reinforcement learning in neural networks. Dans *Proceedings of the First International Conference on Neural Networks (ICNN'87)*, 1987.

R. Williams. Simple statistical gradient-following algorithms for connectionnist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.

C. Wu et Y. Lin. Minimizing risk models in Markov decision processes with policies depending on target values. *Journal of Mathematical Analysis and Applications*, (231):47–67, 1999.

Z. Xing, J. Pei, et E. Keogh. A brief survey on sequence classification. *ACM SIGKDD Explorations*, 12(1):40–48, June 2010.

S. Yoon, A. Fern, et B. Givan. FF-Replan : a baseline for probabilistic planning. Dans *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling (ICAPS'07)*, September 2007.

H. Younes. Extending PDDL to model stochastic decision processes. Dans *Proceedings of the ICAPS'03 Workshop on PDDL*, 2003.

H. L. S. Younes et M. L. Littman. PPDDL1.0 : An extension to PDDL for expressing planning domains with probabilistic effects. Technical Report CMU-CS-04-167, Carnegie Mellon University, October 2004.

H. L. S. Younes, M. L. Littman, D. Weissman, et J. Asmuth. The first probabilistic track of the international planning competition. *Journal of Artificial Intelligence Research*, 24:851–887, 2005.

Z. Zamani, S. Sanner, P. Poupart, et K. Kersting. Symbolic dynamic programming for continuous state and observation POMDPs. Dans *Advances in Neural Information Processing Systems 25 (NIPS'12)*, 2012.

M. Zinkevich, A. Greenwald, et M. L. Littman. Cyclic equilibria in Markov games. Dans *Advances in Neural Information Processing Systems (NIPS'05)*, pages 1641–1648.

Publications

Articles de revues internationales avec comité de lecture

M. Steinmetz, J. Hoffmann, et O. Buffet. Goal probability analysis in MDP probabilistic planning : Exploring and enhancing the state of the art. *Journal of Artificial Intelligence Research*, 57:229–271, 2016a. URL http://www.jair.org/papers/paper5153.html.

J. Dibangoye, C. Amato, O. Buffet, et F. Charpillet. Optimally solving Dec-POMDPs as continuous-state MDPs. *Journal of Artificial Intelligence Research*, 55:443–497, 2016. URL http://www.jair.org/papers/paper4623.html.

O. Buffet et D. Aberdeen. The factored policy-gradient planner. *Artificial Intelligence*, 173(5-6):722–747, 2009. URL http://dx.doi.org/10.1016/j.artint.2008.11.008.

O. Buffet. Reachability analysis for uncertain SSPs. *International Journal on Artificial Intelligence Tools*, 16(4):725–749, 2007a.

O. Buffet, A. Dutech, et F. Charpillet. Shaping multi-agent systems with gradient reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 15(2):197–220, 2007.

Articles de revues nationales avec comité de lecture

M. Tlig, O. Buffet, et O. Simonin. Intersections intelligentes pour le contrôle de véhicules sans pilote. coordination locale et optimisation globale. *Revue d'Intelligence Artificielle*, 30(3):353–382, 2016. URL http://dx.doi.org/10.3166/ria.30.353-382.

O. Buffet, A. Dutech, et F. Charpillet. Etude de différentes combinaisons de comportements adaptatives. *Revue d'Intelligence Artificielle*, 20(2–3):311–344, 2006.

O. Buffet, A. Dutech, et F. Charpillet. Développement autonome des comportements de base d'un agent. *Revue d'Intelligence Artificielle*, 19(4–5):603–632, Septembre 2005.

Chapitres de livres

O. Buffet. *Processus décisionnels de Markov en intelligence artificielle*, volume 2, chapter 3 - Méthodes de gradient pour la recherche de politiques paramétrées. Lavoisier - Hermes Science Publications, 2008.

O. Buffet. *Markov Decision Processes in Artificial Intelligence*, chapter 5 - Policy-Gradient Algorithms. ISTE/Wiley, 2010.

S. Thiébaux et O. Buffet. *Markov Decision Processes in Artificial Intelligence*, chapter 15 - Operations Planning. ISTE/Wiley, 2010.

S. Thiébaux et O. Buffet. *Processus décisionnels de Markov en intelligence artificielle*, volume 2, chapter 8 - Planification d'opérations. Lavoisier - Hermes Science Publications, 2008.

Conférences internationales avec comité de lecture

M. Steinmetz, J. Hoffmann, et O. Buffet. Revisiting goal probability analysis in probabilistic planning. Dans *Proceedings of the 26th International Conference on Automated Planning and Scheduling (ICAPS'16)*, 2016b.

J. Dibangoye, O. Buffet, et O. Simonin. Structural results for cooperative decentralized control models. Dans *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-15)*, 2015a.

J. Dibangoye, C. Amato, O. Buffet, et F. Charpillet. Exploiting separability in multiagent planning with continuous-state MDPs (extended abstract). Dans *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-15) [Best Papers From Sister Conferences Track]*, 2015b.

J. Dibangoye, O. Buffet, et F. Charpillet. Error-bounded approximations for infinite-horizon discounted decentralized POMDPs. Dans *Proceedings of the European Conference on Machine Learning (ECML/PKDD-14)*, 2014a.

M. Krajnanský, J. Hoffmann, O. Buffet, et A. Fern. Learning pruning rules for heuristic search planning. Dans *Proceedings of the Twenty-first European Conference on Artificial Intelligence (ECAI-14)*, 2014a.

A. Fansi Tchango, V. Thomas, O. Buffet, F. Flacher, et A. Dutech. Simultaneous tracking and activity recognition (star) using advanced agent-based behavioral simulations. Dans *Proceedings of the Twenty-first European Conference on Artificial Intelligence (ECAI-14)*, 2014a.

M. Tlig, O. Buffet, et O. Simonin. Stop-free strategies for traffic networks : Decentralized on-line optimization. Dans *Proceedings of the Twenty-first European Conference on Artificial Intelligence (PAIS/ECAI-14)*, 2014a.

A. Fansi Tchango, V. Thomas, O. Buffet, F. Flacher, et A. Dutech. Towards the usage of advanced behavioral simulations for simultaneous tracking and activity recognition. Dans *Proceedings of the Seventh European Starting AI Researcher Symposium (STAIRS-14)*, 2014b.

A. Fansi Tchango, V. Thomas, O. Buffet, A. Dutech, et F. Flacher. Tracking multiple interacting targets using a joint probabilistic data association filter. Dans *Proceedings of the Seventeenth International Conference on Information Fusion (Fusion-14)*, 2014c.

M. Tlig, O. Buffet, et O. Simonin. Decentralized traffic management : A synchronization-based intersection control. Dans *Proceedings of the Third International Conference on Advanced Logistics and Transport (ICALT-14) / Symposium on Intelligent Transportation Systems (ITS)*, 2014b.

J. Dibangoye, C. Amato, O. Buffet, et F. Charpillet. Exploiting separability in multi-agent planning with continuous-state MDPs. Dans *Proceedings of the Thirteenth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-14)*, 2014b.

A. Fansi Tchango, V. Thomas, O. Buffet, F. Flacher, et A. Dutech. Simulation-based behavior tracking of pedestrians in partially observed indoor environments. Dans *Proceedings of the Thirteenth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-14)*, 2014d.

J. Dibangoye, C. Amato, O. Buffet, et F. Charpillet. Optimally solving Dec-POMDPs as continuous-state MDPs. Dans *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI-13)*, 2013a.

S. Nicol, T. Iwamura, O. Buffet, et I. Chadès. Adaptive management of migratory birds under sea level rise. Dans *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI-13)*, 2013.

M. Tagorti, B. Scherrer, O. Buffet, et J. Hoffmann. Abstraction pathologies in Markov decision processes. Dans *Proceedings of the ICAPS-13 workshop on Heuristics and Search for Domain-Independent Planning (HSDIP-13)*, 2013a.

M. Tlig, O. Buffet, et O. Simonin. Reactive coordination rules for traffic optimization in road sharing problems. Dans *Proceedings of the PAAMS Workshop on Agent-based Approaches for the Transportation Modelling and Optimisation (AATMO-13)*, 2013a.

W.-T. Lin, O. Buffet, C.-S. Lee, et O. Teytaud. Optimistic heuristics for minesweeper. Dans *Proceedings of the International Computer Symposium (ICS-12)*, 2012.

M. Tlig, O. Buffet, et O. Simonin. Cooperative behaviors for the self-regulation of autonomous vehicles in space sharing conflicts. Dans *Proceedings of the Twenty-Fourth International Conference on Tools with Artificial Intelligence (ICTAI-12)*, 2012.

M. Araya-López, V. Thomas, et O. Buffet. Near-optimal BRL using optimistic local transitions. Dans *Proceedings of the Twenty-Ninth International Conference on Machine Learning (ICML-12)*, 2012a.

I. Chadès, J. Carwardine, T. G. Martin, S. Nicol, R. Sabbadin, et O. Buffet. MOMDPs : a solution for modelling adaptive management problems. Dans *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI-12)*, 2012.

C. Sarraute, O. Buffet, et J. Hoffmann. POMDPs make better hackers : Accounting for uncertainty in penetration testing. Dans *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI-12)*, 2012a.

D. Maxim, O. Buffet, L. Santinelli, L. Cucu-Grosjean, et R. Davis. Optimal priority assignment algorithms for probabilistic real-time systems. Dans *Proceedings of the 19th International Conference on Real-Time and Network Systems (RTNS-11)*, 2011.

M. Araya-López, O. Buffet, V. Thomas, et F. Charpillet. Active learning of MDP models. Dans *Proceedings of the Ninth European Workshop on Reinforcement Learning (EWRL-11)*, 2011a.

M. Godichaud, E. Chanthery, O. Buffet, et M. Contat. Formalizing and solving information collection problems with autonomous sensor systems. Dans *Proceedings of the Eighteenth IFAC World congress (IFAC-10)*, 2011a.

C. Sarraute, O. Buffet, et J. Hoffmann. Penetration testing == POMDP solving? Dans *Working Notes for the 2011 IJCAI Workshop on Intelligent Security (SecArt-11)*, 2011.

O. Buffet et L. Cucu-Grosjean. Impact of job dropping on the probabilistic schedulability of uniprocessor deterministic real-time systems. Dans *Online Proceedings of the Workshop EVOLVE - A bridge between Probability, Set Oriented Numerics and Evolutionary Computation*, 2011a.

M. Araya-López, O. Buffet, V. Thomas, et F. Charpillet. A POMDP extension with belief-dependent rewards. Dans *Advances in Neural Information Processing Systems 23 (NIPS-10)*, 2010a.

M. Araya-López, V. Thomas, O. Buffet, et F. Charpillet. A closer look at MOMDPs. Dans *Proceedings of the Twenty-Second IEEE International Conference on Tools with Artificial Intelligence (ICTAI-10)*, 2010b.

A. Brun, A. Hamad, O. Buffet, et A. Boyer. From "I like" to "I prefer" in collaborative filtering. Dans *Proceedings of the Twenty-Second IEEE International Conference on Tools with Artificial Intelligence (ICTAI-10)*, 2010a. [poster].

A. Glad, O. Simonin, O. Buffet, et F. Charpillet. Influence of different execution models on patrolling ant behaviors : from agents to robots. Dans *Proceedings of the Ninth International Conference on Autonomous Agents and MultiAgent Systems (AAMAS'10)*, 2010.

A. Glad, O. Buffet, O. Simonin, et F. Charpillet. Self-organization of patrolling-ant algorithms. Dans *Proceedings of the Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO'09)*, 2009a.

A. Glad, O. Simonin, O. Buffet, et F. Charpillet. Theoretical study of ant-based algorithms for multi-agent patrolling. Dans *Proceedings of the Eighteenth European Conference on Artificial Intelligence (ECAI'08)*, 2008.

O. Buffet et D. Aberdeen. FF+FPG : Guiding a policy gradient algorithm for planning. Dans *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling (ICAPS'07)*, September 2007a.

D. Aberdeen et O. Buffet. Temporal probabilistic planning with policy-gradients. Dans *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling (ICAPS'07)*, September 2007.

D. Aberdeen, O. Buffet, et O. Thomas. Policy-gradients for PSRs and POMDPs. Dans *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS'07)*, Mars 2007.

E. Kelareva, O. Buffet, J. Huang, et S. Thiébaux. Factored planning using decomposition trees. Dans *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI'07)*, Janvier 2007.

O. Buffet. Reachability analysis for uncertain SSPs. Dans *Proceedings of the Seventeenth IEEE International Conference on Tools with Artificial Intelligence (ICTAI'05)*, 2005a.

O. Buffet et D. Aberdeen. Robust planning with (L)RTDP. Dans *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*, 2005a.

O. Buffet, A. Dutech, et F. Charpillet. Self-growth of basic behaviors in an action selection based agent. Dans *From Animals to Animats 8 : Proceedings of the 8th International Conference on Simulation of Adaptive Behavior (SAB'04)*, 2004.

O. Buffet, A. Dutech, et F. Charpillet. Automatic generation of an agent's basic behaviors. Dans *Proceedings of the 2nd International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS'03)*, 2003a.

O. Buffet, A. Dutech, et F. Charpillet. Adaptive combination of behaviors in an agent. Dans *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI'02)*, 2002a.

O. Buffet, A. Dutech, et F. Charpillet. Learning to weigh basic behaviors in scalable agents. Dans *Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS'02)*, 2002b. [poster].

A. Dutech, O. Buffet, et F. Charpillet. Multi-agent systems by incremental gradient reinforcement learning. Dans *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI'01)*, 2001.

O. Buffet, A. Dutech, et F. Charpillet. Incremental reinforcement learning for designing multi-agent systems. Dans *Proceedings of the Fifth International Conference on Autonomous Agents (Agents'01)*, pages 31–31, 2001. [poster].

Présentations en conférences internationales

S. Naguleswaran, O. Buffet, et L. White. Dynamic programming using quantum search for optimizing petri net models. Dans *International Federation of Operational Research Societies (IFORS)*, Hawaï, USA, July 2005.

Conférences nationales avec comité de lecture

J. Dibangoye, O. Buffet, et O. Simonin. Résultats structurels pour les modèles de contrôle décentralisé coopératif. Dans *Actes des dixièmes journées francophones planification, décision, apprentissage pour la conduite de systèmes (JFPDA-14)*, 2014c.

M. Krajnanský, J. Hoffmann, O. Buffet, et A. Fern. Learning pruning rules for heuristic search planning. Dans *Actes des neuvièmes journées francophones planification, décision, apprentissage pour la conduite de systèmes (JFPDA-14)*, 2014b.

M. Tagorti, B. Scherrer, O. Buffet, et J. Hoffmann. Abstraction pathologies in Markov decision processes. Dans *Actes des huitièmes journées francophones planification, décision, apprentissage pour la conduite de systèmes (JFPDA-13)*, 2013b.

M. Araya-López, O. Buffet, et V. Thomas. Active diagnosis through belief-lookahead information gathering. Dans *Actes des huitièmes journées francophones planification, décision, apprentissage pour la conduite de systèmes (JFPDA-13)*, 2013.

J. Dibangoye, C. Amato, O. buffet, et F. Charpillet. Résoudre des Dec-POMDP optimalement comme des MDP à espace d'états continu. Dans *Actes des huitièmes journées francophones planification, décision, apprentissage pour la conduite de systèmes (JFPDA-13)*, 2013b.

M. Tlig, O. Buffet, et O. Simonin. Synchronisation de véhicules autonomes aux croisements d'un réseau de routes. Dans *Vingt-et-unième journées francophones sur les systèmes multi-agents (JFSMA-13)*, 2013b. [démonstration].

M. Tlig, O. Buffet, et O. Simonin. Synchronisation de véhicules autonomes aux croisements d'un réseau de routes. Dans *Actes des onzièmes rencontres jeunes chercheurs en intelligence artificielle (RJCIA-13)*, 2013c.

M. Araya-López, V. Thomas, et O. Buffet. BRL quasi-optimal à l'aide de transitions locales optimistes. Dans *Actes des septièmes journées francophones planification, décision, apprentissage pour la conduite de systèmes (JFPDA-12)*, 2012b.

M. Araya-López, O. Buffet, V. Thomas, et F. Charpillet. Apprentissage actif de modèle de MDP. Dans *Actes des sixièmes journées francophones planification, décision, apprentissage pour la conduite de systèmes (JFDPA'11)*, 2011b.

M. Araya-López, O. Buffet, V. Thomas, et F. Charpillet. Une extension des POMDP avec des récompenses dépendant de l'état de croyance. Dans *Actes de la conférence francophone sur l'apprentissage automatique (CAp'11)*, 2011c. [french version of the NIPS-10 paper].

O. Buffet et L. Cucu-Grosjean. Recherche systématique pour l'ordonnancement temps réel global multiprocesseur. Dans *Actes du douzième congrès annuel de la Société française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF'11)*, 2011b.

M. Araya-López, V. Thomas, O. Buffet, et F. Charpillet. Des POMDPs avec des variables d'état visibles. Dans *Actes des cinquièmes journées francophones planification, décision, apprentissage pour la conduite de systèmes (JFPDA'10)*, 2010c.

A. Brun, A. Hamad, O. Buffet, et A. Boyer. Vers l'utilisation de relations de préférence pour le filtrage collaboratif. Dans *Actes du dix-septième congrés francophone AFRIF-AFIA sur la Reconnaissance des Formes et l'Intelligence Artificielle (RFIA'10)*, 2010b.

O. Buffet et D. Aberdeen. Ff+fpg : Guider un planificateur basé sur une méthode de gradient. Dans *Actes des deuxièmes journées francophones planification, décision, apprentissage pour la conduite de systèmes (JFPDA'07)*, 2007b.

O. Buffet et D. Aberdeen. Planification robuste à l'aide d'une montée de gradient. Dans *Actes de la conférence francophone sur l'apprentissage automatique (CAp'06)*, 2006a.

O. Buffet et D. Aberdeen. Planification robuste avec (L)RTDP. Dans *Actes de la conférence francophone sur l'apprentissage automatique (CAp'05)*, 2005b.

C. Sarraute, O. Buffet, et J. Hoffmann. Les POMDP font de meilleurs hackers : Tenir compte de l'incertitude dans les tests de pénétration. Dans *Actes des septièmes journées francophones planification, décision, apprentissage pour la conduite de systèmes (JFPDA-12)*, 2012b.

I. Chadès, J. Carwardine, T. G. Martin, S. Nicol, et O. Buffet. Les POMDP : une solution pour modéliser des problèmes de gestion adaptative en biologie de la conservation. Dans *Actes des sixièmes journées francophones planification, décision, apprentissage pour la conduite de systèmes (JFDPA'11)*, 2011.

L. Sarzyniec, O. Buffet, et A. Dutech. Apprentissage par renforcement développemental en robotique autonome. Dans *Actes de la conférence francophone sur l'apprentissage automatique (CAp'11)*, 2011.

M. Godichaud, E. Chanthery, O. Buffet, et M. Contat. Formalisation et résolution de problèmes d'acquisition d'informations par des systèmes autonomes. Dans *Actes du douzième congrès annuel de la Société française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF'11)*, 2011b.

A. Glad, O. Buffet, O. Simonin, et F. Charpillet. Auto-organisation dans les algorithmes fourmis pour la patrouille multi-agent. Dans *Actes des quatrièmes journées francophones planification, décision, apprentissage pour la conduite de systèmes (JFPDA'09)*, 2009b. (french version of SASO'09 paper).

A. Dutech, O. Buffet, et F. Charpillet. Développement autonome des comportements de base d'un agent. Dans *Actes de la Conférence d'Apprentissage (CAp'04), Montpellier, France*, 2004.

O. Buffet, A. Dutech, et F. Charpillet. Apprentissage par renforcement pour la conception de systèmes multi-agents réactifs. Dans *Actes des Journées Francophones sur les Systèmes Multi-Agents (JFSMA'03)*, 2003b.

Workshops internationaux avec comité de lecture

A. Brun, A. Hamad, O. Buffet, et A. Boyer. Towards preference relations in recommender systems. Dans *ECML/PKDD Workshop on Preference Learning (PL-10)*, 2010c.

O. Buffet et L. Cucu-Grosjean. Impact of job dropping on the schedulability of uniprocessor probabilistic real-time systems with variable execution times. Dans *Proceedings of the First International Real-Time Scheduling Open Problems Seminar (RTSOPS 2010a), joint workshop with the 22nd Euromicro International Conference on Real-Time Systems (ECRTS 2010a)*, July 2010a.

O. Buffet et J. Hoffmann. All that glitters is not gold : Using landmarks for reward shaping in FPG. Dans *Proceedings of the ICAPS'10 Workshop on Planning and Scheduling under Uncertainty (PSUWS)*, May 2010.

L. Cucu-Grosjean et O. Buffet. Global multiprocessor real-time scheduling as a constraint satisfaction problem. Dans *Proceedings of the ICPP'09 Workshop on Real-time systems on multicore platforms : Theory and Practice (XRTS'09)*, September 2009.

D. Aberdeen et O. Buffet. Simulation methods for uncertain decision-theoretic planning. Dans *Proceedings of the IJCAI 2005 Workshop on Planning and Learning in A Priori Unknown or Dynamic Domains*, 2005.

O. Buffet et D. Aberdeen. The factored policy gradient planner (IPC-06 version). Dans A. Gerevini, B. Bonet, et B. Givan, éditeurs, *Proceedings of the Fifth International Planning Competition (IPC-5)*, pages 69–71, June 2006b. [Winner, probabilistic track of the 5th International Planning Competition].

O. Buffet et D. Aberdeen. Policy-gradient for robust planning. Dans *Proceedings of the ECAI'06 Workshop on Planning, Learning and Monitoring with Uncertainty and Dynamic Worlds (PLMUDW'06)*, 2006c.

O. Buffet. Fast reachability analysis for uncertain SSPs. Dans *Proceedings of the IJCAI 2005b Workshop on Planning and Learning in A Priori Unknown or Dynamic Domains*, 2005b.

O. Buffet et D. Aberdeen. A two-teams approach for robust probabilistic temporal planning. Dans *Proceedings of the ECML'05 workshop on Reinforcement Learning in Non-Stationary Environments*, 2005c.

O. Buffet et A. Dutech. A self-made agent based on action-selection. Dans *Proceedings of the 6th European Workshop on Reinforcement Learning (EWRL-6)*, 2003.

O. Buffet et A. Dutech. Looking for scalable agents. Dans *Proceedings of the Fifth European Workshop on Reinforcement Learning (EWRL-5)*, 2001.

Rapports de recherche

J. S. Dibangoye, C. Amato, O. Buffet, et F. Charpillet. Optimally solving Dec-POMDPs as continuous-state MDPs : Theory and algorithms. Technical Report RR-8517, INRIA, 2014d.

M. Tlig, O. Buffet, et O. Simonin. Decentralized traffic management : A synchronization-based intersection control – extended version. Technical Report RR-8500, INRIA, 2014c. [extended version of ICALT'14 paper].

M. Araya-López, V. Thomas, et O. Buffet. Near-optimal BRL using optimistic local transitions (extended version). Technical Report RR-7965, INRIA, 2012c. [extended version of ICML'12 paper].

M. Araya-López, O. Buffet, V. Thomas, et F. Charpillet. A POMDP extension with belief-dependent rewards (extended version). Technical Report RR-7433, INRIA, 2010d. [extended version of NIPS'10 paper].

O. Buffet. Rapport Agata : Proposition de modélisation pour le suivi de situation et la prise de décision. Technical Report projet Agata, novembre 2007b.

O. Buffet. Rapport Agata $t_0 + 5$ mois : Proposition de modélisation pour le suivi de situation et la prise de décision. Technical Report projet Agata, juin 2007c.

O. Buffet et D. Aberdeen. Robust probabilistic temporal planning : Dynamic programming vs policy-search. Technical report, 2005d.

O. Buffet. Robust (L)RTDP : Reachability analysis. Technical report, National ICT Australia, december 2004.

O. Buffet et D. Aberdeen. Planning with robust (L)RTDP. Technical report, National ICT Australia, november 2004.

O. Buffet et L. Cucu-Grosjean. Systematic searches for global multiprocessor real-time scheduling. Technical Report RR-7386, INRIA, 09 2010b. URL `http://hal.inria.fr/inria-00519324/en/`.

O. Buffet. Apprentissage par renforcement pour la conception de systèmes multi-agents. [rapport d'avancement de thèse], Laboratoire LOrrain de Recherche en Informatique et ses Applications (LORIA), 2002.

Logiciels distribués

O. Buffet et D. Aberdeen. [FPG-IPC] the factored policy gradient planner (IPC version). `http://fpg.loria.fr/`.

D. Aberdeen, O. Buffet, J. Yu, O. Thomas, et S. Richter. [libPG] the policy-gradient library. `http://code.google.com/p/libpgrl/`.

Mémoires

O. Buffet. *Une double approche modulaire de l'apprentissage par renforcement pour des agents intelligents adaptatifs*. Thèse de doctorat, Université Henri Poincaré, Nancy 1, septembre 2003. Laboratoire Lorrain de recherche en informatique et ses applications (LORIA).

O. Buffet. Apprentissage par renforcement dans un système multi-agents. Mémoire de D.E.A., Université Henri Poincaré (Nancy), 2000. Mémoire de DEA.

Editions (actes / livres)

O. Sigaud et O. Buffet, éditeurs. *Processus décisionnels de Markov en intelligence artificielle*, volume 1. Lavoisier - Hermes Science Publications, 2008.

O. Buffet et O. Sigaud, éditeurs. *Processus décisionnels de Markov en intelligence artificielle*, volume 2. Lavoisier - Hermes Science Publications, 2008.

J. Bidot, D. Bryce, O. Buffet, H. Palacios, et S. Sanner, éditeurs. *Proceedings of the ICAPS'10 Workshop on Planning and Scheduling under Uncertainty (PSUWS'10)*, Mai 2010.

O. Buffet et O. Sigaud, éditeurs. *Markov Decision Processes in Artificial Intelligence*. ISTE/Wiley, 2010. [version anglaise de Sigaud et Buffet (2008); Buffet et Sigaud (2008)].

O. Buffet et D. Bryce, éditeurs. *Booklet of the International Planning Competition 2008 : Uncertainty Part*, Septembre 2008.

U. Kuter, D. Aberdeen, O. Buffet, et P. Stone, éditeurs. *Proceedings of the ICAPS'07 Workshop on Artificial Intelligence Planning and Learning (AIPL'07)*, Septembre 2007.

A. Botea, O. Buffet, et M. Zanella, éditeurs. *Proceedings of the ECAI'06 Workshop on Planning, Learning and Monitoring with Uncertainty and Dynamic Worlds (PLMUDW'06)*, Août 2006. Università di Trento.

A. Dutech et O. Buffet, éditeurs. *Proceedings of the Sixth European Workshop on Reinforcement Learning (EWRL'03)*, Septembre 2003. INRIA.

Résumé

Ce mémoire présente des travaux dans le domaine de la prise de décision séquentielle (automatique) dans l'incertain. Il décrit d'abord les fondements scientifiques sur lesquels reposent la majeure partie des dits travaux : le formalisme des processus de décision markoviens (MDP) et certaines de ses extensions. Les contributions sont ensuite organisées en trois ensembles : (i) un premier sur la recherche et l'exploitation de structure dans les MDP, mais aussi sur la résolution de MDP par optimisation directe d'un contrôleur (par opposition à l'emploi de la programmation dynamique) ; (ii) le suivant sur la résolution de problèmes à observabilité partielle (POMDP) par optimisation directe d'un contrôleur, sur la résolution d'extensions de POMDP en se ramenant à un POMDP, et sur une approche optimiste pour l'apprentissage par renforcement bayésien ; et (iii) le dernier sur des travaux s'éloignant des MDP, mais mettant en jeu des systèmes dynamiques, des algorithmes de recherche, ou la théorie de l'utilité.

Mots-clés: processus de décision markoviens, apprentissage par renforcement, planification probabiliste, MDP, POMDP, DecPOMDP, AR.